openGPS ISO 5436-2 XML

0.02

Generated by Doxygen 1.5.6

Thu Oct 23 15:43:23 2008

Contents

1 openGPS ISO 5436-2 XML Library Documentation **1**

2 Todo List **1**

3 Module Index **1**

3.1 Modules 1

4 Directory Hierarchy **2**

4.1 Directories 2

5 Namespace Index **2**

5.1 Namespace List 2

6 Class Index **3**

6.1 Class Hierarchy 3

7 Class Index **6**

7.1 Class List 6

8 File Index **10**

8.1 File List 10

9 Module Documentation **15**

9.1 Ansi-C Interface for X3P library 15

9.1.1 Detailed Description 15

9.1.2 Define Documentation 25

9.1.3 Typedef Documentation 29

9.1.4 Enumeration Type Documentation 31

9.1.5 Function Documentation 31

9.2 Sample files demonstrating X3P file format. 62

9.3 C++ Interface for X3P-library 63

10 Directory Documentation **64**

10.1 ISO5436_2_XML/c/ Directory Reference 64

10.2 S:/openGPS/ISO5436_XML/trunk/include/opengps/cxx/ Directory Reference 65

10.3 ISO5436_2_XML/cxx/ Directory Reference 66

10.4 S:/openGPS/ISO5436_XML/trunk/include/ Directory Reference 69

10.5 tools/IncRev/ Directory Reference 70

10.6 ISO5436_2_XML/ Directory Reference 70

10.7 ISO5436_2_XML_Demo/ Directory Reference 71
10.8 S:/openGPS/ISO5436_XML/trunk/include/opengps/ Directory Reference 71
10.9 ISO5436_2_XML_Demo/samplefiles/ Directory Reference 72
10.10 tools/ Directory Reference 73
10.11 ISO5436_2_XML/xyssl/ Directory Reference 73

11 Namespace Documentation **73**
11.1 OpenGPS Namespace Reference 73
11.1.1 Detailed Description 73
11.1.2 Typedef Documentation 78
11.2 OpenGPS::Schemas Namespace Reference 79
11.2.1 Detailed Description 79
11.3 OpenGPS::Schemas::ISO5436_2 Namespace Reference 79
11.3.1 Detailed Description 79
11.3.2 Function Documentation 85
11.4 std Namespace Reference 111
11.4.1 Detailed Description 111
11.5 xml_schema Namespace Reference 115
11.5.1 Detailed Description 115
11.5.2 Typedef Documentation 117
11.5.3 Variable Documentation 120

12 Class Documentation **120**
12.1 OpenGPS::Schemas::ISO5436_2::AxesType Class Reference 120
12.1.1 Detailed Description 121
12.1.2 Member Typedef Documentation 126
12.1.3 Constructor & Destructor Documentation 127
12.1.4 Member Function Documentation 129
12.1.5 Member Data Documentation 136
12.2 OpenGPS::Schemas::ISO5436_2::AxisDescriptionType Class Reference 136
12.2.1 Detailed Description 136
12.2.2 Member Typedef Documentation 142
12.2.3 Constructor & Destructor Documentation 144
12.2.4 Member Function Documentation 146
12.2.5 Member Data Documentation 153
12.3 OpenGPS::Schemas::ISO5436_2::AxisType Class Reference 154
12.3.1 Detailed Description 154

12.3.2 Member Enumeration Documentation 156
12.3.3 Constructor & Destructor Documentation 156
12.3.4 Member Function Documentation 159
12.3.5 Member Data Documentation 161
12.4 OpenGPS::BinaryLSBPointVectorReaderContext Class Reference 161
12.4.1 Detailed Description 162
12.4.2 Constructor & Destructor Documentation 163
12.4.3 Member Function Documentation 164
12.5 OpenGPS::BinaryLSBPointVectorWriterContext Class Reference 167
12.5.1 Detailed Description 168
12.5.2 Constructor & Destructor Documentation 169
12.5.3 Member Function Documentation 169
12.6 OpenGPS::BinaryMSBPointVectorReaderContext Class Reference 173
12.6.1 Detailed Description 174
12.6.2 Constructor & Destructor Documentation 175
12.6.3 Member Function Documentation 175
12.7 OpenGPS::BinaryMSBPointVectorWriterContext Class Reference 179
12.7.1 Detailed Description 180
12.7.2 Constructor & Destructor Documentation 181
12.7.3 Member Function Documentation 181
12.8 OpenGPS::BinaryPointVectorReaderContext Class Reference 185
12.8.1 Detailed Description 186
12.8.2 Constructor & Destructor Documentation 187
12.8.3 Member Function Documentation 187
12.8.4 Member Data Documentation 190
12.9 OpenGPS::BinaryPointVectorWriterContext Class Reference 190
12.9.1 Detailed Description 191
12.9.2 Constructor & Destructor Documentation 192
12.9.3 Member Function Documentation 192
12.9.4 Member Data Documentation 195
12.10 OpenGPS::Schemas::ISO5436_2::DataLinkType Class Reference 196
12.10.1 Detailed Description 196
12.10.2 Member Typedef Documentation 202
12.10.3 Constructor & Destructor Documentation 204
12.10.4 Member Function Documentation 205
12.10.5 Member Data Documentation 213

12.11 OpenGPS::Schemas::ISO5436_2::DataListType Class Reference 214
12.11.1 Detailed Description 214
12.11.2 Member Typedef Documentation 216
12.11.3 Constructor & Destructor Documentation 217
12.11.4 Member Function Documentation 218
12.11.5 Member Data Documentation 221
12.12 OpenGPS::DataPoint Class Reference 221
12.12.1 Detailed Description 221
12.12.2 Constructor & Destructor Documentation 222
12.12.3 Member Function Documentation 223
12.13 OpenGPS::DataPointImpl Class Reference 227
12.13.1 Detailed Description 227
12.13.2 Constructor & Destructor Documentation 229
12.13.3 Member Function Documentation 229
12.13.4 Member Data Documentation 233
12.14 OpenGPS::DataPointParser Class Reference 234
12.14.1 Detailed Description 234
12.14.2 Constructor & Destructor Documentation 235
12.14.3 Member Function Documentation 235
12.15 OpenGPS::Schemas::ISO5436_2::DataType Class Reference 236
12.15.1 Detailed Description 236
12.15.2 Member Enumeration Documentation 238
12.15.3 Constructor & Destructor Documentation 238
12.15.4 Member Function Documentation 241
12.15.5 Member Data Documentation 243
12.16 OpenGPS::Schemas::ISO5436_2::Datum Class Reference 243
12.16.1 Detailed Description 243
12.16.2 Constructor & Destructor Documentation 245
12.16.3 Member Function Documentation 247
12.17 OpenGPS::DoubleDataPointParser Class Reference 248
12.17.1 Detailed Description 248
12.17.2 Constructor & Destructor Documentation 248
12.17.3 Member Function Documentation 249
12.18 OpenGPS::DoubleInlineValidity Class Reference 249
12.18.1 Detailed Description 250
12.18.2 Constructor & Destructor Documentation 251

12.18.3 Member Function Documentation 251
12.19 OpenGPS::DoublePointBuffer Class Reference 252
12.19.1 Detailed Description 253
12.19.2 Constructor & Destructor Documentation 254
12.19.3 Member Function Documentation 254
12.19.4 Member Data Documentation 258
12.20 OpenGPS::Environment Class Reference 258
12.20.1 Detailed Description 259
12.20.2 Constructor & Destructor Documentation 262
12.20.3 Member Function Documentation 262
12.20.4 Member Data Documentation 271
12.21 OpenGPS::Exception Class Reference 271
12.21.1 Detailed Description 272
12.21.2 Constructor & Destructor Documentation 273
12.21.3 Member Function Documentation 273
12.21.4 Member Data Documentation 274
12.22 OpenGPS::ExceptionHistory Class Reference 274
12.22.1 Detailed Description 275
12.22.2 Constructor & Destructor Documentation 276
12.22.3 Member Function Documentation 276
12.22.4 Member Data Documentation 278
12.23 OpenGPS::Schemas::ISO5436_2::FeatureType Class Reference 279
12.23.1 Detailed Description 279
12.23.2 Constructor & Destructor Documentation 280
12.23.3 Member Function Documentation 282
12.24 OpenGPS::FloatDataPointParser Class Reference 283
12.24.1 Detailed Description 283
12.24.2 Constructor & Destructor Documentation 284
12.24.3 Member Function Documentation 284
12.25 OpenGPS::FloatInlineValidity Class Reference 285
12.25.1 Detailed Description 285
12.25.2 Constructor & Destructor Documentation 286
12.25.3 Member Function Documentation 286
12.26 OpenGPS::FloatPointBuffer Class Reference 288
12.26.1 Detailed Description 288
12.26.2 Constructor & Destructor Documentation 289

12.26.3 Member Function Documentation 290
12.26.4 Member Data Documentation 294
12.27 OpenGPS::Info Class Reference 294
12.27.1 Detailed Description 294
12.27.2 Constructor & Destructor Documentation 295
12.27.3 Member Function Documentation 295
12.28 OpenGPS::InputBinaryFileStream Class Reference 297
12.28.1 Detailed Description 298
12.28.2 Member Typedef Documentation 299
12.28.3 Constructor & Destructor Documentation 299
12.28.4 Member Data Documentation 299
12.29 OpenGPS::Schemas::ISO5436_2::InstrumentType Class Reference 299
12.29.1 Detailed Description 299
12.29.2 Member Typedef Documentation 304
12.29.3 Constructor & Destructor Documentation 306
12.29.4 Member Function Documentation 308
12.29.5 Member Data Documentation 315
12.30 OpenGPS::Int16DataPointParser Class Reference 315
12.30.1 Detailed Description 316
12.30.2 Constructor & Destructor Documentation 316
12.30.3 Member Function Documentation 316
12.31 OpenGPS::Int16PointBuffer Class Reference 317
12.31.1 Detailed Description 317
12.31.2 Constructor & Destructor Documentation 319
12.31.3 Member Function Documentation 319
12.31.4 Member Data Documentation 323
12.32 OpenGPS::Int16ValidBuffer Class Reference 323
12.32.1 Detailed Description 324
12.32.2 Constructor & Destructor Documentation 325
12.32.3 Member Function Documentation 325
12.33 OpenGPS::Int32DataPointParser Class Reference 328
12.33.1 Detailed Description 329
12.33.2 Constructor & Destructor Documentation 329
12.33.3 Member Function Documentation 329
12.34 OpenGPS::Int32PointBuffer Class Reference 330
12.34.1 Detailed Description 330

12.34.2 Constructor & Destructor Documentation 332
12.34.3 Member Function Documentation 332
12.34.4 Member Data Documentation 336
12.35 OpenGPS::Int32ValidBuffer Class Reference 336
12.35.1 Detailed Description 337
12.35.2 Constructor & Destructor Documentation 338
12.35.3 Member Function Documentation 338
12.36 OpenGPS::ISO5436_2 Class Reference 341
12.36.1 Detailed Description 342
12.36.2 Constructor & Destructor Documentation 344
12.36.3 Member Function Documentation 345
12.36.4 Member Data Documentation 357
12.37 OpenGPS::ISO5436_2Container Class Reference 357
12.37.1 Detailed Description 358
12.37.2 Member Typedef Documentation 366
12.37.3 Constructor & Destructor Documentation 366
12.37.4 Member Function Documentation 366
12.37.5 Member Data Documentation 412
12.38 OpenGPS::ISO5436_2Container::PointIteratorImpl Class Reference 414
12.38.1 Detailed Description 415
12.38.2 Constructor & Destructor Documentation 416
12.38.3 Member Function Documentation 417
12.38.4 Member Data Documentation 422
12.39 OpenGPS::Schemas::ISO5436_2::ISO5436_2Type Class Reference 423
12.39.1 Detailed Description 423
12.39.2 Member Typedef Documentation 430
12.39.3 Constructor & Destructor Documentation 432
12.39.4 Member Function Documentation 434
12.39.5 Member Data Documentation 444
12.40 OpenGPS::Schemas::ISO5436_2::MatrixDimensionType Class Reference 445
12.40.1 Detailed Description 445
12.40.2 Member Typedef Documentation 449
12.40.3 Constructor & Destructor Documentation 450
12.40.4 Member Function Documentation 451
12.40.5 Member Data Documentation 456
12.41 md5_context Struct Reference 456

12.41.1 Detailed Description 456
12.41.2 Member Data Documentation 457
12.42 OpenGPS::MissingDataPointParser Class Reference 457
12.42.1 Detailed Description 458
12.42.2 Constructor & Destructor Documentation 458
12.42.3 Member Function Documentation 458
12.43 OGPS_DataPoint Struct Reference 459
12.43.1 Detailed Description 460
12.43.2 Member Data Documentation 460
12.44 OGPS_DataPointValue Union Reference 460
12.44.1 Detailed Description 460
12.44.2 Member Data Documentation 460
12.45 OGPS_ISO5436_2 Struct Reference 461
12.45.1 Detailed Description 461
12.45.2 Member Data Documentation 462
12.46 OGPS_PointIterator Struct Reference 462
12.46.1 Detailed Description 462
12.46.2 Member Data Documentation 462
12.47 OGPS_PointVector Struct Reference 462
12.47.1 Detailed Description 463
12.47.2 Member Data Documentation 463
12.48 OpenGPS::OutputBinaryFileStream Class Reference 464
12.48.1 Detailed Description 465
12.48.2 Member Typedef Documentation 466
12.48.3 Constructor & Destructor Documentation 466
12.48.4 Member Data Documentation 466
12.49 OpenGPS::PointBuffer Class Reference 466
12.49.1 Detailed Description 467
12.49.2 Constructor & Destructor Documentation 468
12.49.3 Member Function Documentation 468
12.49.4 Member Data Documentation 472
12.50 OpenGPS::PointIterator Class Reference 473
12.50.1 Detailed Description 473
12.50.2 Constructor & Destructor Documentation 474
12.50.3 Member Function Documentation 474
12.51 OpenGPS::PointValidityProvider Class Reference 477

12.51.1 Detailed Description 478
12.51.2 Constructor & Destructor Documentation 479
12.51.3 Member Function Documentation 479
12.51.4 Member Data Documentation 480
12.52 OpenGPS::PointVector Class Reference 480
12.52.1 Detailed Description 481
12.52.2 Constructor & Destructor Documentation 483
12.52.3 Member Function Documentation 484
12.52.4 Member Data Documentation 494
12.53 OpenGPS::PointVectorBase Class Reference 495
12.53.1 Detailed Description 495
12.53.2 Constructor & Destructor Documentation 496
12.53.3 Member Function Documentation 496
12.54 OpenGPS::PointVectorInputStringStream Class Reference 497
12.54.1 Detailed Description 498
12.54.2 Member Typedef Documentation 498
12.54.3 Constructor & Destructor Documentation 499
12.54.4 Member Data Documentation 499
12.55 OpenGPS::PointVectorInvariantLocale Class Reference 499
12.55.1 Detailed Description 499
12.55.2 Member Typedef Documentation 500
12.55.3 Constructor & Destructor Documentation 500
12.56 OpenGPS::PointVectorOutputStringStream Class Reference 500
12.56.1 Detailed Description 501
12.56.2 Member Typedef Documentation 502
12.56.3 Constructor & Destructor Documentation 502
12.56.4 Member Data Documentation 502
12.57 OpenGPS::PointVectorParser Class Reference 502
12.57.1 Detailed Description 502
12.57.2 Constructor & Destructor Documentation 503
12.57.3 Member Function Documentation 504
12.57.4 Member Data Documentation 506
12.58 OpenGPS::PointVectorParserBuilder Class Reference 506
12.58.1 Detailed Description 507
12.58.2 Constructor & Destructor Documentation 508
12.58.3 Member Function Documentation 508

12.58.4 Member Data Documentation 509
12.59 OpenGPS::PointVectorProxy Class Reference 510
12.59.1 Detailed Description 510
12.59.2 Constructor & Destructor Documentation 512
12.59.3 Member Function Documentation 512
12.59.4 Member Data Documentation 514
12.60 OpenGPS::PointVectorProxy::DataPointProxy Class Reference 515
12.60.1 Detailed Description 515
12.60.2 Constructor & Destructor Documentation 517
12.60.3 Member Function Documentation 517
12.60.4 Member Data Documentation 523
12.61 OpenGPS::PointVectorProxyContext Class Reference 524
12.61.1 Detailed Description 524
12.61.2 Constructor & Destructor Documentation 524
12.61.3 Member Function Documentation 525
12.62 OpenGPS::PointVectorProxyContextList Class Reference 525
12.62.1 Detailed Description 526
12.62.2 Constructor & Destructor Documentation 527
12.62.3 Member Function Documentation 527
12.62.4 Member Data Documentation 528
12.63 OpenGPS::PointVectorProxyContextMatrix Class Reference 528
12.63.1 Detailed Description 529
12.63.2 Constructor & Destructor Documentation 530
12.63.3 Member Function Documentation 530
12.63.4 Member Data Documentation 531
12.64 OpenGPS::PointVectorReaderContext Class Reference 532
12.64.1 Detailed Description 532
12.64.2 Constructor & Destructor Documentation 533
12.64.3 Member Function Documentation 533
12.65 OpenGPS::PointVectorWhitespaceFacet Class Reference 535
12.65.1 Detailed Description 536
12.65.2 Member Typedef Documentation 536
12.65.3 Constructor & Destructor Documentation 536
12.65.4 Member Function Documentation 536
12.66 OpenGPS::PointVectorWriterContext Class Reference 537
12.66.1 Detailed Description 537

12.66.2 Constructor & Destructor Documentation 538
12.66.3 Member Function Documentation 538
12.67 OpenGPS::Schemas::ISO5436_2::ProbingSystemType Class Reference 540
12.67.1 Detailed Description 540
12.67.2 Member Typedef Documentation 543
12.67.3 Constructor & Destructor Documentation 544
12.67.4 Member Function Documentation 545
12.67.5 Member Data Documentation 549
12.68 OpenGPS::Schemas::ISO5436_2::Record1Type Class Reference 550
12.68.1 Detailed Description 550
12.68.2 Member Typedef Documentation 554
12.68.3 Constructor & Destructor Documentation 555
12.68.4 Member Function Documentation 556
12.68.5 Member Data Documentation 562
12.69 OpenGPS::Schemas::ISO5436_2::Record2Type Class Reference 563
12.69.1 Detailed Description 563
12.69.2 Member Typedef Documentation 571
12.69.3 Constructor & Destructor Documentation 573
12.69.4 Member Function Documentation 575
12.69.5 Member Data Documentation 587
12.70 OpenGPS::Schemas::ISO5436_2::Record3Type Class Reference 587
12.70.1 Detailed Description 587
12.70.2 Member Typedef Documentation 593
12.70.3 Constructor & Destructor Documentation 596
12.70.4 Member Function Documentation 597
12.70.5 Member Data Documentation 606
12.71 OpenGPS::Schemas::ISO5436_2::Record4Type Class Reference 606
12.71.1 Detailed Description 607
12.71.2 Member Typedef Documentation 609
12.71.3 Constructor & Destructor Documentation 609
12.71.4 Member Function Documentation 610
12.71.5 Member Data Documentation 613
12.72 OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType Class Reference 613
12.72.1 Detailed Description 613
12.72.2 Constructor & Destructor Documentation 615
12.72.3 Member Function Documentation 617

12.73 OpenGPS::Schemas::ISO5436_2::RotationType Class Reference 618
12.73.1 Detailed Description 618
12.73.2 Member Typedef Documentation 628
12.73.3 Constructor & Destructor Documentation 631
12.73.4 Member Function Documentation 633
12.73.5 Member Data Documentation 646
12.74 OpenGPS::String Class Reference 647
12.74.1 Detailed Description 648
12.74.2 Member Typedef Documentation 649
12.74.3 Constructor & Destructor Documentation 649
12.74.4 Member Function Documentation 650
12.75 OpenGPS::Schemas::ISO5436_2::Type Class Reference 651
12.75.1 Detailed Description 651
12.75.2 Member Enumeration Documentation 653
12.75.3 Constructor & Destructor Documentation 654
12.75.4 Member Function Documentation 657
12.75.5 Member Data Documentation 658
12.76 OpenGPS::ValidBuffer Class Reference 659
12.76.1 Detailed Description 659
12.76.2 Constructor & Destructor Documentation 661
12.76.3 Member Function Documentation 661
12.76.4 Member Data Documentation 664
12.77 OpenGPS::VectorBuffer Class Reference 665
12.77.1 Detailed Description 665
12.77.2 Constructor & Destructor Documentation 667
12.77.3 Member Function Documentation 667
12.77.4 Member Data Documentation 671
12.78 OpenGPS::VectorBufferBuilder Class Reference 671
12.78.1 Detailed Description 672
12.78.2 Constructor & Destructor Documentation 673
12.78.3 Member Function Documentation 673
12.78.4 Member Data Documentation 676
12.79 OpenGPS::XmlPointVectorReaderContext Class Reference 676
12.79.1 Detailed Description 677
12.79.2 Member Typedef Documentation 679
12.79.3 Constructor & Destructor Documentation 679

12.79.4 Member Function Documentation 679
12.79.5 Member Data Documentation 682
12.80 OpenGPS::XmlPointVectorWriterContext Class Reference 683
12.80.1 Detailed Description 684
12.80.2 Member Typedef Documentation 685
12.80.3 Constructor & Destructor Documentation 685
12.80.4 Member Function Documentation 685
12.80.5 Member Data Documentation 689
12.81 OpenGPS::ZipOutputStream Class Reference 689
12.81.1 Detailed Description 690
12.81.2 Member Typedef Documentation 690
12.81.3 Constructor & Destructor Documentation 690
12.81.4 Member Function Documentation 691
12.82 OpenGPS::ZipStreamBuffer Class Reference 691
12.82.1 Detailed Description 691
12.82.2 Member Typedef Documentation 692
12.82.3 Constructor & Destructor Documentation 692
12.82.4 Member Function Documentation 693
12.82.5 Member Data Documentation 693

13 File Documentation **694**
13.1 auto_ptr_types.hxx File Reference 694
13.1.1 Detailed Description 694
13.2 binary_lsb_point_vector_reader_context.cxx File Reference 694
13.2.1 Define Documentation 695
13.3 binary_lsb_point_vector_reader_context.hxx File Reference 696
13.3.1 Detailed Description 696
13.4 binary_lsb_point_vector_writer_context.cxx File Reference 697
13.4.1 Define Documentation 697
13.5 binary_lsb_point_vector_writer_context.hxx File Reference 698
13.5.1 Detailed Description 698
13.6 binary_msb_point_vector_reader_context.cxx File Reference 699
13.6.1 Define Documentation 700
13.7 binary_msb_point_vector_reader_context.hxx File Reference 700
13.7.1 Detailed Description 700
13.8 binary_msb_point_vector_writer_context.cxx File Reference 701
13.8.1 Define Documentation 702

13.9 binary_msb_point_vector_writer_context.hxx File Reference 703
13.9.1 Detailed Description 703
13.10 binary_point_vector_reader_context.cxx File Reference 703
13.10.1 Define Documentation 704
13.11 binary_point_vector_reader_context.hxx File Reference 704
13.11.1 Detailed Description 704
13.12 binary_point_vector_writer_context.cxx File Reference 705
13.12.1 Define Documentation 706
13.13 binary_point_vector_writer_context.hxx File Reference 707
13.13.1 Detailed Description 707
13.14 data_point.h File Reference 707
13.14.1 Detailed Description 707
13.15 data_point.hxx File Reference 709
13.15.1 Detailed Description 709
13.16 data_point_c.cxx File Reference 710
13.17 data_point_c.hxx File Reference 711
13.17.1 Detailed Description 711
13.17.2 Typedef Documentation 711
13.18 data_point_impl.cxx File Reference 711
13.19 data_point_impl.hxx File Reference 712
13.19.1 Detailed Description 712
13.20 data_point_parser.hxx File Reference 713
13.20.1 Detailed Description 713
13.21 data_point_proxy.cxx File Reference 713
13.21.1 Define Documentation 714
13.22 data_point_type.h File Reference 715
13.22.1 Detailed Description 715
13.23 double_data_point_parser.cxx File Reference 715
13.24 double_data_point_parser.hxx File Reference 716
13.24.1 Detailed Description 716
13.25 double_point_buffer.cxx File Reference 717
13.26 double_point_buffer.hxx File Reference 717
13.26.1 Detailed Description 717
13.27 Doxygen.cpp File Reference 718
13.27.1 Detailed Description 718
13.28 environment.cxx File Reference 718

13.28.1 Define Documentation 719
13.29 environment.hxx File Reference 719
13.29.1 Detailed Description 719
13.30 exceptions.cxx File Reference 720
13.31 exceptions.hxx File Reference 720
13.31.1 Detailed Description 720
13.32 float_data_point_parser.cxx File Reference 721
13.33 float_data_point_parser.hxx File Reference 721
13.33.1 Detailed Description 721
13.34 float_point_buffer.cxx File Reference 722
13.35 float_point_buffer.hxx File Reference 723
13.35.1 Detailed Description 723
13.36 IncRev.c File Reference 723
13.36.1 Detailed Description 723
13.36.2 Define Documentation 724
13.36.3 Function Documentation 725
13.37 info.cxx File Reference 726
13.38 info.h File Reference 726
13.38.1 Detailed Description 726
13.39 info.hxx File Reference 727
13.39.1 Detailed Description 727
13.40 info_c.cxx File Reference 728
13.41 inline_validity.cxx File Reference 729
13.42 inline_validity.hxx File Reference 729
13.42.1 Detailed Description 729
13.43 int16_data_point_parser.cxx File Reference 730
13.44 int16_data_point_parser.hxx File Reference 731
13.44.1 Detailed Description 731
13.45 int16_point_buffer.cxx File Reference 732
13.46 int16_point_buffer.hxx File Reference 732
13.46.1 Detailed Description 732
13.47 int32_data_point_parser.cxx File Reference 733
13.48 int32_data_point_parser.hxx File Reference 733
13.48.1 Detailed Description 733
13.49 int32_point_buffer.cxx File Reference 734
13.50 int32_point_buffer.hxx File Reference 735

13.50.1 Detailed Description 735
13.51 ISO5436-sample1.x3p File Reference 735
13.51.1 Detailed Description 735
13.52 ISO5436-sample1_bin.x3p File Reference 736
13.52.1 Detailed Description 736
13.53 ISO5436-sample2.x3p File Reference 736
13.53.1 Detailed Description 736
13.54 ISO5436-sample3.x3p File Reference 736
13.54.1 Detailed Description 736
13.55 ISO5436-sample4.x3p File Reference 736
13.55.1 Detailed Description 736
13.55.2 Variable Documentation 737
13.56 ISO5436-sample4_bin.x3p File Reference 737
13.56.1 Detailed Description 737
13.56.2 Variable Documentation 737
13.57 iso5436_2.cxx File Reference 738
13.58 iso5436_2.h File Reference 738
13.58.1 Detailed Description 738
13.59 iso5436_2.hxx File Reference 740
13.59.1 Detailed Description 740
13.60 iso5436_2_c.cxx File Reference 742
13.61 iso5436_2_container.cxx File Reference 744
13.61.1 Define Documentation 745
13.61.2 Typedef Documentation 745
13.62 iso5436_2_container.hxx File Reference 746
13.62.1 Detailed Description 746
13.63 iso5436_2_handle.hxx File Reference 746
13.63.1 Detailed Description 746
13.64 iso5436_2_handle_c.hxx File Reference 748
13.64.1 Detailed Description 748
13.65 ISO5436_2_XML_Demo.cxx File Reference 748
13.65.1 Detailed Description 748
13.65.2 Function Documentation 749
13.66 iso5436_2_xsd.cxx File Reference 760
13.67 iso5436_2_xsd.hxx File Reference 766
13.67.1 Detailed Description 766

13.68 iso5436_2_xsd.hxx File Reference . . . 776
13.68.1 Detailed Description . . . 776
13.69 md5.c File Reference . . . 783
13.69.1 Define Documentation . . . 785
13.69.2 Function Documentation . . . 786
13.69.3 Variable Documentation . . . 789
13.70 md5.h File Reference . . . 790
13.70.1 Detailed Description . . . 790
13.70.2 Function Documentation . . . 791
13.71 messages.h File Reference . . . 795
13.71.1 Detailed Description . . . 795
13.72 messages_c.cxx File Reference . . . 796
13.73 messages_c.hxx File Reference . . . 797
13.73.1 Detailed Description . . . 797
13.73.2 Define Documentation . . . 798
13.74 missing_data_point_parser.cxx File Reference . . . 800
13.75 missing_data_point_parser.hxx File Reference . . . 800
13.75.1 Detailed Description . . . 800
13.76 opengps.h File Reference . . . 801
13.76.1 Detailed Description . . . 801
13.77 opengps.hxx File Reference . . . 803
13.77.1 Detailed Description . . . 803
13.78 point_buffer.cxx File Reference . . . 804
13.79 point_buffer.hxx File Reference . . . 804
13.79.1 Detailed Description . . . 804
13.80 point_iterator.cxx File Reference . . . 805
13.81 point_iterator.h File Reference . . . 805
13.81.1 Detailed Description . . . 805
13.82 point_iterator.hxx File Reference . . . 807
13.82.1 Detailed Description . . . 807
13.83 point_iterator_c.cxx File Reference . . . 808
13.84 point_iterator_c.hxx File Reference . . . 809
13.84.1 Detailed Description . . . 809
13.84.2 Typedef Documentation . . . 810
13.85 point_validity_provider.cxx File Reference . . . 810
13.86 point_validity_provider.hxx File Reference . . . 810

13.86.1 Detailed Description 810
13.87 point_vector.cxx File Reference 811
13.88 point_vector.h File Reference 811
13.88.1 Detailed Description 811
13.89 point_vector.hxx File Reference 815
13.89.1 Detailed Description 815
13.90 point_vector_base.hxx File Reference 816
13.90.1 Detailed Description 816
13.91 point_vector_c.cxx File Reference 816
13.92 point_vector_c.hxx File Reference 819
13.92.1 Detailed Description 819
13.92.2 Typedef Documentation 820
13.93 point_vector_iostream.cxx File Reference 820
13.94 point_vector_iostream.hxx File Reference 820
13.94.1 Detailed Description 820
13.95 point_vector_parser.cxx File Reference 821
13.96 point_vector_parser.hxx File Reference 822
13.96.1 Detailed Description 822
13.97 point_vector_parser_builder.cxx File Reference 823
13.98 point_vector_parser_builder.hxx File Reference 823
13.98.1 Detailed Description 823
13.99 point_vector_proxy.cxx File Reference 824
13.100 point_vector_proxy.hxx File Reference 824
13.100.1 Detailed Description 824
13.101 point_vector_proxy_context.cxx File Reference 825
13.102 point_vector_proxy_context.hxx File Reference 826
13.102.1 Detailed Description 826
13.103 point_vector_proxy_context_list.cxx File Reference 827
13.104 point_vector_proxy_context_list.hxx File Reference 827
13.104.1 Detailed Description 827
13.105 point_vector_proxy_context_matrix.cxx File Reference 828
13.106 point_vector_proxy_context_matrix.hxx File Reference 828
13.106.1 Detailed Description 828
13.107 point_vector_reader_context.hxx File Reference 829
13.107.1 Detailed Description 829
13.108 point_vector_writer_context.hxx File Reference 830

13.108.1 Detailed Description 830
13.109 Sample_Files_Description_for_Doxygen.h File Reference 831
13.110 stdafx.hxx File Reference 831
13.110.1 Detailed Description 831
13.110.2 Define Documentation 832
13.111 string.cxx File Reference 837
13.112 string.hxx File Reference 837
13.112.1 Detailed Description 837
13.113 valid_buffer.cxx File Reference 838
13.114 valid_buffer.hxx File Reference 838
13.114.1 Detailed Description 838
13.115 vector_buffer.cxx File Reference 839
13.116 vector_buffer.hxx File Reference 840
13.116.1 Detailed Description 840
13.117 vector_buffer_builder.cxx File Reference 841
13.118 vector_buffer_builder.hxx File Reference 841
13.118.1 Detailed Description 841
13.119 Version.h File Reference 842
13.119.1 Detailed Description 842
13.120 win32_environment.cxx File Reference 843
13.121 win32_environment.hxx File Reference 843
13.121.1 Detailed Description 843
13.122 xml_point_vector_reader_context.cxx File Reference 843
13.122.1 Define Documentation 844
13.123 xml_point_vector_reader_context.hxx File Reference 844
13.123.1 Detailed Description 844
13.124 xml_point_vector_writer_context.cxx File Reference 845
13.124.1 Define Documentation 846
13.125 xml_point_vector_writer_context.hxx File Reference 846
13.125.1 Detailed Description 846
13.126 zsd_Licence_Header.c File Reference 847
13.127 zip_stream_buffer.cxx File Reference 847
13.128 zip_stream_buffer.hxx File Reference 847
13.128.1 Detailed Description 847

1 openGPS ISO 5436-2 XML Library Documentation

Figure 1: width=10cm

The openGPS ISO5436-2 XML Library contains an implementation of the X3P file format according to the ISO 5436-2 standard.

The homepage of openGPS is www.opengps.eu.

2 Todo List

Class OpenGPS::String Add a streaming input and output operator

3 Module Index

3.1 Modules

Here is a list of all modules:

Sample files demonstrating X3P file format. 62

C++ Interface for X3P-library 63

Ansi-C Interface for X3P library 15

4 Directory Hierarchy

4.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

include 69

opengps 71

cxx 65

ISO5436_2_XML 70

c 64

cxx 66

xyssl 73

ISO5436_2_XML_Demo 71

samplefiles 72

tools 73

IncRev 70

5 Namespace Index

5.1 Namespace List

Here is a list of all namespaces with brief descriptions:

OpenGPS (The standard namespace for the openGPS software library) 73

OpenGPS::Schemas (Holds the C++ representations of the structure of all supported XML documents) 79

OpenGPS::Schemas::ISO5436_2 (C++ namespace for the http://www.opengps.eu/2008/ISO5436_2 schema namespace) 79

std (The standard namespace for the C++ library) 111

xml_schema (The namespace for basic XML schemas) 115

6 Class Index

6.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

OpenGPS::Schemas::ISO5436_2::AxesType 120

OpenGPS::Schemas::ISO5436_2::AxisDescriptionType 136

OpenGPS::Schemas::ISO5436_2::AxisType 154

OpenGPS::Schemas::ISO5436_2::DataLinkType 196

OpenGPS::Schemas::ISO5436_2::DataListType 214

OpenGPS::DataPoint 221

OpenGPS::DataPointImpl 227

OpenGPS::PointVectorProxy::DataPointProxy 515

OpenGPS::DataPointParser 234
OpenGPS::DoubleDataPointParser 248
OpenGPS::FloatDataPointParser 283
OpenGPS::Int16DataPointParser 315
OpenGPS::Int32DataPointParser 328
OpenGPS::MissingDataPointParser 457
OpenGPS::Schemas::ISO5436_2::DataType 236
OpenGPS::Schemas::ISO5436_2::Datum 243
OpenGPS::Environment 258
OpenGPS::Exception 271
OpenGPS::ExceptionHistory 274
OpenGPS::Schemas::ISO5436_2::FeatureType 279
OpenGPS::Info 294
OpenGPS::InputBinaryFileStream 297
OpenGPS::Schemas::ISO5436_2::InstrumentType 299
OpenGPS::ISO5436_2 341
OpenGPS::ISO5436_2Container 357
OpenGPS::Schemas::ISO5436_2::ISO5436_2Type 423
OpenGPS::Schemas::ISO5436_2::MatrixDimensionType 445
md5_context 456
OGPS_DataPoint 459
OGPS_DataPointValue 460
OGPS_ISO5436_2 461
OGPS_PointIterator 462
OGPS_PointVector 462
OpenGPS::OutputBinaryFileStream 464
OpenGPS::PointBuffer 466
OpenGPS::DoublePointBuffer 252
OpenGPS::FloatPointBuffer 288

OpenGPS::Int16PointBuffer 317

OpenGPS::Int32PointBuffer 330

OpenGPS::PointIterator 473

OpenGPS::ISO5436_2Container::PointIteratorImpl 414

OpenGPS::PointValidityProvider 477

OpenGPS::DoubleInlineValidity 249

OpenGPS::FloatInlineValidity 285

OpenGPS::ValidBuffer 659

OpenGPS::Int16ValidBuffer 323

OpenGPS::Int32ValidBuffer 336

OpenGPS::PointVectorBase 495

OpenGPS::PointVector 480

OpenGPS::PointVectorProxy 510

OpenGPS::PointVectorInputStringStream 497

OpenGPS::PointVectorInvariantLocale 499

OpenGPS::PointVectorOutputStringStream 500

OpenGPS::PointVectorParser 502

OpenGPS::PointVectorParserBuilder 506

OpenGPS::PointVectorProxyContext 524

OpenGPS::PointVectorProxyContextList 525

OpenGPS::PointVectorProxyContextMatrix 528

OpenGPS::PointVectorReaderContext 532

OpenGPS::BinaryPointVectorReaderContext 185

OpenGPS::BinaryLSBPointVectorReaderContext 161

OpenGPS::BinaryMSBPointVectorReaderContext 173

OpenGPS::XmlPointVectorReaderContext 676

OpenGPS::PointVectorWhitespaceFacet 535

OpenGPS::PointVectorWriterContext 537

OpenGPS::BinaryPointVectorWriterContext 190

OpenGPS::BinaryLSBPointVectorWriterContext 167

OpenGPS::BinaryMSBPointVectorWriterContext 179

OpenGPS::XmlPointVectorWriterContext 683

OpenGPS::Schemas::ISO5436_2::ProbingSystemType 540

OpenGPS::Schemas::ISO5436_2::Record1Type 550

OpenGPS::Schemas::ISO5436_2::Record2Type 563

OpenGPS::Schemas::ISO5436_2::Record3Type 587

OpenGPS::Schemas::ISO5436_2::Record4Type 606

OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType 613

OpenGPS::Schemas::ISO5436_2::RotationType 618

OpenGPS::String 647

OpenGPS::Schemas::ISO5436_2::Type 651

OpenGPS::VectorBuffer 665

OpenGPS::VectorBufferBuilder 671

OpenGPS::ZipOutputStream 689

OpenGPS::ZipStreamBuffer 691

7 Class Index

7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

OpenGPS::Schemas::ISO5436_2::AxesType (Class corresponding to the AxesType schema type) 120

OpenGPS::Schemas::ISO5436_2::AxisDescriptionType (Class corresponding to the AxisDescriptionType schema type) 136

OpenGPS::Schemas::ISO5436_2::AxisType (Enumeration class corresponding to the AxisType schema type) 154

OpenGPS::BinaryLSBPointVectorReaderContext (Implements OpenGPS::BinaryPointVectorReaderContext for binary files to be parsed on machines reading in least significant byte order) 161

OpenGPS::BinaryLSBPointVectorWriterContext (Implements OpenGPS::BinaryPointVectorWriterContext for binary files to be written on machines operating in least significant byte order) 167

OpenGPS::BinaryMSBPointVectorReaderContext (Implements OpenGPS::BinaryPointVectorReaderContext for binary files to be parsed on machines reading in most significant byte order) 173

OpenGPS::BinaryMSBPointVectorWriterContext (Implements OpenGPS::BinaryPointVectorWriterContext for binary files to be written on machines operating in most significant byte order) 179

OpenGPS::BinaryPointVectorReaderContext (Specialized OpenGPS::PointVectorReaderContext for binary streams) 185

OpenGPS::BinaryPointVectorWriterContext (Implements OpenGPS::PointVectorWriterContext for writing to compressed binary streams of point vectors) 190

OpenGPS::Schemas::ISO5436_2::DataLinkType (Class corresponding to the DataLinkType schema type) 196

OpenGPS::Schemas::ISO5436_2::DataListType (Class corresponding to the DataListType schema type) 214

OpenGPS::DataPoint (Typesafe representation of a single data point value) 221

OpenGPS::DataPointImpl (A straightforward implementation of OpenGPS::DataPoint) 227

OpenGPS::DataPointParser (Interface for reading/writing point data) 234

OpenGPS::Schemas::ISO5436_2::DataType (Enumeration class corresponding to the DataType schema type) 236

OpenGPS::Schemas::ISO5436_2::Datum (Class corresponding to the Datum schema type) 243

OpenGPS::DoubleDataPointParser (Reads/Writes instances of OpenGPS::DataPoint of OGPS_Double point data) 248

OpenGPS::DoubleInlineValidity (Implements OpenGPS::PointValidityProvider as a lookup of a special IEEE754 value) 249

OpenGPS::DoublePointBuffer (Manages static memory and typesafe access) 252

OpenGPS::Environment (Interface for communicating with the operating system and related subjects) 258

OpenGPS::Exception (Describes a general exception) 271

OpenGPS::ExceptionHistory (Maintains a history of exceptions) 274

OpenGPS::Schemas::ISO5436_2::FeatureType (Class corresponding to the FeatureType schema type) 279

OpenGPS::FloatDataPointParser (Reads/Writes instances of OpenGPS::DataPoint of OGPS_Float point data) 283

OpenGPS::FloatInlineValidity (Implements OpenGPS::PointValidityProvider as a lookup of a special IEEE754 value) 285

OpenGPS::FloatPointBuffer (Manages static memory and typesafe access) 288

OpenGPS::Info (Publishes license text, ownership and similar information) 294

OpenGPS::InputBinaryFileStream (A binary stream class used for reading from binary files) 297

OpenGPS::Schemas::ISO5436_2::InstrumentType (Class corresponding to the InstrumentType schema type) 299

OpenGPS::Int16DataPointParser (Reads/Writes instances of OpenGPS::DataPoint of OGPS_Int16 point data) 315

OpenGPS::Int16PointBuffer (Manages static memory and typesafe access) 317

OpenGPS::Int16ValidBuffer (Implementation of OpenGPS::ValidBuffer for Z axis of OGPS_Int16 data type) 323

OpenGPS::Int32DataPointParser (Reads/Writes instances of OpenGPS::DataPoint of OGPS_Int32 point data) 328

OpenGPS::Int32PointBuffer (Manages static memory and typesafe access) 330

OpenGPS::Int32ValidBuffer (Implementation of OpenGPS::ValidBuffer for Z axis of OGPS_Int32 data type) 336

OpenGPS::ISO5436_2 (Represents the ISO5436-2 XML X3P file format container) 341

OpenGPS::ISO5436_2Container (This is the main gate to this software library) 357

OpenGPS::ISO5436_2Container::PointIteratorImpl (Implementation of the point iterator interface) 414

OpenGPS::Schemas::ISO5436_2::ISO5436_2Type (Class corresponding to the ISO5436_2Type schema type) 423

OpenGPS::Schemas::ISO5436_2::MatrixDimensionType (Class corresponding to the MatrixDimensionType schema type) 445

md5_context (MD5 context structure) 456

OpenGPS::MissingDataPointParser (Reads/Writes instances of OpenGPS::DataPoint of missing point data) 457

OGPS_DataPoint (Encapsulates the C++ structure of a data point handle used internally within the C interface) 459

OGPS_DataPointValue (Typesafe storage for every possible type of point data) 460

OGPS_ISO5436_2 (Encapsulates the internal C++ structure of an ISO5436-2 handle used within the C interface) 461

OGPS_PointIterator (Encapsulates the internal C++ structure of a point iterator handle used within the C interface) 462

OGPS_PointVector (Encapsulates the internal C++ structure of a point vector handle used within the C interface) 462

OpenGPS::OutputBinaryFileStream (A binary stream class used for writing to binary files) 464

OpenGPS::PointBuffer (A static memory buffer that stores all point data belonging to a single axis) 466

OpenGPS::PointIterator (Interface to a point iterator) 473

OpenGPS::PointValidityProvider (Interface to communicate the validity of a point vector at a given location) 477

OpenGPS::PointVector (Typesafe representation of three-dimensional point measurement data) 480

OpenGPS::PointVectorBase (Typesafe and very fundamental representation of three-dimensional point measurement data) 495

OpenGPS::PointVectorInputStringStream (A string stream used for parsing a point vector as defined in ISO5436-2 XML) 497

OpenGPS::PointVectorInvariantLocale (The culture invariant locale) 499

OpenGPS::PointVectorOutputStringStream (A locale invariant string stream used to convert a point vector object to its representation as text according to the ISO5436-2 XML specification) 500

OpenGPS::PointVectorParser (Generic parser of a point vector) 502

OpenGPS::PointVectorParserBuilder (Component which handles the building process of a specialized parser object for reading and writing typed point data as a three-vector) 506

OpenGPS::PointVectorProxy (Serves one row of the three distinct buffers of data points managed by OpenGPS::VectorBuffer and raises the expression of accessing a single point vector) 510

OpenGPS::PointVectorProxy::DataPointProxy (Proxies one single data point pointed to by the current context information) 515

OpenGPS::PointVectorProxyContext (Indexing of point data managed by OpenGPS::VectorBuffer) 524

OpenGPS::PointVectorProxyContextList (Indexing of point data managed by OpenGPS::VectorBuffer) 525

OpenGPS::PointVectorProxyContextMatrix (Indexing of point data managed by OpenGPS::VectorBuffer) 528

OpenGPS::PointVectorReaderContext (Encapsulates an underlying stream of point vector data) 532

OpenGPS::PointVectorWhitespaceFacet (Redefines the white space character set to make parsing of a point vector easier) 535

OpenGPS::PointVectorWriterContext (Encapsulates an underlying stream of point vector data) 537

OpenGPS::Schemas::ISO5436_2::ProbingSystemType (Class corresponding to the ProbingSystemType schema type) 540

OpenGPS::Schemas::ISO5436_2::Record1Type (Class corresponding to the Record1Type schema type) 550

OpenGPS::Schemas::ISO5436_2::Record2Type (Class corresponding to the Record2Type schema type) 563

OpenGPS::Schemas::ISO5436_2::Record3Type (Class corresponding to the Record3Type schema type) 587

OpenGPS::Schemas::ISO5436_2::Record4Type (Class corresponding to the Record4Type schema type) 606

OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType (Class corresponding to the RotationMatrixElementType schema type) 613

OpenGPS::Schemas::ISO5436_2::RotationType (Class corresponding to the RotationType schema type) 618

OpenGPS::String (Stores an OGPS_Character sequence) 647

OpenGPS::Schemas::ISO5436_2::Type (Enumeration class corresponding to the Type schema type) 651

OpenGPS::ValidBuffer (Implements the OpenGPS::PointValidityProvider as an external binary file) 659

OpenGPS::VectorBuffer (Implements the memory structure of point measurement data) 665

OpenGPS::VectorBufferBuilder (Creates an object which is able to assemble a OpenGPS::VectorBuffer instance) 671

OpenGPS::XmlPointVectorReaderContext (Specialized OpenGPS::PointVectorReaderContext for point vectors stored as list of strings) 676

OpenGPS::XmlPointVectorWriterContext (Specialized OpenGPS::PointVectorWriterContext for point vectors stored as list of strings) 683

OpenGPS::ZipOutputStream (Provides an output stream interface to write binary data to Info-Zip archives) 689

OpenGPS::ZipStreamBuffer (Provides a buffer interface suitable for streaming the zipFile handle defined in the zlib/minizip package) 691

8 File Index

8.1 File List

Here is a list of all files with brief descriptions:

auto_ptr_types.hxx (Agglomerated definition of std::auto_ptr commonly used) 694

binary_lsb_point_vector_reader_context.cxx 694

binary_lsb_point_vector_reader_context.hxx (Implementation of access methods for reading typed point data from a binary file of point vectors) 696

binary_lsb_point_vector_writer_context.cxx 697

binary_lsb_point_vector_writer_context.hxx (Implementation of access methods for writing typed point data to a binary file of point vectors) 698

binary_msb_point_vector_reader_context.cxx 699

binary_msb_point_vector_reader_context.hxx (Implementation of access methods for reading typed point data from a binary file of point vectors) 700

binary_msb_point_vector_writer_context.cxx 701

binary_msb_point_vector_writer_context.hxx (Implementation of access methods for writing typed point data to a binary file of point vectors) 703

binary_point_vector_reader_context.cxx 703

binary_point_vector_reader_context.hxx (Interface for reading typed point data from an underlying binary file stream of point vectors) 704

binary_point_vector_writer_context.cxx 705

binary_point_vector_writer_context.hxx (Interface for writing typed point data to an underlying binary of point vectors) 707

data_point.h (The abstract data type of the typesafe representation of a single data point value) 707

data_point.hxx (Typesafe representation of a single data point value) 709

data_point_c.cxx 710

data_point_c.hxx (Handle mechanism that makes a C++ OpenGPS::DataPoint object accessible through its corresponding C interface) 711

data_point_impl.cxx 711

data_point_impl.hxx (An implementation of a data point to be used for typesafe storage and access of point data) 712

data_point_parser.hxx (Generic interface of a data point parser for reading/writing point data of any supported data type from/to any supported media) 713

data_point_proxy.cxx 713

data_point_type.h (An enumeration type which may describe the data type of a current data point value) 715

double_data_point_parser.cxx 715

double_data_point_parser.hxx (A data point parser for point data of type OGPS_DoublePointType) 716

double_point_buffer.cxx 717

double_point_buffer.hxx (Allocate static memory to store point data) 717

Doxygen.cpp (Title page of documentation, no source code) 718

environment.cxx 718

environment.hxx (An interface to provide access to the environment of different architectures and operating systems) 719

exceptions.cxx 720

exceptions.hxx (Exception types thrown within this software library) 720

float_data_point_parser.cxx 721

float_data_point_parser.hxx (A data point parser for point data of type OGPS_-FloatPointType) 721

float_point_buffer.cxx 722

float_point_buffer.hxx (Allocate static memory to store point data) 723

IncRev.c (Tool to increment version numbers in text files) 723

info.cxx 726

info.h (Copyright and license information) 726

info.hxx (Copyright and license information) 727

info_c.cxx 728

inline_validity.cxx 729

inline_validity.hxx (Communicate validity of point vectors through special IEEE754 values) 729

int16_data_point_parser.cxx 730

int16_data_point_parser.hxx (A data point parser for point data of type OGPS_-Int16PointType) 731

int16_point_buffer.cxx 732

int16_point_buffer.hxx (Allocate static memory to store point data) 732

int32_data_point_parser.cxx 733

int32_data_point_parser.hxx (A data point parser for point data of type OGPS_-Int32PointType) 733

int32_point_buffer.cxx 734

int32_point_buffer.hxx (Allocate static memory to store point data) 735

ISO5436-sample1.x3p (Simple matrix example with documented XML-main file) 735

ISO5436-sample1_bin.x3p (Simple binary matrix example with documented XML-main file) 736

ISO5436-sample2.x3p (Simple unordered list example with documented XML-main file) 736

ISO5436-sample3.x3p (Matrix example mixed incremental/absolute axis) 736

ISO5436-sample4.x3p (Matrix example with integer encoded z-values) 736

ISO5436-sample4_bin.x3p (Matrix example with integer encoded z-values in binary format) 737

iso5436_2.cxx 738

iso5436_2.h (The abstract data type of the ISO 5436-2 X3P file specification) 738

iso5436_2.hxx (Represents the ISO5436-2 XML X3P file format container) 740

iso5436_2_c.cxx 742

iso5436_2_container.cxx 744

iso5436_2_container.hxx (Concrete implementation of the interface of an X3P container) 746

iso5436_2_handle.hxx (Enhancing C++ part of the C interface of the abstract data type of the ISO 5436-2 X3P file format) 746

iso5436_2_handle_c.hxx (Handle mechanism which makes a C++ OpenGPS::ISO5436_2 object accessible through the corresponding C interface of an ISO5436-2 X3P file container) 748

ISO5436_2_XML_Demo.cxx (Example programm demonstrating some of the features of the X3P-data format) 748

iso5436_2_xsd.cxx 760

include/opengps/cxx/iso5436_2_xsd.hxx (Generated from iso5436_2.xsd) 766

src/ISO5436_2_XML/iso5436_2_xsd.hxx (Generated from iso5436_2.xsd) 776

md5.c 783

md5.h 790

messages.h (Global handling of error and warning messages) 795

messages_c.cxx 796

messages_c.hxx (Error and warning messages during application) 797

missing_data_point_parser.cxx 800

missing_data_point_parser.hxx (A data point parser for point data of type OGPS_MissingPointType) 800

opengps.h (Common define's and typedef's) 801

opengps.hxx (Common define's and typedef's) 803

point_buffer.cxx 804

point_buffer.hxx (Allocate static memory to store point data) 804

point_iterator.cxx 805

point_iterator.h (The abstract data type of a point iterator used for direct manipulation of point data stored in an ISO 5436-2 X3P file) 805

point_iterator.hxx (Interface to a point iterator used for direct manipulation of point data stored in an ISO 5436-2 X3P file) 807

point_iterator_c.cxx 808

point_iterator_c.hxx (Handle mechanism which makes a C++ OpenGPS::PointIterator object accessible through the corresponding C interface of a point iterator) 809

point_validity_provider.cxx 810

point_validity_provider.hxx (Interface to communicate whether the point vector at a given location contains valid data) 810

point_vector.cxx 811

point_vector.h (The abstract data type of a point vector which holds values for all components of 3D point data stored in an ISO 5436-2 X3P file) 811

point_vector.hxx (Typesafe representation of three-dimensional point measurement data) 815

point_vector_base.hxx (Very fundamental representation of three-dimensional point measurement data) 816

point_vector_c.cxx 816

point_vector_c.hxx (Handle mechanism which makes a C++ OpenGPS::PointVector object accessible through the corresponding C interface of a point vector) 819

point_vector_iostream.cxx 820

point_vector_iostream.hxx (Locale related stuff to support locale invariant parsing of point vector data) 820

point_vector_parser.cxx 821

point_vector_parser.hxx (Generic parser of a point vector) 822

point_vector_parser_builder.cxx 823

point_vector_parser_builder.hxx (Builder which assembles a concrete instance of the generic point vector parser) 823

point_vector_proxy.cxx 824

point_vector_proxy.hxx (Make the internal memory structure of distinct data point buffers accessable as point vector data) 824

point_vector_proxy_context.cxx 825

point_vector_proxy_context.hxx (Indexing of point data managed by OpenGPS::VectorBuffer) 826

point_vector_proxy_context_list.cxx 827

point_vector_proxy_context_list.hxx (Indexing of point data managed by OpenGPS::VectorBuffer) 827

point_vector_proxy_context_matrix.cxx 828

point_vector_proxy_context_matrix.hxx (Indexing of point data managed by OpenGPS::VectorBuffer) 828

point_vector_reader_context.hxx (Interface for reading typed point data from an underlying stream of point vectors) 829

point_vector_writer_context.hxx (Interface for writing typed point data to an underlying stream of point vectors) 830

Sample_Files_Description_for_Doxygen.h 831

stdafx.hxx (Standard includes and defines) 831

string.cxx 837

string.hxx (An enhanced std::string string type) 837

valid_buffer.cxx 838

valid_buffer.hxx (Communicate validity of point vectors through external binary file) 838

vector_buffer.cxx 839

vector_buffer.hxx (The internal memory structure that stores all point measurement data) 840

vector_buffer_builder.cxx 841

vector_buffer_builder.hxx (Methods to assemble a OpenGPS::VectorBuffer instance) 841

Version.h (Include file generated from "Version.in" containing the current version) 842

win32_environment.cxx 843

win32_environment.hxx (The environment of Microsoft Windows operating systems) 843

xml_point_vector_reader_context.cxx 843

xml_point_vector_reader_context.hxx (Implementation of access methods for reading typed point data from a string list of point vectors) 844

xml_point_vector_writer_context.cxx 845

xml_point_vector_writer_context.hxx (Implementation of access methods for writing typed point data to a string list of point vectors) 846

xsd_Licence_Header.c 847

zip_stream_buffer.cxx 847

zip_stream_buffer.hxx (Integrates Info-Zip handles into the common standard io framework) 847

9 Module Documentation

9.1 Ansi-C Interface for X3P library

Collaboration diagram for Ansi-C Interface for X3P library:

9.1.1 Detailed Description

Ansi-C Interface for X3P library.

Files

- file data_point.h

  The abstract data type of the typesafe representation of a single data point value.

- file data_point_type.h

  An enumeration type which may describe the data type of a current data point value.

- file info.h

  Copyright and license information.

- file iso5436_2.h

  The abstract data type of the ISO 5436-2 X3P file specification.

- file iso5436_2_handle.hxx

  Enhancing C++ part of the C interface of the abstract data type of the ISO 5436-2 X3P file format.

- file iso5436_2_handle_c.hxx

  Handle mechanism which makes a C++ OpenGPS::ISO5436_2 object accessible through the corresponding C interface of an ISO5436-2 X3P file container.

- file messages.h

  Global handling of error and warning messages.

- file opengps.h

  Common define's and typedef's.

- file point_iterator.h

  The abstract data type of a point iterator used for direct manipulation of point data stored in an ISO 5436-2 X3P file.

- file point_vector.h

  The abstract data type of a point vector which holds values for all components of 3D point data stored in an ISO 5436-2 X3P file.

- file Version.h

  Include file generated from "Version.in" containing the current version.

Namespaces

- namespace OpenGPS

  The standard namespace for the openGPS software library.

Classes

- struct OGPS_ISO5436_2

  Encapsulates the internal C++ structure of an ISO5436-2 handle used within the C interface.

Defines

- #define _EX_T(x) (x)

  Comprises text used to describe raised exceptions, that might be localized.

- #define _OPENGPS_BUILD 15

  Build number is automatically incremented after each succesfull release build.

- #define _OPENGPS_BUILD_DATE "2008/10/23 15:38:40"

  Build date.

- #define _OPENGPS_CLEAN_BUILD "Warning! Contains locally modified code!"

  Warning message "Warning! Contains locally modified code!" or empty.

- #define _OPENGPS_CLEAN_BUILD_FLAG 0

  0 if locally modified sources exist, 1 otherwise

- #define _OPENGPS_CLEAN_REVISION

  Warning message "Warning! Contains mixed revisions!" or empty.

- #define _OPENGPS_CLEAN_REVISION_FLAG 1

  0 if it contains mixed revisions, 1 otherwise

- #define _OPENGPS_EXPORT

  Manages export and import of symbols when used as shared library.

- #define _OPENGPS_MINVERSION 1

  Minor program revision. This is increased manually in the release process.

- #define _OPENGPS_REPOSITORY_URL "https://open-gps.svn.sourceforge.net/svnroot/open-gps/ISO5436_XML/trunk/src/ISO5436_2_XML"

  URL of project archive.

- #define _OPENGPS_REVISION "186"

  Version or Version range, automatically extracted from SVN repository.

- #define _OPENGPS_REVISION_DATE "2008/10/23 13:12:49"

  Date of highest revision.

- #define _OPENGPS_VERSION 0

  Major program version. This is increased manually in the release process.

- #define FALSE 0

  The FALSE boolean value.

- #define NULL 0

  Initial value for pointers when they do not point to any memory location.

- #define OGPS_FEATURE_TYPE_PROFILE_NAME _T("PRO")

  Gives the name of the profile feature type in an ISO 5436-2 document.

- #define OGPS_FEATURE_TYPE_SURFACE_NAME _T("SUR")

  Gives the name of the surface feature type in an ISO 5436-2 document.

- #define OGPS_ISO5436_2000_REVISION_NAME _T("ISO5436 - 2000")

  Gives the revision name of an ISO 5436-2 document.

- #define TRUE 1

  The TRUE boolean value.

Typedefs

- typedef int OGPS_Boolean

  Holds a boolean value.

- typedef char OGPS_Character

  The current type of characters.

- typedef struct _OGPS_DATA_POINT * OGPS_DataPointPtr

  Typesafe representation of a single data point value.

- typedef double OGPS_Double

  Represents measurement data of type double.

- typedef float OGPS_Float

  Represents measurement data of type float.

- typedef short OGPS_Int16

  Represents measurement data of type short.

- typedef int OGPS_Int32

  Represents measurement data of type int.

- typedef struct OGPS_ISO5436_2 * OGPS_ISO5436_2Handle
- typedef struct _OGPS_ISO5436_2_HANDLE * OGPS_ISO5436_2Handle

  Represents the ISO5436-2 XML X3P file format container.

- typedef struct _OGPS_POINT_ITERATOR * OGPS_PointIteratorPtr

  Interface to a point iterator.

- typedef struct _OGPS_POINT_VECTOR * OGPS_PointVectorPtr

  Typesafe representation of three-dimensional point measurement data.

- typedef unsigned long OGPS_ULong

  Represents most index ranges.

Enumerations

- enum OGPS_DataPointType {
  OGPS_MissingPointType, OGPS_Int16PointType, OGPS_Int32PointType, OGPS_FloatPointType, OGPS_DoublePointType }

  Possible types of data stored in an OGPS_DataPointPtr instance.

- enum OGPS_ExceptionId {
  OGPS_ExNone, OGPS_ExGeneral, OGPS_ExInvalidArgument, OGPS_ExInvalidOperation, OGPS_ExNotImplemented, OGPS_ExOverflow, OGPS_ExWarning }

  Possible failure conditions.

Functions

- _OPENGPS_EXPORT void ogps_AppendVendorSpecific (const OGPS_ISO5436_2Handle handle, const OGPS_Character *vendorURI, const OGPS_Character *filePath)

  Add vendorspecific file content to the X3P archive.

- _OPENGPS_EXPORT void ogps_CloseISO5436_2 (OGPS_ISO5436_2Handle *handle)

  Closes an OGPS_ISO5436_2Handle file handle and releases its resources.

- _OPENGPS_EXPORT OGPS_ISO5436_2Handle ogps_CreateListISO5436_2 (const OGPS_Character *const file, const OGPS_Character *const temp, const OpenGPS::Schemas::ISO5436_2::Record1Type &record1, const OpenGPS::Schemas::ISO5436_2::Record2Type *record2, const OGPS_ULong listDimension, const OGPS_Boolean useBinaryData=TRUE) throw ()

  Creates a new ISO5436-2 XML X3P file.

- _OPENGPS_EXPORT OGPS_ISO5436_2Handle ogps_CreateMatrixISO5436_2 (const OGPS_Character *const file, const OGPS_Character *const temp, const OpenGPS::Schemas::ISO5436_2::Record1Type &record1, const OpenGPS::Schemas::ISO5436_2::Record2Type *record2, const OpenGPS::Schemas::ISO5436_2::MatrixDimensionType &matrixDimension, const OGPS_Boolean useBinaryData=TRUE) throw ()

  Creates a new ISO5436-2 XML X3P file.

- _OPENGPS_EXPORT OGPS_PointIteratorPtr ogps_CreateNextPointIterator (const OGPS_ISO5436_2Handle handle)

  Creates an iterator to access point data contained in an ISO5436-2 X3P file.

- _OPENGPS_EXPORT OGPS_PointVectorPtr ogps_CreatePointVector (void)

  Creates an instance of type OGPS_PointVectorPtr.

- _OPENGPS_EXPORT OGPS_PointIteratorPtr ogps_CreatePrevPointIterator (const OGPS_-ISO5436_2Handle handle)

  Creates an iterator to access point data contained in an ISO5436-2 X3P file.

- _OPENGPS_EXPORT void ogps_FreePointIterator (OGPS_PointIteratorPtr *const iterator)

  Frees the point iterator instance.

- _OPENGPS_EXPORT void ogps_FreePointVector (OGPS_PointVectorPtr *const vector)

  Frees an instance of a point vector.

- _OPENGPS_EXPORT size_t ogps_GetAboutInfo (OGPS_Character *const text, const size_t size)

  Gets a short message describing the purpose of this software library.

- _OPENGPS_EXPORT size_t ogps_GetCopyrightInfo (OGPS_Character *const text, const size_t size)

  Gets the copyright information of this software library.

- _OPENGPS_EXPORT void ogps_GetCurrentPoint (const OGPS_PointIteratorPtr iterator, OGPS_-PointVectorPtr const vector)

  Gets the value of the current point vector.

- _OPENGPS_EXPORT OpenGPS::Schemas::ISO5436_2::ISO5436_2Type *const ogps_-GetDocument (const OGPS_ISO5436_2Handle handle) throw ()

  Provides access to the ISO5436_2 XML document.

- _OPENGPS_EXPORT OGPS_Double ogps_GetDouble (const OGPS_DataPointPtr dataPoint)

  Gets the stored value of type OGPS_Double out of a given data point instance.

- _OPENGPS_EXPORT OGPS_Double ogps_GetDoubleX (const OGPS_PointVectorPtr vector)

  Gets the value of the x component of the given vector.

- _OPENGPS_EXPORT OGPS_Double ogps_GetDoubleY (const OGPS_PointVectorPtr vector)

  Gets the value of the y component of the given vector.

- _OPENGPS_EXPORT OGPS_Double ogps_GetDoubleZ (const OGPS_PointVectorPtr vector)

  Gets the value of the z component of the given vector.

- _OPENGPS_EXPORT const OGPS_Character * ogps_GetErrorDescription ()

  Gets the last handled detailed error description or NULL.

- _OPENGPS_EXPORT OGPS_ExceptionId ogps_GetErrorId ()

  Gets the type of the last handled error condition.

- _OPENGPS_EXPORT const OGPS_Character * ogps_GetErrorMessage ()

  Possible failure conditions.

- _OPENGPS_EXPORT OGPS_Float ogps_GetFloat (const OGPS_DataPointPtr dataPoint)

Gets the stored value of type OGPS_Float out of a given data point instance.

- _OPENGPS_EXPORT OGPS_Float ogps_GetFloatX (const OGPS_PointVectorPtr vector)

  Gets the value of the x component of the given vector.

- _OPENGPS_EXPORT OGPS_Float ogps_GetFloatY (const OGPS_PointVectorPtr vector)

  Gets the value of the y component of the given vector.

- _OPENGPS_EXPORT OGPS_Float ogps_GetFloatZ (const OGPS_PointVectorPtr vector)

  Gets the value of the z component of the given vector.

- _OPENGPS_EXPORT OGPS_Int16 ogps_GetInt16 (const OGPS_DataPointPtr dataPoint)

  Gets the stored value of type OGPS_Int16 out of a given data point instance.

- _OPENGPS_EXPORT OGPS_Int16 ogps_GetInt16X (const OGPS_PointVectorPtr vector)

  Gets the value of the x component of the given vector.

- _OPENGPS_EXPORT OGPS_Int16 ogps_GetInt16Y (const OGPS_PointVectorPtr vector)

  Gets the value of the y component of the given vector.

- _OPENGPS_EXPORT OGPS_Int16 ogps_GetInt16Z (const OGPS_PointVectorPtr vector)

  Gets the value of the z component of the given vector.

- _OPENGPS_EXPORT OGPS_Int32 ogps_GetInt32 (const OGPS_DataPointPtr dataPoint)

  Gets the stored value of type OGPS_Int32 out of a given data point instance.

- _OPENGPS_EXPORT OGPS_Int32 ogps_GetInt32X (const OGPS_PointVectorPtr vector)

  Gets the value of the x component of the given vector.

- _OPENGPS_EXPORT OGPS_Int32 ogps_GetInt32Y (const OGPS_PointVectorPtr vector)

  Gets the value of the y component of the given vector.

- _OPENGPS_EXPORT OGPS_Int32 ogps_GetInt32Z (const OGPS_PointVectorPtr vector)

  Gets the value of the z component of the given vector.

- _OPENGPS_EXPORT size_t ogps_GetLicenseInfo (OGPS_Character *const text, const size_t size)

  Gets the license information of this software library.

- _OPENGPS_EXPORT void ogps_GetListCoord (const OGPS_ISO5436_2Handle handle, const OGPS_ULong index, OGPS_Double *const x, OGPS_Double *const y, OGPS_Double *const z)

  Gets the fully transformed value of a data point vector at a given index position.

- _OPENGPS_EXPORT OGPS_ULong ogps_GetListDimension (const OGPS_ISO5436_2Handle handle)

  Gets information on the list dimension of the current data structure.

- _OPENGPS_EXPORT void ogps_GetListPoint (const OGPS_ISO5436_2Handle handle, const OGPS_ULong index, OGPS_PointVectorPtr const vector)

  Gets the raw value of a data point vector at a given index position.

- _OPENGPS_EXPORT OGPS_Boolean ogps_GetListPosition (const OGPS_PointIteratorPtr iterator, OGPS_ULong *index)

  Gets the current position of the iterator.

- _OPENGPS_EXPORT void ogps_GetMatrixCoord (const OGPS_ISO5436_2Handle handle, const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w, OGPS_Double *const x, OGPS_Double *const y, OGPS_Double *const z)

  Gets the fully transformed value of a data point vector at a given surface position.

- _OPENGPS_EXPORT void ogps_GetMatrixDimensions (const OGPS_ISO5436_2Handle handle, OGPS_ULong *const size_u, OGPS_ULong *const size_v, OGPS_ULong *const size_w)

  Gets information on the matrix dimensions of the current data structure The dimensions are written to the arguments size_u, size_v, size_w.

- _OPENGPS_EXPORT void ogps_GetMatrixPoint (const OGPS_ISO5436_2Handle handle, const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w, OGPS_PointVectorPtr const vector)

  Gets the raw value of a data point vector at a given surface position.

- _OPENGPS_EXPORT OGPS_Boolean ogps_GetMatrixPosition (const OGPS_PointIteratorPtr iterator, OGPS_ULong *u, OGPS_ULong *v, OGPS_ULong *w)

  Gets the current position of the iterator in topology coordinates.

- _OPENGPS_EXPORT size_t ogps_GetNameInfo (OGPS_Character *const text, const size_t size)

  Gets a short name identifying this software library.

- _OPENGPS_EXPORT OGPS_DataPointType ogps_GetPointType (const OGPS_DataPointPtr dataPoint)

  Gets type information about the current value stored in a given data point instance.

- _OPENGPS_EXPORT OGPS_DataPointType ogps_GetPointTypeX (const OGPS_PointVectorPtr vector)

  Gets the type of points of the x component of the given vector.

- _OPENGPS_EXPORT OGPS_DataPointType ogps_GetPointTypeY (const OGPS_PointVectorPtr vector)

  Gets the type of points of the y component of the given vector.

- _OPENGPS_EXPORT OGPS_DataPointType ogps_GetPointTypeZ (const OGPS_PointVectorPtr vector)

  Gets the type of points of the z component of the given vector.

- _OPENGPS_EXPORT OGPS_Boolean ogps_GetVendorSpecific (const OGPS_ISO5436_2Handle handle, const OGPS_Character *vendorURI, const OGPS_Character *fileName, const OGPS_Character *targetPath)

  Extracts vendorspecific data from the current archive to a given file location.

- _OPENGPS_EXPORT size_t ogps_GetVersionInfo (OGPS_Character *const text, const size_t size)

  Gets the version identifier of this software library.

- _OPENGPS_EXPORT OGPS_DataPointPtr const ogps_GetX (OGPS_PointVectorPtr const vector)

  Gets the x component of the given vector.

- _OPENGPS_EXPORT void ogps_GetXYZ (const OGPS_PointVectorPtr vector, OGPS_Double *const x, OGPS_Double *const y, OGPS_Double *const z)

  Gets the values of each component of the given point vector.

- _OPENGPS_EXPORT OGPS_DataPointPtr const ogps_GetY (OGPS_PointVectorPtr const vector)

  Gets the y component of the given vector.

- _OPENGPS_EXPORT OGPS_DataPointPtr const ogps_GetZ (OGPS_PointVectorPtr const vector)

  Gets the z component of the given vector.

- _OPENGPS_EXPORT OGPS_Boolean ogps_HasError ()

  Returns TRUE when the statement executed previously failed.

- _OPENGPS_EXPORT OGPS_Boolean ogps_HasNextPoint (const OGPS_PointIteratorPtr iterator)

  Asks if there is another point available to iterate.

- _OPENGPS_EXPORT OGPS_Boolean ogps_HasPrevPoint (const OGPS_PointIteratorPtr iterator)

  Asks if there is another point available to iterate.

- _OPENGPS_EXPORT OGPS_Boolean ogps_IsMatrix (const OGPS_ISO5436_2Handle handle)

  Gets information on the structure with which the point measurement data is stored.

- _OPENGPS_EXPORT OGPS_Boolean ogps_IsMatrixCoordValid (const OGPS_ISO5436_2Handle handle, const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w)

  Asks if there is point vector data stored at the given matrix position.

- _OPENGPS_EXPORT OGPS_Boolean ogps_IsValidPoint (const OGPS_PointVectorPtr vector)

  Asks if the given point vector stores a valid data point.

- _OPENGPS_EXPORT OGPS_Boolean ogps_MoveNextPoint (OGPS_PointIteratorPtr const iterator)

  Moves the iterator forward.

- _OPENGPS_EXPORT OGPS_Boolean ogps_MovePrevPoint (OGPS_PointIteratorPtr const iterator)

  Moves the iterator backward.

- _OPENGPS_EXPORT OGPS_ISO5436_2Handle ogps_OpenISO5436_2 (const OGPS_Character *const file, const OGPS_Character *const temp=NULL, const OGPS_Boolean readOnly=FALSE)

  Opens an existing ISO5436-2 XML X3P file.

- _OPENGPS_EXPORT void ogps_PrintCopyrightInfo ()

Prints the copyright information of this software library to standard output.

- _OPENGPS_EXPORT void ogps_PrintLicenseInfo ()

  Prints the license information of this software library to standard output.

- _OPENGPS_EXPORT void ogps_PrintVersionInfo ()

  Prints the version identifier of this software library to standard output.

- _OPENGPS_EXPORT void ogps_ResetNextPointIterator (OGPS_PointIteratorPtr const iterator)

  Resets the iterator to the beginning and turns this iterator instance into a forward iterator.

- _OPENGPS_EXPORT void ogps_ResetPrevPointIterator (OGPS_PointIteratorPtr const iterator)

  Resets the iterator to the beginning and turns this iterator instance into a backward iterator.

- _OPENGPS_EXPORT void ogps_SetCurrentPoint (const OGPS_PointIteratorPtr iterator, const OGPS_PointVectorPtr vector)

  Sets the value of the current point vector.

- _OPENGPS_EXPORT void ogps_SetDouble (OGPS_DataPointPtr const dataPoint, const OGPS_-Double value)

  Stores a new value within a given data point instance.

- _OPENGPS_EXPORT void ogps_SetDoubleX (OGPS_PointVectorPtr const vector, const OGPS_-Double value)

  Sets the new value for the x component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetDoubleY (OGPS_PointVectorPtr const vector, const OGPS_-Double value)

  Sets the new value for the y component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetDoubleZ (OGPS_PointVectorPtr const vector, const OGPS_-Double value)

  Sets the new value for the z component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetFloat (OGPS_DataPointPtr const dataPoint, const OGPS_-Float value)

  Stores a new value within a given data point instance.

- _OPENGPS_EXPORT void ogps_SetFloatX (OGPS_PointVectorPtr const vector, const OGPS_-Float value)

  Sets the new value for the x component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetFloatY (OGPS_PointVectorPtr const vector, const OGPS_-Float value)

  Sets the new value for the y component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetFloatZ (OGPS_PointVectorPtr const vector, const OGPS_-Float value)

  Sets the new value for the z component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetInt16 (OGPS_DataPointPtr const dataPoint, const OGPS_-Int16 value)

  Stores a new value within a given data point instance.

- _OPENGPS_EXPORT void ogps_SetInt16X (OGPS_PointVectorPtr const vector, const OGPS_-Int16 value)

  Sets the new value for the x component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetInt16Y (OGPS_PointVectorPtr const vector, const OGPS_-Int16 value)

  Sets the new value for the y component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetInt16Z (OGPS_PointVectorPtr const vector, const OGPS_-Int16 value)

  Sets the new value for the z component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetInt32 (OGPS_DataPointPtr const dataPoint, const OGPS_-Int32 value)

  Stores a new value within a given data point instance.

- _OPENGPS_EXPORT void ogps_SetInt32X (OGPS_PointVectorPtr const vector, const OGPS_-Int32 value)

  Sets the new value for the x component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetInt32Y (OGPS_PointVectorPtr const vector, const OGPS_-Int32 value)

  Sets the new value for the y component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetInt32Z (OGPS_PointVectorPtr const vector, const OGPS_-Int32 value)

  Sets the new value for the z component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetListPoint (const OGPS_ISO5436_2Handle handle, const OGPS_ULong index, const OGPS_PointVectorPtr vector)

  Sets the value of a three-dimensional data point vector at a given index position.

- _OPENGPS_EXPORT void ogps_SetMatrixPoint (const OGPS_ISO5436_2Handle handle, const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w, const OGPS_PointVectorPtr vec-tor)

  Sets the value of a three-dimensional data point vector at a given surface position.

- _OPENGPS_EXPORT void ogps_WriteISO5436_2 (const OGPS_ISO5436_2Handle handle, const int compressionLevel=-1)

  Writes any changes back to the X3P file.

9.1.2 Define Documentation

9.1.2.1 #define _EX_T(x) (x)

Comprises text used to describe raised exceptions, that might be localized.

Referenced by OpenGPS::PointBuffer::Allocate(), OpenGPS::ValidBuffer::AllocateRaw(), OpenGPS::ISO5436_2Container::AppendVendorSpecific(), OpenGPS::ISO5436_2Container::CheckDocumentInstance(), OpenGPS::ISO5436_2Container::Compress(), OpenGPS::ISO5436_2Container::ConvertULongLongToULong(), OpenGPS::ISO5436_2Container::ConvertULongToInt32(), OpenGPS::ISO5436_2Container::Create(), OpenGPS::ISO5436_2Container::CreateTempDir(), OpenGPS::ISO5436_2Container::Decompress(), OpenGPS::PointBuffer::Get(), OpenGPS::PointVectorProxy::DataPointProxy::Get(), OpenGPS::DataPointImpl::Get(), OpenGPS::ISO5436_2Container::GetListDimension(), OpenGPS::ISO5436_2Container::GetListPoint(), OpenGPS::ISO5436_2Container::GetMatrixDimensions(), OpenGPS::ISO5436_2Container::GetMatrixPoint(), OpenGPS::ISO5436_2Container::GetVendorSpecific(), OpenGPS::ISO5436_2Container::IsMatrix(), OpenGPS::ISO5436_2Container::IsMatrixCoordValid(), OpenGPS::ISO5436_2Container::Open(), OpenGPS::ValidBuffer::Read(), OpenGPS::ISO5436_2Container::ReadXmlDocument(), OpenGPS::PointVectorProxy::DataPointProxy::Reset(), OpenGPS::ISO5436_2Container::SafeMultipilcation(), OpenGPS::ISO5436_2Container::SaveChecksumFile(), OpenGPS::ISO5436_2Container::SavePointBuffer(), OpenGPS::ISO5436_2Container::SaveValidPointsLink(), OpenGPS::ISO5436_2Container::SaveXmlDocument(), OpenGPS::PointBuffer::Set(), OpenGPS::PointVectorProxy::DataPointProxy::Set(), OpenGPS::DataPointImpl::Set(), OpenGPS::PointVectorProxyContextMatrix::SetIndex(), OpenGPS::PointVectorProxyContextList::SetIndex(), OpenGPS::ISO5436_2Container::SetListPoint(), OpenGPS::ISO5436_2Container::SetMatrixPoint(), OpenGPS::BinaryPointVectorReaderContext::Skip(), OpenGPS::ISO5436_2Container::TestChecksums(), OpenGPS::ValidBuffer::Write(), and OpenGPS::ISO5436_2Container::WriteVendorSpecific().

9.1.2.2 #define _OPENGPS_BUILD 15

Build number is automatically incremented after each succesfull release build.

9.1.2.3 #define _OPENGPS_BUILD_DATE "2008/10/23 15:38:40"

Build date.

9.1.2.4 #define _OPENGPS_CLEAN_BUILD "Warning! Contains locally modified code!"

Warning message "Warning! Contains locally modified code!" or empty.

9.1.2.5 #define _OPENGPS_CLEAN_BUILD_FLAG 0

0 if locally modified sources exist, 1 otherwise

9.1.2.6 #define _OPENGPS_CLEAN_REVISION

Warning message "Warning! Contains mixed revisions!" or empty.

9.1.2.7 #define _OPENGPS_CLEAN_REVISION_FLAG 1

0 if it contains mixed revisions, 1 otherwise

9.1.2.8 #define _OPENGPS_EXPORT

Manages export and import of symbols when used as shared library.

Remarks:

If you use openGPS as a shared library within you project you must set the SHARED_OPENGPS_-LIBRARY flag when compiling your source code!

9.1.2.9 #define _OPENGPS_MINVERSION 1

Minor program revision. This is increased manually in the release process.

9.1.2.10 #define _OPENGPS_REPOSITORY_URL "https://open-gps.svn.sourceforge.net/svnroot/open-gps/ISO5436_XML/trunk/src/ISO5436_2_XML"

URL of project archive.

9.1.2.11 #define _OPENGPS_REVISION "186"

Version or Version range, automatically extracted from SVN repository.

9.1.2.12 #define _OPENGPS_REVISION_DATE "2008/10/23 13:12:49"

Date of highest revision.

9.1.2.13 #define _OPENGPS_VERSION 0

Major program version. This is increased manually in the release process.

9.1.2.14 #define FALSE 0

The FALSE boolean value.

See also:

OGPS_Boolean

Referenced by _tmain(), OpenGPS::VectorBufferBuilder::BuildValidityProvider(), OpenGPS::VectorBufferBuilder::BuildX(), OpenGPS::VectorBufferBuilder::BuildY(), OpenGPS::VectorBufferBuilder::BuildZ(), OpenGPS::PointVectorProxyContext::CanIncrementIndex(), OpenGPS::ISO5436_2Container::Compress(), OpenGPS::ISO5436_2Container::ConfigureNamespaceMap(), OpenGPS::String::ConvertFromMd5(), OpenGPS::String::ConvertToMd5(), OpenGPS::PointVectorParser::CreateDataPointParser(), OpenGPS::ISO5436_-2Container::CreateDocument(), OpenGPS::VectorBufferBuilder::CreatePointBuffer(), OpenGPS::ISO5436_2Container::CreatePointBuffer(), OpenGPS::ISO5436_-2Container::CreatePrevPointIterator(), OpenGPS::ISO5436_2Container::CreateTempDir(), OpenGPS::ISO5436_2Container::Decompress(), OpenGPS::ISO5436_-2Container::GetAxisDataType(), OpenGPS::ZipStreamBuffer::GetMd5(), OpenGPS::ISO5436_-2Container::PointIteratorImpl::GetPosition(), OpenGPS::ISO5436_2Container::GetVendorSpecific(), OpenGPS::ISO5436_2Container::GetZaxisDataType(), OpenGPS::ValidBuffer::HasInvalidMarks(), OpenGPS::ISO5436_2Container::PointIteratorImpl::HasNext(), OpenGPS::ISO5436_-2Container::PointIteratorImpl::HasPrev(), OpenGPS::BinaryPointVectorWriterContext::HasStream(), OpenGPS::PointVectorProxyContextMatrix::IncrementIndex(), OpenGPS::PointVectorProxyContextList::IncrementIndex(), OpenGPS::PointVectorProxyContext::IncrementIndex(), OpenGPS::PointVectorProxyContextList::IsMatrix(), OpenGPS::ISO5436_2Container::IsMatrix(), OpenGPS::ISO5436_2Container::ISO5436_-2Container(), OpenGPS::XmlPointVectorReaderContext::MoveNext(), OpenGPS::ISO5436_-2Container::PointIteratorImpl::MoveNext(), OpenGPS::ISO5436_2Container::PointIteratorImpl::MovePrev(),

ogps_GetVendorSpecific(), ogps_IsMatrix(), ogps_IsValidPoint(), OpenGPS::ISO5436_2Container::Open(), OpenGPS::PointValidityProvider::PointValidityProvider(), PrintDimensions(), OpenGPS::ValidBuffer::Read(), OpenGPS::XmlPointVectorWriterContext::Reset(), OpenGPS::ISO5436_2Container::Reset(), OpenGPS::ISO5436_2Container::PointIteratorImpl::ResetPrev(), OpenGPS::ISO5436_2Container::SaveChecksumFile(), simpleExample(), OpenGPS::ValidBuffer::ValidBuffer(), OpenGPS::ISO5436_2Container::VerifyChecksum(), OpenGPS::ISO5436_2Container::VerifyDataBinChecksum(), OpenGPS::ISO5436_2Container::VerifyMainChecksum(), OpenGPS::ISO5436_2Container::VerifyValidBinChecksum(), OpenGPS::ISO5436_2Container::WriteVendorSpecific(), and OpenGPS::XmlPointVectorWriterContext::XmlPointVectorWrite

9.1.2.15 #define NULL 0

Initial value for pointers when they do not point to any memory location.

Referenced by OpenGPS::VectorBufferBuilder::BuildValidityProvider(), OpenGPS::ISO5436_2Container::Compress(), convertFormat(), OpenGPS::ISO5436_2Container::ConvertPointToCoord(), OpenGPS::ISO5436_2Container::Create(), OpenGPS::PointVectorParser::CreateDataPointParser(), OpenGPS::ISO5436_2Container::CreateDocument(), OpenGPS::VectorBufferBuilder::CreatePointBuffer(), OpenGPS::ISO5436_2Container::CreatePointVectorReaderContext(), OpenGPS::ISO5436_2Container::CreatePointVectorWriterContext(), OpenGPS::ISO5436_2Container::Decompress(), OpenGPS::DoublePointBuffer::DoublePointBuffer(), findline(), OpenGPS::FloatPointBuffer::FloatPointBuffer(), OpenGPS::ExceptionHistory::GetLastErrorDescription(), OpenGPS::ExceptionHistory::GetLastErrorMessage(), OpenGPS::PointVector::GetXYZ(), OpenGPS::ISO5436_2Container::HasDocument(), OpenGPS::BinaryPointVectorReaderContext::HasStream(), OpenGPS::VectorBuffer::HasValidityBuffer(), OpenGPS::ISO5436_2Container::HasVectorBuffer(), OpenGPS::Int16PointBuffer::Int16PointBuffer(), OpenGPS::Int32PointBuffer::Int32PointBuffer(), OpenGPS::ValidBuffer::IsAllocated(), OpenGPS::ISO5436_2Container::ISO5436_2Container(), OpenGPS::PointVectorProxy::DataPointProxy::IsValid(), main(), md5_file(), mediumComplexExample(), ogps_CreateListISO5436_2(), ogps_CreateMatrixISO5436_2(), ogps_CreateNextPointIterator(), ogps_CreatePointVector(), ogps_CreatePrevPointIterator(), ogps_GetDocument(), ogps_GetX(), ogps_GetY(), ogps_GetZ(), ogps_OpenISO5436_2(), ogps_SetCurrentPoint(), ogps_SetMatrixPoint(), performanceDouble(), performanceInt16(), OpenGPS::PointValidityProvider::PointValidityProvider(), OpenGPS::PointVectorParser::PointVectorParser(), OpenGPS::PointVectorParserBuilder::PointVectorParserBuilder(), readonlyExample(), readonlyExample2(), readonlyExampleMatrix(), OpenGPS::ISO5436_2Container::ReadXmlDocument(), OpenGPS::ISO5436_2Container::SaveChecksumFile(), OpenGPS::ISO5436_2Container::SavePointBuffer(), OpenGPS::ISO5436_2Container::SaveValidPointsLink(), OpenGPS::ISO5436_2Container::SaveXmlDocument(), OpenGPS::ISO5436_2Container::SetMatrixPoint(), OpenGPS::VectorBuffer::SetValidityProvider(), simpleExample(), OpenGPS::String::String(), OpenGPS::ValidBuffer::ValidBuffer(), OpenGPS::VectorBuffer::VectorBuffer(), OpenGPS::VectorBufferBuilder::VectorBufferBuilder(), OpenGPS::ISO5436_2Container::WriteVendorSpecific(), OpenGPS::XmlPointVectorReaderContext::XmlPointVectorReaderC and OpenGPS::ZipStreamBuffer::ZipStreamBuffer().

9.1.2.16 #define OGPS_FEATURE_TYPE_PROFILE_NAME _T("PRO")

Gives the name of the profile feature type in an ISO 5436-2 document.

The xerces tool does not generate an appropriate enum value here.

Referenced by performanceDouble(), and performanceInt16().

9.1.2.17 #define OGPS_FEATURE_TYPE_SURFACE_NAME _T("SUR")

Gives the name of the surface feature type in an ISO 5436-2 document.

The xerces tool does not generate an appropriate enum value here.

Referenced by mediumComplexExample(), and simpleExample().

9.1.2.18 #define OGPS_ISO5436_2000_REVISION_NAME _T("ISO5436 - 2000")

Gives the revision name of an ISO 5436-2 document.

The xerces tool does not generate an appropriate enum value here.

Referenced by mediumComplexExample(), performanceDouble(), performanceInt16(), and simpleExample().

9.1.2.19 #define TRUE 1

The TRUE boolean value.

See also:

OGPS_Boolean

Referenced by _tmain(), OpenGPS::XmlPointVectorWriterContext::AppendSeparator(), OpenGPS::BinaryPointVectorWriterContext::BinaryPointVectorWriterContext(), OpenGPS::VectorBufferBuilder::BuildBuffer(), OpenGPS::VectorBufferBuilder::BuildValidityProvider(), OpenGPS::ISO5436_2Container::Compress(), OpenGPS::ISO5436_2Container::ConfigureNamespaceMap(), convertFormat(), OpenGPS::String::ConvertFromMd5(), OpenGPS::ISO5436_2Container::CreateDocument(), OpenGPS::ISO5436_2Container::CreateNextPointIterator(), OpenGPS::VectorBufferBuilder::CreatePointBuffer(), OpenGPS::ISO5436_2Container::Decompress(), OpenGPS::ISO5436_2Container::DecompressChecksum(), OpenGPS::ISO5436_2Container::DecompressDataBin(), OpenGPS::ISO5436_2Container::DecompressMain(), OpenGPS::ZipStreamBuffer::GetMd5(), OpenGPS::ISO5436_2Container::PointIteratorImpl::GetPosition(), OpenGPS::ISO5436_2Container::GetVendorSpecific(), OpenGPS::ISO5436_2Container::GetXaxisDataType(), OpenGPS::ISO5436_2Container::GetYaxisDataType(), OpenGPS::ValidBuffer::HasInvalidMarks(), OpenGPS::BinaryPointVectorWriterContext::HasStream(), OpenGPS::PointVectorProxyContextMatrix::IncrementIndex(), OpenGPS::PointVectorProxyContextList::IncrementIndex(), OpenGPS::PointVectorProxyContextMatrix::IsMatrix(), OpenGPS::ISO5436_2Container::IsMatrix(), OpenGPS::ISO5436_2Container::ISO5436_2Container(), OpenGPS::ValidBuffer::IsValid(), OpenGPS::BinaryPointVectorReaderContext::IsValid(), mediumComplexExample(), OpenGPS::XmlPointVectorReaderContext::MoveNext(), OpenGPS::ISO5436_2Container::PointIteratorImpl::MoveNext(), OpenGPS::ISO5436_2Container::PointIteratorImpl::MovePrev(), OpenGPS::ISO5436_2Container::PointIteratorImpl::PointIteratorImpl(), PrintDimensions(), OpenGPS::ValidBuffer::Read(), readonlyExample(), readonlyExample2(), readonlyExample3(), readonlyExample4(), readonlyExampleMatrix(), OpenGPS::ISO5436_2Container::Reset(), OpenGPS::ISO5436_2Container::PointIteratorImpl::ResetNext(), OpenGPS::ISO5436_2Container::PointIteratorImpl::ResetPrev(), OpenGPS::ISO5436_2Container::SaveChecksumFile(), OpenGPS::ISO5436_2Container::SaveValidPointsLink(), OpenGPS::ISO5436_2Container::SaveXmlDocument(), OpenGPS::ISO5436_2Container::VerifyChecksum(), and OpenGPS::ISO5436_2Container::WriteVendorSpecific().

9.1.3 Typedef Documentation

9.1.3.1 typedef int OGPS_Boolean

Holds a boolean value.

See also:

FALSE, TRUE

9.1.3.2 typedef char OGPS_Character

The current type of characters.

This is either unicode (wchar_t) or char.

9.1.3.3 typedef struct _OGPS_DATA_POINT∗ OGPS_DataPointPtr

Typesafe representation of a single data point value.

In an OGPS_PointVectorPtr every component of that three-vector is accessible through its own OGPS_-DataPointPtr instance.

Remarks:

An instance of OGPS_DataPointPtr cannot be created of its own. Indirectly a handle of this type can be accessed through an OGPS_PointVectorPtr object.

The corresponding C++ implementation is provided by OpenGPS::DataPoint.

9.1.3.4 typedef double OGPS_Double

Represents measurement data of type double.

9.1.3.5 typedef float OGPS_Float

Represents measurement data of type float.

9.1.3.6 typedef short OGPS_Int16

Represents measurement data of type short.

9.1.3.7 typedef int OGPS_Int32

Represents measurement data of type int.

9.1.3.8 typedef struct OGPS_ISO5436_2 ∗ OGPS_ISO5436_2Handle

9.1.3.9 typedef struct _OGPS_ISO5436_2_HANDLE∗ OGPS_ISO5436_2Handle

Represents the ISO5436-2 XML X3P file format container.

Provides access to the content of an already existing or newly created X3P file container. You can obtain an instance of this handle by calling ogps_OpenISO5436_2, ogps_CreateMatrixISO5436_2 or ogps_-CreateListISO5436_2.

Remarks:

An instance of OGPS_DataPointPtr cannot be created of its own. Instead refer to ogps_-OpenISO5436_2, ogps_CreateMatrixISO5436_2 or ogps_CreateListISO5436_2.

The corresponding C++ implementation is given by OpenGPS::ISO5436_2.

9.1.3.10 typedef struct _OGPS_POINT_ITERATOR* OGPS_PointIteratorPtr

Interface to a point iterator.

The point iterator may be used to traverse all point vectors contained in an ISO5436-2 XML X3P file format container.

Remarks:

An instance of OGPS_PointIteratorPtr can be obtained from ogps_CreateNextPointIterator or ogps_-CreatePrevPointIterator. You must free an instance of type OGPS_PointIteratorPtr with ogps_-FreePointIterator when you done with it.

The corresponding C++ implementation is provided by OpenGPS::PointIterator.

9.1.3.11 typedef struct _OGPS_POINT_VECTOR* OGPS_PointVectorPtr

Typesafe representation of three-dimensional point measurement data.

The corresponding C++ implementation is provided by OpenGPS::PointVector.

9.1.3.12 typedef unsigned long OGPS_ULong

Represents most index ranges.

9.1.4 Enumeration Type Documentation

9.1.4.1 enum OGPS_DataPointType

Possible types of data stored in an OGPS_DataPointPtr instance.

Enumerator:

OGPS_MissingPointType Describes an invalid type.

OGPS_Int16PointType Describes an instance of type OGPS_Int16.

OGPS_Int32PointType Describes an instance of type OGPS_Int32.

OGPS_FloatPointType Describes an instance of type OGPS_Float.

OGPS_DoublePointType Describes an instance of type OGPS_Double.

9.1.4.2 enum OGPS_ExceptionId

Possible failure conditions.

Enumerator:

OGPS_ExNone No failure condition trapped.
This serves as some kind of default value.

OGPS_ExGeneral A failure condition occured, but it has not been specified in detail.

OGPS_ExInvalidArgument The value of at least one of the parameters passed to a function is invalid in the current context.

OGPS_ExInvalidOperation Due to the state of the object an operation could not be performed.

OGPS_ExNotImplemented A specific implementation of an interface does not implement this operation.

OGPS_ExOverflow An overflow occured.
There is no guarantee of the integrity of further processing steps.

OGPS_ExWarning Indicates a non-fatal error that may be ignored.
This is for informational purpose only.

9.1.5 Function Documentation

9.1.5.1 _OPENGPS_EXPORT void ogps_AppendVendorSpecific (const OGPS_ISO5436_2Handle *handle*, const OGPS_Character * *vendorURI*, const OGPS_Character * *filePath*)

Add vendorspecific file content to the X3P archive.

Call this before ogps_WriteISO5436_2 and the content of a file will be added to the X3P file when written. This can be called multiple times to add more than one file of your choice. These files must exist at the time when ogps_WriteISO5436_2 is being executed. The file will be added to the root of the archive with the given file name from the full path specified.

See also:

ogps_GetVendorSpecific

Parameters:

handle Operate on this handle object.

vendorURI Your very own vendor specifier in a URI conformant format.

filePath The absolute path to the file to be added to the document container.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by simpleExample().

9.1.5.2 _OPENGPS_EXPORT void ogps_CloseISO5436_2 (OGPS_ISO5436_2Handle * *handle*)

Closes an OGPS_ISO5436_2Handle file handle and releases its resources.

Remarks:

This does not save any changes you made! You must call ogps_WriteISO5436_2 before if your changes should be saved.

See also:

ogps_CreateMatrixISO5436_2, ogps_CreateListISO5436_2, ogps_OpenISO5436_2, ogps_WriteISO5436_2

Parameters:

handle Handle object to close.

References _ASSERT, _OPENGPS_DELETE, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by convertFormat(), mediumComplexExample(), performanceDouble(), performanceInt16(), readonlyExample(), readonlyExample2(), readonlyExampleMatrix(), and simpleExample().

9.1.5.3 _OPENGPS_EXPORT OGPS_ISO5436_2Handle ogps_CreateListISO5436_2 (const OGPS_Character *const *file*, const OGPS_Character *const *temp*, const OpenGPS::Schemas::ISO5436_2::Record1Type & *record1*, const OpenGPS::Schemas::ISO5436_2::Record2Type * *record2*, const OGPS_ULong *listDimension*, const OGPS_Boolean *useBinaryData* = TRUE) throw ()

Creates a new ISO5436-2 XML X3P file.

The Record3 object defined in the ISO5436_2 XML specification will be created automatically.

Remarks:

You must release the returned handle object with gps_CloseISO5436_2 when done with it.

Parameters:

file Full path to the ISO5436-2 XML X3P to be created.

temp Specifies the new absolute path to the directory where unpacked X3P data gets stored temporarily. If set to NULL the default directory for temporary files specified by your system is used.

record1 The Record1 object defined in the ISO5436_2 XML specification. The given object instance must be valid.

record2 The Record2 object defined in the ISO5436_2 XML specification. This is optional, so the parameter can be NULL. But if set, it must point to a valid instance.

listDimension Specifies the size of point measurement data that will be processed.

useBinaryData Defines whether point measurement data will be directly stored into the xml document as tag elements or if it is separately stored in a binary file within the X3P container.

Returns:

Returns the file handle or NULL on failure.

References _ASSERT, _OPENGPS_DELETE, _OPENGPS_GENERIC_EXCEPTION_HANDLER_CLEANUP, and NULL.

Referenced by convertFormat(), performanceDouble(), and performanceInt16().

9.1.5.4 _OPENGPS_EXPORT OGPS_ISO5436_2Handle ogps_CreateMatrixISO5436_2 (const OGPS_Character *const *file*, const OGPS_Character *const *temp*, const OpenGPS::Schemas::ISO5436_2::Record1Type & *record1*, const OpenGPS::Schemas::ISO5436_2::Record2Type * *record2*, const OpenGPS::Schemas::ISO5436_2::MatrixDimensionType & *matrixDimension*, const OGPS_Boolean *useBinaryData* = TRUE) throw ()

Creates a new ISO5436-2 XML X3P file.

The Record3 object defined in the ISO5436_2 XML specification will be created automatically.

Remarks:

You must release the returned handle object with gps_CloseISO5436_2 when done with it.

Parameters:

file Full path to the ISO5436-2 XML X3P to be created.

temp Specifies the new absolute path to the directory where unpacked X3P data gets stored temporarily. If set to NULL the default directory for temporary files specified by your system is used.

record1 The Record1 object defined in the ISO5436_2 XML specification. The given object instance must be valid.

record2 The Record2 object defined in the ISO5436_2 XML specification. This is optional, so the parameter can be NULL. But if set, it must point to a valid instance.

matrixDimension Specifies the topology for which point measurement data will be processed.

useBinaryData Defines whether point measurement data will be directly stored into the xml document as tag elements or if it is separately stored in a binary file within the X3P container.

Returns:

Returns the file handle or NULL on failure.

References _ASSERT, _OPENGPS_DELETE, _OPENGPS_GENERIC_EXCEPTION_HANDLER_-CLEANUP, and NULL.

Referenced by convertFormat(), mediumComplexExample(), and simpleExample().

9.1.5.5 _OPENGPS_EXPORT OGPS_PointIteratorPtr ogps_CreateNextPointIterator (const OGPS_ISO5436_2Handle *handle*)

Creates an iterator to access point data contained in an ISO5436-2 X3P file.

Iterates the point data in forward direction.

Remarks:

You must free the resources occupied by the returned iterator handle by calling ogps_FreePointIterator.

Parameters:

handle Operate on this handle object.

Returns:

Returns an iterator handle on success otherwise NULL.

References _ASSERT, _OPENGPS_DELETE, _OPENGPS_GENERIC_EXCEPTION_HANDLER_-CLEANUP, and NULL.

Referenced by convertFormat(), mediumComplexExample(), readonlyExample(), readonlyExample2(), and readonlyExampleMatrix().

9.1.5.6 _OPENGPS_EXPORT OGPS_PointVectorPtr ogps_CreatePointVector (void)

Creates an instance of type OGPS_PointVectorPtr.

Remarks:

If your create an OGPS_PointVectorPtr object using this function, you must call ogps_FreePointVector to release the object.

Returns:

Returns an instance of type OGPS_PointVectorPtr.

References NULL.

Referenced by convertFormat(), mediumComplexExample(), performanceDouble(), performanceInt16(), readonlyExample(), readonlyExample2(), readonlyExampleMatrix(), and simpleExample().

9.1.5.7 _OPENGPS_EXPORT OGPS_PointIteratorPtr ogps_CreatePrevPointIterator (const OGPS_ISO5436_2Handle *handle*)

Creates an iterator to access point data contained in an ISO5436-2 X3P file.

Iterates the point data in backward direction.

Remarks:

You must free the resources occupied by the returned iterator handle by calling ogps_FreePointIterator.

Parameters:

handle Operate on this handle object.

Returns:

Returns an iterator handle on success otherwise NULL.

References _ASSERT, _OPENGPS_DELETE, _OPENGPS_GENERIC_EXCEPTION_HANDLER_-CLEANUP, and NULL.

9.1.5.8 _OPENGPS_EXPORT void ogps_FreePointIterator (OGPS_PointIteratorPtr *const *itera-tor*)

Frees the point iterator instance.

Remarks:

Must be called for every point iterator handle obtained through ogps_CreateNextPointIterator or ogps_CreatePrevPointIterator.

Parameters:

iterator Operate on this iterator handle.

References _ASSERT, _OPENGPS_DELETE, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by convertFormat(), mediumComplexExample(), readonlyExample(), readonlyExample2(), and readonlyExampleMatrix().

9.1.5.9 _OPENGPS_EXPORT void ogps_FreePointVector (OGPS_PointVectorPtr *const *vector*)

Frees an instance of a point vector.

See also:

ogps_CreatePointVector

Parameters:

vector Operate on this point vector instance.

References _OPENGPS_DELETE, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by convertFormat(), mediumComplexExample(), performanceDouble(), performanceInt16(), readonlyExample(), readonlyExample2(), readonlyExampleMatrix(), and simpleExample().

9.1.5.10 _OPENGPS_EXPORT size_t ogps_GetAboutInfo (OGPS_Character *const *text*, const size_t *size*)

Gets a short message describing the purpose of this software library.

Parameters:

text Allocated string buffer. Will contain the resulting message on success.

size The size of the allocated buffer in characters.

Returns:

On success returns the number of characters written, otherwise returns the required size in characters of an allocated text buffer to store the about message including the terminating null character.

References _OPENGPS_GENERIC_EXCEPTION_HANDLER, and OpenGPS::Info::GetAbout().

Here is the call graph for this function:

9.1.5.11 _OPENGPS_EXPORT size_t ogps_GetCopyrightInfo (OGPS_Character *const *text*, const size_t *size*)

Gets the copyright information of this software library.

Parameters:

text Allocated string buffer. Will contain the resulting message on success.

size The size of the allocated buffer in characters.

Returns:

On success returns the number of characters written, otherwise returns the required size in characters of an allocated text buffer to store the copyright information including the terminating null character.

References _OPENGPS_GENERIC_EXCEPTION_HANDLER, and OpenGPS::Info::GetCopyright().

Here is the call graph for this function:

9.1.5.12 _OPENGPS_EXPORT void ogps_GetCurrentPoint (const OGPS_PointIteratorPtr *iterator*, OGPS_PointVectorPtr const *vector*)

Gets the value of the current point vector.

See also:

ogps_MoveNext

On failure you may get further information by calling ogps_GetErrorMessage hereafter.

Parameters:

iterator Operate on this iterator handle.

vector Gets a copy of the vector at the current iterator position.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by convertFormat(), readonlyExample(), readonlyExample2(), and readonlyExampleMatrix().

9.1.5.13 _OPENGPS_EXPORT OpenGPS::Schemas::ISO5436_2::ISO5436_2Type* const ogps_GetDocument (const OGPS_ISO5436_2Handle *handle*) throw ()

Provides access to the ISO5436_2 XML document.

Parameters:

handle Handle object to operate on.

Returns:

Returns ISO5436_2 XML document handle or NULL on failure.

References _ASSERT, _OPENGPS_GENERIC_EXCEPTION_HANDLER, and NULL.

Referenced by convertFormat(), and readonlyExample2().

9.1.5.14 _OPENGPS_EXPORT OGPS_Double ogps_GetDouble (const OGPS_DataPointPtr *dataPoint*)

Gets the stored value of type OGPS_Double out of a given data point instance.

Remarks:

If the current type does not equal OGPS_Double, this function returns the value 0.0. You may check ogps_HasError whether this operation has been successfully executed.

See also:

ogps_GetPointType

Parameters:

dataPoint Operate on this data point instance.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.15 _OPENGPS_EXPORT OGPS_Double ogps_GetDoubleX (const OGPS_PointVectorPtr *vector*)

Gets the value of the x component of the given vector.

Remarks:

Returns 0.0 if there is a type mismatch and the value of the component is stored as a data type other than OGPS_Double.

See also:

OGPS_DataPointType

Parameters:

vector Operate on this point vector instance.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by readonlyExample2().

9.1.5.16 _OPENGPS_EXPORT OGPS_Double ogps_GetDoubleY (const OGPS_PointVectorPtr *vector*)

Gets the value of the y component of the given vector.

Remarks:

Returns 0.0 if there is a type mismatch and the value of the component is stored as a data type other than OGPS_Double.

See also:

OGPS_DataPointType

Parameters:

vector Operate on this point vector instance.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by readonlyExample2().

9.1.5.17 _OPENGPS_EXPORT OGPS_Double ogps_GetDoubleZ (const OGPS_PointVectorPtr *vector*)

Gets the value of the z component of the given vector.

Remarks:

Returns 0.0 if there is a type mismatch and the value of the component is stored as a data type other than OGPS_Double.

See also:

OGPS_DataPointType

Parameters:

vector Operate on this point vector instance.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by readonlyExample2().

9.1.5.18 _OPENGPS_EXPORT const OGPS_Character* ogps_GetErrorDescription ()

Gets the last handled detailed error description or NULL.

Remarks:

This may return NULL even though an error occured. This is because the description of a particular error condition may not be available.

See also:

ogps_GetErrorId

References OpenGPS::ExceptionHistory::GetLastErrorDescription().

Here is the call graph for this function:

9.1.5.19 _OPENGPS_EXPORT OGPS_ExceptionId ogps_GetErrorId ()

Gets the type of the last handled error condition.

See also:

ogps_GetErrorMessage

References OpenGPS::ExceptionHistory::GetLastErrorId().

Here is the call graph for this function:

9.1.5.20 _OPENGPS_EXPORT const OGPS_Character* ogps_GetErrorMessage ()

Possible failure conditions.

Gets the last handled error message or NULL.

Remarks:

This may return NULL even though an error occured. This is because the description of a particular error condition may not be available.

Use ogps_HasError to check whether an error condition has been trapped.

See also:

ogps_GetErrorId

References OpenGPS::ExceptionHistory::GetLastErrorMessage().

Here is the call graph for this function:

9.1.5.21 _OPENGPS_EXPORT OGPS_Float ogps_GetFloat (const OGPS_DataPointPtr *dataPoint*)

Gets the stored value of type OGPS_Float out of a given data point instance.

Remarks:

If the current type does not equal OGPS_Float, this function returns the value 0.0. You may check ogps_HasError whether this operation has been successfully executed.

See also:

ogps_GetPointType

Parameters:

dataPoint Operate on this data point instance.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.22 _OPENGPS_EXPORT OGPS_Float ogps_GetFloatX (const OGPS_PointVectorPtr *vector*)

Gets the value of the x component of the given vector.

Remarks:

Returns 0.0 if there is a type mismatch and the value of the component is stored as a data type other than OGPS_Float.

See also:

OGPS_DataPointType

Parameters:

vector Operate on this point vector instance.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by readonlyExample2().

9.1.5.23 _OPENGPS_EXPORT OGPS_Float ogps_GetFloatY (const OGPS_PointVectorPtr *vector*)

Gets the value of the y component of the given vector.

Remarks:

Returns 0.0 if there is a type mismatch and the value of the component is stored as a data type other than OGPS_Float.

See also:

OGPS_DataPointType

Parameters:

vector Operate on this point vector instance.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by readonlyExample2().

9.1.5.24 _OPENGPS_EXPORT OGPS_Float ogps_GetFloatZ (const OGPS_PointVectorPtr *vector*)

Gets the value of the z component of the given vector.

Remarks:

Returns 0.0 if there is a type mismatch and the value of the component is stored as a data type other than OGPS_Float.

See also:

OGPS_DataPointType

Parameters:

vector Operate on this point vector instance.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by readonlyExample2().

9.1.5.25 _OPENGPS_EXPORT OGPS_Int16 ogps_GetInt16 (const OGPS_DataPointPtr *dataPoint*)

Gets the stored value of type OGPS_Int16 out of a given data point instance.

Remarks:

If the current type does not equal OGPS_Int16, this function returns the value 0. You may check ogps_HasError whether this operation has been successfully executed.

See also:

ogps_GetPointType

Parameters:

dataPoint Operate on this data point instance.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.26 _OPENGPS_EXPORT OGPS_Int16 ogps_GetInt16X (const OGPS_PointVectorPtr *vector*)

Gets the value of the x component of the given vector.

Remarks:

Returns 0 if there is a type mismatch and the value of the component is stored as a data type other than OGPS_Int16.

See also:

OGPS_DataPointType

Parameters:

vector Operate on this point vector instance.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by readonlyExample2().

9.1.5.27 _OPENGPS_EXPORT OGPS_Int16 ogps_GetInt16Y (const OGPS_PointVectorPtr *vector*)

Gets the value of the y component of the given vector.

Remarks:

Returns 0 if there is a type mismatch and the value of the component is stored as a data type other than OGPS_Int16.

See also:

OGPS_DataPointType

Parameters:

vector Operate on this point vector instance.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by readonlyExample2().

9.1.5.28 _OPENGPS_EXPORT OGPS_Int16 ogps_GetInt16Z (const OGPS_PointVectorPtr *vector*)

Gets the value of the z component of the given vector.

Remarks:

Returns 0 if there is a type mismatch and the value of the component is stored as a data type other than OGPS_Int16.

See also:

OGPS_DataPointType

Parameters:

vector Operate on this point vector instance.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by readonlyExample2().

9.1.5.29 _OPENGPS_EXPORT OGPS_Int32 ogps_GetInt32 (const OGPS_DataPointPtr *dataPoint*)

Gets the stored value of type OGPS_Int32 out of a given data point instance.

Remarks:

If the current type does not equal OGPS_Int32, this function returns the value 0. You may check ogps_HasError whether this operation has been successfully executed.

See also:

ogps_GetPointType

Parameters:

dataPoint Operate on this data point instance.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.30 _OPENGPS_EXPORT OGPS_Int32 ogps_GetInt32X (const OGPS_PointVectorPtr *vector*)

Gets the value of the x component of the given vector.

Remarks:

Returns 0 if there is a type mismatch and the value of the component is stored as a data type other than OGPS_Int32.

See also:

OGPS_DataPointType

Parameters:

vector Operate on this point vector instance.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by readonlyExample2().

9.1.5.31 _OPENGPS_EXPORT OGPS_Int32 ogps_GetInt32Y (const OGPS_PointVectorPtr *vector*)

Gets the value of the y component of the given vector.

Remarks:

Returns 0 if there is a type mismatch and the value of the component is stored as a data type other than OGPS_Int32.

See also:

OGPS_DataPointType

Parameters:

vector Operate on this point vector instance.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by readonlyExample2().

9.1.5.32 _OPENGPS_EXPORT OGPS_Int32 ogps_GetInt32Z (const OGPS_PointVectorPtr *vector*)

Gets the value of the z component of the given vector.

Remarks:

Returns 0 if there is a type mismatch and the value of the component is stored as a data type other than OGPS_Int32.

See also:

OGPS_DataPointType

Parameters:

vector Operate on this point vector instance.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by readonlyExample2().

9.1.5.33 _OPENGPS_EXPORT size_t ogps_GetLicenseInfo (OGPS_Character *const *text*, const size_t *size*)

Gets the license information of this software library.

Parameters:

text Allocated string buffer. Will contain the resulting message on success.

size The size of the allocated buffer in characters.

Returns:

On success returns the number of characters written, otherwise returns the required size in characters of an allocated text buffer to store the license information including the terminating null character.

References _OPENGPS_GENERIC_EXCEPTION_HANDLER, and OpenGPS::Info::GetLicense().

Here is the call graph for this function:

9.1.5.34 _OPENGPS_EXPORT void ogps_GetListCoord (const OGPS_ISO5436_2Handle *handle*, const OGPS_ULong *index*, OGPS_Double *const *x*, OGPS_Double *const *y*, OGPS_Double *const *z*)

Gets the fully transformed value of a data point vector at a given index position.

Other than with ogps_GetListPoint this function also applies the axes transformation specified in the axes definition area of the ISO 5436-2 XML document.

Remarks:

ogps_GetListCoord is the valid access method only if point vectors are stored in list format. If this is not the case and point vectors are stored in matrix format with encoded topology information, then you must use ogps_GetMatrixCoord instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ogps_GetDocument.

See also:

ogps_GetListPoint, ogps_GetMatrixCoord

On failure you may get further information by calling ogps_GetErrorMessage hereafter.

Parameters:

handle Operate on this handle object.

index The index of the surface position.

x Returns the fully transformed x component of the point value at the given index position. If this parameter is set to NULL, the x axis component will be safely ignored.

y Returns the fully transformed y component of the point value at the given index position. If this parameter is set to NULL, the y axis component will be safely ignored.

z Returns the fully transformed z component of the point value at the given index position. If this parameter is set to NULL, the z axis component will be safely ignored.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.35 _OPENGPS_EXPORT OGPS_ULong ogps_GetListDimension (const OGPS_ISO5436_2Handle *handle*)

Gets information on the list dimension of the current data structure.

Be sure that the current handle points to a list structure.

Parameters:

handle Operate on this handle object.

Returns:

The dimension of the current list structure.

Remarks:

Important: After execution check with ogps_HasError() whether the request was processed correctly, otherwise future behavior of your program is undefined!

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by PrintDimensions().

9.1.5.36 _OPENGPS_EXPORT void ogps_GetListPoint (const OGPS_ISO5436_2Handle *handle*, const OGPS_ULong *index*, OGPS_PointVectorPtr const *vector*)

Gets the raw value of a data point vector at a given index position.

Remarks:

ogps_GetListPoint is the valid access method only if point vectors are stored in list format. If this is not the case and point vectors are stored in matrix format with encoded topology information, then you must use ogps_GetMatrixPoint instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ogps_GetDocument.

See also:

ogps_GetListCoord, ogps_GetMatrixPoint

On failure you may get further information by calling ogps_GetErrorMessage hereafter.

Parameters:

handle Operate on this handle object.

index The index of the surface position.

vector Returns the raw point value at the given position.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.37 _OPENGPS_EXPORT OGPS_Boolean ogps_GetListPosition (const OGPS_PointIteratorPtr *iterator*, OGPS_ULong * *index*)

Gets the current position of the iterator.

See also:

ogps_MoveNext, ogps_GetMatrixPosition

Parameters:

iterator Operate on this iterator handle.

index Gets the position index.

Returns:

Returns TRUE on success, FALSE otherwise.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER_RETVALBOOL.

9.1.5.38 _OPENGPS_EXPORT void ogps_GetMatrixCoord (const OGPS_ISO5436_2Handle *handle*, const OGPS_ULong *u*, const OGPS_ULong *v*, const OGPS_ULong *w*, OGPS_Double *const *x*, OGPS_Double *const *y*, OGPS_Double *const *z*)

Gets the fully transformed value of a data point vector at a given surface position.

Other than with ogps_GetMatrixPoint this function also applies the axes transformation specified in the axes definition area of the ISO 5436-2 XML document.

Remarks:

ogps_GetMatrixCoord is the valid access method only if point vectors are stored in matrix format with encoded topology information. If this is not the case point vectors are stored in list format. Then you must use ogps_GetListCoord instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ogps_GetDocument.

See also:

ogps_GetMatrixPoint, ogps_GetListCoord

On failure you may get further information by calling ogps_GetErrorMessage hereafter.

Parameters:

handle Operate on this handle object.

u The u-direction of the surface position.

v The v-direction of the surface position.

w The w-direction of the surface position.

x Returns the fully transformed x component of the point value at the given u,v,w position. If this parameter is set to NULL, the x axis component will be safely ignored.

y Returns the fully transformed y component of the point value at the given u,v,w position. If this parameter is set to NULL, the y axis component will be safely ignored.

z Returns the fully transformed z component of the point value at the given u,v,w position. If this parameter is set to NULL, the z axis component will be safely ignored.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.39 _OPENGPS_EXPORT void ogps_GetMatrixDimensions (const OGPS_ISO5436_2Handle *handle*, OGPS_ULong *const *size_u*, OGPS_ULong *const *size_v*, OGPS_ULong *const *size_w*)

Gets information on the matrix dimensions of the current data structure The dimensions are written to the arguments size_u, size_v, size_w.

Be sure that the current handle points to a matrix structure.

Parameters:

handle Operate on this handle object.

size_u Pointer to the return value of the matrix dimension in u-direction

size_v Pointer to the return value of the matrix dimension in v-direction

size_w Pointer to the return value of the matrix dimension in w-direction Also see remarks.

Remarks:

Important: After execution check with ogps_HasError() whether the request was processed correctly, otherwise future behavior of your program is undefined!

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by PrintDimensions().

9.1.5.40 _OPENGPS_EXPORT void ogps_GetMatrixPoint (const OGPS_ISO5436_2Handle *handle*, const OGPS_ULong *u*, const OGPS_ULong *v*, const OGPS_ULong *w*, OGPS_PointVectorPtr const *vector*)

Gets the raw value of a data point vector at a given surface position.

Remarks:

ogps_GetMatrixPoint is the valid access method only if point vectors are stored in matrix format with encoded topology information. If this is not the case point vectors are stored in list format. Then you must use ogps_GetListPoint instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ogps_GetDocument.

See also:

ogps_GetMatrixCoord, ogps_GetListPoint

On failure you may get further information by calling ogps_GetErrorMessage hereafter.

Parameters:

handle Operate on this handle object.

u The u-direction of the surface position.

v The v-direction of the surface position.

w The w-direction of the surface position.

vector Returns the raw point value at the given u,v,w position.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.41 _OPENGPS_EXPORT OGPS_Boolean ogps_GetMatrixPosition (const OGPS_PointIteratorPtr *iterator*, OGPS_ULong * *u*, OGPS_ULong * *v*, OGPS_ULong * *w*)

Gets the current position of the iterator in topology coordinates.

See also:

ogps_MoveNext, ogps_GetListPosition

Parameters:

iterator Operate on this iterator handle.

u Gets the position index of the u component of the surface. If this parameter is set to NULL, it will be safely ignored and nothing returns here.

v Gets the position index of the v component of the surface. If this parameter is set to NULL, it will be safely ignored and nothing returns here.

w Gets the position index of the w component of the surface. If this parameter is set to NULL, it will be safely ignored and nothing returns here.

Returns:

Returns TRUE on success, FALSE otherwise.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER_RETVALBOOL.

Referenced by readonlyExampleMatrix().

9.1.5.42 _OPENGPS_EXPORT size_t ogps_GetNameInfo (OGPS_Character *const *text*, const size_t *size*)

Gets a short name identifying this software library.

Parameters:

text Allocated string buffer. Will contain the resulting message on success.

size The size of the allocated buffer in characters.

Returns:

On success returns the number of characters written, otherwise returns the required size in characters of an allocated text buffer to store the name including the terminating null character.

References _OPENGPS_GENERIC_EXCEPTION_HANDLER, and OpenGPS::Info::GetName().

Here is the call graph for this function:

9.1.5.43 _OPENGPS_EXPORT OGPS_DataPointType ogps_GetPointType (const OGPS_DataPointPtr *dataPoint*)

Gets type information about the current value stored in a given data point instance.

Returns OGPS_MissingPointType if this instance does not store any value at all. You may check ogps_HasError whether this operation has been successfully executed.

Parameters:

dataPoint Operate on this data point instance.

References _ASSERT, _OPENGPS_GENERIC_EXCEPTION_HANDLER, and OGPS_MissingPointType.

9.1.5.44 _OPENGPS_EXPORT OGPS_DataPointType ogps_GetPointTypeX (const OGPS_PointVectorPtr *vector*)

Gets the type of points of the x component of the given vector.

Parameters:

vector Operate on this point vector instance.

References _ASSERT, _OPENGPS_GENERIC_EXCEPTION_HANDLER, and OGPS_MissingPointType.

Referenced by readonlyExample2().

9.1.5.45 _OPENGPS_EXPORT OGPS_DataPointType ogps_GetPointTypeY (const OGPS_PointVectorPtr *vector*)

Gets the type of points of the y component of the given vector.

Parameters:

vector Operate on this point vector instance.

References _ASSERT, _OPENGPS_GENERIC_EXCEPTION_HANDLER, and OGPS_-MissingPointType.

Referenced by readonlyExample2().

9.1.5.46 _OPENGPS_EXPORT OGPS_DataPointType ogps_GetPointTypeZ (const OGPS_-PointVectorPtr *vector*)

Gets the type of points of the z component of the given vector.

Parameters:

vector Operate on this point vector instance.

References _ASSERT, _OPENGPS_GENERIC_EXCEPTION_HANDLER, and OGPS_-MissingPointType.

Referenced by readonlyExample2().

9.1.5.47 _OPENGPS_EXPORT OGPS_Boolean ogps_GetVendorSpecific (const OGPS_ISO5436_-2Handle *handle*, const OGPS_Character ∗ *vendorURI*, const OGPS_Character ∗ *fileName*, const OGPS_Character ∗ *targetPath*)

Extracts vendorspecific data from the current archive to a given file location.

If the current X3P archive contains vendorspecific data registered for a vendorURI under the given filename in the root directory of the archive, the compressed file will be extracted to the given location.

See also:

ogps_AppendVendorSpecific

Parameters:

handle Operate on this handle object.

vendorURI Your very own vendor specifier in a URI conformant format.

fileName The name of the file to be expected in the root of the archive which is to be decompressed.

targetPath The file in the archive will get extracted here.

Return values:

FALSE if there is no file registered for the given vendorURI within the archive, TRUE if the file has been found and extracted.

References _ASSERT, _OPENGPS_GENERIC_EXCEPTION_HANDLER, and FALSE.

9.1.5.48 _OPENGPS_EXPORT size_t ogps_GetVersionInfo (OGPS_Character ∗const *text*, const size_t *size*)

Gets the version identifier of this software library.

Parameters:

text Allocated string buffer. Will contain the resulting message on success.

size The size of the allocated buffer in characters.

Returns:

On success returns the number of characters written, otherwise returns the required size in characters of an allocated text buffer to store the version identifier including the terminating null character.

References _OPENGPS_GENERIC_EXCEPTION_HANDLER, and OpenGPS::Info::GetVersion().

Here is the call graph for this function:

9.1.5.49 _OPENGPS_EXPORT OGPS_DataPointPtr const ogps_GetX (OGPS_PointVectorPtr const *vector*)

Gets the x component of the given vector.

Parameters:

vector Operate on this point vector instance.

References _ASSERT, _OPENGPS_GENERIC_EXCEPTION_HANDLER, and NULL.

9.1.5.50 _OPENGPS_EXPORT void ogps_GetXYZ (const OGPS_PointVectorPtr *vector*, OGPS_Double *const *x*, OGPS_Double *const *y*, OGPS_Double *const *z*)

Gets the values of each component of the given point vector.

Parameters:

vector Operate on this point vector instance.

x Gets the value of the x component. If this parameter is set to NULL, this does nothing.

y Gets the value of the y component. If this parameter is set to NULL, this does nothing.

z Gets the value of the z component. If this parameter is set to NULL, this does nothing.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by readonlyExample(), and readonlyExampleMatrix().

9.1.5.51 _OPENGPS_EXPORT OGPS_DataPointPtr const ogps_GetY (OGPS_PointVectorPtr const *vector*)

Gets the y component of the given vector.

Parameters:

vector Operate on this point vector instance.

References _ASSERT, _OPENGPS_GENERIC_EXCEPTION_HANDLER, and NULL.

9.1.5.52 _OPENGPS_EXPORT OGPS_DataPointPtr const ogps_GetZ (OGPS_PointVectorPtr const *vector*)

Gets the z component of the given vector.

Parameters:

vector Operate on this point vector instance.

References _ASSERT, _OPENGPS_GENERIC_EXCEPTION_HANDLER, and NULL.

9.1.5.53 _OPENGPS_EXPORT OGPS_Boolean ogps_HasError ()

Returns TRUE when the statement executed previously failed.

Use ogps_GetErrorId to obtain more information about the failure.

References OpenGPS::ExceptionHistory::GetLastErrorId(), and OGPS_ExNone.

Referenced by convertFormat(), mediumComplexExample(), PrintDimensions(), readonlyExample(), readonlyExample2(), readonlyExample3(), readonlyExample4(), and readonlyExampleMatrix().

Here is the call graph for this function:

9.1.5.54 _OPENGPS_EXPORT OGPS_Boolean ogps_HasNextPoint (const OGPS_-PointIteratorPtr *iterator*)

Asks if there is another point available to iterate.

Remarks:

Use this function with an iterator handle obtained from ogps_CreateNextPointIterator.

See also:

ogps_MoveNextPoint, ogps_HasPrevPoint

Parameters:

iterator Operate on this iterator handle.

Returns:

Returns TRUE if there is at least one more point available to iterate, FALSE otherwise.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER_RETVALBOOL.

Referenced by convertFormat(), readonlyExample(), readonlyExample2(), and readonlyExampleMatrix().

9.1.5.55 _OPENGPS_EXPORT OGPS_Boolean ogps_HasPrevPoint (const OGPS_-PointIteratorPtr *iterator*)

Asks if there is another point available to iterate.

Remarks:

Use this function with an iterator handle obtained from ogps_CreatePrevPointIterator.

See also:

ogps_MovePrevPoint, ogps_HasNextPoint

Parameters:

iterator Operate on this iterator handle.

Returns:

Returns TRUE if there is at least one more point available to iterate, FALSE otherwise.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER_RETVALBOOL.

9.1.5.56 _OPENGPS_EXPORT OGPS_Boolean ogps_IsMatrix (const OGPS_ISO5436_2Handle *handle*)

Gets information on the structure with which the point measurement data is stored.

Parameters:

handle Operate on this handle object.

Returns:

Returns TRUE if the underlying document structure represents the matrix topology, otherwise FALSE, which corresponds to unordered list structure. Also see remarks.

Remarks:

Important: After execution check with ogps_HasError() whether the request was processed correctly, otherwise futurue behavior of your program is undefined!

References _ASSERT, _OPENGPS_GENERIC_EXCEPTION_HANDLER, and FALSE.

Referenced by convertFormat(), and PrintDimensions().

9.1.5.57 _OPENGPS_EXPORT OGPS_Boolean ogps_IsMatrixCoordValid (const OGPS_ISO5436_2Handle *handle*, const OGPS_ULong *u*, const OGPS_ULong *v*, const OGPS_ULong *w*)

Asks if there is point vector data stored at the given matrix position.

Since the matrix storage format encodes topology information there may not exist valid point vector data for every u,v,w position because there was no measurement data available.

See also:

ogps_GetMatrixPoint, ogps_GetMatrixCoord, ogps_SetMatrixPoint

Parameters:

handle Operate on this handle object.

u The u-direction of the surface position.

v The v-direction of the surface position.

w The w-direction of the surface position.

Returns:

Returns TRUE if the vector point data at the given position is valid, otherwise return FALSE to indicate there is no measurement data available at this particular position.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER_RETVALBOOL.

9.1.5.58 _OPENGPS_EXPORT OGPS_Boolean ogps_IsValidPoint (const OGPS_PointVectorPtr *vector*)

Asks if the given point vector stores a valid data point.

A valid point vector does not have components where some or all of its values are missing. Missing or invalid points are indicated by OGPS_MissingPointType.

Parameters:

vector Operate on this point vector instance.

Returns:

Returns TRUE if this point vector contains valid point components only, FALSE otherwise.

References _ASSERT, _OPENGPS_GENERIC_EXCEPTION_HANDLER, and FALSE.

Referenced by convertFormat(), readonlyExample(), readonlyExample2(), and readonlyExampleMatrix().

9.1.5.59 _OPENGPS_EXPORT OGPS_Boolean ogps_MoveNextPoint (OGPS_PointIteratorPtr const *iterator*)

Moves the iterator forward.

Remarks:

Use this function directly after initialising the iterator object with ogps_CreateNextPointIterator to move to the first point.

See also:

ogps_HasNextPoint

Parameters:

iterator Operate on this iterator handle.

Returns:

Returns TRUE on success, FALSE otherwise.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER_RETVALBOOL.

Referenced by convertFormat(), mediumComplexExample(), readonlyExample(), readonlyExample2(), and readonlyExampleMatrix().

9.1.5.60 _OPENGPS_EXPORT OGPS_Boolean ogps_MovePrevPoint (OGPS_PointIteratorPtr const *iterator*)

Moves the iterator backward.

Remarks:

Use this function directly after initialising the iterator object with ogps_CreatePrevPointIterator to move to the first point.

See also:

ogps_HasPrevPoint

Parameters:

iterator Operate on this iterator handle.

Returns:

Returns TRUE on success, FALSE otherwise.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER_RETVALBOOL.

9.1.5.61 _OPENGPS_EXPORT OGPS_ISO5436_2Handle ogps_OpenISO5436_2 (const OGPS_Character *const *file*, const OGPS_Character *const *temp* = `NULL`, const OGPS_Boolean *readOnly* = `FALSE`)

Opens an existing ISO5436-2 XML X3P file.

Remarks:

You must free the returned handle by calling ogps_CloseISO5436_2 when done with it.

See also:

ogps_CloseISO5436_2

Parameters:

file Full path to the ISO5436-2 XML X3P to open.

temp Optionally specifies the new absolute path to the directory where unpacked X3P data gets stored temporarily. If this parameter is set to NULL the default directory for temporary files will be used as specified by your system.

readOnly If set to TRUE subsequend operations on the returned handle object assume that you will access any obtained data as read-only and won't make any changes. This may speed up some operations. If unsure set this parameter to FALSE.

Returns:

On success returns the handle object to the opened file, otherwise a NULL pointer is returned. You may get further information about the failure by calling ogps_GetErrorMessage hereafter.

References _ASSERT, _OPENGPS_DELETE, _OPENGPS_GENERIC_EXCEPTION_HANDLER_CLEANUP, and NULL.

Referenced by convertFormat(), readonlyExample(), readonlyExample2(), and readonlyExampleMatrix().

9.1.5.62 _OPENGPS_EXPORT void ogps_PrintCopyrightInfo ()

Prints the copyright information of this software library to standard output.

References _OPENGPS_GENERIC_EXCEPTION_HANDLER, and OpenGPS::Info::PrintCopyright().

Here is the call graph for this function:

9.1.5.63 _OPENGPS_EXPORT void ogps_PrintLicenseInfo ()

Prints the license information of this software library to standard output.

References _OPENGPS_GENERIC_EXCEPTION_HANDLER, and OpenGPS::Info::PrintLicense().

Here is the call graph for this function:

9.1.5.64 _OPENGPS_EXPORT void ogps_PrintVersionInfo ()

Prints the version identifier of this software library to standard output.

References _OPENGPS_GENERIC_EXCEPTION_HANDLER, and OpenGPS::Info::PrintVersion().

Here is the call graph for this function:

9.1.5.65 _OPENGPS_EXPORT void ogps_ResetNextPointIterator (OGPS_PointIteratorPtr const *iterator*)

Resets the iterator to the beginning and turns this iterator instance into a forward iterator.

Parameters:

iterator Operate on this iterator handle.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.66 _OPENGPS_EXPORT void ogps_ResetPrevPointIterator (OGPS_PointIteratorPtr const *iterator*)

Resets the iterator to the beginning and turns this iterator instance into a backward iterator.

Parameters:

iterator Operate on this iterator handle.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.67 _OPENGPS_EXPORT void ogps_SetCurrentPoint (const OGPS_PointIteratorPtr *iterator*, const OGPS_PointVectorPtr *vector*)

Sets the value of the current point vector.

See also:

ogps_MoveNext

On failure you may get further information by calling ogps_GetErrorMessage hereafter.

Parameters:

iterator Operate on this iterator handle.

vector New value of the current point vector. May be NULL to indicate an invalid point.

References _ASSERT, _OPENGPS_GENERIC_EXCEPTION_HANDLER, and NULL.

Referenced by convertFormat(), and mediumComplexExample().

9.1.5.68 _OPENGPS_EXPORT void ogps_SetDouble (OGPS_DataPointPtr const *dataPoint*, const OGPS_Double *value*)

Stores a new value within a given data point instance.

Also adjusts the current type information reflecting this set operation, if necessary. You may check ogps_-HasError whether this operation has been successfully executed.

Parameters:

dataPoint Operate on this data point instance.

value The new value to be stored.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.69 _OPENGPS_EXPORT void ogps_SetDoubleX (OGPS_PointVectorPtr const *vector*, const OGPS_Double *value*)

Sets the new value for the x component of a given point vector.

Parameters:

vector Operate on this point vector instance.

value The new value.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.70 _OPENGPS_EXPORT void ogps_SetDoubleY (OGPS_PointVectorPtr const *vector*, const OGPS_Double *value*)

Sets the new value for the y component of a given point vector.

Parameters:

vector Operate on this point vector instance.

value The new value.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.71 _OPENGPS_EXPORT void ogps_SetDoubleZ (OGPS_PointVectorPtr const *vector*, const OGPS_Double *value*)

Sets the new value for the z component of a given point vector.

Parameters:

vector Operate on this point vector instance.

value The new value.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by mediumComplexExample(), performanceDouble(), and simpleExample().

9.1.5.72 _OPENGPS_EXPORT void ogps_SetFloat (OGPS_DataPointPtr const *dataPoint*, const OGPS_Float *value*)

Stores a new value within a given data point instance.

Also adjusts the current type information reflecting this set operation, if necessary. You may check ogps_-HasError whether this operation has been successfully executed.

Parameters:

dataPoint Operate on this data point instance.

value The new value to be stored.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.73 _OPENGPS_EXPORT void ogps_SetFloatX (OGPS_PointVectorPtr const *vector*, const OGPS_Float *value*)

Sets the new value for the x component of a given point vector.

Parameters:

vector Operate on this point vector instance.

value The new value.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.74 _OPENGPS_EXPORT void ogps_SetFloatY (OGPS_PointVectorPtr const *vector*, const OGPS_Float *value*)

Sets the new value for the y component of a given point vector.

Parameters:

vector Operate on this point vector instance.

value The new value.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by mediumComplexExample().

9.1.5.75 _OPENGPS_EXPORT void ogps_SetFloatZ (OGPS_PointVectorPtr const *vector*, const OGPS_Float *value*)

Sets the new value for the z component of a given point vector.

Parameters:

vector Operate on this point vector instance.

value The new value.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.76 _OPENGPS_EXPORT void ogps_SetInt16 (OGPS_DataPointPtr const *dataPoint*, const OGPS_Int16 *value*)

Stores a new value within a given data point instance.

Also adjusts the current type information reflecting this set operation, if necessary. You may check ogps_-HasError whether this operation has been successfully executed.

Parameters:

dataPoint Operate on this data point instance.

value The new value to be stored.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.77 _OPENGPS_EXPORT void ogps_SetInt16X (OGPS_PointVectorPtr const *vector*, const OGPS_Int16 *value*)

Sets the new value for the x component of a given point vector.

Parameters:

vector Operate on this point vector instance.

value The new value.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.78 _OPENGPS_EXPORT void ogps_SetInt16Y (OGPS_PointVectorPtr const *vector*, const OGPS_Int16 *value*)

Sets the new value for the y component of a given point vector.

Parameters:

vector Operate on this point vector instance.

value The new value.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.79 _OPENGPS_EXPORT void ogps_SetInt16Z (OGPS_PointVectorPtr const *vector*, const OGPS_Int16 *value*)

Sets the new value for the z component of a given point vector.

Parameters:

vector Operate on this point vector instance.

value The new value.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by performanceInt16().

9.1.5.80 _OPENGPS_EXPORT void ogps_SetInt32 (OGPS_DataPointPtr const *dataPoint*, const OGPS_Int32 *value*)

Stores a new value within a given data point instance.

Also adjusts the current type information reflecting this set operation, if necessary. You may check ogps_-HasError whether this operation has been successfully executed.

Parameters:

dataPoint Operate on this data point instance.

value The new value to be stored.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.81 _OPENGPS_EXPORT void ogps_SetInt32X (OGPS_PointVectorPtr const *vector*, const OGPS_Int32 *value*)

Sets the new value for the x component of a given point vector.

Parameters:

vector Operate on this point vector instance.

value The new value.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by mediumComplexExample().

9.1.5.82 _OPENGPS_EXPORT void ogps_SetInt32Y (OGPS_PointVectorPtr const *vector*, const OGPS_Int32 *value*)

Sets the new value for the y component of a given point vector.

Parameters:

vector Operate on this point vector instance.

value The new value.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.83 _OPENGPS_EXPORT void ogps_SetInt32Z (OGPS_PointVectorPtr const *vector*, const OGPS_Int32 *value*)

Sets the new value for the z component of a given point vector.

Parameters:

vector Operate on this point vector instance.

value The new value.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

9.1.5.84 _OPENGPS_EXPORT void ogps_SetListPoint (const OGPS_ISO5436_2Handle *handle*, const OGPS_ULong *index*, const OGPS_PointVectorPtr *vector*)

Sets the value of a three-dimensional data point vector at a given index position.

Manipulates the point vector data stored in an ISO5436-2 X3P file directly.

Remarks:

The data types of the values stored in the given vector parameter must correspond with the data types specified in the ISO5436-2 XML document within the axes definition area. Also ogps_SetListPoint is the valid access method only if point vectors are stored in list format. If this is not the case and point vectors are stored in matrix format with encoded topology information then you must use ogps_-SetMatrixPoint instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ogps_GetDocument.

See also:

ogps_SetMatrixPoint

On failure you may get further information by calling ogps_GetErrorMessage hereafter.

Parameters:

handle Operate on this handle object.

index The index position of the point vector to manipulate.

vector Set this point value at the given index position.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by performanceDouble(), and performanceInt16().

9.1.5.85 _OPENGPS_EXPORT void ogps_SetMatrixPoint (const OGPS_ISO5436_2Handle *handle*, const OGPS_ULong *u*, const OGPS_ULong *v*, const OGPS_ULong *w*, const OGPS_PointVectorPtr *vector*)

Sets the value of a three-dimensional data point vector at a given surface position.

Manipulates the point vector data stored in an ISO5436-2 X3P file directly.

Remarks:

The data types of the values stored in the given vector parameter must correspond with the data types specified in the ISO5436-2 XML document within the axes definition area. Also ogps_SetMatrixPoint is the valid access method only if point vectors are stored in matrix format with encoded topology information. If this is not the case point vectors are stored in list format. Then you must use ogps_SetListPoint instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ogps_GetDocument.

See also:

ogps_SetListPoint

On failure you may get further information by calling ogps_GetErrorMessage hereafter.

Parameters:

handle Operate on this handle object.

u The u-direction of the surface position.

v The v-direction of the surface position.

w The w-direction of the surface position.

vector Set this point value at the given u,v,w position. If this parameter is set to NULL, this indicates there is no measurement data available for this position. This is due to the topology encoding properties of the matrix format storage.

References _ASSERT, _OPENGPS_GENERIC_EXCEPTION_HANDLER, and NULL.

Referenced by simpleExample().

9.1.5.86 _OPENGPS_EXPORT void ogps_WriteISO5436_2 (const OGPS_ISO5436_2Handle *handle*, const int *compressionLevel* = `-1`)

Writes any changes back to the X3P file.

Call this function before ogps_CloseISO5436_2 if you want to store the changes you have made.

See also:

ogps_CreateMatrixISO5436_2, ogps_CreateListISO5436_2, ogps_OpenISO5436_2, ogps_CloseISO5436_2

Parameters:

handle Operate on this handle object.

compressionLevel Optionally specifies the compression level used when writing the X3P file which is nothing else than a simple zip file container. The default value for this parameter is (-1) which enables standard compression level as a good trade-off between processing time and compression ratio. Values between 0 and 9 are possible. A value of 0 means "no compression" and a value of 9 enables the highest level compression rate at the cost of highest computation time.

Returns:

Returns TRUE on success and FALSE if anything went wrong. When FALSE no changes will be saved. You may get further information about the failure by calling ogps_GetErrorMessage hereafter.

References _ASSERT, and _OPENGPS_GENERIC_EXCEPTION_HANDLER.

Referenced by convertFormat(), mediumComplexExample(), performanceDouble(), performanceInt16(), and simpleExample().

9.2 Sample files demonstrating X3P file format.

Files

- file ISO5436-sample1.x3p

  Simple matrix example with documented XML-main file.

- file ISO5436-sample1_bin.x3p

  Simple binary matrix example with documented XML-main file.

- file ISO5436-sample2.x3p

  Simple unordered list example with documented XML-main file.

- file ISO5436-sample3.x3p

  Matrix example mixed incremental/absolute axis.

- file ISO5436-sample4.x3p

  Matrix example with integer encoded z-values.

- file ISO5436-sample4_bin.x3p

  Matrix example with integer encoded z-values in binary format.

9.3 C++ Interface for X3P-library

Collaboration diagram for C++ Interface for X3P-library:

Files

- file info.hxx

  Copyright and license information.

- file iso5436_2.hxx

  Represents the ISO5436-2 XML X3P file format container.

- file opengps.hxx

  Common define's and typedef's.

- file point_iterator.hxx

  Interface to a point iterator used for direct manipulation of point data stored in an ISO 5436-2 X3P file.

- file point_vector.hxx

  Typesafe representation of three-dimensional point measurement data.

- file point_vector_base.hxx

  Very fundamental representation of three-dimensional point measurement data.

- file string.hxx

  An enhanced std::string string type.

Namespaces

- namespace OpenGPS

  The standard namespace for the openGPS software library.

Modules

- Ansi-C Interface for X3P library

  Ansi-C Interface for X3P library.

10 Directory Documentation

10.1 ISO5436_2_XML/c/ Directory Reference

Files

- file data_point_c.cxx
- file data_point_c.hxx

Handle mechanism that makes a C++ OpenGPS::DataPoint object accessible through its corresponding C interface.

- file info_c.cxx
- file iso5436_2_c.cxx
- file iso5436_2_handle_c.hxx

  Handle mechanism which makes a C++ OpenGPS::ISO5436_2 object accessible through the corresponding C interface of an ISO5436-2 X3P file container.

- file messages_c.cxx
- file messages_c.hxx

  Error and warning messages during application.

- file point_iterator_c.cxx
- file point_iterator_c.hxx

  Handle mechanism which makes a C++ OpenGPS::PointIterator object accessible through the corresponding C interface of a point iterator.

- file point_vector_c.cxx
- file point_vector_c.hxx

  Handle mechanism which makes a C++ OpenGPS::PointVector object accessible through the corresponding C interface of a point vector.

10.2 S:/openGPS/ISO5436_XML/trunk/include/opengps/cxx/ Directory Reference

Files

- file data_point.hxx

  Typesafe representation of a single data point value.

- file exceptions.hxx

  Exception types thrown within this software library.

- file info.hxx

  Copyright and license information.

- file iso5436_2.hxx

  Represents the ISO5436-2 XML X3P file format container.

- file iso5436_2_handle.hxx

  Enhancing C++ part of the C interface of the abstract data type of the ISO 5436-2 X3P file format.

- file include/opengps/cxx/iso5436_2_xsd.hxx

  Generated from iso5436_2.xsd.

- file opengps.hxx

  Common define's and typedef's.

- file point_iterator.hxx

  Interface to a point iterator used for direct manipulation of point data stored in an ISO 5436-2 X3P file.

- file point_vector.hxx

  Typesafe representation of three-dimensional point measurement data.

- file point_vector_base.hxx

  Very fundamental representation of three-dimensional point measurement data.

- file string.hxx

  An enhanced std::string string type.

10.3 ISO5436_2_XML/cxx/ Directory Reference

Files

- file auto_ptr_types.hxx

  Agglomerated definition of std::auto_ptr commonly used.

- file binary_lsb_point_vector_reader_context.cxx
- file binary_lsb_point_vector_reader_context.hxx

  Implementation of access methods for reading typed point data from a binary file of point vectors.

- file binary_lsb_point_vector_writer_context.cxx
- file binary_lsb_point_vector_writer_context.hxx

  Implementation of access methods for writing typed point data to a binary file of point vectors.

- file binary_msb_point_vector_reader_context.cxx
- file binary_msb_point_vector_reader_context.hxx

  Implementation of access methods for reading typed point data from a binary file of point vectors.

- file binary_msb_point_vector_writer_context.cxx
- file binary_msb_point_vector_writer_context.hxx

  Implementation of access methods for writing typed point data to a binary file of point vectors.

- file binary_point_vector_reader_context.cxx
- file binary_point_vector_reader_context.hxx

  Interface for reading typed point data from an underlying binary file stream of point vectors.

- file binary_point_vector_writer_context.cxx
- file binary_point_vector_writer_context.hxx

  Interface for writing typed point data to an underlying binary of point vectors.

- file data_point_impl.cxx
- file data_point_impl.hxx

  An implementation of a data point to be used for typesafe storage and access of point data.

- file data_point_parser.hxx

  Generic interface of a data point parser for reading/writing point data of any supported data type from/to any supported media.

- file data_point_proxy.cxx
- file double_data_point_parser.cxx
- file double_data_point_parser.hxx

  A data point parser for point data of type OGPS_DoublePointType.

- file double_point_buffer.cxx
- file double_point_buffer.hxx

  Allocate static memory to store point data.

- file environment.cxx
- file environment.hxx

  An interface to provide access to the environment of different architectures and operating systems.

- file exceptions.cxx
- file float_data_point_parser.cxx
- file float_data_point_parser.hxx

  A data point parser for point data of type OGPS_FloatPointType.

- file float_point_buffer.cxx
- file float_point_buffer.hxx

  Allocate static memory to store point data.

- file info.cxx
- file inline_validity.cxx
- file inline_validity.hxx

  Communicate validity of point vectors through special IEEE754 values.

- file int16_data_point_parser.cxx
- file int16_data_point_parser.hxx

  A data point parser for point data of type OGPS_Int16PointType.

- file int16_point_buffer.cxx
- file int16_point_buffer.hxx

  Allocate static memory to store point data.

- file int32_data_point_parser.cxx
- file int32_data_point_parser.hxx

  A data point parser for point data of type OGPS_Int32PointType.

- file int32_point_buffer.cxx
- file int32_point_buffer.hxx

  Allocate static memory to store point data.

- file iso5436_2.cxx
- file iso5436_2_container.cxx
- file iso5436_2_container.hxx

  Concrete implementation of the interface of an X3P container.

- file missing_data_point_parser.cxx
- file missing_data_point_parser.hxx

  A data point parser for point data of type OGPS_MissingPointType.

- file point_buffer.cxx
- file point_buffer.hxx

  Allocate static memory to store point data.

- file point_iterator.cxx
- file point_validity_provider.cxx
- file point_validity_provider.hxx

  Interface to communicate whether the point vector at a given location contains valid data.

- file point_vector.cxx
- file point_vector_iostream.cxx
- file point_vector_iostream.hxx

  Locale related stuff to support locale invariant parsing of point vector data .

- file point_vector_parser.cxx
- file point_vector_parser.hxx

  Generic parser of a point vector.

- file point_vector_parser_builder.cxx
- file point_vector_parser_builder.hxx

  Builder which assembles a concrete instance of the generic point vector parser.

- file point_vector_proxy.cxx
- file point_vector_proxy.hxx

  Make the internal memory structure of distinct data point buffers accessable as point vector data.

- file point_vector_proxy_context.cxx
- file point_vector_proxy_context.hxx

Indexing of point data managed by OpenGPS::VectorBuffer.

- file point_vector_proxy_context_list.cxx
- file point_vector_proxy_context_list.hxx

  Indexing of point data managed by OpenGPS::VectorBuffer.

- file point_vector_proxy_context_matrix.cxx
- file point_vector_proxy_context_matrix.hxx

  Indexing of point data managed by OpenGPS::VectorBuffer.

- file point_vector_reader_context.hxx

  Interface for reading typed point data from an underlying stream of point vectors.

- file point_vector_writer_context.hxx

  Interface for writing typed point data to an underlying stream of point vectors.

- file stdafx.hxx

  Standard includes and defines.

- file string.cxx
- file valid_buffer.cxx
- file valid_buffer.hxx

  Communicate validity of point vectors through external binary file.

- file vector_buffer.cxx
- file vector_buffer.hxx

  The internal memory structure that stores all point measurement data.

- file vector_buffer_builder.cxx
- file vector_buffer_builder.hxx

  Methods to assemble a OpenGPS::VectorBuffer instance.

- file Version.h

  Include file generated from "Version.in" containing the current version.

- file win32_environment.cxx
- file win32_environment.hxx

  The environment of Microsoft Windows operating systems.

- file xml_point_vector_reader_context.cxx
- file xml_point_vector_reader_context.hxx

  Implementation of access methods for reading typed point data from a string list of point vectors.

- file xml_point_vector_writer_context.cxx
- file xml_point_vector_writer_context.hxx

  Implementation of access methods for writing typed point data to a string list of point vectors.

- file zip_stream_buffer.cxx
- file zip_stream_buffer.hxx

  Integrates Info-Zip handles into the common standard io framework.

10.4 S:/openGPS/ISO5436_XML/trunk/include/ Directory Reference

Directories

- directory opengps

10.5 tools/IncRev/ Directory Reference

Files

- file IncRev.c

  Tool to increment version numbers in text files.

10.6 ISO5436_2_XML/ Directory Reference

Directories

- directory c
- directory cxx
- directory xyssl

Files

- file iso5436_2_xsd.cxx
- file src/ISO5436_2_XML/iso5436_2_xsd.hxx

  Generated from iso5436_2.xsd.

- file xsd_Licence_Header.c

10.7 ISO5436_2_XML_Demo/ Directory Reference

Directories

- directory samplefiles

Files

- file ISO5436_2_XML_Demo.cxx

  Example programm demonstrating some of the features of the X3P-data format.

10.8 S:/openGPS/ISO5436_XML/trunk/include/opengps/ Directory Reference

Directories

- directory cxx

Files

- file data_point.h

  The abstract data type of the typesafe representation of a single data point value.

- file data_point_type.h

  An enumeration type which may describe the data type of a current data point value.

- file info.h

  Copyright and license information.

- file iso5436_2.h

  The abstract data type of the ISO 5436-2 X3P file specification.

- file messages.h

  Global handling of error and warning messages.

- file opengps.h

  Common define's and typedef's.

- file point_iterator.h

  The abstract data type of a point iterator used for direct manipulation of point data stored in an ISO 5436-2 X3P file.

- file point_vector.h

  The abstract data type of a point vector which holds values for all components of 3D point data stored in an ISO 5436-2 X3P file.

10.9 ISO5436_2_XML_Demo/samplefiles/ Directory Reference

Files

- file ISO5436-sample1.x3p

  Simple matrix example with documented XML-main file.

- file ISO5436-sample1_bin.x3p

  Simple binary matrix example with documented XML-main file.

- file ISO5436-sample2.x3p

  Simple unordered list example with documented XML-main file.

- file ISO5436-sample3.x3p

  Matrix example mixed incremental/absolute axis.

- file ISO5436-sample4.x3p

  Matrix example with integer encoded z-values.

- file ISO5436-sample4_bin.x3p
  Matrix example with integer encoded z-values in binary format.
- file Sample_Files_Description_for_Doxygen.h

10.10 tools/ Directory Reference

Directories

- directory IncRev

10.11 ISO5436_2_XML/xyssl/ Directory Reference

Files

- file md5.c
- file md5.h

11 Namespace Documentation

11.1 OpenGPS Namespace Reference

11.1.1 Detailed Description

The standard namespace for the openGPS software library.

The main namespace of this library.

Classes

- class BinaryLSBPointVectorReaderContext

Implements OpenGPS::BinaryPointVectorReaderContext for binary files to be parsed on machines reading in least significant byte order.

- class BinaryLSBPointVectorWriterContext

  Implements OpenGPS::BinaryPointVectorWriterContext for binary files to be written on machines operating in least significant byte order.

- class BinaryMSBPointVectorReaderContext

  Implements OpenGPS::BinaryPointVectorReaderContext for binary files to be parsed on machines reading in most significant byte order.

- class BinaryMSBPointVectorWriterContext

  Implements OpenGPS::BinaryPointVectorWriterContext for binary files to be written on machines operating in most significant byte order.

- class BinaryPointVectorReaderContext

  Specialized OpenGPS::PointVectorReaderContext for binary streams.

- class BinaryPointVectorWriterContext

  Implements OpenGPS::PointVectorWriterContext for writing to compressed binary streams of point vectors.

- class DataPoint

  Typesafe representation of a single data point value.

- class DataPointImpl

  A straightforward implementation of OpenGPS::DataPoint.

- class DataPointParser

  Interface for reading/writing point data.

- class DoubleDataPointParser

  Reads/Writes instances of OpenGPS::DataPoint of OGPS_Double point data.

- class DoubleInlineValidity

  Implements OpenGPS::PointValidityProvider as a lookup of a special IEEE754 value.

- class DoublePointBuffer

  Manages static memory and typesafe access.

- class Environment

  Interface for communicating with the operating system and related subjects.

- class Exception

  Describes a general exception.

- class ExceptionHistory

  Maintains a history of exceptions.

- class FloatDataPointParser

  Reads/Writes instances of OpenGPS::DataPoint of OGPS_Float point data.

- class FloatInlineValidity

  Implements OpenGPS::PointValidityProvider as a lookup of a special IEEE754 value.

- class FloatPointBuffer

  Manages static memory and typesafe access.

- class Info

  Publishes license text, ownership and similar information.

- class InputBinaryFileStream

  A binary stream class used for reading from binary files.

- class Int16DataPointParser

  Reads/Writes instances of OpenGPS::DataPoint of OGPS_Int16 point data.

- class Int16PointBuffer

  Manages static memory and typesafe access.

- class Int16ValidBuffer

  Implementation of OpenGPS::ValidBuffer for Z axis of OGPS_Int16 data type.

- class Int32DataPointParser

  Reads/Writes instances of OpenGPS::DataPoint of OGPS_Int32 point data.

- class Int32PointBuffer

  Manages static memory and typesafe access.

- class Int32ValidBuffer

  Implementation of OpenGPS::ValidBuffer for Z axis of OGPS_Int32 data type.

- class ISO5436_2

  Represents the ISO5436-2 XML X3P file format container.

- class ISO5436_2Container

  This is the main gate to this software library.

- class MissingDataPointParser

  Reads/Writes instances of OpenGPS::DataPoint of missing point data.

- class OutputBinaryFileStream

  A binary stream class used for writing to binary files.

- class PointBuffer

  A static memory buffer that stores all point data belonging to a single axis.

- class PointIterator

  Interface to a point iterator.

- class PointValidityProvider

  Interface to communicate the validity of a point vector at a given location.

- class PointVector

  Typesafe representation of three-dimensional point measurement data.

- class PointVectorBase

  Typesafe and very fundamental representation of three-dimensional point measurement data.

- class PointVectorInputStringStream

  A string stream used for parsing a point vector as defined in ISO5436-2 XML.

- class PointVectorInvariantLocale

  The culture invariant locale.

- class PointVectorOutputStringStream

  A locale invariant string stream used to convert a point vector object to its representation as text according to the ISO5436-2 XML specification.

- class PointVectorParser

  Generic parser of a point vector.

- class PointVectorParserBuilder

  Component which handles the building process of a specialized parser object for reading and writing typed point data as a three-vector.

- class PointVectorProxy

  Serves one row of the three distinct buffers of data points managed by OpenGPS::VectorBuffer and raises the expression of accessing a single point vector.

- class PointVectorProxyContext

  Indexing of point data managed by OpenGPS::VectorBuffer.

- class PointVectorProxyContextList

  Indexing of point data managed by OpenGPS::VectorBuffer.

- class PointVectorProxyContextMatrix

  Indexing of point data managed by OpenGPS::VectorBuffer.

- class PointVectorReaderContext

  Encapsulates an underlying stream of point vector data.

- class PointVectorWhitespaceFacet

  Redefines the white space character set to make parsing of a point vector easier.

- class PointVectorWriterContext

  Encapsulates an underlying stream of point vector data.

- class String

  Stores an OGPS_Character sequence.

- class ValidBuffer

Implements the OpenGPS::PointValidityProvider as an external binary file.

- class VectorBuffer

  Implements the memory structure of point measurement data.

- class VectorBufferBuilder

  Creates an object which is able to assemble a OpenGPS::VectorBuffer instance.

- class XmlPointVectorReaderContext

  Specialized OpenGPS::PointVectorReaderContext for point vectors stored as list of strings.

- class XmlPointVectorWriterContext

  Specialized OpenGPS::PointVectorWriterContext for point vectors stored as list of strings.

- class ZipOutputStream

  Provides an output stream interface to write binary data to Info-Zip archives.

- class ZipStreamBuffer

  Provides a buffer interface suitable for streaming the zipFile handle defined in the zlib/minizip package.

Namespaces

- namespace Schemas

  Holds the C++ representations of the structure of all supported XML documents.

Typedefs

- typedef char Byte

  Data type of the size of one (possibly signed) byte.

- typedef Byte * BytePtr

  Pointer to a data type of the size of one (possibly signed) byte.

- typedef char OGPS_ExceptionChar

  Character type used to provide more information about an exception.

- typedef std::auto_ptr< PointBuffer > PointBufferAutoPtr

  std::auto_ptr for usage with OpenGPS::PointBuffer type.

- typedef std::auto_ptr< PointIterator > PointIteratorAutoPtr

  std::auto_ptr for usage with OpenGPS::PointIterator type.

- typedef std::auto_ptr< PointVectorBase > PointVectorAutoPtr

  std::auto_ptr for usage with OpenGPS::PointVectorBase type.

- typedef std::auto_ptr< PointVectorParserBuilder > PointVectorParserBuilderAutoPtr

  std::auto_ptr for usage with OpenGPS::PointVectorParserBuilder type.

- typedef std::auto_ptr< PointVectorProxyContext > PointVectorProxyContextAutoPtr
- typedef unsigned char UnsignedByte

  Data type of the size of one unsigned byte.

- typedef UnsignedByte * UnsignedBytePtr

  Pointer to a data type of the size of one unsigned byte.

- typedef std::auto_ptr< VectorBuffer > VectorBufferAutoPtr

  std::auto_ptr for usage with OpenGPS::VectorBuffer type.

- typedef std::auto_ptr< VectorBufferBuilder > VectorBufferBuilderAutoPtr

  std::auto_ptr for usage with OpenGPS::VectorBufferBuilder type.

11.1.2 Typedef Documentation

11.1.2.1 typedef char OpenGPS::Byte

Data type of the size of one (possibly signed) byte.

11.1.2.2 typedef Byte* OpenGPS::BytePtr

Pointer to a data type of the size of one (possibly signed) byte.

11.1.2.3 typedef char OpenGPS::OGPS_ExceptionChar

Character type used to provide more information about an exception.

11.1.2.4 typedef std::auto_ptr<PointBuffer> OpenGPS::PointBufferAutoPtr

std::auto_ptr for usage with OpenGPS::PointBuffer type.

11.1.2.5 typedef std::auto_ptr< PointIterator > OpenGPS::PointIteratorAutoPtr

std::auto_ptr for usage with OpenGPS::PointIterator type.

11.1.2.6 typedef std::auto_ptr< PointVectorBase > OpenGPS::PointVectorAutoPtr

std::auto_ptr for usage with OpenGPS::PointVectorBase type.

11.1.2.7 typedef std::auto_ptr< PointVectorParserBuilder > OpenGPS::PointVectorParserBuilderAutoPtr

std::auto_ptr for usage with OpenGPS::PointVectorParserBuilder type.

11.1.2.8 typedef std::auto_ptr<PointVectorProxyContext> OpenGPS::PointVectorProxyContextAutoPtr

11.1.2.9 typedef unsigned char OpenGPS::UnsignedByte

Data type of the size of one unsigned byte.

11.1.2.10 typedef UnsignedByte* OpenGPS::UnsignedBytePtr

Pointer to a data type of the size of one unsigned byte.

11.1.2.11 typedef std::auto_ptr< VectorBuffer > OpenGPS::VectorBufferAutoPtr

std::auto_ptr for usage with OpenGPS::VectorBuffer type.

11.1.2.12 typedef std::auto_ptr< VectorBufferBuilder > OpenGPS::VectorBufferBuilderAutoPtr

std::auto_ptr for usage with OpenGPS::VectorBufferBuilder type.

11.2 OpenGPS::Schemas Namespace Reference

11.2.1 Detailed Description

Holds the C++ representations of the structure of all supported XML documents.

The schema namespace for the XML-implementation.

Namespaces

- namespace ISO5436_2

  C++ namespace for the http://www.opengps.eu/2008/ISO5436_2 schema namespace.

11.3 OpenGPS::Schemas::ISO5436_2 Namespace Reference

11.3.1 Detailed Description

C++ namespace for the http://www.opengps.eu/2008/ISO5436_2 schema namespace.

Classes

- class AxesType

  Class corresponding to the AxesType schema type.

- class AxisDescriptionType

  Class corresponding to the AxisDescriptionType schema type.

- class AxisType

  Enumeration class corresponding to the AxisType schema type.

- class DataLinkType

  Class corresponding to the DataLinkType schema type.

- class DataListType

  Class corresponding to the DataListType schema type.

- class DataType

  Enumeration class corresponding to the DataType schema type.

- class Datum

  Class corresponding to the Datum schema type.

- class FeatureType

  Class corresponding to the FeatureType schema type.

- class InstrumentType

  Class corresponding to the InstrumentType schema type.

- class ISO5436_2Type

  Class corresponding to the ISO5436_2Type schema type.

- class MatrixDimensionType

  Class corresponding to the MatrixDimensionType schema type.

- class ProbingSystemType

  Class corresponding to the ProbingSystemType schema type.

- class Record1Type

  Class corresponding to the Record1Type schema type.

- class Record2Type

  Class corresponding to the Record2Type schema type.

- class Record3Type

  Class corresponding to the Record3Type schema type.

- class Record4Type

  Class corresponding to the Record4Type schema type.

- class RotationMatrixElementType

  Class corresponding to the RotationMatrixElementType schema type.

- class RotationType

  Class corresponding to the RotationType schema type.

- class Type

  Enumeration class corresponding to the Type schema type.

Functions

- _OPENGPS_EXPORT bool operator!= (const RotationType &, const RotationType &)
- _OPENGPS_EXPORT bool operator!= (const MatrixDimensionType &, const MatrixDimensionType &)
- _OPENGPS_EXPORT bool operator!= (const DataLinkType &, const DataLinkType &)
- _OPENGPS_EXPORT bool operator!= (const DataListType &, const DataListType &)
- _OPENGPS_EXPORT bool operator!= (const ProbingSystemType &, const ProbingSystemType &)
- _OPENGPS_EXPORT bool operator!= (const InstrumentType &, const InstrumentType &)
- _OPENGPS_EXPORT bool operator!= (const AxisDescriptionType &, const AxisDescriptionType &)
- _OPENGPS_EXPORT bool operator!= (const AxesType &, const AxesType &)
- _OPENGPS_EXPORT bool operator!= (const Record4Type &, const Record4Type &)
- _OPENGPS_EXPORT bool operator!= (const Record3Type &, const Record3Type &)
- _OPENGPS_EXPORT bool operator!= (const Record2Type &, const Record2Type &)
- _OPENGPS_EXPORT bool operator!= (const ISO5436_2Type &, const ISO5436_2Type &)
- _OPENGPS_EXPORT bool operator!= (const Record1Type &, const Record1Type &)
- _OPENGPS_EXPORT void operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const Datum &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMAttr &, const Datum &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const Datum &)
- _OPENGPS_EXPORT void operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const Type &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMAttr &, const Type &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const Type &)
- _OPENGPS_EXPORT void operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const DataType &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMAttr &, const DataType &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const DataType &)
- _OPENGPS_EXPORT void operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const AxisType &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMAttr &, const AxisType &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const AxisType &)
- _OPENGPS_EXPORT void operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const FeatureType &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMAttr &, const FeatureType &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const FeatureType &)
- _OPENGPS_EXPORT void operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const RotationMatrixElementType &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMAttr &, const RotationMatrixElementType &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const RotationMatrixElementType &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const RotationType &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const MatrixDimensionType &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const DataLinkType &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const DataListType &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const ProbingSystemType &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const InstrumentType &)

- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const AxisDescriptionType &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const AxesType &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const Record4Type &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const Record3Type &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const Record2Type &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const ISO5436_2Type &)
- _OPENGPS_EXPORT void operator<< (::xercesc::DOMElement &, const Record1Type &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const Datum &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const Type &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, Type::value)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const DataType &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, DataType::value)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const AxisType &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, AxisType::value)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const FeatureType &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const RotationMatrixElementType &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const RotationType &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const MatrixDimensionType &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const DataLinkType &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const DataListType &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const ProbingSystemType &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const InstrumentType &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const AxisDescriptionType &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const AxesType &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const Record4Type &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const Record3Type &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const Record2Type &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const ISO5436_2Type &)
- _OPENGPS_EXPORT::std::wostream & operator<< (::std::wostream &, const Record1Type &)
- _OPENGPS_EXPORT bool operator== (const RotationType &, const RotationType &)
- _OPENGPS_EXPORT bool operator== (const MatrixDimensionType &, const MatrixDimensionType &)
- _OPENGPS_EXPORT bool operator== (const DataLinkType &, const DataLinkType &)
- _OPENGPS_EXPORT bool operator== (const DataListType &, const DataListType &)
- _OPENGPS_EXPORT bool operator== (const ProbingSystemType &, const ProbingSystemType &)
- _OPENGPS_EXPORT bool operator== (const InstrumentType &, const InstrumentType &)
- _OPENGPS_EXPORT bool operator== (const AxisDescriptionType &, const AxisDescriptionType &)
- _OPENGPS_EXPORT bool operator== (const AxesType &, const AxesType &)
- _OPENGPS_EXPORT bool operator== (const Record4Type &, const Record4Type &)
- _OPENGPS_EXPORT bool operator== (const Record3Type &, const Record3Type &)
- _OPENGPS_EXPORT bool operator== (const Record2Type &, const Record2Type &)
- _OPENGPS_EXPORT bool operator== (const ISO5436_2Type &, const ISO5436_2Type &)
- _OPENGPS_EXPORT bool operator== (const Record1Type &, const Record1Type &)

Serialization functions for the ISO5436_2 document root.

The only global element: The root node

- _OPENGPS_EXPORT::xsd::cxx::xml::dom::auto_ptr< ::xercesc::DOMDocument > ISO5436_2 (const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xml_schema::flags f=0)

  Serialize to a new Xerces-C++ DOM document.

- _OPENGPS_EXPORT void ISO5436_2 (::xercesc::DOMDocument &d, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x,::xml_schema::flags f=0)

  Serialize to an existing Xerces-C++ DOM document.

- _OPENGPS_EXPORT void ISO5436_2 (::xercesc::XMLFormatTarget &ft, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xercesc::DOMErrorHandler &eh, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a Xerces-C++ XML format target with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT void ISO5436_2 (::xercesc::XMLFormatTarget &ft, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xml_schema::error_handler &eh, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a Xerces-C++ XML format target with an error handler.

- _OPENGPS_EXPORT void ISO5436_2 (::xercesc::XMLFormatTarget &ft, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a Xerces-C++ XML format target.

- _OPENGPS_EXPORT void ISO5436_2 (::std::ostream &os, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xercesc::DOMErrorHandler &eh, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a standard output stream with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT void ISO5436_2 (::std::ostream &os, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xml_schema::error_handler &eh, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a standard output stream with an error handler.

- _OPENGPS_EXPORT void ISO5436_2 (::std::ostream &os, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a standard output stream.

Parsing functions for the ISO5436_2 document root.

The only global element: The root node

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > ISO5436_2 (::xercesc::DOMDocument *d,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > ISO5436_2 (const ::xercesc::DOMDocument &d,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > ISO5436_2 (const ::xercesc::DOMInputSource &is,::xercesc::DOMErrorHandler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM input source with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > ISO5436_2 (const ::xercesc::DOMInputSource &is,::xml_schema::error_handler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM input source with an error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > ISO5436_2 (const ::xercesc::DOMInputSource &is,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM input source.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > ISO5436_2 (::std::istream &is, const ::std::wstring &id,::xercesc::DOMErrorHandler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > ISO5436_2 (::std::istream &is, const ::std::wstring &id,::xml_schema::error_handler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > ISO5436_2 (::std::istream &is, const ::std::wstring &id,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > ISO5436_2 (::std::istream &is,::xercesc::DOMErrorHandler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > ISO5436_2 (::std::istream &is,::xml_schema::error_handler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > ISO5436_2 (::std::istream &is,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > ISO5436_2 (const ::std::wstring &uri,::xercesc::DOMErrorHandler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > ISO5436_2 (const ::std::wstring &uri,::xml_schema::error_handler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a URI or a local file with an error handler.
- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > ISO5436_2 (const ::std::wstring &uri,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a URI or a local file.

11.3.2 Function Documentation

11.3.2.1 _OPENGPS_EXPORT::xsd::cxx::xml::dom::auto_ptr<::xercesc::DOMDocument > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type & *x*, const ::xml_schema::namespace_infomap & *m*, ::xml_schema::flags *f* = 0)

Serialize to a new Xerces-C++ DOM document.

Parameters:

x An object model to serialize.

m A namespace information map.

f Serialization flags.

Returns:

A pointer to the new Xerces-C++ DOM document.

Referenced by ISO5436_2(), OpenGPS::ISO5436_2Container::ReadXmlDocument(), and OpenGPS::ISO5436_2Container::SaveXmlDocument().

11.3.2.2 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::DOMDocument & *d*, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type & *x*, ::xml_schema::flags *f* = 0)

Serialize to an existing Xerces-C++ DOM document.

Parameters:

d A Xerces-C++ DOM document.

x An object model to serialize.

f Serialization flags.

Note that it is your responsibility to create the DOM document with the correct root element as well as set the necessary namespace mapping attributes.

References n.

11.3.2.3 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::XMLFormatTarget & *ft*, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type & *x*, const ::xml_schema::namespace_infomap & *m*, ::xercesc::DOMErrorHandler & *eh*, const ::std::wstring & *e* = L"UTF-8", ::xml_schema::flags *f* = 0)

Serialize to a Xerces-C++ XML format target with a Xerces-C++ DOM error handler.

Parameters:

ft A Xerces-C++ XML format target.

x An object model to serialize.

m A namespace information map.

eh A Xerces-C++ DOM error handler.

e A character encoding to produce XML in.

f Serialization flags.

This function reports serialization errors by calling the error handler.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.4 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::XMLFormatTarget & *ft*, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type & *x*, const ::xml_schema::namespace_infomap & *m*, ::xml_schema::error_handler & *eh*, const ::std::wstring & *e* = `L"UTF-8"`, ::xml_schema::flags *f* = `0`)

Serialize to a Xerces-C++ XML format target with an error handler.

Parameters:

ft A Xerces-C++ XML format target.

x An object model to serialize.

m A namespace information map.

eh An error handler.

e A character encoding to produce XML in.

f Serialization flags.

This function reports serialization errors by calling the error handler.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.5 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::XMLFormatTarget & *ft*, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type & *x*, const ::xml_schema::namespace_infomap & *m*, const ::std::wstring & *e* = `L"UTF-8"`, ::xml_schema::flags *f* = `0`)

Serialize to a Xerces-C++ XML format target.

Parameters:

ft A Xerces-C++ XML format target.

x An object model to serialize.

m A namespace information map.

e A character encoding to produce XML in.

f Serialization flags.

This function uses exceptions to report serialization errors.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.6 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::ostream & *os*, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type & *x*, const ::xml_schema::namespace_infomap & *m*, ::xercesc::DOMErrorHandler & *eh*, const ::std::wstring & *e* = `L"UTF-8"`, ::xml_schema::flags *f* = `0`)

Serialize to a standard output stream with a Xerces-C++ DOM error handler.

Parameters:

os A standrad output stream.

x An object model to serialize.

m A namespace information map.

eh A Xerces-C++ DOM error handler.

e A character encoding to produce XML in.

f Serialization flags.

This function reports serialization errors by calling the error handler.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.7 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::ostream & *os*, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type & *x*, const ::xml_schema::namespace_infomap & *m*, ::xml_schema::error_handler & *eh*, const ::std::wstring & *e* = `L"UTF-8"`, ::xml_schema::flags *f* = `0`)

Serialize to a standard output stream with an error handler.

Parameters:

os A standrad output stream.

x An object model to serialize.

m A namespace information map.

eh An error handler.

e A character encoding to produce XML in.

f Serialization flags.

This function reports serialization errors by calling the error handler.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.8 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::ostream & *os*, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type & *x*, const ::xml_schema::namespace_infomap & *m*, const ::std::wstring & *e* = `L"UTF-8"`, ::xml_schema::flags *f* = `0`)

Serialize to a standard output stream.

Parameters:

os A standrad output stream.

x An object model to serialize.

m A namespace information map.

e A character encoding to produce XML in.

f Serialization flags.

This function uses exceptions to report serialization errors.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.9 _OPENGPS_EXPORT::std::auto_ptr<::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::DOMDocument * *d*, ::xml_schema::flags *f* = `0`, const ::xml_schema::properties & *p* = `::xml_schema::properties()`)

Parse a Xerces-C++ DOM document.

Parameters:

d A pointer to the Xerces-C++ DOM document.

f Parsing flags.

p Parsing properties.

Returns:

A pointer to the root of the object model.

This function is normally used together with the keep_dom and own_dom parsing flags to assign ownership of the DOM document to the object model.

References n.

11.3.2.10 _OPENGPS_EXPORT::std::auto_ptr<::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::xercesc::DOMDocument & *d*, ::xml_schema::flags *f* = 0, const ::xml_schema::properties & *p* = ::xml_schema::properties ())

Parse a Xerces-C++ DOM document.

Parameters:

d A Xerces-C++ DOM document.

f Parsing flags.

p Parsing properties.

Returns:

A pointer to the root of the object model.

References ISO5436_2(), and n.

Here is the call graph for this function:

11.3.2.11 _OPENGPS_EXPORT::std::auto_ptr<::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::xercesc::DOMInputSource & *is*, ::xercesc::DOMErrorHandler & *eh*, ::xml_schema::flags *f* = 0, const ::xml_schema::properties & *p* = ::xml_schema::properties ())

Parse a Xerces-C++ DOM input source with a Xerces-C++ DOM error handler.

Parameters:

is A Xerces-C++ DOM input source.

eh A Xerces-C++ DOM error handler.

f Parsing flags.

p Parsing properties.

Returns:

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.12 _OPENGPS_EXPORT::std::auto_ptr<::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::xercesc::DOMInputSource & *is*, ::xml_schema::error_handler & *eh*, ::xml_schema::flags *f* = 0, const ::xml_schema::properties & *p* = ::xml_schema::properties())

Parse a Xerces-C++ DOM input source with an error handler.

Parameters:

- ***is*** A Xerces-C++ DOM input source.
- ***eh*** An error handler.
- ***f*** Parsing flags.
- ***p*** Parsing properties.

Returns:

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.13 _OPENGPS_EXPORT::std::auto_ptr<::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::xercesc::DOMInputSource & *is*, ::xml_schema::flags *f* = 0, const ::xml_schema::properties & *p* = ::xml_schema::properties())

Parse a Xerces-C++ DOM input source.

Parameters:

- ***is*** A Xerces-C++ DOM input source.
- ***f*** Parsing flags.
- ***p*** Parsing properties.

Returns:

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.14 _OPENGPS_EXPORT::std::auto_ptr<::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream & *is*, const ::std::wstring & *id*, ::xercesc::DOMErrorHandler & *eh*, ::xml_schema::flags *f* = 0, const ::xml_schema::properties & *p* = ::xml_schema::properties ())

Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

Parameters:

- ***is*** A standrad input stream.
- ***id*** A resource id.
- ***eh*** A Xerces-C++ DOM error handler.
- ***f*** Parsing flags.
- ***p*** Parsing properties.

Returns:

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.15 _OPENGPS_EXPORT::std::auto_ptr<::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream & *is*, const ::std::wstring & *id*, ::xml_schema::error_handler & *eh*, ::xml_schema::flags *f* = 0, const ::xml_schema::properties & *p* = ::xml_schema::properties ())

Parse a standard input stream with a resource id and an error handler.

Parameters:

- ***is*** A standrad input stream.
- ***id*** A resource id.
- ***eh*** An error handler.
- ***f*** Parsing flags.
- ***p*** Parsing properties.

Returns:

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.16 _OPENGPS_EXPORT::std::auto_ptr<::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream & *is*, const ::std::wstring & *id*, ::xml_schema::flags *f* = 0, const ::xml_schema::properties & *p* = ::xml_schema::properties ())

Parse a standard input stream with a resource id.

Parameters:

is A standrad input stream.

id A resource id.

f Parsing flags.

p Parsing properties.

Returns:

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function uses exceptions to report parsing errors.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.17 _OPENGPS_EXPORT::std::auto_ptr<::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream & *is*, ::xercesc::DOMErrorHandler & *eh*, ::xml_schema::flags *f* = 0, const ::xml_schema::properties & *p* = ::xml_schema::properties ())

Parse a standard input stream with a Xerces-C++ DOM error handler.

Parameters:

is A standrad input stream.

eh A Xerces-C++ DOM error handler.

f Parsing flags.

p Parsing properties.

Returns:

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.18 _OPENGPS_EXPORT::std::auto_ptr<::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream & *is*, ::xml_schema::error_handler & *eh*, ::xml_schema::flags *f* = 0, const ::xml_schema::properties & *p* = ::xml_schema::properties ())

Parse a standard input stream with an error handler.

Parameters:

is A standrad input stream.

eh An error handler.

f Parsing flags.

p Parsing properties.

Returns:

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.19 _OPENGPS_EXPORT::std::auto_ptr<::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream & *is*, ::xml_schema::flags *f* = 0, const ::xml_schema::properties & *p* = ::xml_schema::properties ())

Parse a standard input stream.

Parameters:

is A standrad input stream.

f Parsing flags.

p Parsing properties.

Returns:

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.20 _OPENGPS_EXPORT::std::auto_ptr<::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::std::wstring & *uri*, ::xercesc::DOMErrorHandler & *eh*, ::xml_schema::flags *f* = 0, const ::xml_schema::properties & *p* = ::xml_schema::properties ())

Parse a URI or a local file with a Xerces-C++ DOM error handler.

Parameters:

uri A URI or a local file name.

eh A Xerces-C++ DOM error handler.

f Parsing flags.

p Parsing properties.

Returns:

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.21 _OPENGPS_EXPORT::std::auto_ptr<::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::std::wstring & *uri*, ::xml_schema::error_handler & *eh*, ::xml_schema::flags *f* = 0, const ::xml_schema::properties & *p* = ::xml_schema::properties ())

Parse a URI or a local file with an error handler.

Parameters:

uri A URI or a local file name.

eh An error handler.

f Parsing flags.

p Parsing properties.

Returns:

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

References ISO5436_2().

Here is the call graph for this function:

11.3.2.22 _OPENGPS_EXPORT::std::auto_ptr<::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::std::wstring & *uri*, ::xml_schema::flags *f* = 0, const ::xml_schema::properties & *p* = ::xml_schema::properties())

Parse a URI or a local file.

Parameters:

uri A URI or a local file name.

f Parsing flags.

p Parsing properties.

Returns:

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

Referenced by ISO5436_2().

11.3.2.23 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const RotationType &, const RotationType &)

11.3.2.24 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const MatrixDimensionType &, const MatrixDimensionType &)

11.3.2.25 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const DataLinkType &, const DataLinkType &)

11.3.2.26 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const DataListType &, const DataListType &)

11.3.2.27 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const ProbingSystemType &, const ProbingSystemType &)

11.3.2.28 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const InstrumentType &, const InstrumentType &)

11.3.2.29 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const AxisDescriptionType &, const AxisDescriptionType &)

11.3.2.30 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const AxesType &, const AxesType &)

11.3.2.31 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record4Type &, const Record4Type &)

11.3.2.32 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record3Type &, const Record3Type &)

11.3.2.33 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record2Type &, const Record2Type &)

11.3.2.34 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const ISO5436_2Type &, const ISO5436_2Type &)

11.3.2.35 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record1Type &, const Record1Type &)

11.3.2.36 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const Datum &)

11.3.2.37 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const Datum &)

11.3.2.38 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Datum &)

11.3.2.39 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const Type &)

11.3.2.40 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const Type &)

11.3.2.41 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Type &)

11.3.2.42 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const DataType &)

11.3.2.43 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const DataType &)

11.3.2.44 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const DataType &)

11.3.2.45 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const AxisType &)

11.3.2.46 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const AxisType &)

11.3.2.47 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const AxisType &)

11.3.2.48 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const FeatureType &)

11.3.2.49 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const FeatureType &)

11.3.2.50 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const FeatureType &)

11.3.2.51 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const RotationMatrixElementType &)

11.3.2.52 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const RotationMatrixElementType &)

11.3.2.53 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const RotationMatrixElementType &)

11.3.2.54 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const RotationType &)

References OpenGPS::Schemas::ISO5436_2::RotationType::r11(), OpenGPS::Schemas::ISO5436_2::RotationType::r12(), OpenGPS::Schemas::ISO5436_2::RotationType::r13(), OpenGPS::Schemas::ISO5436_2::RotationType::r21(), OpenGPS::Schemas::ISO5436_2::RotationType::r22(), OpenGPS::Schemas::ISO5436_2::RotationType::r23(), OpenGPS::Schemas::ISO5436_2::RotationType::r31(), OpenGPS::Schemas::ISO5436_2::RotationType::r32(), and OpenGPS::Schemas::ISO5436_2::RotationType::r33().

Here is the call graph for this function:

11.3.2.55 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const MatrixDimensionType &)

References OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX(), OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY(), and OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType::SizeZ().

Here is the call graph for this function:

11.3.2.56 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const DataLinkType &)

References OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData(), OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints(), OpenGPS::Schemas::ISO5436_2::DataLinkType::PointDataLink(), and OpenGPS::Schemas::ISO5436_-2::DataLinkType::ValidPointsLink().

Here is the call graph for this function:

11.3.2.57 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const DataListType &)

References OpenGPS::Schemas::ISO5436_2::DataListType::Datum(), and n.

Here is the call graph for this function:

11.3.2.58 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const ProbingSystemType &)

References OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification(), and OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Type().

Here is the call graph for this function:

11.3.2.59 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const InstrumentType &)

References OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer(), OpenGPS::Schemas::ISO5436_2::InstrumentType::Model(), OpenGPS::Schemas::ISO5436_2::InstrumentType::Serial(), and OpenGPS::Schemas::ISO5436_2::InstrumentType::Version().

Here is the call graph for this function:

11.3.2.60 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const AxisDescriptionType &)

References OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType(), OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType(), OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment(), and OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Offset().

Here is the call graph for this function:

11.3.2.61 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const AxesType &)

References OpenGPS::Schemas::ISO5436_2::AxesType::CX(), OpenGPS::Schemas::ISO5436_-2::AxesType::CY(), OpenGPS::Schemas::ISO5436_2::AxesType::CZ(), and OpenGPS::Schemas::ISO5436_2::AxesType::Rotation().

Here is the call graph for this function:

11.3.2.62 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Record4Type &)

References OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile().

Here is the call graph for this function:

11.3.2.63 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Record3Type &)

References OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink(), OpenGPS::Schemas::ISO5436_-2::Record3Type::DataList(), OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension(), and OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension().

Here is the call graph for this function:

11.3.2.64 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Record2Type &)

References OpenGPS::Schemas::ISO5436_2::Record2Type::CalibrationDate(), OpenGPS::Schemas::ISO5436_2::Record2Type::Comment(), OpenGPS::Schemas::ISO5436_-2::Record2Type::Creator(), OpenGPS::Schemas::ISO5436_2::Record2Type::Date(), OpenGPS::Schemas::ISO5436_2::Record2Type::Instrument(), and OpenGPS::Schemas::ISO5436_-2::Record2Type::ProbingSystem().

Here is the call graph for this function:

11.3.2.65 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const ISO5436_2Type &)

References OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1(), OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2(), OpenGPS::Schemas::ISO5436_-2::ISO5436_2Type::Record3(), OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record4(), and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID().

Here is the call graph for this function:

11.3.2.66 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Record1Type &)

References OpenGPS::Schemas::ISO5436_2::Record1Type::Axes(), OpenGPS::Schemas::ISO5436_-2::Record1Type::FeatureType(), and OpenGPS::Schemas::ISO5436_2::Record1Type::Revision().

Here is the call graph for this function:

11.3.2.67 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_-2::operator<< (::std::wostream &, const Datum &)

11.3.2.68 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_-2::operator<< (::std::wostream &, const Type &)

11.3.2.69 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_-2::operator<< (::std::wostream &, Type::value)

References OpenGPS::Schemas::ISO5436_2::Type::_xsd_Type_literals_.

11.3.2.70 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_-2::operator<< (::std::wostream &, const DataType &)

11.3.2.71 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_-2::operator<< (::std::wostream &, DataType::value)

References OpenGPS::Schemas::ISO5436_2::DataType::_xsd_DataType_literals_.

11.3.2.72 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_-2::operator<< (::std::wostream &, const AxisType &)

11.3.2.73 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_-2::operator<< (::std::wostream &, AxisType::value)

References OpenGPS::Schemas::ISO5436_2::AxisType::_xsd_AxisType_literals_.

11.3.2.74 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_-2::operator<< (::std::wostream &, const FeatureType &)

11.3.2.75 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_-2::operator<< (::std::wostream &, const RotationMatrixElementType &)

11.3.2.76 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const RotationType &)

References OpenGPS::Schemas::ISO5436_2::RotationType::r11(), OpenGPS::Schemas::ISO5436_2::RotationType::r12(), OpenGPS::Schemas::ISO5436_2::RotationType::r13(), OpenGPS::Schemas::ISO5436_2::RotationType::r21(), OpenGPS::Schemas::ISO5436_2::RotationType::r22(), OpenGPS::Schemas::ISO5436_2::RotationType::r23(), OpenGPS::Schemas::ISO5436_2::RotationType::r31(), OpenGPS::Schemas::ISO5436_2::RotationType::r32(), and OpenGPS::Schemas::ISO5436_2::RotationType::r33().

Here is the call graph for this function:

11.3.2.77 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const MatrixDimensionType &)

References OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX(), OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY(), and OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ().

Here is the call graph for this function:

11.3.2.78 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const DataLinkType &)

References OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData(), OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints(), OpenGPS::Schemas::ISO5436_2::DataLinkType::PointDataLink(), and OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink().

Here is the call graph for this function:

11.3.2.79 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const DataListType &)

References OpenGPS::Schemas::ISO5436_2::DataListType::Datum().

Here is the call graph for this function:

11.3.2.80 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const ProbingSystemType &)

References OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification(), and OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Type().

Here is the call graph for this function:

11.3.2.81 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const InstrumentType &)

References OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer(), OpenGPS::Schemas::ISO5436_2::InstrumentType::Model(), OpenGPS::Schemas::ISO5436_2::InstrumentType::Serial(), and OpenGPS::Schemas::ISO5436_2::InstrumentType::Version().

Here is the call graph for this function:

11.3.2.82 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const AxisDescriptionType &)

References OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType(), OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType(), OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment(), and OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Offset().

Here is the call graph for this function:

11.3.2.83 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const AxesType &)

References OpenGPS::Schemas::ISO5436_2::AxesType::CX(), OpenGPS::Schemas::ISO5436_2::AxesType::CY(), OpenGPS::Schemas::ISO5436_2::AxesType::CZ(), and OpenGPS::Schemas::ISO5436_2::AxesType::Rotation().

Here is the call graph for this function:

11.3.2.84 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Record4Type &)

References OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile().

Here is the call graph for this function:

11.3.2.85 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Record3Type &)

References OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink(), OpenGPS::Schemas::ISO5436_2::Record3Type::DataList(), OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension(), and OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension().

Here is the call graph for this function:

11.3.2.86 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Record2Type &)

References OpenGPS::Schemas::ISO5436_2::Record2Type::CalibrationDate(), OpenGPS::Schemas::ISO5436_2::Record2Type::Comment(), OpenGPS::Schemas::ISO5436_2::Record2Type::Creator(), OpenGPS::Schemas::ISO5436_2::Record2Type::Date(), OpenGPS::Schemas::ISO5436_2::Record2Type::Instrument(), and OpenGPS::Schemas::ISO5436_2::Record2Type::ProbingSystem().

Here is the call graph for this function:

11.3.2.87 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const ISO5436_2Type &)

References OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1(), OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2(), OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3(), OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record4(), and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID().

Here is the call graph for this function:

11.3.2.88 _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Record1Type &)

References OpenGPS::Schemas::ISO5436_2::Record1Type::Axes(), OpenGPS::Schemas::ISO5436_2::Record1Type::FeatureType(), and OpenGPS::Schemas::ISO5436_2::Record1Type::Revision().

Here is the call graph for this function:

11.3.2.89 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const RotationType &, const RotationType &)

References OpenGPS::Schemas::ISO5436_2::RotationType::r11(), OpenGPS::Schemas::ISO5436_2::RotationType::r12(), OpenGPS::Schemas::ISO5436_2::RotationType::r13(), OpenGPS::Schemas::ISO5436_2::RotationType::r21(), OpenGPS::Schemas::ISO5436_2::RotationType::r22(), OpenGPS::Schemas::ISO5436_2::RotationType::r23(), OpenGPS::Schemas::ISO5436_2::RotationType::r31(), OpenGPS::Schemas::ISO5436_2::RotationType::r32(), and OpenGPS::Schemas::ISO5436_2::RotationType::r33().

Here is the call graph for this function:

11.3.2.90 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const MatrixDimensionType &, const MatrixDimensionType &)

References OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX(), OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY(), and OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ().

Here is the call graph for this function:

11.3.2.91 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const DataLinkType &, const DataLinkType &)

References OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData(), OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints(), OpenGPS::Schemas::ISO5436_2::DataLinkType::PointDataLink(), and OpenGPS::Schemas::ISO5436_-2::DataLinkType::ValidPointsLink().

Here is the call graph for this function:

11.3.2.92 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const DataL-istType &, const DataListType &)

References OpenGPS::Schemas::ISO5436_2::DataListType::Datum().

Here is the call graph for this function:

11.3.2.93 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Prob-ingSystemType &, const ProbingSystemType &)

References OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification(), and OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Type().

Here is the call graph for this function:

11.3.2.94 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const InstrumentType &, const InstrumentType &)

References OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer(), OpenGPS::Schemas::ISO5436_2::InstrumentType::Model(), OpenGPS::Schemas::ISO5436_2::InstrumentType::Serial(), and OpenGPS::Schemas::ISO5436_2::InstrumentType::Version().

Here is the call graph for this function:

11.3.2.95 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const AxisDescriptionType &, const AxisDescriptionType &)

References OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType(), OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType(), OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment(), and OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Offset().

Here is the call graph for this function:

11.3.2.96 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const AxesType &, const AxesType &)

References OpenGPS::Schemas::ISO5436_2::AxesType::CX(), OpenGPS::Schemas::ISO5436_2::AxesType::CY(), OpenGPS::Schemas::ISO5436_2::AxesType::CZ(), and OpenGPS::Schemas::ISO5436_2::AxesType::Rotation().

Here is the call graph for this function:

11.3.2.97 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Record4Type &, const Record4Type &)

References OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile().

Here is the call graph for this function:

11.3.2.98 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Record3Type &, const Record3Type &)

References OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink(), OpenGPS::Schemas::ISO5436_-2::Record3Type::DataList(), OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension(), and OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension().

Here is the call graph for this function:

11.3.2.99 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Record2Type &, const Record2Type &)

References OpenGPS::Schemas::ISO5436_2::Record2Type::CalibrationDate(), OpenGPS::Schemas::ISO5436_2::Record2Type::Comment(), OpenGPS::Schemas::ISO5436_-2::Record2Type::Creator(), OpenGPS::Schemas::ISO5436_2::Record2Type::Date(), OpenGPS::Schemas::ISO5436_2::Record2Type::Instrument(), and OpenGPS::Schemas::ISO5436_-2::Record2Type::ProbingSystem().

Here is the call graph for this function:

11.3.2.100 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const ISO5436_2Type &, const ISO5436_2Type &)

References OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1(), OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2(), OpenGPS::Schemas::ISO5436_-2::ISO5436_2Type::Record3(), OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record4(), and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID().

Here is the call graph for this function:

11.3.2.101 _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Record1Type &, const Record1Type &)

References OpenGPS::Schemas::ISO5436_2::Record1Type::Axes(), OpenGPS::Schemas::ISO5436_-2::Record1Type::FeatureType(), and OpenGPS::Schemas::ISO5436_2::Record1Type::Revision().

Here is the call graph for this function:

11.4 std Namespace Reference

11.4.1 Detailed Description

The standard namespace for the C++ library.

STL namespace.

Classes

- class **allocator**

  STL class.

- class **auto_ptr**

  STL class.

- class **bad_alloc**

  STL class.

- class **bad_cast**

STL class.

- class **bad_exception**

  STL class.

- class **bad_typeid**

  STL class.

- class **basic_fstream**

  STL class.

- class **basic_ifstream**

  STL class.

- class **basic_ios**

  STL class.

- class **basic_iostream**

  STL class.

- class **basic_istream**

  STL class.

- class **basic_istringstream**

  STL class.

- class **basic_ofstream**

  STL class.

- class **basic_ostream**

  STL class.

- class **basic_ostringstream**

  STL class.

- class **basic_string**

  STL class.

- class **basic_stringstream**

  STL class.

- class **bitset**

  STL class.

- class **complex**

  STL class.

- class **deque**

  STL class.

- class **domain_error**

  STL class.

- class **exception**

  STL class.

- class **fstream**

  STL class.

- class **ifstream**

  STL class.

- class **invalid_argument**

  STL class.

- class **ios**

  STL class.

- class **ios_base**

  STL class.

- class **istream**

  STL class.

- class **istringstream**

  STL class.

- class **length_error**

  STL class.

- class **list**

  STL class.

- class **logic_error**

  STL class.

- class **map**

  STL class.

- class **multimap**

  STL class.

- class **multiset**

  STL class.

- class **ofstream**

  STL class.

- class **ostream**

  STL class.

- class **ostringstream**
  STL class.
- class **out_of_range**
  STL class.
- class **overflow_error**
  STL class.
- class **priority_queue**
  STL class.
- class **queue**
  STL class.
- class **range_error**
  STL class.
- class **runtime_error**
  STL class.
- class **set**
  STL class.
- class **stack**
  STL class.
- class **string**
  STL class.
- class **stringstream**
  STL class.
- class **underflow_error**
  STL class.
- class **valarray**
  STL class.
- class **vector**
  STL class.
- class **wfstream**
  STL class.
- class **wifstream**
  STL class.
- class **wios**

STL class.

- class **wistream**

  STL class.

- class **wistringstream**

  STL class.

- class **wofstream**

  STL class.

- class **wostream**

  STL class.

- class **wostringstream**

  STL class.

- class **wstring**

  STL class.

- class **wstringstream**

  STL class.

11.5 xml_schema Namespace Reference

11.5.1 Detailed Description

The namespace for basic XML schemas.

Typedefs

- typedef ::xsd::cxx::tree::base64_binary< wchar_t, simple_type > base64_binary
- typedef bool boolean
- typedef ::xsd::cxx::tree::bounds< wchar_t > bounds
- typedef ::xsd::cxx::tree::buffer< wchar_t > buffer
- typedef signed char byte
- typedef ::xsd::cxx::tree::date< wchar_t, simple_type > date
- typedef ::xsd::cxx::tree::date_time< wchar_t, simple_type > date_time
- typedef ::xsd::cxx::tree::day< wchar_t, simple_type > day
- typedef double decimal
- typedef ::xsd::cxx::tree::diagnostics< wchar_t > diagnostics
- typedef double double_
- typedef ::xsd::cxx::tree::duplicate_id< wchar_t > duplicate_id
- typedef ::xsd::cxx::tree::duration< wchar_t, simple_type > duration
- typedef ::xsd::cxx::tree::entities< wchar_t, simple_type, entity > entities
- typedef ::xsd::cxx::tree::entity< wchar_t, ncname > entity
- typedef ::xsd::cxx::tree::error< wchar_t > error
- typedef ::xsd::cxx::xml::error_handler< wchar_t > error_handler

- typedef ::xsd::cxx::tree::exception< wchar_t > exception
- typedef ::xsd::cxx::tree::expected_attribute< wchar_t > expected_attribute
- typedef ::xsd::cxx::tree::expected_element< wchar_t > expected_element
- typedef ::xsd::cxx::tree::expected_text_content< wchar_t > expected_text_content
- typedef ::xsd::cxx::tree::flags flags
- typedef float float_
- typedef ::xsd::cxx::tree::hex_binary< wchar_t, simple_type > hex_binary
- typedef ::xsd::cxx::tree::id< wchar_t, ncname > id
- typedef ::xsd::cxx::tree::idref< type, wchar_t, ncname > idref
- typedef ::xsd::cxx::tree::idrefs< wchar_t, simple_type, idref > idrefs
- typedef int int_
- typedef long long integer
- typedef ::xsd::cxx::tree::language< wchar_t, token > language
- typedef long long long_
- typedef ::xsd::cxx::tree::month< wchar_t, simple_type > month
- typedef ::xsd::cxx::tree::month_day< wchar_t, simple_type > month_day
- typedef ::xsd::cxx::tree::name< wchar_t, token > name
- typedef ::xsd::cxx::xml::dom::namespace_info< wchar_t > namespace_info
- typedef ::xsd::cxx::xml::dom::namespace_infomap< wchar_t > namespace_infomap
- typedef ::xsd::cxx::tree::ncname< wchar_t, name > ncname
- typedef integer negative_integer
- typedef ::xsd::cxx::tree::nmtoken< wchar_t, token > nmtoken
- typedef ::xsd::cxx::tree::nmtokens< wchar_t, simple_type, nmtoken > nmtokens
- typedef ::xsd::cxx::tree::no_namespace_mapping< wchar_t > no_namespace_mapping
- typedef ::xsd::cxx::tree::no_prefix_mapping< wchar_t > no_prefix_mapping
- typedef ::xsd::cxx::tree::no_type_info< wchar_t > no_type_info
- typedef integer non_negative_integer
- typedef integer non_positive_integer
- typedef ::xsd::cxx::tree::normalized_string< wchar_t, string > normalized_string
- typedef ::xsd::cxx::tree::not_derived< wchar_t > not_derived
- typedef ::xsd::cxx::tree::parsing< wchar_t > parsing
- typedef integer positive_integer
- typedef ::xsd::cxx::tree::properties< wchar_t > properties
- typedef ::xsd::cxx::tree::qname< wchar_t, simple_type, uri, ncname > qname
- typedef ::xsd::cxx::tree::serialization< wchar_t > serialization
- typedef ::xsd::cxx::tree::severity severity
- typedef short short_
- typedef ::xsd::cxx::tree::simple_type< type > simple_type
- typedef ::xsd::cxx::tree::string< wchar_t, simple_type > string
- typedef ::xsd::cxx::tree::time< wchar_t, simple_type > time
- typedef ::xsd::cxx::tree::token< wchar_t, normalized_string > token
- typedef ::xsd::cxx::tree::type type
- typedef ::xsd::cxx::tree::unexpected_element< wchar_t > unexpected_element
- typedef ::xsd::cxx::tree::unexpected_enumerator< wchar_t > unexpected_enumerator
- typedef unsigned char unsigned_byte
- typedef unsigned int unsigned_int
- typedef unsigned long long unsigned_long
- typedef unsigned short unsigned_short
- typedef ::xsd::cxx::tree::uri< wchar_t, simple_type > uri
- typedef ::xsd::cxx::tree::xsi_already_in_use< wchar_t > xsi_already_in_use
- typedef ::xsd::cxx::tree::year< wchar_t, simple_type > year
- typedef ::xsd::cxx::tree::year_month< wchar_t, simple_type > year_month

Variables

- const XMLCh *const tree_node_key = ::xsd::cxx::tree::user_data_keys::node

11.5.2 Typedef Documentation

11.5.2.1 typedef::xsd::cxx::tree::base64_binary< wchar_t, simple_type > xml_schema::base64_binary

11.5.2.2 typedef bool xml_schema::boolean

11.5.2.3 typedef::xsd::cxx::tree::bounds< wchar_t > xml_schema::bounds

11.5.2.4 typedef::xsd::cxx::tree::buffer< wchar_t > xml_schema::buffer

11.5.2.5 typedef signed char xml_schema::byte

11.5.2.6 typedef::xsd::cxx::tree::date< wchar_t, simple_type > xml_schema::date

11.5.2.7 typedef::xsd::cxx::tree::date_time< wchar_t, simple_type > xml_schema::date_time

11.5.2.8 typedef::xsd::cxx::tree::day< wchar_t, simple_type > xml_schema::day

11.5.2.9 typedef double xml_schema::decimal

11.5.2.10 typedef::xsd::cxx::tree::diagnostics< wchar_t > xml_schema::diagnostics

11.5.2.11 typedef double xml_schema::double_

11.5.2.12 typedef::xsd::cxx::tree::duplicate_id< wchar_t > xml_schema::duplicate_id

11.5.2.13 typedef::xsd::cxx::tree::duration< wchar_t, simple_type > xml_schema::duration

11.5.2.14 typedef::xsd::cxx::tree::entities< wchar_t, simple_type, entity > xml_schema::entities

11.5.2.15 typedef::xsd::cxx::tree::entity< wchar_t, ncname > xml_schema::entity

11.5.2.16 typedef::xsd::cxx::tree::error< wchar_t > xml_schema::error

11.5.2.17 typedef::xsd::cxx::xml::error_handler< wchar_t > xml_schema::error_handler

11.5.2.18 typedef::xsd::cxx::tree::exception< wchar_t > xml_schema::exception

11.5.2.19 typedef::xsd::cxx::tree::expected_attribute< wchar_t > xml_schema::expected_-attribute

11.5.2.20 typedef::xsd::cxx::tree::expected_element< wchar_t > xml_schema::expected_element

11.5.2.21 typedef::xsd::cxx::tree::expected_text_content< wchar_t > xml_schema::expected_-text_content

11.5.2.22 typedef::xsd::cxx::tree::flags xml_schema::flags

11.5.2.23 typedef float xml_schema::float_

11.5.2.24 typedef::xsd::cxx::tree::hex_binary< wchar_t, simple_type > xml_schema::hex_binary

11.5.2.25 typedef::xsd::cxx::tree::id< wchar_t, ncname > xml_schema::id

11.5.2.26 typedef::xsd::cxx::tree::idref< type, wchar_t, ncname > xml_schema::idref

11.5.2.27 typedef::xsd::cxx::tree::idrefs< wchar_t, simple_type, idref > xml_schema::idrefs

11.5.2.28 typedef int xml_schema::int_

11.5.2.29 typedef long long xml_schema::integer

11.5.2.30 typedef::xsd::cxx::tree::language< wchar_t, token > xml_schema::language

11.5.2.31 typedef long long xml_schema::long_

11.5.2.32 typedef::xsd::cxx::tree::month< wchar_t, simple_type > xml_schema::month

11.5.2.33 typedef::xsd::cxx::tree::month_day< wchar_t, simple_type > xml_schema::month_day

11.5.2.34 typedef::xsd::cxx::tree::name< wchar_t, token > xml_schema::name

11.5.2.35 typedef::xsd::cxx::xml::dom::namespace_info< wchar_t > xml_schema::namespace_-info

11.5.2.36 typedef::xsd::cxx::xml::dom::namespace_infomap< wchar_t > xml_-schema::namespace_infomap

11.5.2.37 typedef::xsd::cxx::tree::ncname< wchar_t, name > xml_schema::ncname

11.5.2.38 typedef integer xml_schema::negative_integer

11.5.2.39 typedef::xsd::cxx::tree::nmtoken< wchar_t, token > xml_schema::nmtoken

11.5.2.40 typedef::xsd::cxx::tree::nmtokens< wchar_t, simple_type, nmtoken > xml_schema::nmtokens

11.5.2.41 typedef::xsd::cxx::tree::no_namespace_mapping< wchar_t > xml_schema::no_namespace_mapping

11.5.2.42 typedef::xsd::cxx::tree::no_prefix_mapping< wchar_t > xml_schema::no_prefix_mapping

11.5.2.43 typedef::xsd::cxx::tree::no_type_info< wchar_t > xml_schema::no_type_info

11.5.2.44 typedef integer xml_schema::non_negative_integer

11.5.2.45 typedef integer xml_schema::non_positive_integer

11.5.2.46 typedef::xsd::cxx::tree::normalized_string< wchar_t, string > xml_schema::normalized_string

11.5.2.47 typedef::xsd::cxx::tree::not_derived< wchar_t > xml_schema::not_derived

11.5.2.48 typedef::xsd::cxx::tree::parsing< wchar_t > xml_schema::parsing

11.5.2.49 typedef integer xml_schema::positive_integer

11.5.2.50 typedef::xsd::cxx::tree::properties< wchar_t > xml_schema::properties

11.5.2.51 typedef::xsd::cxx::tree::qname< wchar_t, simple_type, uri, ncname > xml_schema::qname

11.5.2.52 typedef::xsd::cxx::tree::serialization< wchar_t > xml_schema::serialization

11.5.2.53 typedef::xsd::cxx::tree::severity xml_schema::severity

11.5.2.54 typedef short xml_schema::short_

11.5.2.55 typedef::xsd::cxx::tree::simple_type< type > xml_schema::simple_type

11.5.2.56 typedef::xsd::cxx::tree::string< wchar_t, simple_type > xml_schema::string

11.5.2.57 typedef::xsd::cxx::tree::time< wchar_t, simple_type > xml_schema::time

11.5.2.58 typedef::xsd::cxx::tree::token< wchar_t, normalized_string > xml_schema::token

11.5.2.59 typedef::xsd::cxx::tree::type xml_schema::type

11.5.2.60 typedef::xsd::cxx::tree::unexpected_element< wchar_t > xml_schema::unexpected_element

11.5.2.61 typedef::xsd::cxx::tree::unexpected_enumerator< wchar_t > xml_schema::unexpected_enumerator

11.5.2.62 typedef unsigned char xml_schema::unsigned_byte

11.5.2.63 typedef unsigned int xml_schema::unsigned_int

11.5.2.64 typedef unsigned long long xml_schema::unsigned_long

11.5.2.65 typedef unsigned short xml_schema::unsigned_short

11.5.2.66 typedef::xsd::cxx::tree::uri< wchar_t, simple_type > xml_schema::uri

11.5.2.67 typedef::xsd::cxx::tree::xsi_already_in_use< wchar_t > xml_schema::xsi_already_in_use

11.5.2.68 typedef::xsd::cxx::tree::year< wchar_t, simple_type > xml_schema::year

11.5.2.69 typedef::xsd::cxx::tree::year_month< wchar_t, simple_type > xml_schema::year_month

11.5.3 Variable Documentation

11.5.3.1 const XMLCh *const xml_schema::tree_node_key = ::xsd::cxx::tree::user_data_keys::node

12 Class Documentation

12.1 OpenGPS::Schemas::ISO5436_2::AxesType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.1.1 Detailed Description

Class corresponding to the AxesType schema type.

CX

Accessor and modifier functions for the CX required element.

Description of X-Axis

- typedef ::xsd::cxx::tree::traits< CX_type, wchar_t > CX_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::AxisDescriptionType CX_type
  Element type.
- void CX (::std::auto_ptr< CX_type > p)
  Set the element value without copying.
- void CX (const CX_type &x)
  Set the element value.
- CX_type & CX ()
  Return a read-write reference to the element.
- const CX_type & CX () const
  Return a read-only (constant) reference to the element.

CX

Accessor and modifier functions for the CX required element.

Description of X-Axis

- typedef ::xsd::cxx::tree::traits< CX_type, wchar_t > CX_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::AxisDescriptionType CX_type
  Element type.
- void CX (::std::auto_ptr< CX_type > p)
  Set the element value without copying.
- void CX (const CX_type &x)
  Set the element value.
- CX_type & CX ()
  Return a read-write reference to the element.
- const CX_type & CX () const
  Return a read-only (constant) reference to the element.

CY

Accessor and modifier functions for the CY required element.

Description of Y-Axis

- typedef ::xsd::cxx::tree::traits< CY_type, wchar_t > CY_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::AxisDescriptionType CY_type
  Element type.
- void CY (::std::auto_ptr< CY_type > p)
  Set the element value without copying.
- void CY (const CY_type &x)
  Set the element value.
- CY_type & CY ()
  Return a read-write reference to the element.
- const CY_type & CY () const
  Return a read-only (constant) reference to the element.

CY

Accessor and modifier functions for the CY required element.

Description of Y-Axis

- typedef ::xsd::cxx::tree::traits< CY_type, wchar_t > CY_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::AxisDescriptionType CY_type
  Element type.
- void CY (::std::auto_ptr< CY_type > p)
  Set the element value without copying.
- void CY (const CY_type &x)
  Set the element value.
- CY_type & CY ()
  Return a read-write reference to the element.
- const CY_type & CY () const
  Return a read-only (constant) reference to the element.

CZ

Accessor and modifier functions for the CZ required element.

Description of Z-Axis

- typedef ::xsd::cxx::tree::traits< CZ_type, wchar_t > CZ_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::AxisDescriptionType CZ_type

  Element type.

- void CZ (::std::auto_ptr< CZ_type > p)

  Set the element value without copying.

- void CZ (const CZ_type &x)

  Set the element value.

- CZ_type & CZ ()

  Return a read-write reference to the element.

- const CZ_type & CZ () const

  Return a read-only (constant) reference to the element.

CZ

Accessor and modifier functions for the CZ required element.

Description of Z-Axis

- typedef ::xsd::cxx::tree::traits< CZ_type, wchar_t > CZ_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::AxisDescriptionType CZ_type

  Element type.

- void CZ (::std::auto_ptr< CZ_type > p)

  Set the element value without copying.

- void CZ (const CZ_type &x)

  Set the element value.

- CZ_type & CZ ()

  Return a read-write reference to the element.

- const CZ_type & CZ () const

  Return a read-only (constant) reference to the element.

Rotation

Accessor and modifier functions for the Rotation optional element.

An optional rotation of the data points. If this element is missing a unit transformation is assumed.

- typedef ::xsd::cxx::tree::optional< Rotation_type > Rotation_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< Rotation_type, wchar_t > Rotation_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::RotationType Rotation_type
  Element type.
- void Rotation (::std::auto_ptr< Rotation_type > p)
  Set the element value without copying.
- void Rotation (const Rotation_optional &x)
  Set the element value.
- void Rotation (const Rotation_type &x)
  Set the element value.
- Rotation_optional & Rotation ()
  Return a read-write reference to the element container.
- const Rotation_optional & Rotation () const
  Return a read-only (constant) reference to the element container.

Rotation

Accessor and modifier functions for the Rotation optional element.

An optional rotation of the data points. If this element is missing a unit transformation is assumed.

- typedef ::xsd::cxx::tree::optional< Rotation_type > Rotation_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< Rotation_type, wchar_t > Rotation_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::RotationType Rotation_type
  Element type.
- void Rotation (::std::auto_ptr< Rotation_type > p)
  Set the element value without copying.
- void Rotation (const Rotation_optional &x)
  Set the element value.

- void Rotation (const Rotation_type &x)

  Set the element value.

- Rotation_optional & Rotation ()

  Return a read-write reference to the element container.

- const Rotation_optional & Rotation () const

  Return a read-only (constant) reference to the element container.

Constructors

- virtual AxesType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- AxesType (const AxesType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- AxesType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- AxesType (const CX_type &, const CY_type &, const CZ_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Constructors

- virtual AxesType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- AxesType (const AxesType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- AxesType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- AxesType (const CX_type &, const CY_type &, const CZ_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Protected Member Functions

- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)
- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)

Private Attributes

- ::xsd::cxx::tree::one< CX_type > CX_
- ::xsd::cxx::tree::one< CY_type > CY_
- ::xsd::cxx::tree::one< CZ_type > CZ_
- Rotation_optional Rotation_

12.1.2 Member Typedef Documentation

12.1.2.1 typedef ::xsd::cxx::tree::traits< CX_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxesType::CX_traits

Element traits type.

12.1.2.2 typedef ::xsd::cxx::tree::traits< CX_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxesType::CX_traits

Element traits type.

12.1.2.3 typedef ::OpenGPS::Schemas::ISO5436_2::AxisDescriptionType OpenGPS::Schemas::ISO5436_2::AxesType::CX_type

Element type.

12.1.2.4 typedef ::OpenGPS::Schemas::ISO5436_2::AxisDescriptionType OpenGPS::Schemas::ISO5436_2::AxesType::CX_type

Element type.

12.1.2.5 typedef ::xsd::cxx::tree::traits< CY_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxesType::CY_traits

Element traits type.

12.1.2.6 typedef ::xsd::cxx::tree::traits< CY_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxesType::CY_traits

Element traits type.

12.1.2.7 typedef ::OpenGPS::Schemas::ISO5436_2::AxisDescriptionType OpenGPS::Schemas::ISO5436_2::AxesType::CY_type

Element type.

12.1.2.8 typedef ::OpenGPS::Schemas::ISO5436_2::AxisDescriptionType OpenGPS::Schemas::ISO5436_2::AxesType::CY_type

Element type.

12.1.2.9 typedef ::xsd::cxx::tree::traits< CZ_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxesType::CZ_traits

Element traits type.

12.1.2.10 typedef ::xsd::cxx::tree::traits< CZ_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxesType::CZ_traits

Element traits type.

12.1.2.11 typedef ::OpenGPS::Schemas::ISO5436_2::AxisDescriptionType OpenGPS::Schemas::ISO5436_2::AxesType::CZ_type

Element type.

12.1.2.12 typedef ::OpenGPS::Schemas::ISO5436_2::AxisDescriptionType OpenGPS::Schemas::ISO5436_2::AxesType::CZ_type

Element type.

12.1.2.13 typedef ::xsd::cxx::tree::optional< Rotation_type > OpenGPS::Schemas::ISO5436_2::AxesType::Rotation_optional

Element optional container type.

12.1.2.14 typedef ::xsd::cxx::tree::optional< Rotation_type > OpenGPS::Schemas::ISO5436_2::AxesType::Rotation_optional

Element optional container type.

12.1.2.15 typedef ::xsd::cxx::tree::traits< Rotation_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxesType::Rotation_traits

Element traits type.

12.1.2.16 typedef ::xsd::cxx::tree::traits< Rotation_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxesType::Rotation_traits

Element traits type.

12.1.2.17 typedef ::OpenGPS::Schemas::ISO5436_2::RotationType OpenGPS::Schemas::ISO5436_2::AxesType::Rotation_type

Element type.

12.1.2.18 typedef ::OpenGPS::Schemas::ISO5436_2::RotationType OpenGPS::Schemas::ISO5436_2::AxesType::Rotation_type

Element type.

12.1.3 Constructor & Destructor Documentation

12.1.3.1 OpenGPS::Schemas::ISO5436_2::AxesType::AxesType (const CX_type & *CX*, const CY_type & *CY*, const CZ_type & *CZ*)

Construct an instance from the ultimate base and initializers for required elements and attributes.

Referenced by _clone().

12.1.3.2 OpenGPS::Schemas::ISO5436_2::AxesType::AxesType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

References parse().

Here is the call graph for this function:

12.1.3.3 OpenGPS::Schemas::ISO5436_2::AxesType::AxesType (const AxesType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.1.3.4 OpenGPS::Schemas::ISO5436_2::AxesType::AxesType (const CX_type &, const CY_type &, const CZ_type &)

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.1.3.5 OpenGPS::Schemas::ISO5436_2::AxesType::AxesType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.1.3.6 OpenGPS::Schemas::ISO5436_2::AxesType::AxesType (const AxesType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.1.4 Member Function Documentation

12.1.4.1 virtual AxesType* OpenGPS::Schemas::ISO5436_2::AxesType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.1.4.2 AxesType * OpenGPS::Schemas::ISO5436_2::AxesType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References AxesType().

Here is the call graph for this function:

12.1.4.3 void OpenGPS::Schemas::ISO5436_2::AxesType::CX (::std::auto_ptr< CX_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.1.4.4 void OpenGPS::Schemas::ISO5436_2::AxesType::CX (const CX_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.1.4.5 CX_type& OpenGPS::Schemas::ISO5436_2::AxesType::CX ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.1.4.6 const CX_type& OpenGPS::Schemas::ISO5436_2::AxesType::CX () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.1.4.7 void OpenGPS::Schemas::ISO5436_2::AxesType::CX (::std::auto_ptr< CX_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.1.4.8 void OpenGPS::Schemas::ISO5436_2::AxesType::CX (const CX_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.1.4.9 CX_type& OpenGPS::Schemas::ISO5436_2::AxesType::CX ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.1.4.10 const AxesType::CX_type & OpenGPS::Schemas::ISO5436_2::AxesType::CX () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

References CX_.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.1.4.11 void OpenGPS::Schemas::ISO5436_2::AxesType::CY (::std::auto_ptr< CY_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.1.4.12 void OpenGPS::Schemas::ISO5436_2::AxesType::CY (const CY_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.1.4.13 CY_type& OpenGPS::Schemas::ISO5436_2::AxesType::CY ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.1.4.14 const CY_type& OpenGPS::Schemas::ISO5436_2::AxesType::CY () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.1.4.15 void OpenGPS::Schemas::ISO5436_2::AxesType::CY (::std::auto_ptr< CY_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.1.4.16 void OpenGPS::Schemas::ISO5436_2::AxesType::CY (const CY_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.1.4.17 CY_type& OpenGPS::Schemas::ISO5436_2::AxesType::CY ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.1.4.18 const CY_type& OpenGPS::Schemas::ISO5436_2::AxesType::CY () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_-2::operator==(), and parse().

12.1.4.19 void OpenGPS::Schemas::ISO5436_2::AxesType::CZ (::std::auto_ptr< CZ_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.1.4.20 void OpenGPS::Schemas::ISO5436_2::AxesType::CZ (const CZ_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.1.4.21 CZ_type& OpenGPS::Schemas::ISO5436_2::AxesType::CZ ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.1.4.22 const CZ_type& OpenGPS::Schemas::ISO5436_2::AxesType::CZ () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.1.4.23 void OpenGPS::Schemas::ISO5436_2::AxesType::CZ (::std::auto_ptr< CZ_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.1.4.24 void OpenGPS::Schemas::ISO5436_2::AxesType::CZ (const CZ_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.1.4.25 CZ_type& OpenGPS::Schemas::ISO5436_2::AxesType::CZ ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.1.4.26 const CZ_type& OpenGPS::Schemas::ISO5436_2::AxesType::CZ () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.1.4.27 void OpenGPS::Schemas::ISO5436_2::AxesType::parse (::xsd::cxx::xml::dom::parser< wchar_t > &, ::xml_schema::flags) [protected]

12.1.4.28 void OpenGPS::Schemas::ISO5436_2::AxesType::parse (::xsd::cxx::xml::dom::parser< wchar_t > & *p*, ::xml_schema::flags *f*) [protected]

References CX(), CX_, CY(), CY_, CZ(), CZ_, n, and Rotation().

Referenced by AxesType().

Here is the call graph for this function:

12.1.4.29 void OpenGPS::Schemas::ISO5436_2::AxesType::Rotation (::std::auto_ptr< Rotation_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.1.4.30 void OpenGPS::Schemas::ISO5436_2::AxesType::Rotation (const Rotation_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.1.4.31 void OpenGPS::Schemas::ISO5436_2::AxesType::Rotation (const Rotation_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.1.4.32 Rotation_optional& OpenGPS::Schemas::ISO5436_2::AxesType::Rotation ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.1.4.33 const Rotation_optional& OpenGPS::Schemas::ISO5436_2::AxesType::Rotation () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

12.1.4.34 void OpenGPS::Schemas::ISO5436_2::AxesType::Rotation (::std::auto_ptr< Rotation_-type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.1.4.35 void OpenGPS::Schemas::ISO5436_2::AxesType::Rotation (const Rotation_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.1.4.36 void OpenGPS::Schemas::ISO5436_2::AxesType::Rotation (const Rotation_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.1.4.37 Rotation_optional& OpenGPS::Schemas::ISO5436_2::AxesType::Rotation ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.1.4.38 const Rotation_optional& OpenGPS::Schemas::ISO5436_2::AxesType::Rotation () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.1.5 Member Data Documentation

12.1.5.1 xsd::cxx::tree::one< CX_type > OpenGPS::Schemas::ISO5436_2::AxesType::CX_ [private]

Referenced by CX(), and parse().

12.1.5.2 xsd::cxx::tree::one< CY_type > OpenGPS::Schemas::ISO5436_2::AxesType::CY_ [private]

Referenced by parse().

12.1.5.3 xsd::cxx::tree::one< CZ_type > OpenGPS::Schemas::ISO5436_2::AxesType::CZ_ [private]

Referenced by parse().

12.1.5.4 Rotation_optional OpenGPS::Schemas::ISO5436_2::AxesType::Rotation_ [private]

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.2 OpenGPS::Schemas::ISO5436_2::AxisDescriptionType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.2.1 Detailed Description

Class corresponding to the AxisDescriptionType schema type.

AxisType

Accessor and modifier functions for the AxisType required element.

Type of axis can be "I" for Incremental, "A" for Absolute.The z-axis must be absolute!

- typedef ::xsd::cxx::tree::traits< AxisType_type, wchar_t > AxisType_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::AxisType AxisType_type

  Element type.

- void AxisType (::std::auto_ptr< AxisType_type > p)

  Set the element value without copying.

- void AxisType (const AxisType_type &x)

  Set the element value.

- AxisType_type & AxisType ()

  Return a read-write reference to the element.

- const AxisType_type & AxisType () const

  Return a read-only (constant) reference to the element.

AxisType

Accessor and modifier functions for the AxisType required element.

Type of axis can be "I" for Incremental, "A" for Absolute.The z-axis must be absolute!

- typedef ::xsd::cxx::tree::traits< AxisType_type, wchar_t > AxisType_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::AxisType AxisType_type

  Element type.

- void AxisType (::std::auto_ptr< AxisType_type > p)

  Set the element value without copying.

- void AxisType (const AxisType_type &x)

  Set the element value.

- AxisType_type & AxisType ()

  Return a read-write reference to the element.

- const AxisType_type & AxisType () const

  Return a read-only (constant) reference to the element.

DataType

Accessor and modifier functions for the DataType optional element.

Data type for absolute axis: "I" for int16, "L" for int32, "F" for float32, "D" for float64. Incremental axes do not have/need a data type

- typedef ::xsd::cxx::tree::optional< DataType_type > DataType_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< DataType_type, wchar_t > DataType_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::DataType DataType_type

  Element type.

- void DataType (::std::auto_ptr< DataType_type > p)

  Set the element value without copying.

- void DataType (const DataType_optional &x)

  Set the element value.

- void DataType (const DataType_type &x)

  Set the element value.

- DataType_optional & DataType ()

  Return a read-write reference to the element container.

- const DataType_optional & DataType () const

  Return a read-only (constant) reference to the element container.

DataType

Accessor and modifier functions for the DataType optional element.

Data type for absolute axis: "I" for int16, "L" for int32, "F" for float32, "D" for float64. Incremental axes do not have/need a data type

- typedef ::xsd::cxx::tree::optional< DataType_type > DataType_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< DataType_type, wchar_t > DataType_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::DataType DataType_type

  Element type.

- void DataType (::std::auto_ptr< DataType_type > p)

  Set the element value without copying.

- void DataType (const DataType_optional &x)

  Set the element value.

- void DataType (const DataType_type &x)

  Set the element value.

- DataType_optional & DataType ()

  Return a read-write reference to the element container.

- const DataType_optional & DataType () const

  Return a read-only (constant) reference to the element container.

Increment

Accessor and modifier functions for the Increment optional element.

Needed for incremental axis and integer data types: Increment is the multiplyer of the integer coordinate for the computation of the real coordinate: Xreal = Xoffset + Xinteger∗XIncrement. The unit of increment and offset is metre.

- typedef ::xsd::cxx::tree::optional< Increment_type > Increment_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< Increment_type, wchar_t > Increment_traits

  Element traits type.

- typedef ::xml_schema::double_ Increment_type

  Element type.

- void Increment (const Increment_optional &x)

  Set the element value.

- void Increment (const Increment_type &x)

  Set the element value.

- Increment_optional & Increment ()

  Return a read-write reference to the element container.

- const Increment_optional & Increment () const

  Return a read-only (constant) reference to the element container.

Increment

Accessor and modifier functions for the Increment optional element.

Needed for incremental axis and integer data types: Increment is the multiplyer of the integer coordinate for the computation of the real coordinate: Xreal = Xoffset + Xinteger∗XIncrement. The unit of increment and offset is metre.

- typedef ::xsd::cxx::tree::optional< Increment_type > Increment_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< Increment_type, wchar_t > Increment_traits

  Element traits type.

- typedef ::xml_schema::double_ Increment_type
  Element type.
- void Increment (const Increment_optional &x)
  Set the element value.
- void Increment (const Increment_type &x)
  Set the element value.
- Increment_optional & Increment ()
  Return a read-write reference to the element container.
- const Increment_optional & Increment () const
  Return a read-only (constant) reference to the element container.

Offset

Accessor and modifier functions for the Offset optional element.

The offset of axis in meter.

- typedef ::xsd::cxx::tree::optional< Offset_type > Offset_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< Offset_type, wchar_t > Offset_traits
  Element traits type.
- typedef ::xml_schema::double_ Offset_type
  Element type.
- void Offset (const Offset_optional &x)
  Set the element value.
- void Offset (const Offset_type &x)
  Set the element value.
- Offset_optional & Offset ()
  Return a read-write reference to the element container.
- const Offset_optional & Offset () const
  Return a read-only (constant) reference to the element container.

Offset

Accessor and modifier functions for the Offset optional element.

The offset of axis in meter.

- typedef ::xsd::cxx::tree::optional< Offset_type > Offset_optional

Element optional container type.

- typedef ::xsd::cxx::tree::traits< Offset_type, wchar_t > Offset_traits

  Element traits type.

- typedef ::xml_schema::double_ Offset_type

  Element type.

- void Offset (const Offset_optional &x)

  Set the element value.

- void Offset (const Offset_type &x)

  Set the element value.

- Offset_optional & Offset ()

  Return a read-write reference to the element container.

- const Offset_optional & Offset () const

  Return a read-only (constant) reference to the element container.

Constructors

- virtual AxisDescriptionType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- AxisDescriptionType (const AxisDescriptionType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- AxisDescriptionType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- AxisDescriptionType (const AxisType_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Constructors

- virtual AxisDescriptionType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- AxisDescriptionType (const AxisDescriptionType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- AxisDescriptionType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- AxisDescriptionType (const AxisType_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Protected Member Functions

- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)
- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)

Private Attributes

- ::xsd::cxx::tree::one< AxisType_type > AxisType_
- DataType_optional DataType_
- Increment_optional Increment_
- Offset_optional Offset_

12.2.2 Member Typedef Documentation

12.2.2.1 typedef ::xsd::cxx::tree::traits< AxisType_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType_traits

Element traits type.

12.2.2.2 typedef ::xsd::cxx::tree::traits< AxisType_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType_traits

Element traits type.

12.2.2.3 typedef ::OpenGPS::Schemas::ISO5436_2::AxisType OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType_type

Element type.

12.2.2.4 typedef ::OpenGPS::Schemas::ISO5436_2::AxisType OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType_type

Element type.

12.2.2.5 typedef ::xsd::cxx::tree::optional< DataType_type > OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType_optional

Element optional container type.

12.2.2.6 typedef ::xsd::cxx::tree::optional< DataType_type > OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType_optional

Element optional container type.

12.2.2.7 typedef ::xsd::cxx::tree::traits< DataType_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType_traits

Element traits type.

12.2.2.8 typedef ::xsd::cxx::tree::traits< DataType_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType_traits

Element traits type.

12.2.2.9 typedef ::OpenGPS::Schemas::ISO5436_2::DataType OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType::DataType_type

Element type.

12.2.2.10 typedef ::OpenGPS::Schemas::ISO5436_2::DataType OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType::DataType_type

Element type.

12.2.2.11 typedef ::xsd::cxx::tree::optional< Increment_type > OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType::Increment_optional

Element optional container type.

12.2.2.12 typedef ::xsd::cxx::tree::optional< Increment_type > OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType::Increment_optional

Element optional container type.

12.2.2.13 typedef ::xsd::cxx::tree::traits< Increment_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment_traits

Element traits type.

12.2.2.14 typedef ::xsd::cxx::tree::traits< Increment_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment_traits

Element traits type.

12.2.2.15 typedef ::xml_schema::double_ OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType::Increment_type

Element type.

12.2.2.16 typedef ::xml_schema::double_ OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType::Increment_type

Element type.

12.2.2.17 typedef ::xsd::cxx::tree::optional< Offset_type > OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType::Offset_optional

Element optional container type.

12.2.2.18 typedef ::xsd::cxx::tree::optional< Offset_type > OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType::Offset_optional

Element optional container type.

12.2.2.19 typedef ::xsd::cxx::tree::traits< Offset_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Offset_traits

Element traits type.

12.2.2.20 typedef ::xsd::cxx::tree::traits< Offset_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Offset_traits

Element traits type.

12.2.2.21 typedef ::xml_schema::double_ OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType::Offset_type

Element type.

12.2.2.22 typedef ::xml_schema::double_ OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType::Offset_type

Element type.

12.2.3 Constructor & Destructor Documentation

12.2.3.1 OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisDescriptionType (const AxisType_type & *AxisType*)

Construct an instance from the ultimate base and initializers for required elements and attributes.

Referenced by _clone().

12.2.3.2 OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisDescriptionType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

References parse().

Here is the call graph for this function:

12.2.3.3 OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisDescriptionType (const AxisDescriptionType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.2.3.4 OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisDescriptionType (const AxisType_type &)

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.2.3.5 OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisDescriptionType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.2.3.6 OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisDescriptionType (const AxisDescriptionType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.2.4 Member Function Documentation

12.2.4.1 virtual AxisDescriptionType* OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.2.4.2 AxisDescriptionType * OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References AxisDescriptionType().

Here is the call graph for this function:

12.2.4.3 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType (::std::auto_ptr< AxisType_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.2.4.4 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType (const AxisType_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.2.4.5 AxisType_type& OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.2.4.6 const AxisType_type& OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.2.4.7 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType (::std::auto_ptr< AxisType_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.2.4.8 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType (const AxisType_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.2.4.9 AxisType_type& OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.2.4.10 const AxisType_type& OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::ISO5436_2Container::GetAxisDataType(), OpenGPS::ISO5436_2Container::GetIncrementX(), OpenGPS::ISO5436_2Container::GetIncrementY(), OpenGPS::ISO5436_2Container::IsIncrementalX(), OpenGPS::ISO5436_2Container::IsIncrementalY(), OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.2.4.11 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType (::std::auto_ptr< DataType_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.2.4.12 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType (const DataType_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.2.4.13 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType (const DataType_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.2.4.14 DataType_optional& OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.2.4.15 const DataType_optional& OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

12.2.4.16 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType (::std::auto_ptr< DataType_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.2.4.17 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType (const DataType_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.2.4.18 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType (const DataType_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.2.4.19 DataType_optional& OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.2.4.20 const DataType_optional& OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

Referenced by OpenGPS::ISO5436_2Container::GetAxisDataType(), mediumComplexExample(), OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), parse(), performanceDouble(), performanceInt16(), and simpleExample().

12.2.4.21 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment (const Increment_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.2.4.22 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment (const Increment_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.2.4.23 Increment_optional& OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.2.4.24 const Increment_optional& OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

12.2.4.25 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment (const Increment_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.2.4.26 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment (const Increment_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.2.4.27 Increment_optional& OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.2.4.28 const Increment_optional& OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

Referenced by OpenGPS::ISO5436_2Container::GetIncrementX(), OpenGPS::ISO5436_2Container::GetIncrementY(), OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), parse(), and simpleExample().

12.2.4.29 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Offset (const Offset_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.2.4.30 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Offset (const Offset_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.2.4.31 Offset_optional& OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Offset ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.2.4.32 const Offset_optional& OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Offset () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

12.2.4.33 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Offset (const Offset_-optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.2.4.34 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Offset (const Offset_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.2.4.35 Offset_optional& OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Offset ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.2.4.36 const Offset_optional& OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Offset () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

Referenced by OpenGPS::ISO5436_2Container::GetOffsetX(), OpenGPS::ISO5436_2Container::GetOffsetY(), OpenGPS::ISO5436_2Container::GetOffsetZ(), OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), parse(), and simpleExample().

12.2.4.37 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::parse (::xsd::cxx::xml::dom::parser< wchar_t > &, ::xml_schema::flags) [protected]

12.2.4.38 void OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::parse (::xsd::cxx::xml::dom::parser< wchar_t > & *p*, ::xml_schema::flags *f*) [protected]

References AxisType(), AxisType_, DataType(), Increment(), n, and Offset().

Referenced by AxisDescriptionType().

Here is the call graph for this function:

12.2.5 Member Data Documentation

12.2.5.1 xsd::cxx::tree::one< AxisType_type > OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType_ [private]

Referenced by parse().

12.2.5.2 DataType_optional OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType_ [private]

12.2.5.3 Increment_optional OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment_- [private]

12.2.5.4 Offset_optional OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Offset_- [private]

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.3 OpenGPS::Schemas::ISO5436_2::AxisType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.3.1 Detailed Description

Enumeration class corresponding to the AxisType schema type.

Public Types

- enum value { A, I, A, I }

  Underlying enum type.

- enum value { A, I, A, I }

  Underlying enum type.

Public Member Functions

- virtual AxisType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- virtual AxisType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- AxisType (const AxisType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- AxisType (const ::std::wstring &s, const ::xercesc::DOMElement *e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a string fragment.

- AxisType (const ::xercesc::DOMAttr &a,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM attribute.

- AxisType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- AxisType (const ::xml_schema::token &v)

  Construct an instance from the base value.

- AxisType (value v)

  Construct an instance from the underlying enum value.

- AxisType (const AxisType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- AxisType (const ::std::wstring &s, const ::xercesc::DOMElement *e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a string fragment.

- AxisType (const ::xercesc::DOMAttr &a,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM attribute.

- AxisType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- AxisType (const ::xml_schema::token &v)

  Construct an instance from the base value.

- AxisType (value v)

  Construct an instance from the underlying enum value.

- virtual operator value () const

  Implicit conversion operator to the underlying enum value.

- virtual operator value () const

  Implicit conversion operator to the underlying enum value.

- AxisType & operator= (value v)

  Assign the underlying enum value.

- AxisType & operator= (value v)

  Assign the underlying enum value.

Static Public Attributes

- static const value _xsd_AxisType_indexes_ [2]
- static const wchar_t *const _xsd_AxisType_literals_ [2]

Protected Member Functions

- value _xsd_AxisType_convert () const
- value _xsd_AxisType_convert () const

12.3.2 Member Enumeration Documentation

12.3.2.1 enum OpenGPS::Schemas::ISO5436_2::AxisType::value

Underlying enum type.

Enumerator:

A

I

A

I

12.3.2.2 enum OpenGPS::Schemas::ISO5436_2::AxisType::value

Underlying enum type.

Enumerator:

A

I

A

I

12.3.3 Constructor & Destructor Documentation

12.3.3.1 OpenGPS::Schemas::ISO5436_2::AxisType::AxisType (value *v*)

Construct an instance from the underlying enum value.

Parameters:

v A enum value.

Referenced by _clone().

12.3.3.2 OpenGPS::Schemas::ISO5436_2::AxisType::AxisType (const ::xml_schema::token & *v*)

Construct an instance from the base value.

Parameters:

v A base value.

12.3.3.3 OpenGPS::Schemas::ISO5436_2::AxisType::AxisType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

e A DOM element to extract the data from.

f Flags to construct the new instance with.

c A pointer to the object that will contain the new instance.

References _xsd_AxisType_convert().

Here is the call graph for this function:

12.3.3.4 OpenGPS::Schemas::ISO5436_2::AxisType::AxisType (const ::xercesc::DOMAttr & *a*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM attribute.

Parameters:

a A DOM attribute to extract the data from.

f Flags to construct the new instance with.

c A pointer to the object that will contain the new instance.

References _xsd_AxisType_convert().

Here is the call graph for this function:

12.3.3.5 OpenGPS::Schemas::ISO5436_2::AxisType::AxisType (const ::std::wstring & *s*, const ::xercesc::DOMElement * *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a string fragment.

Parameters:

s A string fragment to extract the data from.

e A DOM element containing the string fragment.

f Flags to construct the new instance with.

c A pointer to the object that will contain the new instance.

References _xsd_AxisType_convert().

Here is the call graph for this function:

12.3.3.6 OpenGPS::Schemas::ISO5436_2::AxisType::AxisType (const AxisType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.3.3.7 OpenGPS::Schemas::ISO5436_2::AxisType::AxisType (value *v*)

Construct an instance from the underlying enum value.

Parameters:

- ***v*** A enum value.

12.3.3.8 OpenGPS::Schemas::ISO5436_2::AxisType::AxisType (const ::xml_schema::token & *v*)

Construct an instance from the base value.

Parameters:

- ***v*** A base value.

12.3.3.9 OpenGPS::Schemas::ISO5436_2::AxisType::AxisType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.3.3.10 OpenGPS::Schemas::ISO5436_2::AxisType::AxisType (const ::xercesc::DOMAttr & *a*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM attribute.

Parameters:

- ***a*** A DOM attribute to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.3.3.11 OpenGPS::Schemas::ISO5436_2::AxisType::AxisType (const ::std::wstring & *s*, const ::xercesc::DOMElement * *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a string fragment.

Parameters:

- ***s*** A string fragment to extract the data from.
- ***e*** A DOM element containing the string fragment.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.3.3.12 OpenGPS::Schemas::ISO5436_2::AxisType::AxisType (const AxisType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.3.4 Member Function Documentation

12.3.4.1 virtual AxisType* OpenGPS::Schemas::ISO5436_2::AxisType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.3.4.2 AxisType * OpenGPS::Schemas::ISO5436_2::AxisType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References AxisType().

Here is the call graph for this function:

12.3.4.3 value OpenGPS::Schemas::ISO5436_2::AxisType::_xsd_AxisType_convert () const [protected]

12.3.4.4 AxisType::value OpenGPS::Schemas::ISO5436_2::AxisType::_xsd_AxisType_convert () const [protected]

References _xsd_AxisType_indexes_, and _xsd_AxisType_literals_.

Referenced by AxisType().

12.3.4.5 virtual OpenGPS::Schemas::ISO5436_2::AxisType::operator value () const [inline, virtual]

Implicit conversion operator to the underlying enum value.

Returns:

A enum value.

12.3.4.6 virtual OpenGPS::Schemas::ISO5436_2::AxisType::operator value () const [inline, virtual]

Implicit conversion operator to the underlying enum value.

Returns:

A enum value.

12.3.4.7 AxisType& OpenGPS::Schemas::ISO5436_2::AxisType::operator= (value *v*)

Assign the underlying enum value.

Parameters:

v A enum value.

Returns:

A refernce to the instance.

12.3.4.8 AxisType& OpenGPS::Schemas::ISO5436_2::AxisType::operator= (value *v*)

Assign the underlying enum value.

Parameters:

v A enum value.

Returns:

A refernce to the instance.

12.3.5 Member Data Documentation

12.3.5.1 static const value OpenGPS::Schemas::ISO5436_2::AxisType::_xsd_AxisType_indexes_- [static]

Referenced by _xsd_AxisType_convert().

12.3.5.2 static const wchar_t *const OpenGPS::Schemas::ISO5436_2::AxisType::_xsd_AxisType_-literals_ [static]

Referenced by _xsd_AxisType_convert(), and OpenGPS::Schemas::ISO5436_2::operator<<().

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.4 OpenGPS::BinaryLSBPointVectorReaderContext Class Reference

```
#include <binary_lsb_point_vector_reader_context.hxx>
```

Inheritance diagram for OpenGPS::BinaryLSBPointVectorReaderContext:

Collaboration diagram for OpenGPS::BinaryLSBPointVectorReaderContext:

12.4.1 Detailed Description

Implements OpenGPS::BinaryPointVectorReaderContext for binary files to be parsed on machines reading in least significant byte order.

Public Member Functions

- BinaryLSBPointVectorReaderContext (const OpenGPS::String &filePath)

  Creates a new instance.

- virtual OGPS_Boolean IsValid () const throw (...)

  Asks if there is readable point vector stored at the current position.

- virtual OGPS_Boolean MoveNext () throw (...)

  Move the current reading position of the stream to the next three-vector.

- virtual void Read (OGPS_Double *const value) throw (...)

Reads the currently underlying data as OGPS_Double.

- virtual void Read (OGPS_Float *const value) throw (...)

  Reads the currently underlying data as OGPS_Float.

- virtual void Read (OGPS_Int32 *const value) throw (...)

  Reads the currently underlying data as OGPS_Int32.

- virtual void Read (OGPS_Int16 *const value) throw (...)

  Reads the currently underlying data as OGPS_Int16.

- virtual void Skip () throw (...)

  Skips reading of the currently underlying data.

- virtual ~BinaryLSBPointVectorReaderContext ()

  Destroys this instance.

Protected Member Functions

- virtual void Close ()

  Closes the internal handle to the binary file and releases its resources.

- InputBinaryFileStream * GetStream ()

  Returns the underlying data stream for read access.

- OGPS_Boolean HasStream () const

  Returns TRUE if the underlying stream object had successfully been allocated, FALSE otherwise.

- virtual OGPS_Boolean IsGood () const

  Asks if the underlying data stream is still valid.

12.4.2 Constructor & Destructor Documentation

12.4.2.1 BinaryLSBPointVectorReaderContext::BinaryLSBPointVectorReaderContext (const OpenGPS::String & *filePath*)

Creates a new instance.

Parameters:

filePath Absolute path to the binary file streamed herein.

12.4.2.2 BinaryLSBPointVectorReaderContext::~BinaryLSBPointVectorReaderContext () `[virtual]`

Destroys this instance.

12.4.3 Member Function Documentation

12.4.3.1 void BinaryPointVectorReaderContext::Close () [protected, virtual, inherited]

Closes the internal handle to the binary file and releases its resources.

References _OPENGPS_DELETE, and OpenGPS::BinaryPointVectorReaderContext::m_Stream.

Referenced by OpenGPS::BinaryPointVectorReaderContext::~BinaryPointVectorReaderContext().

12.4.3.2 InputBinaryFileStream * BinaryPointVectorReaderContext::GetStream () [protected, inherited]

Returns the underlying data stream for read access.

References OpenGPS::BinaryPointVectorReaderContext::m_Stream.

Referenced by OpenGPS::BinaryMSBPointVectorReaderContext::Read(), and Read().

12.4.3.3 OGPS_Boolean BinaryPointVectorReaderContext::HasStream () const [protected, inherited]

Returns TRUE if the underlying stream object had successfully been allocated, FALSE otherwise.

References OpenGPS::BinaryPointVectorReaderContext::m_Stream, and NULL.

12.4.3.4 OGPS_Boolean BinaryPointVectorReaderContext::IsGood () const [protected, virtual, inherited]

Asks if the underlying data stream is still valid.

Returns:

Returns TRUE if no previous access to the underlying data stream was harmful. Returns FALSE if any damage occured.

References _ASSERT, and OpenGPS::BinaryPointVectorReaderContext::m_Stream.

Referenced by OpenGPS::BinaryPointVectorReaderContext::Skip().

12.4.3.5 OGPS_Boolean BinaryPointVectorReaderContext::IsValid () const throw (...) [virtual, inherited]

Asks if there is readable point vector stored at the current position.

A point vector may be invalid if it had not been measured, but is needed for topological consistency though.

Remarks:

This must be checked before one attempts to PointVectorReaderContext::Read/ PointVectorReaderContext::Skip any of the point data of one of the three coordinates.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns TRUE if the current point is readable, FALSE otherwise.

Implements OpenGPS::PointVectorReaderContext.

References TRUE.

12.4.3.6 OGPS_Boolean BinaryPointVectorReaderContext::MoveNext () throw (...) [virtual, inherited]

Move the current reading position of the stream to the next three-vector.

This is possible only if all three coordinates of the current point vector data had been fully read (see PointVectorReaderContext::Read and PointVectorReaderContext::Skip member functions). If PointVectorReaderContext::IsValid returns FALSE, call this method directly to move to the next point vector in storage.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns TRUE when there is more data to be parsed, FALSE otherwise.

Implements OpenGPS::PointVectorReaderContext.

References _CHECK_STREAM_AND_THROW_EXCEPTION, and OpenGPS::BinaryPointVectorReaderContext::m_Stream.

12.4.3.7 void BinaryLSBPointVectorReaderContext::Read (OGPS_Double *const *value*) throw (...) [virtual]

Reads the currently underlying data as OGPS_Double.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorReaderContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_STREAM_AND_THROW_EXCEPTION, _OPENGPS_BINFORMAT_DOUBLE_SIZE, and OpenGPS::BinaryPointVectorReaderContext::GetStream().

Here is the call graph for this function:

12.4.3.8 void BinaryLSBPointVectorReaderContext::Read (OGPS_Float *const *value*) throw (...) [virtual]

Reads the currently underlying data as OGPS_Float.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorReaderContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_-STREAM_AND_THROW_EXCEPTION, _OPENGPS_BINFORMAT_FLOAT_SIZE, and OpenGPS::BinaryPointVectorReaderContext::GetStream().

Here is the call graph for this function:

12.4.3.9 void BinaryLSBPointVectorReaderContext::Read (OGPS_Int32 *const *value*) throw (...) [virtual]

Reads the currently underlying data as OGPS_Int32.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorReaderContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_-STREAM_AND_THROW_EXCEPTION, _OPENGPS_BINFORMAT_INT32_SIZE, and OpenGPS::BinaryPointVectorReaderContext::GetStream().

Here is the call graph for this function:

12.4.3.10 void BinaryLSBPointVectorReaderContext::Read (OGPS_Int16 *const *value*) throw (...) [virtual]

Reads the currently underlying data as OGPS_Int16.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorReaderContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_-STREAM_AND_THROW_EXCEPTION, _OPENGPS_BINFORMAT_INT16_SIZE, and OpenGPS::BinaryPointVectorReaderContext::GetStream().

Here is the call graph for this function:

12.4.3.11 void BinaryPointVectorReaderContext::Skip () throw (...) [virtual, inherited]

Skips reading of the currently underlying data.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

This must be called for integrity if it is expected from the reading process that there is no point data available for reading, i.e. for the current coordinate no point data was saved because the corresponding axis is of incremental type.

Implements OpenGPS::PointVectorReaderContext.

References _CHECK_STREAM_AND_THROW_EXCEPTION, _EX_T, OpenGPS::BinaryPointVectorReaderContext::IsGood(), and OGPS_ExInvalidOperation.

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- binary_lsb_point_vector_reader_context.hxx
- binary_lsb_point_vector_reader_context.cxx

12.5 OpenGPS::BinaryLSBPointVectorWriterContext Class Reference

```
#include <binary_lsb_point_vector_writer_context.hxx>
```

Inheritance diagram for OpenGPS::BinaryLSBPointVectorWriterContext:

Collaboration diagram for OpenGPS::BinaryLSBPointVectorWriterContext:

12.5.1 Detailed Description

Implements OpenGPS::BinaryPointVectorWriterContext for binary files to be written on machines operating in least significant byte order.

Writes binary data to a compressed stream. Normally this stream points to a file descriptor within a zip archive.

Public Member Functions

- BinaryLSBPointVectorWriterContext (zipFile handle)

  Creates a new instance.

- virtual void Close ()

  Closes the internal handle to the binary stream and frees its resources.

- void GetMd5 (OpenGPS::UnsignedByte md5[16])

  Gets the md5 checksum of all bytes written.

- virtual void MoveNext () throw (...)

  Complete the transaction of the current point vector.

- virtual void Skip () throw (...)

  There is no point data to be written to the underlying stream for the current axis.

- virtual void Write (const OGPS_Double *const value) throw (...)

  Writes a single point of type OGPS_Double to the underlying stream.

- virtual void Write (const OGPS_Float *const value) throw (...)

  Writes a single point of type OGPS_Float to the underlying stream.

- virtual void Write (const OGPS_Int32 *const value) throw (...)

  Writes a single point of type OGPS_Int32 to the underlying stream.

- virtual void Write (const OGPS_Int16 *const value) throw (...)

  Writes a single point of type OGPS_Int16 to the underlying stream.

- virtual ~BinaryLSBPointVectorWriterContext ()

  Destroys this instance.

Protected Member Functions

- std::ostream * GetStream ()
  Gets the internal binary stream.
- OGPS_Boolean HasStream () const
  Asks whether there is a target stream available.
- OGPS_Boolean IsGood () const
  Asks if the underlying data stream is still valid.

12.5.2 Constructor & Destructor Documentation

12.5.2.1 BinaryLSBPointVectorWriterContext::BinaryLSBPointVectorWriterContext (zipFile *handle*)

Creates a new instance.

Parameters:

handle The zip-stream where binary data is written to.

12.5.2.2 BinaryLSBPointVectorWriterContext::~BinaryLSBPointVectorWriterContext () `[virtual]`

Destroys this instance.

12.5.3 Member Function Documentation

12.5.3.1 void BinaryPointVectorWriterContext::Close () `[virtual, inherited]`

Closes the internal handle to the binary stream and frees its resources.

References _OPENGPS_DELETE, OpenGPS::BinaryPointVectorWriterContext::m_Buffer, and OpenGPS::BinaryPointVectorWriterContext::m_Stream.

Referenced by OpenGPS::BinaryPointVectorWriterContext::~BinaryPointVectorWriterContext().

12.5.3.2 void BinaryPointVectorWriterContext::GetMd5 (OpenGPS::UnsignedByte *md5*[16]) `[inherited]`

Gets the md5 checksum of all bytes written.

When called this resets the currently computed md5 sum. Future calls to this method will ignore older md5 data.

References _ASSERT, OpenGPS::ZipStreamBuffer::GetMd5(), OpenGPS::BinaryPointVectorWriterContext::HasStream(), and OpenGPS::BinaryPointVectorWriterContext::m_Buffer.

Here is the call graph for this function:

12.5.3.3 std::ostream * BinaryPointVectorWriterContext::GetStream () [protected, inherited]

Gets the internal binary stream.

Returns:

Returns the target binary stream or NULL.

References _ASSERT, OpenGPS::BinaryPointVectorWriterContext::m_Buffer, and OpenGPS::BinaryPointVectorWriterContext::m_Stream.

Referenced by OpenGPS::BinaryMSBPointVectorWriterContext::Write(), and Write().

12.5.3.4 OGPS_Boolean BinaryPointVectorWriterContext::HasStream () const [protected, inherited]

Asks whether there is a target stream available.

Returns:

Returns TRUE if there is an operable target stream for point data, FALSE otherwise.

See also:

BinaryPointVectorWriterContext::IsGood

References _ASSERT, FALSE, OpenGPS::BinaryPointVectorWriterContext::m_Buffer, OpenGPS::BinaryPointVectorWriterContext::m_Stream, and TRUE.

Referenced by OpenGPS::BinaryPointVectorWriterContext::GetMd5().

12.5.3.5 OGPS_Boolean BinaryPointVectorWriterContext::IsGood () const [protected, virtual, inherited]

Asks if the underlying data stream is still valid.

Returns:

Returns TRUE if no previous access to the underlying data stream was harmful. Returns FALSE if any damage occured.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, OpenGPS::BinaryPointVectorWriterContext::m_Buffer, and OpenGPS::BinaryPointVectorWriterContext::m_Stream.

12.5.3.6 void BinaryPointVectorWriterContext::MoveNext () throw (...) [virtual, inherited]

Complete the transaction of the current point vector.

One point vector is made up of three components (data points). After three subsequent PointVectorWriterContext::Write/ PointVectorWriterContext::Skip calls, call this method to indicate that the whole point vector has been written.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Implements OpenGPS::PointVectorWriterContext.

12.5.3.7 void BinaryPointVectorWriterContext::Skip () throw (...) [virtual, inherited]

There is no point data to be written to the underlying stream for the current axis.

One point vector has three components, whereas only the Z component needs to be set explicitly in major cases because of implicit axis definitions of X or Y. Use this skip method then (with the X or Y components) to indicate that, because a successfully written point vector must comprise all three component values.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

This must be called for integrity if there is no point data to be stored because the corresponding axis is of incremental type.

Implements OpenGPS::PointVectorWriterContext.

References _CHECK_ISGOOD_AND_THROW_EXCEPTION, and _CHECK_STREAM_AND_-THROW_EXCEPTION.

12.5.3.8 void BinaryLSBPointVectorWriterContext::Write (const OGPS_Double *const *value*) throw (...) [virtual]

Writes a single point of type OGPS_Double to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_-STREAM_AND_THROW_EXCEPTION, _OPENGPS_BINFORMAT_DOUBLE_SIZE, and OpenGPS::BinaryPointVectorWriterContext::GetStream().

Here is the call graph for this function:

12.5.3.9 void BinaryLSBPointVectorWriterContext::Write (const OGPS_Float *const *value*) throw (...) [virtual]

Writes a single point of type OGPS_Float to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_-STREAM_AND_THROW_EXCEPTION, _OPENGPS_BINFORMAT_FLOAT_SIZE, and OpenGPS::BinaryPointVectorWriterContext::GetStream().

Here is the call graph for this function:

12.5.3.10 void BinaryLSBPointVectorWriterContext::Write (const OGPS_Int32 *const *value*) throw (...) [virtual]

Writes a single point of type OGPS_Int32 to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_-STREAM_AND_THROW_EXCEPTION, _OPENGPS_BINFORMAT_INT32_SIZE, and OpenGPS::BinaryPointVectorWriterContext::GetStream().

Here is the call graph for this function:

12.5.3.11 void BinaryLSBPointVectorWriterContext::Write (const OGPS_Int16 *const *value*) throw (...) [virtual]

Writes a single point of type OGPS_Int16 to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_-STREAM_AND_THROW_EXCEPTION, _OPENGPS_BINFORMAT_INT16_SIZE, and OpenGPS::BinaryPointVectorWriterContext::GetStream().

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- binary_lsb_point_vector_writer_context.hxx
- binary_lsb_point_vector_writer_context.cxx

12.6 OpenGPS::BinaryMSBPointVectorReaderContext Class Reference

```
#include <binary_msb_point_vector_reader_context.hxx>
```

Inheritance diagram for OpenGPS::BinaryMSBPointVectorReaderContext:

Collaboration diagram for OpenGPS::BinaryMSBPointVectorReaderContext:

12.6.1 Detailed Description

Implements OpenGPS::BinaryPointVectorReaderContext for binary files to be parsed on machines reading in most significant byte order.

Public Member Functions

- BinaryMSBPointVectorReaderContext (const OpenGPS::String &filePath)

  Creates a new instance.

- virtual OGPS_Boolean IsValid () const throw (...)

  Asks if there is readable point vector stored at the current position.

- virtual OGPS_Boolean MoveNext () throw (...)

  Move the current reading position of the stream to the next three-vector.

- virtual void Read (OGPS_Double *const value) throw (...)

  Reads the currently underlying data as OGPS_Double.

- virtual void Read (OGPS_Float *const value) throw (...)

  Reads the currently underlying data as OGPS_Float.

- virtual void Read (OGPS_Int32 *const value) throw (...)

  Reads the currently underlying data as OGPS_Int32.

- virtual void Read (OGPS_Int16 *const value) throw (...)

  Reads the currently underlying data as OGPS_Int16.

- virtual void Skip () throw (...)

  Skips reading of the currently underlying data.

- virtual ~BinaryMSBPointVectorReaderContext ()

  Destroys this instance.

Protected Member Functions

- virtual void Close ()

  Closes the internal handle to the binary file and releases its resources.

- InputBinaryFileStream * GetStream ()

  Returns the underlying data stream for read access.

- OGPS_Boolean HasStream () const

  Returns TRUE if the underlying stream object had successfully been allocated, FALSE otherwise.

- virtual OGPS_Boolean IsGood () const

  Asks if the underlying data stream is still valid.

12.6.2 Constructor & Destructor Documentation

12.6.2.1 BinaryMSBPointVectorReaderContext::BinaryMSBPointVectorReaderContext (const OpenGPS::String & *filePath*)

Creates a new instance.

Parameters:

filePath Absolute path to the binary file streamed herein.

12.6.2.2 BinaryMSBPointVectorReaderContext::~BinaryMSBPointVectorReaderContext () [virtual]

Destroys this instance.

References OpenGPS::Environment::Reset().

Here is the call graph for this function:

12.6.3 Member Function Documentation

12.6.3.1 void BinaryPointVectorReaderContext::Close () [protected, virtual, inherited]

Closes the internal handle to the binary file and releases its resources.

References _OPENGPS_DELETE, and OpenGPS::BinaryPointVectorReaderContext::m_Stream.

Referenced by OpenGPS::BinaryPointVectorReaderContext::~BinaryPointVectorReaderContext().

12.6.3.2 InputBinaryFileStream * BinaryPointVectorReaderContext::GetStream () [protected, inherited]

Returns the underlying data stream for read access.

References OpenGPS::BinaryPointVectorReaderContext::m_Stream.

Referenced by Read(), and OpenGPS::BinaryLSBPointVectorReaderContext::Read().

12.6.3.3 OGPS_Boolean BinaryPointVectorReaderContext::HasStream () const [protected, inherited]

Returns TRUE if the underlying stream object had successfully been allocated, FALSE otherwise.

References OpenGPS::BinaryPointVectorReaderContext::m_Stream, and NULL.

12.6.3.4 OGPS_Boolean BinaryPointVectorReaderContext::IsGood () const [protected, virtual, inherited]

Asks if the underlying data stream is still valid.

Returns:

Returns TRUE if no previous access to the underlying data stream was harmful. Returns FALSE if any damage occured.

References _ASSERT, and OpenGPS::BinaryPointVectorReaderContext::m_Stream.

Referenced by OpenGPS::BinaryPointVectorReaderContext::Skip().

12.6.3.5 OGPS_Boolean BinaryPointVectorReaderContext::IsValid () const throw (...) [virtual, inherited]

Asks if there is readable point vector stored at the current position.

A point vector may be invalid if it had not been measured, but is needed for topological consistency though.

Remarks:

This must be checked before one attempts to PointVectorReaderContext::Read/ PointVectorReaderContext::Skip any of the point data of one of the three coordinates.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns TRUE if the current point is readable, FALSE otherwise.

Implements OpenGPS::PointVectorReaderContext.

References TRUE.

12.6.3.6 OGPS_Boolean BinaryPointVectorReaderContext::MoveNext () throw (...) [virtual, inherited]

Move the current reading position of the stream to the next three-vector.

This is possible only if all three coordinates of the current point vector data had been fully read (see PointVectorReaderContext::Read and PointVectorReaderContext::Skip member functions). If PointVectorReaderContext::IsValid returns FALSE, call this method directly to move to the next point vector in storage.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns TRUE when there is more data to be parsed, FALSE otherwise.

Implements OpenGPS::PointVectorReaderContext.

References _CHECK_STREAM_AND_THROW_EXCEPTION, and OpenGPS::BinaryPointVectorReaderContext::m_Stream.

12.6.3.7 void BinaryMSBPointVectorReaderContext::Read (OGPS_Double *const *value*) throw (...) [virtual]

Reads the currently underlying data as OGPS_Double.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorReaderContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_-STREAM_AND_THROW_EXCEPTION, _OPENGPS_BINFORMAT_DOUBLE_SIZE, OpenGPS::Environment::ByteSwap64(), OpenGPS::Environment::GetInstance(), and OpenGPS::BinaryPointVectorReaderContext::GetStream().

Here is the call graph for this function:

12.6.3.8 void BinaryMSBPointVectorReaderContext::Read (OGPS_Float *const *value*) throw (...) [virtual]

Reads the currently underlying data as OGPS_Float.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorReaderContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_-STREAM_AND_THROW_EXCEPTION, _OPENGPS_BINFORMAT_FLOAT_SIZE, OpenGPS::Environment::ByteSwap32(), OpenGPS::Environment::GetInstance(), and OpenGPS::BinaryPointVectorReaderContext::GetStream().

Here is the call graph for this function:

12.6.3.9 void BinaryMSBPointVectorReaderContext::Read (OGPS_Int32 *const *value*) throw (...) [virtual]

Reads the currently underlying data as OGPS_Int32.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorReaderContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_-STREAM_AND_THROW_EXCEPTION, _OPENGPS_BINFORMAT_INT32_SIZE, OpenGPS::Environment::ByteSwap32(), OpenGPS::Environment::GetInstance(), and OpenGPS::BinaryPointVectorReaderContext::GetStream().

Here is the call graph for this function:

12.6.3.10 void BinaryMSBPointVectorReaderContext::Read (OGPS_Int16 *const *value*) throw (...) [virtual]

Reads the currently underlying data as OGPS_Int16.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorReaderContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_-STREAM_AND_THROW_EXCEPTION, _OPENGPS_BINFORMAT_INT16_SIZE, OpenGPS::Environment::ByteSwap16(), OpenGPS::Environment::GetInstance(), and OpenGPS::BinaryPointVectorReaderContext::GetStream().

Here is the call graph for this function:

12.6.3.11 void BinaryPointVectorReaderContext::Skip () throw (...) [virtual, inherited]

Skips reading of the currently underlying data.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

This must be called for integrity if it is expected from the reading process that there is no point data available for reading, i.e. for the current coordinate no point data was saved because the corresponding axis is of incremental type.

Implements OpenGPS::PointVectorReaderContext.

References _CHECK_STREAM_AND_THROW_EXCEPTION, _EX_T, OpenGPS::BinaryPointVectorReaderContext::IsGood(), and OGPS_ExInvalidOperation.

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- binary_msb_point_vector_reader_context.hxx
- binary_msb_point_vector_reader_context.cxx

12.7 OpenGPS::BinaryMSBPointVectorWriterContext Class Reference

```
#include <binary_msb_point_vector_writer_context.hxx>
```

Inheritance diagram for OpenGPS::BinaryMSBPointVectorWriterContext:

Collaboration diagram for OpenGPS::BinaryMSBPointVectorWriterContext:

12.7.1 Detailed Description

Implements OpenGPS::BinaryPointVectorWriterContext for binary files to be written on machines operating in most significant byte order.

Writes binary data to a compressed stream. Normally this stream points to a file descriptor within a zip archive.

Public Member Functions

- BinaryMSBPointVectorWriterContext (zipFile handle)
  Creates a new instance.
- virtual void Close ()
  Closes the internal handle to the binary stream and frees its resources.
- void GetMd5 (OpenGPS::UnsignedByte md5[16])
  Gets the md5 checksum of all bytes written.
- virtual void MoveNext () throw (...)
  Complete the transaction of the current point vector.
- virtual void Skip () throw (...)
  There is no point data to be written to the underlying stream for the current axis.
- virtual void Write (const OGPS_Double *const value) throw (...)
  Writes a single point of type OGPS_Double to the underlying stream.
- virtual void Write (const OGPS_Float *const value) throw (...)
  Writes a single point of type OGPS_Float to the underlying stream.
- virtual void Write (const OGPS_Int32 *const value) throw (...)
  Writes a single point of type OGPS_Int32 to the underlying stream.
- virtual void Write (const OGPS_Int16 *const value) throw (...)
  Writes a single point of type OGPS_Int16 to the underlying stream.
- virtual ~BinaryMSBPointVectorWriterContext ()
  Destroys this instance.

Protected Member Functions

- std::ostream * GetStream ()
  Gets the internal binary stream.
- OGPS_Boolean HasStream () const
  Asks whether there is a target stream available.
- OGPS_Boolean IsGood () const
  Asks if the underlying data stream is still valid.

12.7.2 Constructor & Destructor Documentation

12.7.2.1 BinaryMSBPointVectorWriterContext::BinaryMSBPointVectorWriterContext (zipFile *handle*)

Creates a new instance.

Parameters:

handle The zip-stream where binary data is written to.

12.7.2.2 BinaryMSBPointVectorWriterContext::~BinaryMSBPointVectorWriterContext () [virtual]

Destroys this instance.

References OpenGPS::Environment::Reset().

Here is the call graph for this function:

12.7.3 Member Function Documentation

12.7.3.1 void BinaryPointVectorWriterContext::Close () [virtual, inherited]

Closes the internal handle to the binary stream and frees its resources.

References _OPENGPS_DELETE, OpenGPS::BinaryPointVectorWriterContext::m_Buffer, and OpenGPS::BinaryPointVectorWriterContext::m_Stream.

Referenced by OpenGPS::BinaryPointVectorWriterContext::~BinaryPointVectorWriterContext().

12.7.3.2 void BinaryPointVectorWriterContext::GetMd5 (OpenGPS::UnsignedByte *md5*[16]) [inherited]

Gets the md5 checksum of all bytes written.

When called this resets the currently computed md5 sum. Future calls to this method will ignore older md5 data.

References _ASSERT, OpenGPS::ZipStreamBuffer::GetMd5(), OpenGPS::BinaryPointVectorWriterContext::HasStream(), and OpenGPS::BinaryPointVectorWriterContext::m_Buffer.

Here is the call graph for this function:

12.7.3.3 std::ostream * BinaryPointVectorWriterContext::GetStream () [protected, inherited]

Gets the internal binary stream.

Returns:

Returns the target binary stream or NULL.

References _ASSERT, OpenGPS::BinaryPointVectorWriterContext::m_Buffer, and OpenGPS::BinaryPointVectorWriterContext::m_Stream.

Referenced by Write(), and OpenGPS::BinaryLSBPointVectorWriterContext::Write().

12.7.3.4 OGPS_Boolean BinaryPointVectorWriterContext::HasStream () const [protected, inherited]

Asks whether there is a target stream available.

Returns:

Returns TRUE if there is an operable target stream for point data, FALSE otherwise.

See also:

BinaryPointVectorWriterContext::IsGood

References _ASSERT, FALSE, OpenGPS::BinaryPointVectorWriterContext::m_Buffer, OpenGPS::BinaryPointVectorWriterContext::m_Stream, and TRUE.

Referenced by OpenGPS::BinaryPointVectorWriterContext::GetMd5().

12.7.3.5 OGPS_Boolean BinaryPointVectorWriterContext::IsGood () const [protected, virtual, inherited]

Asks if the underlying data stream is still valid.

Returns:

Returns TRUE if no previous access to the underlying data stream was harmful. Returns FALSE if any damage occured.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, OpenGPS::BinaryPointVectorWriterContext::m_Buffer, and OpenGPS::BinaryPointVectorWriterContext::m_Stream.

12.7.3.6 void BinaryPointVectorWriterContext::MoveNext () throw (...) [virtual, inherited]

Complete the transaction of the current point vector.

One point vector is made up of three components (data points). After three subsequent PointVectorWriterContext::Write/ PointVectorWriterContext::Skip calls, call this method to indicate that the whole point vector has been written.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Implements OpenGPS::PointVectorWriterContext.

12.7.3.7 void BinaryPointVectorWriterContext::Skip () throw (...) [virtual, inherited]

There is no point data to be written to the underlying stream for the current axis.

One point vector has three components, whereas only the Z component needs to be set explicitly in major cases because of implicit axis definitions of X or Y. Use this skip method then (with the X or Y components) to indicate that, because a successfully written point vector must comprise all three component values.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

This must be called for integrity if there is no point data to be stored because the corresponding axis is of incremental type.

Implements OpenGPS::PointVectorWriterContext.

References _CHECK_ISGOOD_AND_THROW_EXCEPTION, and _CHECK_STREAM_AND_-THROW_EXCEPTION.

12.7.3.8 void BinaryMSBPointVectorWriterContext::Write (const OGPS_Double *const *value*) throw (...) [virtual]

Writes a single point of type OGPS_Double to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_-STREAM_AND_THROW_EXCEPTION, _OPENGPS_BINFORMAT_DOUBLE_SIZE, OpenGPS::Environment::GetInstance(), and OpenGPS::BinaryPointVectorWriterContext::GetStream().

Here is the call graph for this function:

12.7.3.9 void BinaryMSBPointVectorWriterContext::Write (const OGPS_Float *const *value*) throw (...) [virtual]

Writes a single point of type OGPS_Float to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_-STREAM_AND_THROW_EXCEPTION, _OPENGPS_BINFORMAT_FLOAT_SIZE, OpenGPS::Environment::GetInstance(), and OpenGPS::BinaryPointVectorWriterContext::GetStream().

Here is the call graph for this function:

12.7.3.10 void BinaryMSBPointVectorWriterContext::Write (const OGPS_Int32 *const *value*) throw (...) [virtual]

Writes a single point of type OGPS_Int32 to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_-STREAM_AND_THROW_EXCEPTION, _OPENGPS_BINFORMAT_INT32_SIZE, OpenGPS::Environment::GetInstance(), and OpenGPS::BinaryPointVectorWriterContext::GetStream().

Here is the call graph for this function:

12.7.3.11 void BinaryMSBPointVectorWriterContext::Write (const OGPS_Int16 *const *value*) throw (...) [virtual]

Writes a single point of type OGPS_Int16 to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_-STREAM_AND_THROW_EXCEPTION, _OPENGPS_BINFORMAT_INT16_SIZE, OpenGPS::Environment::GetInstance(), and OpenGPS::BinaryPointVectorWriterContext::GetStream().

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- binary_msb_point_vector_writer_context.hxx
- binary_msb_point_vector_writer_context.cxx

12.8 OpenGPS::BinaryPointVectorReaderContext Class Reference

```
#include <binary_point_vector_reader_context.hxx>
```

Inheritance diagram for OpenGPS::BinaryPointVectorReaderContext:

Collaboration diagram for OpenGPS::BinaryPointVectorReaderContext:

12.8.1 Detailed Description

Specialized OpenGPS::PointVectorReaderContext for binary streams.

Public Member Functions

- virtual OGPS_Boolean IsValid () const throw (...)
  Asks if there is readable point vector stored at the current position.
- virtual OGPS_Boolean MoveNext () throw (...)
  Move the current reading position of the stream to the next three-vector.
- virtual void Read (OGPS_Double *const value)=0 throw (...)
  Reads the currently underlying data as OGPS_Double.
- virtual void Read (OGPS_Float *const value)=0 throw (...)
  Reads the currently underlying data as OGPS_Float.
- virtual void Read (OGPS_Int32 *const value)=0 throw (...)
  Reads the currently underlying data as OGPS_Int32.
- virtual void Read (OGPS_Int16 *const value)=0 throw (...)
  Reads the currently underlying data as OGPS_Int16.
- virtual void Skip () throw (...)
  Skips reading of the currently underlying data.

Protected Member Functions

- BinaryPointVectorReaderContext (const OpenGPS::String &filePath)
  Creates a new instance.
- virtual void Close ()
  Closes the internal handle to the binary file and releases its resources.
- InputBinaryFileStream * GetStream ()
  Returns the underlying data stream for read access.
- OGPS_Boolean HasStream () const
  Returns TRUE if the underlying stream object had successfully been allocated, FALSE otherwise.
- virtual OGPS_Boolean IsGood () const
  Asks if the underlying data stream is still valid.
- virtual ~BinaryPointVectorReaderContext ()
  Destroys this instance.

Private Attributes

- InputBinaryFileStream * m_Stream

  Pointer to the underlying data stream of binary point vectors.

12.8.2 Constructor & Destructor Documentation

12.8.2.1 BinaryPointVectorReaderContext::BinaryPointVectorReaderContext (const OpenGPS::String & *filePath*) [protected]

Creates a new instance.

Parameters:

filePath Absolute path to the binary file streamed herein.

References m_Stream.

12.8.2.2 BinaryPointVectorReaderContext::~BinaryPointVectorReaderContext () [protected, virtual]

Destroys this instance.

References Close().

Here is the call graph for this function:

12.8.3 Member Function Documentation

12.8.3.1 void BinaryPointVectorReaderContext::Close () [protected, virtual]

Closes the internal handle to the binary file and releases its resources.

References _OPENGPS_DELETE, and m_Stream.

Referenced by ~BinaryPointVectorReaderContext().

12.8.3.2 InputBinaryFileStream * BinaryPointVectorReaderContext::GetStream () [protected]

Returns the underlying data stream for read access.

References m_Stream.

Referenced by OpenGPS::BinaryMSBPointVectorReaderContext::Read(), and OpenGPS::BinaryLSBPointVectorReaderContext::Read().

12.8.3.3 OGPS_Boolean BinaryPointVectorReaderContext::HasStream () const [protected]

Returns TRUE if the underlying stream object had successfully been allocated, FALSE otherwise.

References m_Stream, and NULL.

12.8.3.4 OGPS_Boolean BinaryPointVectorReaderContext::IsGood () const [protected, virtual]

Asks if the underlying data stream is still valid.

Returns:

Returns TRUE if no previous access to the underlying data stream was harmful. Returns FALSE if any damage occured.

References _ASSERT, and m_Stream.

Referenced by Skip().

12.8.3.5 OGPS_Boolean BinaryPointVectorReaderContext::IsValid () const throw (...) [virtual]

Asks if there is readable point vector stored at the current position.

A point vector may be invalid if it had not been measured, but is needed for topological consistency though.

Remarks:

This must be checked before one attempts to PointVectorReaderContext::Read/ PointVectorReaderContext::Skip any of the point data of one of the three coordinates.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns TRUE if the current point is readable, FALSE otherwise.

Implements OpenGPS::PointVectorReaderContext.

References TRUE.

12.8.3.6 OGPS_Boolean BinaryPointVectorReaderContext::MoveNext () throw (...) [virtual]

Move the current reading position of the stream to the next three-vector.

This is possible only if all three coordinates of the current point vector data had been fully read (see PointVectorReaderContext::Read and PointVectorReaderContext::Skip member functions). If PointVectorReaderContext::IsValid returns FALSE, call this method directly to move to the next point vector in storage.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns TRUE when there is more data to be parsed, FALSE otherwise.

Implements OpenGPS::PointVectorReaderContext.

References _CHECK_STREAM_AND_THROW_EXCEPTION, and m_Stream.

12.8.3.7 virtual void OpenGPS::PointVectorReaderContext::Read (OGPS_Double *const *value*) throw (...) [pure virtual, inherited]

Reads the currently underlying data as OGPS_Double.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implemented in OpenGPS::BinaryLSBPointVectorReaderContext, OpenGPS::BinaryMSBPointVectorReaderContext, and OpenGPS::XmlPointVectorReaderContext.

12.8.3.8 virtual void OpenGPS::PointVectorReaderContext::Read (OGPS_Float *const *value*) throw (...) [pure virtual, inherited]

Reads the currently underlying data as OGPS_Float.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implemented in OpenGPS::BinaryLSBPointVectorReaderContext, OpenGPS::BinaryMSBPointVectorReaderContext, and OpenGPS::XmlPointVectorReaderContext.

12.8.3.9 virtual void OpenGPS::PointVectorReaderContext::Read (OGPS_Int32 *const *value*) throw (...) [pure virtual, inherited]

Reads the currently underlying data as OGPS_Int32.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implemented in OpenGPS::BinaryLSBPointVectorReaderContext, OpenGPS::BinaryMSBPointVectorReaderContext, and OpenGPS::XmlPointVectorReaderContext.

12.8.3.10 virtual void OpenGPS::PointVectorReaderContext::Read (OGPS_Int16 *const *value*) throw (...) [pure virtual, inherited]

Reads the currently underlying data as OGPS_Int16.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implemented in OpenGPS::BinaryLSBPointVectorReaderContext, OpenGPS::BinaryMSBPointVectorReaderContext, and OpenGPS::XmlPointVectorReaderContext.

12.8.3.11 void BinaryPointVectorReaderContext::Skip () throw (...) [virtual]

Skips reading of the currently underlying data.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

This must be called for integrity if it is expected from the reading process that there is no point data available for reading, i.e. for the current coordinate no point data was saved because the corresponding axis is of incremental type.

Implements OpenGPS::PointVectorReaderContext.

References _CHECK_STREAM_AND_THROW_EXCEPTION, _EX_T, IsGood(), and OGPS_-ExInvalidOperation.

Here is the call graph for this function:

12.8.4 Member Data Documentation

12.8.4.1 InputBinaryFileStream* OpenGPS::BinaryPointVectorReaderContext::m_Stream [private]

Pointer to the underlying data stream of binary point vectors.

Referenced by BinaryPointVectorReaderContext(), Close(), GetStream(), HasStream(), IsGood(), and MoveNext().

The documentation for this class was generated from the following files:

- binary_point_vector_reader_context.hxx
- binary_point_vector_reader_context.cxx

12.9 OpenGPS::BinaryPointVectorWriterContext Class Reference

```
#include <binary_point_vector_writer_context.hxx>
```

Inheritance diagram for OpenGPS::BinaryPointVectorWriterContext:

Collaboration diagram for OpenGPS::BinaryPointVectorWriterContext:

12.9.1 Detailed Description

Implements OpenGPS::PointVectorWriterContext for writing to compressed binary streams of point vectors.

Usually this points to a file descriptor within a zip archive.

Public Member Functions

- BinaryPointVectorWriterContext (zipFile handle)

  Creates a new instance.

- virtual void Close ()

  Closes the internal handle to the binary stream and frees its resources.

- void GetMd5 (OpenGPS::UnsignedByte md5[16])

  Gets the md5 checksum of all bytes written.

- virtual void MoveNext () throw (...)

  Complete the transaction of the current point vector.

- virtual void Skip () throw (...)

  There is no point data to be written to the underlying stream for the current axis.

- virtual void Write (const OGPS_Double *const value)=0 throw (...)

  Writes a single point of type OGPS_Double to the underlying stream.

- virtual void Write (const OGPS_Float *const value)=0 throw (...)

  Writes a single point of type OGPS_Float to the underlying stream.

- virtual void Write (const OGPS_Int32 *const value)=0 throw (...)

  Writes a single point of type OGPS_Int32 to the underlying stream.

- virtual void Write (const OGPS_Int16 *const value)=0 throw (...)

  Writes a single point of type OGPS_Int16 to the underlying stream.

Protected Member Functions

- std::ostream * GetStream ()

  Gets the internal binary stream.

- OGPS_Boolean HasStream () const

Asks whether there is a target stream available.

- OGPS_Boolean IsGood () const

  Asks if the underlying data stream is still valid.

- virtual ~BinaryPointVectorWriterContext ()

  Destroys this instance.

Private Attributes

- ZipStreamBuffer * m_Buffer

  The target buffer where compressed binary data gets stored.

- ZipOutputStream * m_Stream

  The stream buffer which makes BinaryPointVectorWriterContext::m_Buffer accessible through the more abstract std::ostream interface.

12.9.2 Constructor & Destructor Documentation

12.9.2.1 BinaryPointVectorWriterContext::BinaryPointVectorWriterContext (zipFile *handle*)

Creates a new instance.

Parameters:

handle The zip-stream where binary data is written to.

References m_Buffer, m_Stream, and TRUE.

12.9.2.2 BinaryPointVectorWriterContext::~BinaryPointVectorWriterContext () `[protected, virtual]`

Destroys this instance.

References Close().

Here is the call graph for this function:

12.9.3 Member Function Documentation

12.9.3.1 void BinaryPointVectorWriterContext::Close () `[virtual]`

Closes the internal handle to the binary stream and frees its resources.

References _OPENGPS_DELETE, m_Buffer, and m_Stream.

Referenced by ~BinaryPointVectorWriterContext().

12.9.3.2 void BinaryPointVectorWriterContext::GetMd5 (OpenGPS::UnsignedByte *md5*[16])

Gets the md5 checksum of all bytes written.

When called this resets the currently computed md5 sum. Future calls to this method will ignore older md5 data.

References _ASSERT, OpenGPS::ZipStreamBuffer::GetMd5(), HasStream(), and m_Buffer.

Here is the call graph for this function:

12.9.3.3 std::ostream * BinaryPointVectorWriterContext::GetStream () [protected]

Gets the internal binary stream.

Returns:

Returns the target binary stream or NULL.

References _ASSERT, m_Buffer, and m_Stream.

Referenced by OpenGPS::BinaryMSBPointVectorWriterContext::Write(), and OpenGPS::BinaryLSBPointVectorWriterContext::Write().

12.9.3.4 OGPS_Boolean BinaryPointVectorWriterContext::HasStream () const [protected]

Asks whether there is a target stream available.

Returns:

Returns TRUE if there is an operable target stream for point data, FALSE otherwise.

See also:

BinaryPointVectorWriterContext::IsGood

References _ASSERT, FALSE, m_Buffer, m_Stream, and TRUE.

Referenced by GetMd5().

12.9.3.5 OGPS_Boolean BinaryPointVectorWriterContext::IsGood () const [protected, virtual]

Asks if the underlying data stream is still valid.

Returns:

Returns TRUE if no previous access to the underlying data stream was harmful. Returns FALSE if any damage occured.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, m_Buffer, and m_Stream.

12.9.3.6 void BinaryPointVectorWriterContext::MoveNext () throw (...) [virtual]

Complete the transaction of the current point vector.

One point vector is made up of three components (data points). After three subsequent PointVectorWriterContext::Write/ PointVectorWriterContext::Skip calls, call this method to indicate that the whole point vector has been written.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Implements OpenGPS::PointVectorWriterContext.

12.9.3.7 void BinaryPointVectorWriterContext::Skip () throw (...) [virtual]

There is no point data to be written to the underlying stream for the current axis.

One point vector has three components, whereas only the Z component needs to be set explicitly in major cases because of implicit axis definitions of X or Y. Use this skip method then (with the X or Y components) to indicate that, because a successfully written point vector must comprise all three component values.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

This must be called for integrity if there is no point data to be stored because the corresponding axis is of incremental type.

Implements OpenGPS::PointVectorWriterContext.

References _CHECK_ISGOOD_AND_THROW_EXCEPTION, and _CHECK_STREAM_AND_-THROW_EXCEPTION.

12.9.3.8 virtual void OpenGPS::PointVectorWriterContext::Write (const OGPS_Double *const *value*) throw (...) [pure virtual, inherited]

Writes a single point of type OGPS_Double to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implemented in OpenGPS::BinaryLSBPointVectorWriterContext, OpenGPS::BinaryMSBPointVectorWriterContext, and OpenGPS::XmlPointVectorWriterContext.

12.9.3.9 virtual void OpenGPS::PointVectorWriterContext::Write (const OGPS_Float *const *value*) throw (...) [pure virtual, inherited]

Writes a single point of type OGPS_Float to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implemented in OpenGPS::BinaryLSBPointVectorWriterContext, OpenGPS::BinaryMSBPointVectorWriterContext, and OpenGPS::XmlPointVectorWriterContext.

12.9.3.10 virtual void OpenGPS::PointVectorWriterContext::Write (const OGPS_Int32 *const *value*) throw (...) [pure virtual, inherited]

Writes a single point of type OGPS_Int32 to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implemented in OpenGPS::BinaryLSBPointVectorWriterContext, OpenGPS::BinaryMSBPointVectorWriterContext, and OpenGPS::XmlPointVectorWriterContext.

12.9.3.11 virtual void OpenGPS::PointVectorWriterContext::Write (const OGPS_Int16 *const *value*) throw (...) [pure virtual, inherited]

Writes a single point of type OGPS_Int16 to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implemented in OpenGPS::BinaryLSBPointVectorWriterContext, OpenGPS::BinaryMSBPointVectorWriterContext, and OpenGPS::XmlPointVectorWriterContext.

12.9.4 Member Data Documentation

12.9.4.1 ZipStreamBuffer* OpenGPS::BinaryPointVectorWriterContext::m_Buffer [private]

The target buffer where compressed binary data gets stored.

Referenced by BinaryPointVectorWriterContext(), Close(), GetMd5(), GetStream(), HasStream(), and IsGood().

12.9.4.2 ZipOutputStream* OpenGPS::BinaryPointVectorWriterContext::m_Stream [private]

The stream buffer which makes BinaryPointVectorWriterContext::m_Buffer accessible through the more abstract std::ostream interface.

Referenced by BinaryPointVectorWriterContext(), Close(), GetStream(), HasStream(), and IsGood().

The documentation for this class was generated from the following files:

- binary_point_vector_writer_context.hxx
- binary_point_vector_writer_context.cxx

12.10 OpenGPS::Schemas::ISO5436_2::DataLinkType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.10.1 Detailed Description

Class corresponding to the DataLinkType schema type.

Defines a Link to a binary data file and a binary file containing the information about valid points.

MD5ChecksumPointData

Accessor and modifier functions for the MD5ChecksumPointData required element.

An MD5Checksum of the point data file like calculated by the unix command "md5sum". It consists of 32 hexadecimal digits. The binary representation is a 128 bit number.

- typedef ::xsd::cxx::tree::traits< MD5ChecksumPointData_type, wchar_t > MD5ChecksumPointData_traits

  Element traits type.

- typedef ::xml_schema::hex_binary MD5ChecksumPointData_type

  Element type.

- void MD5ChecksumPointData (::std::auto_ptr< MD5ChecksumPointData_type > p)

  Set the element value without copying.

- void MD5ChecksumPointData (const MD5ChecksumPointData_type &x)

  Set the element value.

- MD5ChecksumPointData_type & MD5ChecksumPointData ()

  Return a read-write reference to the element.

- const MD5ChecksumPointData_type & MD5ChecksumPointData () const

  Return a read-only (constant) reference to the element.

MD5ChecksumPointData

Accessor and modifier functions for the MD5ChecksumPointData required element.

An MD5Checksum of the point data file like calculated by the unix command "md5sum". It consists of 32 hexadecimal digits. The binary representation is a 128 bit number.

- typedef ::xsd::cxx::tree::traits< MD5ChecksumPointData_type, wchar_t > MD5ChecksumPointData_traits

  Element traits type.

- typedef ::xml_schema::hex_binary MD5ChecksumPointData_type

  Element type.

- void MD5ChecksumPointData (::std::auto_ptr< MD5ChecksumPointData_type > p)

  Set the element value without copying.

- void MD5ChecksumPointData (const MD5ChecksumPointData_type &x)

  Set the element value.

- MD5ChecksumPointData_type & MD5ChecksumPointData ()

  Return a read-write reference to the element.

- const MD5ChecksumPointData_type & MD5ChecksumPointData () const

  Return a read-only (constant) reference to the element.

MD5ChecksumValidPoints

Accessor and modifier functions for the MD5ChecksumValidPoints optional element.

An MD5Checksum of the valid points file like calculated by the unix command "md5sum". It consists of 32 hexadecimal digits. The binary representation is a 128 bit number.

- typedef ::xsd::cxx::tree::optional< MD5ChecksumValidPoints_type > MD5ChecksumValidPoints_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< MD5ChecksumValidPoints_type, wchar_t > MD5ChecksumValidPoints_traits

  Element traits type.

- typedef ::xml_schema::hex_binary MD5ChecksumValidPoints_type

  Element type.

- void MD5ChecksumValidPoints (::std::auto_ptr< MD5ChecksumValidPoints_type > p)

  Set the element value without copying.

- void MD5ChecksumValidPoints (const MD5ChecksumValidPoints_optional &x)

  Set the element value.

- void MD5ChecksumValidPoints (const MD5ChecksumValidPoints_type &x)

  Set the element value.

- MD5ChecksumValidPoints_optional & MD5ChecksumValidPoints ()

  Return a read-write reference to the element container.

- const MD5ChecksumValidPoints_optional & MD5ChecksumValidPoints () const

  Return a read-only (constant) reference to the element container.

MD5ChecksumValidPoints

Accessor and modifier functions for the MD5ChecksumValidPoints optional element.

An MD5Checksum of the valid points file like calculated by the unix command "md5sum". It consists of 32 hexadecimal digits. The binary representation is a 128 bit number.

- typedef ::xsd::cxx::tree::optional< MD5ChecksumValidPoints_type > MD5ChecksumValidPoints_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< MD5ChecksumValidPoints_type, wchar_t > MD5ChecksumValidPoints_traits

  Element traits type.

- typedef ::xml_schema::hex_binary MD5ChecksumValidPoints_type

  Element type.

- void MD5ChecksumValidPoints (::std::auto_ptr< MD5ChecksumValidPoints_type > p)

  Set the element value without copying.

- void MD5ChecksumValidPoints (const MD5ChecksumValidPoints_optional &x)

  Set the element value.

- void MD5ChecksumValidPoints (const MD5ChecksumValidPoints_type &x)

  Set the element value.

- MD5ChecksumValidPoints_optional & MD5ChecksumValidPoints ()

  Return a read-write reference to the element container.

- const MD5ChecksumValidPoints_optional & MD5ChecksumValidPoints () const

  Return a read-only (constant) reference to the element container.

PointDataLink

Accessor and modifier functions for the PointDataLink required element.

Relative filename in unix notation to a binary file with point data. Data can be specified directly in the xml file or with a link be stored in an external binary file. The Binary file has the same organisation as the DataList and has the datatypes specified in the axis description.

- typedef ::xsd::cxx::tree::traits< PointDataLink_type, wchar_t > PointDataLink_traits

  Element traits type.

- typedef ::xml_schema::string PointDataLink_type

  Element type.

- void PointDataLink (::std::auto_ptr< PointDataLink_type > p)

  Set the element value without copying.

- void PointDataLink (const PointDataLink_type &x)

  Set the element value.

- PointDataLink_type & PointDataLink ()

  Return a read-write reference to the element.

- const PointDataLink_type & PointDataLink () const

  Return a read-only (constant) reference to the element.

PointDataLink

Accessor and modifier functions for the PointDataLink required element.

Relative filename in unix notation to a binary file with point data. Data can be specified directly in the xml file or with a link be stored in an external binary file. The Binary file has the same organisation as the DataList and has the datatypes specified in the axis description.

- typedef ::xsd::cxx::tree::traits< PointDataLink_type, wchar_t > PointDataLink_traits

  Element traits type.

- typedef ::xml_schema::string PointDataLink_type

  Element type.

- void PointDataLink (::std::auto_ptr< PointDataLink_type > p)

  Set the element value without copying.

- void PointDataLink (const PointDataLink_type &x)

  Set the element value.

- PointDataLink_type & PointDataLink ()

  Return a read-write reference to the element.

- const PointDataLink_type & PointDataLink () const

  Return a read-only (constant) reference to the element.

ValidPointsLink

Accessor and modifier functions for the ValidPointsLink optional element.

Relative filename in unix notation to a binary file that contains a packed array of bools. Each element that is true corresponds to a valid data point in the binary point data file.

If this tag does not exist, all points are valid except for floating point numbers of the special value "NaN" (Not a Number).

- typedef ::xsd::cxx::tree::optional< ValidPointsLink_type > ValidPointsLink_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< ValidPointsLink_type, wchar_t > ValidPointsLink_traits

Element traits type.

- typedef ::xml_schema::string ValidPointsLink_type

  Element type.

- void ValidPointsLink (::std::auto_ptr< ValidPointsLink_type > p)

  Set the element value without copying.

- void ValidPointsLink (const ValidPointsLink_optional &x)

  Set the element value.

- void ValidPointsLink (const ValidPointsLink_type &x)

  Set the element value.

- ValidPointsLink_optional & ValidPointsLink ()

  Return a read-write reference to the element container.

- const ValidPointsLink_optional & ValidPointsLink () const

  Return a read-only (constant) reference to the element container.

ValidPointsLink

Accessor and modifier functions for the ValidPointsLink optional element.

Relative filename in unix notation to a binary file that contains a packed array of bools. Each element that is true corresponds to a valid data point in the binary point data file.

If this tag does not exist, all points are valid except for floating point numbers of the special value "NaN" (Not a Number).

- typedef ::xsd::cxx::tree::optional< ValidPointsLink_type > ValidPointsLink_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< ValidPointsLink_type, wchar_t > ValidPointsLink_traits

  Element traits type.

- typedef ::xml_schema::string ValidPointsLink_type

  Element type.

- void ValidPointsLink (::std::auto_ptr< ValidPointsLink_type > p)

  Set the element value without copying.

- void ValidPointsLink (const ValidPointsLink_optional &x)

  Set the element value.

- void ValidPointsLink (const ValidPointsLink_type &x)

  Set the element value.

- ValidPointsLink_optional & ValidPointsLink ()

  Return a read-write reference to the element container.

- const ValidPointsLink_optional & ValidPointsLink () const

  Return a read-only (constant) reference to the element container.

Constructors

- virtual DataLinkType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- DataLinkType (const DataLinkType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- DataLinkType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- DataLinkType (const PointDataLink_type &, const MD5ChecksumPointData_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Constructors

- virtual DataLinkType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- DataLinkType (const DataLinkType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- DataLinkType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- DataLinkType (const PointDataLink_type &, const MD5ChecksumPointData_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Protected Member Functions

- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)
- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)

Private Attributes

- ::xsd::cxx::tree::one< MD5ChecksumPointData_type > MD5ChecksumPointData_
- MD5ChecksumValidPoints_optional MD5ChecksumValidPoints_
- ::xsd::cxx::tree::one< PointDataLink_type > PointDataLink_
- ValidPointsLink_optional ValidPointsLink_

12.10.2 Member Typedef Documentation

12.10.2.1 typedef ::xsd::cxx::tree::traits< MD5ChecksumPointData_type, wchar_t > OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData_traits

Element traits type.

12.10.2.2 typedef ::xsd::cxx::tree::traits< MD5ChecksumPointData_type, wchar_t > OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData_traits

Element traits type.

12.10.2.3 typedef ::xml_schema::hex_binary OpenGPS::Schemas::ISO5436_-2::DataLinkType::MD5ChecksumPointData_type

Element type.

12.10.2.4 typedef ::xml_schema::hex_binary OpenGPS::Schemas::ISO5436_-2::DataLinkType::MD5ChecksumPointData_type

Element type.

12.10.2.5 typedef ::xsd::cxx::tree::optional< MD5ChecksumValidPoints_type > OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints_optional

Element optional container type.

12.10.2.6 typedef ::xsd::cxx::tree::optional< MD5ChecksumValidPoints_type > OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints_optional

Element optional container type.

12.10.2.7 typedef ::xsd::cxx::tree::traits< MD5ChecksumValidPoints_type, wchar_t > OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints_traits

Element traits type.

12.10.2.8 typedef ::xsd::cxx::tree::traits< MD5ChecksumValidPoints_type, wchar_t > OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints_traits

Element traits type.

12.10.2.9 typedef ::xml_schema::hex_binary OpenGPS::Schemas::ISO5436_-2::DataLinkType::MD5ChecksumValidPoints_type

Element type.

12.10.2.10 typedef ::xml_schema::hex_binary OpenGPS::Schemas::ISO5436_-2::DataLinkType::MD5ChecksumValidPoints_type

Element type.

12.10.2.11 typedef ::xsd::cxx::tree::traits< PointDataLink_type, wchar_t > OpenGPS::Schemas::ISO5436_2::DataLinkType::PointDataLink_traits

Element traits type.

12.10.2.12 typedef ::xsd::cxx::tree::traits< PointDataLink_type, wchar_t > OpenGPS::Schemas::ISO5436_2::DataLinkType::PointDataLink_traits

Element traits type.

12.10.2.13 typedef ::xml_schema::string OpenGPS::Schemas::ISO5436_-2::DataLinkType::PointDataLink_type

Element type.

12.10.2.14 typedef ::xml_schema::string OpenGPS::Schemas::ISO5436_-2::DataLinkType::PointDataLink_type

Element type.

12.10.2.15 typedef ::xsd::cxx::tree::optional< ValidPointsLink_type > OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink_optional

Element optional container type.

12.10.2.16 typedef ::xsd::cxx::tree::optional< ValidPointsLink_type > OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink_optional

Element optional container type.

12.10.2.17 typedef ::xsd::cxx::tree::traits< ValidPointsLink_type, wchar_t > OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink_traits

Element traits type.

12.10.2.18 typedef ::xsd::cxx::tree::traits< ValidPointsLink_type, wchar_t > OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink_traits

Element traits type.

12.10.2.19 typedef ::xml_schema::string OpenGPS::Schemas::ISO5436_-2::DataLinkType::ValidPointsLink_type

Element type.

12.10.2.20 typedef ::xml_schema::string OpenGPS::Schemas::ISO5436_-2::DataLinkType::ValidPointsLink_type

Element type.

12.10.3 Constructor & Destructor Documentation

12.10.3.1 OpenGPS::Schemas::ISO5436_2::DataLinkType::DataLinkType (const PointDataLink_type & *PointDataLink*, const MD5ChecksumPointData_type & *MD5ChecksumPointData*)

Construct an instance from the ultimate base and initializers for required elements and attributes.

Referenced by _clone().

12.10.3.2 OpenGPS::Schemas::ISO5436_2::DataLinkType::DataLinkType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

References parse().

Here is the call graph for this function:

12.10.3.3 OpenGPS::Schemas::ISO5436_2::DataLinkType::DataLinkType (const DataLinkType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.10.3.4 OpenGPS::Schemas::ISO5436_2::DataLinkType::DataLinkType (const PointDataLink_type &, const MD5ChecksumPointData_type &)

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.10.3.5 OpenGPS::Schemas::ISO5436_2::DataLinkType::DataLinkType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.10.3.6 OpenGPS::Schemas::ISO5436_2::DataLinkType::DataLinkType (const DataLinkType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.10.4 Member Function Documentation

12.10.4.1 virtual DataLinkType* OpenGPS::Schemas::ISO5436_2::DataLinkType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.10.4.2 DataLinkType * OpenGPS::Schemas::ISO5436_2::DataLinkType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References DataLinkType().

Here is the call graph for this function:

12.10.4.3 void OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData (::std::auto_ptr< MD5ChecksumPointData_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.10.4.4 void OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData (const MD5ChecksumPointData_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.10.4.5 MD5ChecksumPointData_type& OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.10.4.6 const MD5ChecksumPointData_type& OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.10.4.7 void OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData (::std::auto_ptr< MD5ChecksumPointData_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.10.4.8 void OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData (const MD5ChecksumPointData_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.10.4.9 MD5ChecksumPointData_type& OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.10.4.10 const MD5ChecksumPointData_type& OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.10.4.11 void OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints (::std::auto_ptr< MD5ChecksumValidPoints_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.10.4.12 void OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints (const MD5ChecksumValidPoints_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.10.4.13 void OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints (const MD5ChecksumValidPoints_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.10.4.14 MD5ChecksumValidPoints_optional& OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.10.4.15 const MD5ChecksumValidPoints_optional& OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

12.10.4.16 void OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints (::std::auto_ptr< MD5ChecksumValidPoints_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.10.4.17 void OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints (const MD5ChecksumValidPoints_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.10.4.18 void OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints (const MD5ChecksumValidPoints_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.10.4.19 MD5ChecksumValidPoints_optional& OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.10.4.20 const MD5ChecksumValidPoints_optional& OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.10.4.21 void OpenGPS::Schemas::ISO5436_2::DataLinkType::parse (::xsd::cxx::xml::dom::parser< wchar_t > &, ::xml_schema::flags) [protected]

12.10.4.22 void OpenGPS::Schemas::ISO5436_2::DataLinkType::parse (::xsd::cxx::xml::dom::parser< wchar_t > & *p*, ::xml_schema::flags *f*) [protected]

References MD5ChecksumPointData(), MD5ChecksumPointData_, MD5ChecksumValidPoints(), n, PointDataLink(), PointDataLink_, and ValidPointsLink().

Referenced by DataLinkType().

Here is the call graph for this function:

12.10.4.23 void OpenGPS::Schemas::ISO5436_2::DataLinkType::PointDataLink (::std::auto_ptr< PointDataLink_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.10.4.24 void OpenGPS::Schemas::ISO5436_2::DataLinkType::PointDataLink (const PointDataLink_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.10.4.25 PointDataLink_type& OpenGPS::Schemas::ISO5436_2::DataLinkType::PointDataLink ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.10.4.26 const PointDataLink_type& OpenGPS::Schemas::ISO5436_2::DataLinkType::PointDataLink () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.10.4.27 void OpenGPS::Schemas::ISO5436_2::DataLinkType::PointDataLink (::std::auto_ptr< PointDataLink_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.10.4.28 void OpenGPS::Schemas::ISO5436_2::DataLinkType::PointDataLink (const PointDataLink_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.10.4.29 PointDataLink_type& OpenGPS::Schemas::ISO5436_2::DataLinkType::PointDataLink ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.10.4.30 const PointDataLink_type& OpenGPS::Schemas::ISO5436_2::DataLinkType::PointDataLink () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.10.4.31 void OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink (::std::auto_ptr< ValidPointsLink_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.10.4.32 void OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink (const ValidPointsLink_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.10.4.33 void OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink (const ValidPointsLink_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.10.4.34 ValidPointsLink_optional& OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.10.4.35 const ValidPointsLink_optional& OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

12.10.4.36 void OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink (::std::auto_ptr< ValidPointsLink_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.10.4.37 void OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink (const ValidPointsLink_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.10.4.38 void OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink (const ValidPointsLink_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.10.4.39 ValidPointsLink_optional& OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.10.4.40 const ValidPointsLink_optional& OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.10.5 Member Data Documentation

12.10.5.1 xsd::cxx::tree::one< MD5ChecksumPointData_type > OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData_ [private]

Referenced by parse().

12.10.5.2 MD5ChecksumValidPoints_optional OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints_ [private]

12.10.5.3 xsd::cxx::tree::one< PointDataLink_type > OpenGPS::Schemas::ISO5436_2::DataLinkType::PointDataLink_ [private]

Referenced by parse().

12.10.5.4 ValidPointsLink_optional OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink_ [private]

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.11 OpenGPS::Schemas::ISO5436_2::DataListType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.11.1 Detailed Description

Class corresponding to the DataListType schema type.

The datalist contains the point coordinates in ASCII. A list can by definition not contain invalid points, because it does not define a topological neighbourship. A list is allways ab unsorted list of 3D-points.

Datum

Accessor and modifier functions for the Datum sequence element.

Datum contains a ";" separated list of X,Y,Z floating point or integer coordinates. An empty Datum tag defines an invalid data point.

- typedef Datum_sequence::const_iterator Datum_const_iterator

  Element constant iterator type.

- typedef Datum_sequence::iterator Datum_iterator

  Element iterator type.

- typedef ::xsd::cxx::tree::sequence< Datum_type > Datum_sequence

  Element sequence container type.

- typedef ::xsd::cxx::tree::traits< Datum_type, wchar_t > Datum_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::Datum Datum_type

  Element type.

- void Datum (const Datum_sequence &s)

  Copy elements from a given sequence.

- Datum_sequence & Datum ()

Return a read-write reference to the element sequence.

- const Datum_sequence & Datum () const

  Return a read-only (constant) reference to the element sequence.

Datum

Accessor and modifier functions for the Datum sequence element.

Datum contains a ";" separated list of X,Y,Z floating point or integer coordinates. An empty Datum tag defines an invalid data point.

- typedef Datum_sequence::const_iterator Datum_const_iterator

  Element constant iterator type.

- typedef Datum_sequence::iterator Datum_iterator

  Element iterator type.

- typedef ::xsd::cxx::tree::sequence< Datum_type > Datum_sequence

  Element sequence container type.

- typedef ::xsd::cxx::tree::traits< Datum_type, wchar_t > Datum_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::Datum Datum_type

  Element type.

- void Datum (const Datum_sequence &s)

  Copy elements from a given sequence.

- Datum_sequence & Datum ()

  Return a read-write reference to the element sequence.

- const Datum_sequence & Datum () const

  Return a read-only (constant) reference to the element sequence.

Constructors

- virtual DataListType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- DataListType (const DataListType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- DataListType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- DataListType ()

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Constructors

- virtual DataListType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- DataListType (const DataListType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- DataListType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- DataListType ()

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Protected Member Functions

- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)
- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)

Private Attributes

- Datum_sequence Datum_

12.11.2 Member Typedef Documentation

12.11.2.1 typedef Datum_sequence::const_iterator OpenGPS::Schemas::ISO5436_2::DataListType::Datum_const_iterator

Element constant iterator type.

12.11.2.2 typedef Datum_sequence::const_iterator OpenGPS::Schemas::ISO5436_2::DataListType::Datum_const_iterator

Element constant iterator type.

12.11.2.3 typedef Datum_sequence::iterator OpenGPS::Schemas::ISO5436_2::DataListType::Datum_iterator

Element iterator type.

12.11.2.4 typedef Datum_sequence::iterator OpenGPS::Schemas::ISO5436_2::DataListType::Datum_iterator

Element iterator type.

12.11.2.5 typedef ::xsd::cxx::tree::sequence< Datum_type > OpenGPS::Schemas::ISO5436_2::DataListType::Datum_sequence

Element sequence container type.

12.11.2.6 typedef ::xsd::cxx::tree::sequence< Datum_type > OpenGPS::Schemas::ISO5436_2::DataListType::Datum_sequence

Element sequence container type.

12.11.2.7 typedef ::xsd::cxx::tree::traits< Datum_type, wchar_t > OpenGPS::Schemas::ISO5436_2::DataListType::Datum_traits

Element traits type.

12.11.2.8 typedef ::xsd::cxx::tree::traits< Datum_type, wchar_t > OpenGPS::Schemas::ISO5436_2::DataListType::Datum_traits

Element traits type.

12.11.2.9 typedef ::OpenGPS::Schemas::ISO5436_2::Datum OpenGPS::Schemas::ISO5436_2::DataListType::Datum_type

Element type.

12.11.2.10 typedef ::OpenGPS::Schemas::ISO5436_2::Datum OpenGPS::Schemas::ISO5436_2::DataListType::Datum_type

Element type.

12.11.3 Constructor & Destructor Documentation

12.11.3.1 OpenGPS::Schemas::ISO5436_2::DataListType::DataListType ()

Construct an instance from the ultimate base and initializers for required elements and attributes.

Referenced by _clone().

12.11.3.2 OpenGPS::Schemas::ISO5436_2::DataListType::DataListType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

References parse().

Here is the call graph for this function:

12.11.3.3 OpenGPS::Schemas::ISO5436_2::DataListType::DataListType (const DataListType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.11.3.4 OpenGPS::Schemas::ISO5436_2::DataListType::DataListType ()

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.11.3.5 OpenGPS::Schemas::ISO5436_2::DataListType::DataListType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.11.3.6 OpenGPS::Schemas::ISO5436_2::DataListType::DataListType (const DataListType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.11.4 Member Function Documentation

12.11.4.1 virtual DataListType* OpenGPS::Schemas::ISO5436_2::DataListType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.11.4.2 DataListType * OpenGPS::Schemas::ISO5436_2::DataListType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References DataListType().

Here is the call graph for this function:

12.11.4.3 void OpenGPS::Schemas::ISO5436_2::DataListType::Datum (const Datum_sequence & *s*)

Copy elements from a given sequence.

Parameters:

s A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

12.11.4.4 Datum_sequence& OpenGPS::Schemas::ISO5436_2::DataListType::Datum ()

Return a read-write reference to the element sequence.

Returns:

A reference to the sequence container.

12.11.4.5 const Datum_sequence& OpenGPS::Schemas::ISO5436_2::DataListType::Datum () const

Return a read-only (constant) reference to the element sequence.

Returns:

A constant reference to the sequence container.

12.11.4.6 void OpenGPS::Schemas::ISO5436_2::DataListType::Datum (const Datum_sequence & *s*)

Copy elements from a given sequence.

Parameters:

s A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

12.11.4.7 Datum_sequence& OpenGPS::Schemas::ISO5436_2::DataListType::Datum ()

Return a read-write reference to the element sequence.

Returns:

A reference to the sequence container.

12.11.4.8 const Datum_sequence& OpenGPS::Schemas::ISO5436_2::DataListType::Datum () const

Return a read-only (constant) reference to the element sequence.

Returns:

A constant reference to the sequence container.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_-2::operator==(), and parse().

12.11.4.9 void OpenGPS::Schemas::ISO5436_2::DataListType::parse (::xsd::cxx::xml::dom::parser< wchar_t > &, ::xml_schema::flags) [protected]

12.11.4.10 void OpenGPS::Schemas::ISO5436_2::DataListType::parse (::xsd::cxx::xml::dom::parser< wchar_t > & *p*, ::xml_schema::flags *f*) [protected]

References Datum(), and n.

Referenced by DataListType().

Here is the call graph for this function:

12.11.5 Member Data Documentation

12.11.5.1 Datum_sequence OpenGPS::Schemas::ISO5436_2::DataListType::Datum_- [private]

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.12 OpenGPS::DataPoint Class Reference

`#include <data_point.hxx>`

Inheritance diagram for OpenGPS::DataPoint:

12.12.1 Detailed Description

Typesafe representation of a single data point value.

In an OpenGPS::PointVector every component of that three-vector is accessible as its own OpenGPS::DataPoint instance.

Remarks:

An instance of OpenGPS::DataPoint cannot be created of its own. Indirectly a handle of this type can be accessed through an OpenGPS::PointVector object.

Public Member Functions

- virtual OGPS_Double Get () const =0 throw (...)

  Gets the stored value.

- virtual void Get (OGPS_Double *const value) const =0 throw (...)

  Gets the stored value.

- virtual void Get (OGPS_Float *const value) const =0 throw (...)

  Gets the stored value.

- virtual void Get (OGPS_Int32 *const value) const =0 throw (...)

  Gets the stored value.

- virtual void Get (OGPS_Int16 *const value) const =0 throw (...)
  Gets the stored value.
- virtual OGPS_DataPointType GetPointType () const =0 throw (...)
  Gets type information of the current value stored within this data point.
- virtual OGPS_Boolean IsValid () const =0 throw (...)
  Asks whether there is a point value stored within this instance currently.
- virtual void Reset ()=0 throw (...)
  Resets this instance to its initial state.
- virtual void Set (const DataPoint &src)=0 throw (...)
  Stores a new value.
- virtual void Set (const OGPS_Double value)=0 throw (...)
  Stores a new value.
- virtual void Set (const OGPS_Float value)=0 throw (...)
  Stores a new value.
- virtual void Set (const OGPS_Int32 value)=0 throw (...)
  Stores a new value.
- virtual void Set (const OGPS_Int16 value)=0 throw (...)
  Stores a new value.
- virtual ~DataPoint ()=0
  Destructs this object.

Protected Member Functions

- DataPoint ()=0
  Creates and initializes a new object.

Private Member Functions

- DataPoint (const DataPoint &src)
  The copy-ctor is not implemented.
- DataPoint & operator= (const DataPoint &src)
  The assignment-operator is not implemented.

12.12.2 Constructor & Destructor Documentation

12.12.2.1 DataPoint::DataPoint () `[protected, pure virtual]`

Creates and initializes a new object.

12.12.2.2 DataPoint::~DataPoint () [pure virtual]

Destructs this object.

12.12.2.3 OpenGPS::DataPoint::DataPoint (const DataPoint & *src*) [private]

The copy-ctor is not implemented.

This prevents its usage.

12.12.3 Member Function Documentation

12.12.3.1 virtual OGPS_Double OpenGPS::DataPoint::Get () const throw (...) [pure virtual]

Gets the stored value.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns the stored value or 0.0 if this data point is empty (DataPoint::GetPointType returns OGPS_-MissingPointType in this case).

Implemented in OpenGPS::DataPointImpl, and OpenGPS::PointVectorProxy::DataPointProxy.

12.12.3.2 virtual void OpenGPS::DataPoint::Get (OGPS_Double *const *value*) const throw (...) [pure virtual]

Gets the stored value.

Remarks:

If the current type does not equal OGPS_Double, the behavior is undefined.

See also:

DataPoint::GetPointType

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Destination for the currently stored value.

Implemented in OpenGPS::DataPointImpl, and OpenGPS::PointVectorProxy::DataPointProxy.

12.12.3.3 virtual void OpenGPS::DataPoint::Get (OGPS_Float *const *value*) const throw (...) [pure virtual]

Gets the stored value.

Remarks:

If the current type does not equal OGPS_Float, the behavior is undefined.

See also:

DataPoint::GetPointType

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Destination for the currently stored value.

Implemented in OpenGPS::DataPointImpl, and OpenGPS::PointVectorProxy::DataPointProxy.

12.12.3.4 virtual void OpenGPS::DataPoint::Get (OGPS_Int32 *const *value*) const throw (...) [pure virtual]

Gets the stored value.

Remarks:

If the current type does not equal OGPS_Int32, the behavior is undefined.

See also:

DataPoint::GetPointType

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Destination for the currently stored value.

Implemented in OpenGPS::DataPointImpl, and OpenGPS::PointVectorProxy::DataPointProxy.

12.12.3.5 virtual void OpenGPS::DataPoint::Get (OGPS_Int16 *const *value*) const throw (...) [pure virtual]

Gets the stored value.

Remarks:

If the current type does not equal OGPS_Int16, the behavior is undefined.

See also:

DataPoint::GetPointType

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Destination for the currently stored value.

Implemented in OpenGPS::DataPointImpl, and OpenGPS::PointVectorProxy::DataPointProxy.

Referenced by OpenGPS::PointVector::GetX(), OpenGPS::PointVector::GetXYZ(), OpenGPS::PointVector::GetY(), and OpenGPS::PointVector::GetZ().

12.12.3.6 virtual OGPS_DataPointType OpenGPS::DataPoint::GetPointType () const throw (...) [pure virtual]

Gets type information of the current value stored within this data point.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns OGPS_MissingPointType if this instance does not store any value at all.

Implemented in OpenGPS::DataPointImpl, and OpenGPS::PointVectorProxy::DataPointProxy.

12.12.3.7 virtual OGPS_Boolean OpenGPS::DataPoint::IsValid () const throw (...) [pure virtual]

Asks whether there is a point value stored within this instance currently.

See also:

DataPoint::GetPointType

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns TRUE if a value is stored, FALSE otherwise.

Implemented in OpenGPS::DataPointImpl, and OpenGPS::PointVectorProxy::DataPointProxy.

Referenced by OpenGPS::PointVector::IsValid(), and OpenGPS::PointVectorProxy::Set().

12.12.3.8 DataPoint& OpenGPS::DataPoint::operator= (const DataPoint & *src*) [private]

The assignment-operator is not implemented.

This prevents its usage.

12.12.3.9 virtual void OpenGPS::DataPoint::Reset () throw (...) [pure virtual]

Resets this instance to its initial state.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Implemented in OpenGPS::DataPointImpl, and OpenGPS::PointVectorProxy::DataPointProxy.

12.12.3.10 virtual void OpenGPS::DataPoint::Set (const DataPoint & *src*) throw (...) [pure virtual]

Stores a new value.

Copies its new entry from another DataPoint instance.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

src Object to copy from.

Implemented in OpenGPS::DataPointImpl, and OpenGPS::PointVectorProxy::DataPointProxy.

12.12.3.11 virtual void OpenGPS::DataPoint::Set (const OGPS_Double *value*) throw (...) [pure virtual]

Stores a new value.

Also adjusts the current type information reflecting this set operation, if necessary.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value The new value to be stored.

Implemented in OpenGPS::DataPointImpl, and OpenGPS::PointVectorProxy::DataPointProxy.

12.12.3.12 virtual void OpenGPS::DataPoint::Set (const OGPS_Float *value*) throw (...) [pure virtual]

Stores a new value.

Also adjusts the current type information reflecting this set operation, if necessary.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value The new value to be stored.

Implemented in OpenGPS::DataPointImpl, and OpenGPS::PointVectorProxy::DataPointProxy.

12.12.3.13 virtual void OpenGPS::DataPoint::Set (const OGPS_Int32 *value*) throw (...) [pure virtual]

Stores a new value.

Also adjusts the current type information reflecting this set operation, if necessary.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value The new value to be stored.

Implemented in OpenGPS::DataPointImpl, and OpenGPS::PointVectorProxy::DataPointProxy.

12.12.3.14 virtual void OpenGPS::DataPoint::Set (const OGPS_Int16 *value*) throw (...) [pure virtual]

Stores a new value.

Also adjusts the current type information reflecting this set operation, if necessary.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value The new value to be stored.

Implemented in OpenGPS::DataPointImpl, and OpenGPS::PointVectorProxy::DataPointProxy.

Referenced by OpenGPS::PointVectorProxy::Get(), OpenGPS::PointVectorProxy::Set(), OpenGPS::PointVector::Set(), OpenGPS::PointVector::SetX(), OpenGPS::PointVector::SetY(), and OpenGPS::PointVector::SetZ().

The documentation for this class was generated from the following files:

- data_point.hxx
- data_point_impl.cxx

12.13 OpenGPS::DataPointImpl Class Reference

```
#include <data_point_impl.hxx>
```

Inheritance diagram for OpenGPS::DataPointImpl:

Collaboration diagram for OpenGPS::DataPointImpl:

12.13.1 Detailed Description

A straightforward implementation of OpenGPS::DataPoint.

This mainly supports OpenGPS::PointVector where typesafe storage of arbitrary point data must be maintained for its vector components.

Public Member Functions

- DataPointImpl ()

  Creates a new instance.

- virtual OGPS_Double Get () const throw (...)

  Gets the stored value.

- virtual void Get (OGPS_Double *const value) const throw (...)

  Gets the stored value.

- virtual void Get (OGPS_Float *const value) const throw (...)

  Gets the stored value.

- virtual void Get (OGPS_Int32 *const value) const throw (...)

  Gets the stored value.

- virtual void Get (OGPS_Int16 *const value) const throw (...)

  Gets the stored value.

- virtual OGPS_DataPointType GetPointType () const throw (...)

  Gets type information of the current value stored within this data point.

- virtual OGPS_Boolean IsValid () const throw (...)

  Asks whether there is a point value stored within this instance currently.

- virtual void Set (const DataPoint &src) throw (...)

  Stores a new value.

- virtual void Set (const OGPS_Double value) throw (...)

  Stores a new value.

- virtual void Set (const OGPS_Float value) throw (...)

  Stores a new value.

- virtual void Set (const OGPS_Int32 value) throw (...)

  Stores a new value.

- virtual void Set (const OGPS_Int16 value) throw (...)

  Stores a new value.

- virtual ~DataPointImpl ()

  Destroys this instance.

Protected Member Functions

- virtual void Reset () throw (...)

  Resets this instance to its initial state.

Private Attributes

- OGPS_DataPointType m_Type

  This tag defines which value type is currently valid within DataPointImpl::m_Value.

- OGPS_DataPointValue m_Value

  Typesafe storage for every possible data type.

12.13.2 Constructor & Destructor Documentation

12.13.2.1 DataPointImpl::DataPointImpl ()

Creates a new instance.

References OGPS_DataPointValue::doubleValue, m_Type, m_Value, and OGPS_MissingPointType.

12.13.2.2 DataPointImpl::~DataPointImpl () [virtual]

Destroys this instance.

12.13.3 Member Function Documentation

12.13.3.1 OGPS_Double DataPointImpl::Get () const throw (...) [virtual]

Gets the stored value.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns the stored value or 0.0 if this data point is empty (DataPoint::GetPointType returns OGPS_-MissingPointType in this case).

Implements OpenGPS::DataPoint.

References _EX_T, OGPS_DataPointValue::doubleValue, OGPS_DataPointValue::floatValue, OGPS_-DataPointValue::int16Value, OGPS_DataPointValue::int32Value, m_Type, m_Value, OGPS_-DoublePointType, OGPS_ExInvalidOperation, OGPS_FloatPointType, OGPS_Int16PointType, OGPS_-Int32PointType, and OGPS_MissingPointType.

12.13.3.2 void DataPointImpl::Get (OGPS_Double *const *value*) const throw (...) [virtual]

Gets the stored value.

Remarks:

If the current type does not equal OGPS_Double, the behavior is undefined.

See also:

DataPoint::GetPointType

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Destination for the currently stored value.

Implements OpenGPS::DataPoint.

References _ASSERT, _EX_T, OGPS_DataPointValue::doubleValue, m_Type, m_Value, OGPS_-DoublePointType, and OGPS_ExInvalidOperation.

12.13.3.3 void DataPointImpl::Get (OGPS_Float *const *value*) const throw (...) [virtual]

Gets the stored value.

Remarks:

If the current type does not equal OGPS_Float, the behavior is undefined.

See also:

DataPoint::GetPointType

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Destination for the currently stored value.

Implements OpenGPS::DataPoint.

References _ASSERT, _EX_T, OGPS_DataPointValue::floatValue, m_Type, m_Value, OGPS_-ExInvalidOperation, and OGPS_FloatPointType.

12.13.3.4 void DataPointImpl::Get (OGPS_Int32 *const *value*) const throw (...) [virtual]

Gets the stored value.

Remarks:

If the current type does not equal OGPS_Int32, the behavior is undefined.

See also:

DataPoint::GetPointType

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Destination for the currently stored value.

Implements OpenGPS::DataPoint.

References _ASSERT, _EX_T, OGPS_DataPointValue::int32Value, m_Type, m_Value, OGPS_-ExInvalidOperation, and OGPS_Int32PointType.

12.13.3.5 void DataPointImpl::Get (OGPS_Int16 *const *value*) const throw (...) [virtual]

Gets the stored value.

Remarks:

If the current type does not equal OGPS_Int16, the behavior is undefined.

See also:

DataPoint::GetPointType

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Destination for the currently stored value.

Implements OpenGPS::DataPoint.

References _ASSERT, _EX_T, OGPS_DataPointValue::int16Value, m_Type, m_Value, OGPS_-ExInvalidOperation, and OGPS_Int16PointType.

12.13.3.6 OGPS_DataPointType DataPointImpl::GetPointType () const throw (...) [virtual]

Gets type information of the current value stored within this data point.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns OGPS_MissingPointType if this instance does not store any value at all.

Implements OpenGPS::DataPoint.

References m_Type.

12.13.3.7 OGPS_Boolean DataPointImpl::IsValid () const throw (...) [virtual]

Asks whether there is a point value stored within this instance currently.

See also:

DataPoint::GetPointType

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns TRUE if a value is stored, FALSE otherwise.

Implements OpenGPS::DataPoint.

References m_Type, and OGPS_MissingPointType.

12.13.3.8 void DataPointImpl::Reset () throw (...) [protected, virtual]

Resets this instance to its initial state.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Implements OpenGPS::DataPoint.

References OGPS_DataPointValue::doubleValue, m_Type, m_Value, and OGPS_MissingPointType.

Referenced by Set().

12.13.3.9 void DataPointImpl::Set (const DataPoint & *src*) throw (...) [virtual]

Stores a new value.

Copies its new entry from another DataPoint instance.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

src Object to copy from.

Implements OpenGPS::DataPoint.

References _EX_T, OGPS_DoublePointType, OGPS_ExInvalidArgument, OGPS_FloatPointType, OGPS_Int16PointType, OGPS_Int32PointType, OGPS_MissingPointType, Reset(), and Set().

Here is the call graph for this function:

12.13.3.10 void DataPointImpl::Set (const OGPS_Double *value*) throw (...) [virtual]

Stores a new value.

Also adjusts the current type information reflecting this set operation, if necessary.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value The new value to be stored.

Implements OpenGPS::DataPoint.

References OGPS_DataPointValue::doubleValue, m_Type, m_Value, and OGPS_DoublePointType.

12.13.3.11 void DataPointImpl::Set (const OGPS_Float *value*) throw (...) [virtual]

Stores a new value.

Also adjusts the current type information reflecting this set operation, if necessary.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value The new value to be stored.

Implements OpenGPS::DataPoint.

References OGPS_DataPointValue::floatValue, m_Type, m_Value, and OGPS_FloatPointType.

12.13.3.12 void DataPointImpl::Set (const OGPS_Int32 *value*) throw (...) `[virtual]`

Stores a new value.

Also adjusts the current type information reflecting this set operation, if necessary.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value The new value to be stored.

Implements OpenGPS::DataPoint.

References OGPS_DataPointValue::int32Value, m_Type, m_Value, and OGPS_Int32PointType.

12.13.3.13 void DataPointImpl::Set (const OGPS_Int16 *value*) throw (...) `[virtual]`

Stores a new value.

Also adjusts the current type information reflecting this set operation, if necessary.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value The new value to be stored.

Implements OpenGPS::DataPoint.

References OGPS_DataPointValue::int16Value, m_Type, m_Value, and OGPS_Int16PointType.

Referenced by Set().

12.13.4 Member Data Documentation

12.13.4.1 OGPS_DataPointType OpenGPS::DataPointImpl::m_Type `[private]`

This tag defines which value type is currently valid within DataPointImpl::m_Value.

This restricts access to the only safe element of DataPointImpl::m_Value.

Referenced by DataPointImpl(), Get(), GetPointType(), IsValid(), Reset(), and Set().

12.13.4.2 OGPS_DataPointValue OpenGPS::DataPointImpl::m_Value [private]

Typesafe storage for every possible data type.

The stored value of this data point.

See also:

DataPointImpl::m_Type

Referenced by DataPointImpl(), Get(), Reset(), and Set().

The documentation for this class was generated from the following files:

- data_point_impl.hxx
- data_point_impl.cxx

12.14 OpenGPS::DataPointParser Class Reference

`#include <data_point_parser.hxx>`

Inheritance diagram for OpenGPS::DataPointParser:

12.14.1 Detailed Description

Interface for reading/writing point data.

Reads/writes point data from/to specified media. This interface is implemented for every possible type of point data (OGPS_DataPointType).

Public Member Functions

- virtual void Read (PointVectorReaderContext &context, DataPoint &value)=0 throw (...)

  Reads point data from a given context/media.

- virtual void Write (PointVectorWriterContext &context, const DataPoint &value)=0 throw (...)

  Writes point data to a given context/media.

- virtual ~DataPointParser ()

  Destroys this instance.

Protected Member Functions

- DataPointParser ()

  Creates a new instance.

12.14.2 Constructor & Destructor Documentation

12.14.2.1 DataPointParser::~DataPointParser () [virtual]

Destroys this instance.

12.14.2.2 DataPointParser::DataPointParser () [protected]

Creates a new instance.

12.14.3 Member Function Documentation

12.14.3.1 virtual void OpenGPS::DataPointParser::Read (PointVectorReaderContext & *context*, DataPoint & *value*) throw (...) [pure virtual]

Reads point data from a given context/media.

Using the appropriate access method of the given implementation of OpenGPS::PointVectorReaderContext the current point value read gets stored in a OpenGPS::DataPoint instance. This operation is typesafe.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

context Provides read access to the media.

value Buffer where the point data read gets stored.

Implemented in OpenGPS::DoubleDataPointParser, OpenGPS::FloatDataPointParser, OpenGPS::Int16DataPointParser, OpenGPS::Int32DataPointParser, and OpenGPS::MissingDataPointParser.

Referenced by OpenGPS::PointVectorParser::Read().

12.14.3.2 virtual void OpenGPS::DataPointParser::Write (PointVectorWriterContext & *context*, const DataPoint & *value*) throw (...) [pure virtual]

Writes point data to a given context/media.

Using the appropriate access method of the given implementation of OpenGPS::PointVectorWriterContext the point value currently stored in the OpenGPS::DataPoint instance gets written to the media. This operation is typesafe and will fail if the type of point data stored in the value object does not exacly match the current implementation of the OpenGPS::DataPointParser interface.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

context Provides write access to the media.

value Buffer where the point data to be written is stored.

Implemented in OpenGPS::DoubleDataPointParser, OpenGPS::FloatDataPointParser, OpenGPS::Int16DataPointParser, OpenGPS::Int32DataPointParser, and OpenGPS::MissingDataPointParser.

Referenced by OpenGPS::PointVectorParser::Write().

The documentation for this class was generated from the following files:

- data_point_parser.hxx
- int16_data_point_parser.cxx

12.15 OpenGPS::Schemas::ISO5436_2::DataType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.15.1 Detailed Description

Enumeration class corresponding to the DataType schema type.

Public Types

- enum value {
  I, L, F, D,
  I, L, F, D }
  Underlying enum type.
- enum value {
  I, L, F, D,
  I, L, F, D }
  Underlying enum type.

Public Member Functions

- virtual DataType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const
  Copy the object polymorphically.
- virtual DataType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const
  Copy the object polymorphically.
- DataType (const DataType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)
  Copy constructor.
- DataType (const ::std::wstring &s, const ::xercesc::DOMElement *e,::xml_schema::flags f=0,::xml_schema::type *c=0)
  Construct an instance from a string fragment.

- DataType (const ::xercesc::DOMAttr &a,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM attribute.

- DataType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- DataType (const ::xml_schema::token &v)

  Construct an instance from the base value.

- DataType (value v)

  Construct an instance from the underlying enum value.

- DataType (const DataType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- DataType (const ::std::wstring &s, const ::xercesc::DOMElement *e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a string fragment.

- DataType (const ::xercesc::DOMAttr &a,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM attribute.

- DataType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- DataType (const ::xml_schema::token &v)

  Construct an instance from the base value.

- DataType (value v)

  Construct an instance from the underlying enum value.

- virtual operator value () const

  Implicit conversion operator to the underlying enum value.

- virtual operator value () const

  Implicit conversion operator to the underlying enum value.

- DataType & operator= (value v)

  Assign the underlying enum value.

- DataType & operator= (value v)

  Assign the underlying enum value.

Static Public Attributes

- static const value _xsd_DataType_indexes_ [4]
- static const wchar_t *const _xsd_DataType_literals_ [4]

Protected Member Functions

- value _xsd_DataType_convert () const
- value _xsd_DataType_convert () const

12.15.2 Member Enumeration Documentation

12.15.2.1 enum OpenGPS::Schemas::ISO5436_2::DataType::value

Underlying enum type.

Enumerator:

I

L

F

D

I

L

F

D

12.15.2.2 enum OpenGPS::Schemas::ISO5436_2::DataType::value

Underlying enum type.

Enumerator:

I

L

F

D

I

L

F

D

12.15.3 Constructor & Destructor Documentation

12.15.3.1 OpenGPS::Schemas::ISO5436_2::DataType::DataType (value *v*)

Construct an instance from the underlying enum value.

Parameters:

v A enum value.

Referenced by _clone().

12.15.3.2 OpenGPS::Schemas::ISO5436_2::DataType::DataType (const ::xml_schema::token & *v*)

Construct an instance from the base value.

Parameters:

v A base value.

12.15.3.3 OpenGPS::Schemas::ISO5436_2::DataType::DataType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

e A DOM element to extract the data from.

f Flags to construct the new instance with.

c A pointer to the object that will contain the new instance.

References _xsd_DataType_convert().

Here is the call graph for this function:

12.15.3.4 OpenGPS::Schemas::ISO5436_2::DataType::DataType (const ::xercesc::DOMAttr & *a*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM attribute.

Parameters:

a A DOM attribute to extract the data from.

f Flags to construct the new instance with.

c A pointer to the object that will contain the new instance.

References _xsd_DataType_convert().

Here is the call graph for this function:

12.15.3.5 OpenGPS::Schemas::ISO5436_2::DataType::DataType (const ::std::wstring & *s*, const ::xercesc::DOMElement * *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a string fragment.

Parameters:

s A string fragment to extract the data from.

e A DOM element containing the string fragment.

f Flags to construct the new instance with.

c A pointer to the object that will contain the new instance.

References _xsd_DataType_convert().

Here is the call graph for this function:

12.15.3.6 OpenGPS::Schemas::ISO5436_2::DataType::DataType (const DataType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

x An instance to make a copy of.

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.15.3.7 OpenGPS::Schemas::ISO5436_2::DataType::DataType (value *v*)

Construct an instance from the underlying enum value.

Parameters:

v A enum value.

12.15.3.8 OpenGPS::Schemas::ISO5436_2::DataType::DataType (const ::xml_schema::token & *v*)

Construct an instance from the base value.

Parameters:

v A base value.

12.15.3.9 OpenGPS::Schemas::ISO5436_2::DataType::DataType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

e A DOM element to extract the data from.

f Flags to construct the new instance with.

c A pointer to the object that will contain the new instance.

12.15.3.10 OpenGPS::Schemas::ISO5436_2::DataType::DataType (const ::xercesc::DOMAttr & *a*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM attribute.

Parameters:

- ***a*** A DOM attribute to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.15.3.11 OpenGPS::Schemas::ISO5436_2::DataType::DataType (const ::std::wstring & *s*, const ::xercesc::DOMElement * *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a string fragment.

Parameters:

- ***s*** A string fragment to extract the data from.
- ***e*** A DOM element containing the string fragment.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.15.3.12 OpenGPS::Schemas::ISO5436_2::DataType::DataType (const DataType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.15.4 Member Function Documentation

12.15.4.1 virtual DataType* OpenGPS::Schemas::ISO5436_2::DataType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.15.4.2 DataType * OpenGPS::Schemas::ISO5436_2::DataType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References DataType().

Here is the call graph for this function:

12.15.4.3 value OpenGPS::Schemas::ISO5436_2::DataType::_xsd_DataType_convert () const [protected]

12.15.4.4 DataType::value OpenGPS::Schemas::ISO5436_2::DataType::_xsd_DataType_convert () const [protected]

References _xsd_DataType_indexes_, and _xsd_DataType_literals_.

Referenced by DataType().

12.15.4.5 virtual OpenGPS::Schemas::ISO5436_2::DataType::operator value () const [inline, virtual]

Implicit conversion operator to the underlying enum value.

Returns:

A enum value.

12.15.4.6 virtual OpenGPS::Schemas::ISO5436_2::DataType::operator value () const [inline, virtual]

Implicit conversion operator to the underlying enum value.

Returns:

A enum value.

12.15.4.7 DataType& OpenGPS::Schemas::ISO5436_2::DataType::operator= (value *v*)

Assign the underlying enum value.

Parameters:

v A enum value.

Returns:

A refernce to the instance.

12.15.4.8 DataType& OpenGPS::Schemas::ISO5436_2::DataType::operator= (value *v*)

Assign the underlying enum value.

Parameters:

v A enum value.

Returns:

A refernce to the instance.

12.15.5 Member Data Documentation

12.15.5.1 static const value OpenGPS::Schemas::ISO5436_2::DataType::_xsd_DataType_-indexes_ [static]

Referenced by _xsd_DataType_convert().

12.15.5.2 static const wchar_t *const OpenGPS::Schemas::ISO5436_2::DataType::_xsd_-DataType_literals_ [static]

Referenced by _xsd_DataType_convert(), and OpenGPS::Schemas::ISO5436_2::operator<<().

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.16 OpenGPS::Schemas::ISO5436_2::Datum Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.16.1 Detailed Description

Class corresponding to the Datum schema type.

Constructors

- virtual Datum * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- Datum (const Datum &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- Datum (const ::std::wstring &s, const ::xercesc::DOMElement *e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a string fragment.

- Datum (const ::xercesc::DOMAttr &a,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM attribute.

- Datum (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- Datum (const ::xml_schema::token &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

- Datum ()

  Construct an instance from initializers for required elements and attributes.

Constructors

- virtual Datum * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- Datum (const Datum &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- Datum (const ::std::wstring &s, const ::xercesc::DOMElement *e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a string fragment.

- Datum (const ::xercesc::DOMAttr &a,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM attribute.

- Datum (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- Datum (const ::xml_schema::token &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

- Datum ()

  Construct an instance from initializers for required elements and attributes.

12.16.2 Constructor & Destructor Documentation

12.16.2.1 OpenGPS::Schemas::ISO5436_2::Datum::Datum ()

Construct an instance from initializers for required elements and attributes.

Referenced by _clone().

12.16.2.2 OpenGPS::Schemas::ISO5436_2::Datum::Datum (const ::xml_schema::token & *token*)

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.16.2.3 OpenGPS::Schemas::ISO5436_2::Datum::Datum (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.16.2.4 OpenGPS::Schemas::ISO5436_2::Datum::Datum (const ::xercesc::DOMAttr & *a*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM attribute.

Parameters:

- ***a*** A DOM attribute to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.16.2.5 OpenGPS::Schemas::ISO5436_2::Datum::Datum (const ::std::wstring & *s*, const ::xercesc::DOMElement * *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a string fragment.

Parameters:

- ***s*** A string fragment to extract the data from.
- ***e*** A DOM element containing the string fragment.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.16.2.6 OpenGPS::Schemas::ISO5436_2::Datum::Datum (const Datum & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.16.2.7 OpenGPS::Schemas::ISO5436_2::Datum::Datum ()

Construct an instance from initializers for required elements and attributes.

12.16.2.8 OpenGPS::Schemas::ISO5436_2::Datum::Datum (const ::xml_schema::token &)

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.16.2.9 OpenGPS::Schemas::ISO5436_2::Datum::Datum (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.16.2.10 OpenGPS::Schemas::ISO5436_2::Datum::Datum (const ::xercesc::DOMAttr & *a*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM attribute.

Parameters:

- ***a*** A DOM attribute to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.16.2.11 OpenGPS::Schemas::ISO5436_2::Datum::Datum (const ::std::wstring & *s*, const ::xercesc::DOMElement * *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a string fragment.

Parameters:

- ***s*** A string fragment to extract the data from.
- ***e*** A DOM element containing the string fragment.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.16.2.12 OpenGPS::Schemas::ISO5436_2::Datum::Datum (const Datum & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.16.3 Member Function Documentation

12.16.3.1 virtual Datum* OpenGPS::Schemas::ISO5436_2::Datum::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.16.3.2 Datum * OpenGPS::Schemas::ISO5436_2::Datum::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References Datum().

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.17 OpenGPS::DoubleDataPointParser Class Reference

`#include <double_data_point_parser.hxx>`

Inheritance diagram for OpenGPS::DoubleDataPointParser:

Collaboration diagram for OpenGPS::DoubleDataPointParser:

12.17.1 Detailed Description

Reads/Writes instances of OpenGPS::DataPoint of OGPS_Double point data.

Public Member Functions

- DoubleDataPointParser ()

  Creates a new instance.

- virtual void Read (PointVectorReaderContext &context, DataPoint &value) throw (...)

  Reads point data from a given context/media.

- virtual void Write (PointVectorWriterContext &context, const DataPoint &value) throw (...)

  Writes point data to a given context/media.

- virtual ~DoubleDataPointParser ()

  Destroys this instance.

12.17.2 Constructor & Destructor Documentation

12.17.2.1 DoubleDataPointParser::DoubleDataPointParser ()

Creates a new instance.

12.17.2.2 DoubleDataPointParser::~DoubleDataPointParser () [virtual]

Destroys this instance.

12.17.3 Member Function Documentation

12.17.3.1 void DoubleDataPointParser::Read (PointVectorReaderContext & *context*, DataPoint & *value*) throw (...) [virtual]

Reads point data from a given context/media.

Using the appropriate access method of the given implementation of OpenGPS::PointVectorReaderContext the current point value read gets stored in a OpenGPS::DataPoint instance. This operation is typesafe.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

context Provides read access to the media.

value Buffer where the point data read gets stored.

Implements OpenGPS::DataPointParser.

12.17.3.2 void DoubleDataPointParser::Write (PointVectorWriterContext & *context*, const DataPoint & *value*) throw (...) [virtual]

Writes point data to a given context/media.

Using the appropriate access method of the given implementation of OpenGPS::PointVectorWriterContext the point value currently stored in the OpenGPS::DataPoint instance gets written to the media. This operation is typesafe and will fail if the type of point data stored in the value object does not exacly match the current implementation of the OpenGPS::DataPointParser interface.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

context Provides write access to the media.

value Buffer where the point data to be written is stored.

Implements OpenGPS::DataPointParser.

The documentation for this class was generated from the following files:

- double_data_point_parser.hxx
- double_data_point_parser.cxx

12.18 OpenGPS::DoubleInlineValidity Class Reference

```
#include <inline_validity.hxx>
```

Inheritance diagram for OpenGPS::DoubleInlineValidity:

Collaboration diagram for OpenGPS::DoubleInlineValidity:

12.18.1 Detailed Description

Implements OpenGPS::PointValidityProvider as a lookup of a special IEEE754 value.

If the value of the point data stored for the Z component of a point vector equals the special IEEE754 value of infinity, then the point measurement is identified as invalid.

Public Member Functions

- DoubleInlineValidity (PointBuffer *const value)

  Creates a new instance.

- virtual OGPS_Boolean IsValid (const unsigned int index) const throw (...)

  Gets the validity of a point vector at a given location.

- virtual void SetValid (const unsigned int index, const OGPS_Boolean value) throw (...)

  Sets the validity of a point vector at a given location.

- ~DoubleInlineValidity ()

  Destroys this instance.

Protected Member Functions

- PointBuffer * GetPointBuffer ()

  Returns the point buffer of the Z axis.

- const PointBuffer * GetPointBuffer () const

  Returns the point buffer of the Z axis.

12.18.2 Constructor & Destructor Documentation

12.18.2.1 DoubleInlineValidity::DoubleInlineValidity (PointBuffer *const *value*)

Creates a new instance.

Parameters:

value The point buffer of the Z axis.

References _ASSERT, OpenGPS::PointBuffer::GetPointType(), and OGPS_DoublePointType.

Here is the call graph for this function:

12.18.2.2 DoubleInlineValidity::~DoubleInlineValidity ()

Destroys this instance.

12.18.3 Member Function Documentation

12.18.3.1 PointBuffer * PointValidityProvider::GetPointBuffer () [protected, inherited]

Returns the point buffer of the Z axis.

References OpenGPS::PointValidityProvider::m_PointBuffer.

12.18.3.2 const PointBuffer * PointValidityProvider::GetPointBuffer () const [protected, inherited]

Returns the point buffer of the Z axis.

References OpenGPS::PointValidityProvider::m_PointBuffer.

Referenced by OpenGPS::ValidBuffer::Allocate(), OpenGPS::ValidBuffer::HasInvalidMarks(), OpenGPS::ValidBuffer::IsValid(), IsValid(), OpenGPS::FloatInlineValidity::IsValid(), OpenGPS::Int32ValidBuffer::SetValid(), OpenGPS::Int16ValidBuffer::SetValid(), OpenGPS::ValidBuffer::SetValid(), SetValid(), and OpenGPS::FloatInlineValidity::SetValid().

12.18.3.3 OGPS_Boolean DoubleInlineValidity::IsValid (const unsigned int *index*) const throw (...) [virtual]

Gets the validity of a point vector at a given location.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

index The location of the point data the validity is checked.

Returns:

Returns TRUE if valid, FALSE otherwise.

Implements OpenGPS::PointValidityProvider.

References _ASSERT, OpenGPS::PointBuffer::Get(), and OpenGPS::PointValidityProvider::GetPointBuffer().

Here is the call graph for this function:

12.18.3.4 void DoubleInlineValidity::SetValid (const unsigned int *index*, const OGPS_Boolean *value*) throw (...) `[virtual]`

Sets the validity of a point vector at a given location.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

index The location of the point data the validity is updated.

value The value of the current validity of the point vector at the given position.

Implements OpenGPS::PointValidityProvider.

References _ASSERT, OpenGPS::PointValidityProvider::GetPointBuffer(), and OpenGPS::PointBuffer::Set().

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- inline_validity.hxx
- inline_validity.cxx

12.19 OpenGPS::DoublePointBuffer Class Reference

```
#include <double_point_buffer.hxx>
```

Inheritance diagram for OpenGPS::DoublePointBuffer:

Collaboration diagram for OpenGPS::DoublePointBuffer:

12.19.1 Detailed Description

Manages static memory and typesafe access.

Allocates an internal memory buffer to store point data of type OGPS_Double.

Public Member Functions

- virtual void Allocate (const OGPS_ULong size) throw (...)
  Allocates internal memory.
- DoublePointBuffer ()
  Create a new instance.
- virtual void Get (const OGPS_ULong index, OGPS_Float &value) const throw (...)
  Gets the value of the inernal memory at the given position.
- virtual void Get (const OGPS_ULong index, OGPS_Int32 &value) const throw (...)
  Gets the value of the inernal memory at the given position.
- virtual void Get (const OGPS_ULong index, OGPS_Int16 &value) const throw (...)
  Gets the value of the inernal memory at the given position.
- virtual void Get (const OGPS_ULong index, OGPS_Double &value) const throw (...)
  Gets the value of the inernal memory at the given position.
- virtual OGPS_DataPointType GetPointType () const
  Gets the type of point data that can be stored within this instance.
- virtual OGPS_ULong GetSize () const
  Gets the maximum amount of point data taht can be stored.
- virtual void Set (const OGPS_ULong index, const OGPS_Float value) throw (...)
  Sets the value of the inernal memory at the given position.
- virtual void Set (const OGPS_ULong index, const OGPS_Int32 value) throw (...)
  Sets the value of the inernal memory at the given position.
- virtual void Set (const OGPS_ULong index, const OGPS_Int16 value) throw (...)
  Sets the value of the inernal memory at the given position.
- virtual void Set (const OGPS_ULong index, const OGPS_Double value) throw (...)

Sets the value of the inernal memory at the given position.

- virtual ~DoublePointBuffer ()

  Destroys this instance.

Protected Member Functions

- OpenGPS::UnsignedBytePtr Allocate (const OGPS_ULong size, const size_t typeSize) throw (...)

  Allocates internal memory.

- void Free (OpenGPS::UnsignedBytePtr *value)

  Frees allocated memory.

Private Attributes

- OGPS_Double * m_Buffer

  Pointer to internal memory.

12.19.2 Constructor & Destructor Documentation

12.19.2.1 DoublePointBuffer::DoublePointBuffer ()

Create a new instance.

References m_Buffer, and NULL.

12.19.2.2 DoublePointBuffer::~DoublePointBuffer () [virtual]

Destroys this instance.

References OpenGPS::PointBuffer::Free(), and m_Buffer.

Here is the call graph for this function:

12.19.3 Member Function Documentation

12.19.3.1 OpenGPS::UnsignedBytePtr PointBuffer::Allocate (const OGPS_ULong *size*, const size_t *typeSize*) throw (...) [protected, inherited]

Allocates internal memory.

Remarks:

Allocated memory must be free by PointBuffer::Free manually.

Throws an exception on failure.

Parameters:

size Amount of point data to be stored.

typeSize The size of the data type of the point data stored herein in bytes.

Returns:

Returns a pointer to allocated memory or NULL.

References _ASSERT, _EX_T, OpenGPS::PointBuffer::m_Size, and OGPS_ExGeneral.

12.19.3.2 void DoublePointBuffer::Allocate (const OGPS_ULong *size*) throw (...) [virtual]

Allocates internal memory.

Throws an exception on failure.

Parameters:

size Amount of point data to be stored.

Reimplemented from OpenGPS::PointBuffer.

References _ASSERT, OpenGPS::PointBuffer::Allocate(), and m_Buffer.

Here is the call graph for this function:

12.19.3.3 void PointBuffer::Free (OpenGPS::UnsignedBytePtr * *value*) [protected, inherited]

Frees allocated memory.

See also:

PointBuffer::Allocate

Parameters:

value The pointer to the memory to be freed.

References _OPENGPS_FREE.

Referenced by ~DoublePointBuffer(), OpenGPS::FloatPointBuffer::~FloatPointBuffer(), OpenGPS::Int16PointBuffer::~Int16PointBuffer(), and OpenGPS::Int32PointBuffer::~Int32PointBuffer().

12.19.3.4 void PointBuffer::Get (const OGPS_ULong *index*, OGPS_Float & *value*) const throw (...) [virtual, inherited]

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented in OpenGPS::FloatPointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.19.3.5 void PointBuffer::Get (const OGPS_ULong *index*, OGPS_Int32 & *value*) const throw (...) `[virtual, inherited]`

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented in OpenGPS::Int32PointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.19.3.6 void PointBuffer::Get (const OGPS_ULong *index*, OGPS_Int16 & *value*) const throw (...) `[virtual, inherited]`

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented in OpenGPS::Int16PointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

Referenced by OpenGPS::PointVectorProxy::DataPointProxy::Get(), OpenGPS::DoubleInlineValidity::IsValid(), and OpenGPS::FloatInlineValidity::IsValid().

12.19.3.7 void DoublePointBuffer::Get (const OGPS_ULong *index*, OGPS_Double & *value*) const throw (...) `[virtual]`

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented from OpenGPS::PointBuffer.

References _ASSERT, OpenGPS::PointBuffer::GetSize(), and m_Buffer.

Here is the call graph for this function:

12.19.3.8 OGPS_DataPointType DoublePointBuffer::GetPointType () const [virtual]

Gets the type of point data that can be stored within this instance.

Reimplemented from OpenGPS::PointBuffer.

References OGPS_DoublePointType.

12.19.3.9 OGPS_ULong PointBuffer::GetSize () const [virtual, inherited]

Gets the maximum amount of point data taht can be stored.

References OpenGPS::PointBuffer::m_Size.

Referenced by OpenGPS::ValidBuffer::Allocate(), OpenGPS::Int32PointBuffer::Get(), OpenGPS::Int16PointBuffer::Get(), OpenGPS::FloatPointBuffer::Get(), Get(), OpenGPS::ValidBuffer::HasInvalidMarks(), OpenGPS::Int32PointBuffer::Set(), OpenGPS::Int16PointBuffer::Set(), OpenGPS::FloatPointBuffer::Set(), and Set().

12.19.3.10 void PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Float *value*) throw (...) [virtual, inherited]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented in OpenGPS::FloatPointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.19.3.11 void PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Int32 *value*) throw (...) [virtual, inherited]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented in OpenGPS::Int32PointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.19.3.12 void PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Int16 *value*) throw (...) [virtual, inherited]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented in OpenGPS::Int16PointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

Referenced by OpenGPS::PointVectorProxy::DataPointProxy::Set(), OpenGPS::Int32ValidBuffer::SetValid(), OpenGPS::Int16ValidBuffer::SetValid(), OpenGPS::DoubleInlineValidity::SetValid(), and OpenGPS::FloatInlineValidity::SetValid().

12.19.3.13 void DoublePointBuffer::Set (const OGPS_ULong *index*, const OGPS_Double *value*) throw (...) [virtual]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented from OpenGPS::PointBuffer.

References _ASSERT, OpenGPS::PointBuffer::GetSize(), and m_Buffer.

Here is the call graph for this function:

12.19.4 Member Data Documentation

12.19.4.1 OGPS_Double* OpenGPS::DoublePointBuffer::m_Buffer [private]

Pointer to internal memory.

Referenced by Allocate(), DoublePointBuffer(), Get(), Set(), and ~DoublePointBuffer().

The documentation for this class was generated from the following files:

- double_point_buffer.hxx
- double_point_buffer.cxx

12.20 OpenGPS::Environment Class Reference

```
#include <environment.hxx>
```

Collaboration diagram for OpenGPS::Environment:

12.20.1 Detailed Description

Interface for communicating with the operating system and related subjects.

This encapsulates the differences between hardware architectures, compilers and operation systems this software library will run upon.

Public Member Functions

- void ByteSwap (const OpenGPS::UnsignedBytePtr src, double ∗const value) const

  Swaps endianness.

- OpenGPS::UnsignedBytePtr ByteSwap (const double ∗const value, OpenGPS::UnsignedBytePtr dst) const

  Swaps endianness.

- void ByteSwap (const OpenGPS::UnsignedBytePtr src, float ∗const value) const

  Swaps endianness.

- OpenGPS::UnsignedBytePtr ByteSwap (const float ∗const value, OpenGPS::UnsignedBytePtr dst) const

  Swaps endianness.

- void ByteSwap (const OpenGPS::UnsignedBytePtr src, int ∗const value) const

  Swaps endianness.

- OpenGPS::UnsignedBytePtr ByteSwap (const int ∗const value, OpenGPS::UnsignedBytePtr dst) const

  Swaps endianness.

- void ByteSwap (const OpenGPS::UnsignedBytePtr src, short ∗const value) const

  Swaps endianness.

- OpenGPS::UnsignedBytePtr ByteSwap (const short ∗const value, OpenGPS::UnsignedBytePtr dst) const

  Swaps endianness.

- void ByteSwap16 (const OpenGPS::UnsignedBytePtr src, short ∗const value) const

  Swaps endianness.

- OpenGPS::UnsignedBytePtr ByteSwap16 (const short ∗const value, OpenGPS::UnsignedBytePtr dst) const

  Swaps endianness.

- void ByteSwap32 (const OpenGPS::UnsignedBytePtr src, float ∗const value) const

Swaps endianness.

- OpenGPS::UnsignedBytePtr ByteSwap32 (const float *const value, OpenGPS::UnsignedBytePtr dst) const

  Swaps endianness.

- void ByteSwap32 (const OpenGPS::UnsignedBytePtr src, int *const value) const

  Swaps endianness.

- OpenGPS::UnsignedBytePtr ByteSwap32 (const int *const value, OpenGPS::UnsignedBytePtr dst) const

  Swaps endianness.

- void ByteSwap64 (const OpenGPS::UnsignedBytePtr src, double *const value) const

  Swaps endianness.

- OpenGPS::UnsignedBytePtr ByteSwap64 (const double *const value, OpenGPS::UnsignedBytePtr dst) const

  Swaps endianness.

- virtual OpenGPS::String ConcatPathes (const OpenGPS::String &path1, const OpenGPS::String &path2) const =0

  Concatenates two distinct path names.

- virtual OGPS_Boolean CreateDir (const OpenGPS::String &path) const =0

  Makes a directory.

- virtual OGPS_Character GetAltDirectorySeparator () const =0

  E.g.

- virtual OGPS_Character GetDirectorySeparator () const =0

  E.g.

- virtual OpenGPS::String GetFileName (const OpenGPS::String &path) const =0

  Gets the file name including its extension from a full path.

- virtual OGPS_Boolean GetPathName (const OpenGPS::String &path, OpenGPS::String &clean_-path) const =0

  Converts a given path to a normalized version suitable for the current operating system.

- virtual OpenGPS::String GetTempDir () const =0

  Returns the path to the temporary directory or an empty string on error.

- virtual OpenGPS::String GetUniqueName () const =0

  Gets a random sequence of numeric characters.

- virtual OGPS_Boolean GetVariable (const OpenGPS::String &varName, OpenGPS::String &value) const =0

  Gets the value of a named environment variable.

- virtual OGPS_Boolean IsLittleEndian () const

  Returns TRUE on a system with little endian byte order, FALSE on a system using big endian byte order.

- virtual OGPS_Boolean PathExists (const OpenGPS::String &file) const =0

  Returns TRUE if a given path exists, FALSE otherwise.

- virtual OGPS_Boolean RemoveDir (const OpenGPS::String &path) const =0

  Deletes a given directory.

- virtual OGPS_Boolean RemoveFile (const OpenGPS::String &file) const =0

  Deletes a given file.

- virtual OGPS_Boolean RenameFile (const OpenGPS::String &src, const OpenGPS::String &dst) const =0

  Renames a file.

Static Public Member Functions

- static const Environment *const GetInstance ()

  Creates a new instance of the environment if needed.

- static void Reset ()

  Frees resources obtained through Environment::GetInstance.

Protected Member Functions

- Environment ()

  Creates a new instance.

- virtual ~Environment ()

  Destroys this instance.

Static Protected Member Functions

- static Environment * CreateInstance ()

  Creates the appropriate instance for the current system.

Static Private Attributes

- static Environment * m_Instance = NULL

  Pointer to the environment.

12.20.2 Constructor & Destructor Documentation

12.20.2.1 Environment::Environment () [protected]

Creates a new instance.

12.20.2.2 Environment::~Environment () [protected, virtual]

Destroys this instance.

12.20.3 Member Function Documentation

12.20.3.1 void Environment::ByteSwap (const OpenGPS::UnsignedBytePtr *src*, double *const *value*) const

Swaps endianness.

Parameters:

src The array of bytes to be swapped.

value The swapped bytes as typed data.

References ByteSwap64().

Here is the call graph for this function:

12.20.3.2 OpenGPS::UnsignedBytePtr Environment::ByteSwap (const double *const *value*, OpenGPS::UnsignedBytePtr *dst*) const

Swaps endianness.

Parameters:

value The bytes to be swapped as typed data.

dst The array of bytes in swapped byte order.

Returns:

Pointer to the array of bytes in swapped byte order.

References ByteSwap64().

Here is the call graph for this function:

12.20.3.3 void Environment::ByteSwap (const OpenGPS::UnsignedBytePtr *src*, float *const *value*) const

Swaps endianness.

Parameters:

src The array of bytes to be swapped.

value The swapped bytes as typed data.

References ByteSwap32().

Here is the call graph for this function:

12.20.3.4 OpenGPS::UnsignedBytePtr Environment::ByteSwap (const float *const *value*, OpenGPS::UnsignedBytePtr *dst*) const

Swaps endianness.

Parameters:

value The bytes to be swapped as typed data.

dst The array of bytes in swapped byte order.

Returns:

Pointer to the array of bytes in swapped byte order.

References ByteSwap32().

Here is the call graph for this function:

12.20.3.5 void Environment::ByteSwap (const OpenGPS::UnsignedBytePtr *src*, int *const *value*) const

Swaps endianness.

Parameters:

src The array of bytes to be swapped.

value The swapped bytes as typed data.

References _ASSERT, and _OGPS_INT_SIZE.

12.20.3.6 OpenGPS::UnsignedBytePtr Environment::ByteSwap (const int *const *value*, OpenGPS::UnsignedBytePtr *dst*) const

Swaps endianness.

Parameters:

value The bytes to be swapped as typed data.

dst The array of bytes in swapped byte order.

Returns:

Pointer to the array of bytes in swapped byte order.

References _ASSERT, and _OGPS_INT_SIZE.

12.20.3.7 void Environment::ByteSwap (const OpenGPS::UnsignedBytePtr *src*, short *const *value*) const

Swaps endianness.

Parameters:

src The array of bytes to be swapped.

value The swapped bytes as typed data.

References ByteSwap16().

Here is the call graph for this function:

12.20.3.8 OpenGPS::UnsignedBytePtr Environment::ByteSwap (const short *const *value*, OpenGPS::UnsignedBytePtr *dst*) const

Swaps endianness.

Parameters:

value The bytes to be swapped as typed data.

dst The array of bytes in swapped byte order.

Returns:

Pointer to the array of bytes in swapped byte order.

References ByteSwap16().

Here is the call graph for this function:

12.20.3.9 void Environment::ByteSwap16 (const OpenGPS::UnsignedBytePtr *src*, short *const *value*) const

Swaps endianness.

It is assumed that src is 16 bit long.

Parameters:

src The array of bytes to be swapped.

value The swapped bytes as typed data.

References _ASSERT, and _OGPS_SHORT_SIZE.

12.20.3.10 OpenGPS::UnsignedBytePtr Environment::ByteSwap16 (const short *const *value*, OpenGPS::UnsignedBytePtr *dst*) const

Swaps endianness.

It is assumed that dst is 16 bit long. So data is truncated if the size of the typed data is greater than 16 bit.

Parameters:

value The bytes to be swapped as typed data.

dst The array of bytes in swapped byte order.

Returns:

Pointer to the array of bytes in swapped byte order.

References _ASSERT, and _OGPS_SHORT_SIZE.

Referenced by ByteSwap(), IsLittleEndian(), and OpenGPS::BinaryMSBPointVectorReaderContext::Read().

12.20.3.11 void Environment::ByteSwap32 (const OpenGPS::UnsignedBytePtr *src*, float *const *value*) const

Swaps endianness.

It is assumed that src is 32 bit long.

Parameters:

src The array of bytes to be swapped.

value The swapped bytes as typed data.

References _ASSERT, and _OGPS_FLOAT_SIZE.

12.20.3.12 OpenGPS::UnsignedBytePtr Environment::ByteSwap32 (const float *const *value*, OpenGPS::UnsignedBytePtr *dst*) const

Swaps endianness.

It is assumed that dst is 32 bit long. So data is truncated if the size of the typed data is greater than 32 bit.

Parameters:

value The bytes to be swapped as typed data.

dst The array of bytes in swapped byte order.

Returns:

Pointer to the array of bytes in swapped byte order.

References _ASSERT, and _OGPS_FLOAT_SIZE.

12.20.3.13 void Environment::ByteSwap32 (const OpenGPS::UnsignedBytePtr *src*, int ∗const *value*) const

Swaps endianness.

It is assumed that src is 32 bit long.

Parameters:

src The array of bytes to be swapped.

value The swapped bytes as typed data.

References _ASSERT, and _OGPS_INT_SIZE.

12.20.3.14 OpenGPS::UnsignedBytePtr Environment::ByteSwap32 (const int ∗const *value*, OpenGPS::UnsignedBytePtr *dst*) const

Swaps endianness.

It is assumed that dst is 32 bit long. So data is truncated if the size of the typed data is greater than 32 bit.

Parameters:

value The bytes to be swapped as typed data.

dst The array of bytes in swapped byte order.

Returns:

Pointer to the array of bytes in swapped byte order.

References _ASSERT, and _OGPS_INT_SIZE.

Referenced by ByteSwap(), and OpenGPS::BinaryMSBPointVectorReaderContext::Read().

12.20.3.15 void Environment::ByteSwap64 (const OpenGPS::UnsignedBytePtr *src*, double ∗const *value*) const

Swaps endianness.

It is assumed that src is 64 bit long.

Parameters:

src The array of bytes to be swapped.

value The swapped bytes as typed data.

References _ASSERT, and _OGPS_DOUBLE_SIZE.

12.20.3.16 OpenGPS::UnsignedBytePtr Environment::ByteSwap64 (const double *const *value*, OpenGPS::UnsignedBytePtr *dst*) const

Swaps endianness.

It is assumed that dst is 64 bit long. So data is truncated if the size of the typed data is greater than 64 bit.

Parameters:

value The bytes to be swapped as typed data.

dst The array of bytes in swapped byte order.

Returns:

Pointer to the array of bytes in swapped byte order.

References _ASSERT, and _OGPS_DOUBLE_SIZE.

Referenced by ByteSwap(), and OpenGPS::BinaryMSBPointVectorReaderContext::Read().

12.20.3.17 virtual OpenGPS::String OpenGPS::Environment::ConcatPathes (const OpenGPS::String & *path1*, const OpenGPS::String & *path2*) const `[pure virtual]`

Concatenates two distinct path names.

Parameters:

path1 The first part of the path.

path2 The second part of the path.

Returns:

Returns the concatenated path name or an empty string on error.

Referenced by OpenGPS::ISO5436_2Container::ConfigureNamespaceMap(), OpenGPS::ISO5436_2Container::CreateContainerTempFilePath(), OpenGPS::ISO5436_2Container::CreateTempDir(), OpenGPS::ISO5436_2Container::GetChecksumFileName(), OpenGPS::ISO5436_2Container::GetMainFileName(), OpenGPS::ISO5436_2Container::GetPointDataFileName(), and OpenGPS::ISO5436_2Container::GetValidPointsFileName().

12.20.3.18 virtual OGPS_Boolean OpenGPS::Environment::CreateDir (const OpenGPS::String & *path*) const `[pure virtual]`

Makes a directory.

Parameters:

path The directory to be created.

Returns:

Returns TRUE on success, FALSE otherwise.

Referenced by OpenGPS::ISO5436_2Container::CreateTempDir().

12.20.3.19 static Environment* OpenGPS::Environment::CreateInstance () [static, protected]

Creates the appropriate instance for the current system.

Referenced by GetInstance().

12.20.3.20 virtual OGPS_Character OpenGPS::Environment::GetAltDirectorySeparator () const [pure virtual]

E.g.

gets the forward slash on Microsoft Windows systems.

12.20.3.21 virtual OGPS_Character OpenGPS::Environment::GetDirectorySeparator () const [pure virtual]

E.g.

gets the backslash on Microsoft Windows systems.

12.20.3.22 virtual OpenGPS::String OpenGPS::Environment::GetFileName (const OpenGPS::String & *path*) const [pure virtual]

Gets the file name including its extension from a full path.

Parameters:

The full path of a file.

Returns:

Returns just the filename.

Referenced by OpenGPS::ISO5436_2Container::WriteVendorSpecific().

12.20.3.23 const Environment *const Environment::GetInstance () [static]

Creates a new instance of the environment if needed.

Remarks:

You must call Environment::Reset somewhere within the scope this method has been accessed, e.g. the end of object lifetime within a class scope or at least once for all on termination of the program. Otherwise resources obtained will not get released correctly.

References CreateInstance(), and m_Instance.

Referenced by OpenGPS::ISO5436_2Container::Compress(), OpenGPS::ISO5436_2Container::ConfigureNamespaceMap(), OpenGPS::ISO5436_2Container::CreateContainerTempFilePath(), OpenGPS::ISO5436_2Container::CreatePointVectorReaderContext(), OpenGPS::ISO5436_2Container::CreatePointVectorWriterContext(), OpenGPS::ISO5436_2Container::CreateTempDir(), OpenGPS::ISO5436_2Container::GetChecksumFileName(), OpenGPS::ISO5436_2Container::GetMainFileName(), OpenGPS::ISO5436_2Container::GetPointDataFileName(), OpenGPS::ISO5436_2Container::GetValidPointsFileName(), OpenGPS::BinaryMSBPointVectorReaderContext::Read(), OpenGPS::ISO5436_2Container::RemoveTempDir(), OpenGPS::BinaryMSBPointVectorWriterContext::Write(), and OpenGPS::ISO5436_2Container::WriteVendorSpecific().

Here is the call graph for this function:

12.20.3.24 virtual OGPS_Boolean OpenGPS::Environment::GetPathName (const OpenGPS::String & *path*, OpenGPS::String & *clean_path*) const [pure virtual]

Converts a given path to a normalized version suitable for the current operating system.

Parameters:

path A path name.

clean_path Gets a normalized path name as the result.

Returns:

Returns TRUE on success, FALSE otherwise.

Referenced by OpenGPS::ISO5436_2Container::ConfigureNamespaceMap().

12.20.3.25 virtual OpenGPS::String OpenGPS::Environment::GetTempDir () const [pure virtual]

Returns the path to the temporary directory or an empty string on error.

Referenced by OpenGPS::ISO5436_2Container::CreateTempDir().

12.20.3.26 virtual OpenGPS::String OpenGPS::Environment::GetUniqueName () const [pure virtual]

Gets a random sequence of numeric characters.

Referenced by OpenGPS::ISO5436_2Container::CreateContainerTempFilePath(), OpenGPS::ISO5436_2Container::CreateTempDir(), OpenGPS::ISO5436_2Container::GetPointDataFileName(), and OpenGPS::ISO5436_2Container::GetValidPointsFileName().

12.20.3.27 virtual OGPS_Boolean OpenGPS::Environment::GetVariable (const OpenGPS::String & *varName*, OpenGPS::String & *value*) const [pure virtual]

Gets the value of a named environment variable.

Parameters:

varName The name of the environment variable.

value Gets the value of the given environment variable.

Returns:

Returns TRUE if the value of the environment variable could be obtained, FALSE otherwise.

12.20.3.28 OGPS_Boolean Environment::IsLittleEndian () const [virtual]

Returns TRUE on a system with little endian byte order, FALSE on a system using big endian byte order.

References _ASSERT, and ByteSwap16().

Referenced by OpenGPS::ISO5436_2Container::CreatePointVectorReaderContext().

Here is the call graph for this function:

12.20.3.29 virtual OGPS_Boolean OpenGPS::Environment::PathExists (const OpenGPS::String & *file*) const [pure virtual]

Returns TRUE if a given path exists, FALSE otherwise.

Parameters:

file The path to check.

Referenced by OpenGPS::ISO5436_2Container::CreateTempDir().

12.20.3.30 virtual OGPS_Boolean OpenGPS::Environment::RemoveDir (const OpenGPS::String & *path*) const [pure virtual]

Deletes a given directory.

Parameters:

path The path to the directory to be erased.

12.20.3.31 virtual OGPS_Boolean OpenGPS::Environment::RemoveFile (const OpenGPS::String & *file*) const [pure virtual]

Deletes a given file.

Parameters:

file The path to the file to be erased.

12.20.3.32 virtual OGPS_Boolean OpenGPS::Environment::RenameFile (const OpenGPS::String & *src*, const OpenGPS::String & *dst*) const [pure virtual]

Renames a file.

Parameters:

src The path to the file to be renamed.

dst The new path of the renamed file.

Returns:

Returns TRUE on success, FALSE otherwise.

Referenced by OpenGPS::ISO5436_2Container::Compress().

12.20.3.33 void Environment::Reset () [static]

Frees resources obtained through Environment::GetInstance.

Remarks:

This must be called at least once at the end of some scope in your program when Environment::GetInstance has been called.

See also:

Environment::GetInstance

References _OPENGPS_DELETE, and m_Instance.

Referenced by OpenGPS::BinaryMSBPointVectorReaderContext::~BinaryMSBPointVectorReaderContext(), OpenGPS::BinaryMSBPointVectorWriterContext::~BinaryMSBPointVectorWriterContext(), and OpenGPS::ISO5436_2Container::~ISO5436_2Container().

12.20.4 Member Data Documentation

12.20.4.1 Environment * Environment::m_Instance = NULL [static, private]

Pointer to the environment.

Referenced by GetInstance(), and Reset().

The documentation for this class was generated from the following files:

- environment.hxx
- environment.cxx

12.21 OpenGPS::Exception Class Reference

```
#include <exceptions.hxx>
```

Inheritance diagram for OpenGPS::Exception:

Collaboration diagram for OpenGPS::Exception:

12.21.1 Detailed Description

Describes a general exception.

Public Member Functions

- const OpenGPS::String & details () const throw ()

  Gets a detailed description possible with hints to its avoidance.

- Exception (const Exception &rhs) throw ()

  Creates a new instance.

- Exception (const OGPS_ExceptionId id, const OGPS_ExceptionChar *text, const OGPS_-ExceptionChar *details, const OGPS_ExceptionChar *method) throw ()

  Creates a new instance.

- OGPS_ExceptionId id () const throw ()

  Gets an identifier for the current type of excpetion.

- const OpenGPS::String & method () const throw ()

  Gets the name of the method where the exception occured or NULL.

- virtual ~Exception () throw ()

  Destroys this instance.

Private Attributes

- OpenGPS::String m_Details

  Detailed description of the failure.

- OGPS_ExceptionId m_Id

  Describes the type of failure.

- OpenGPS::String m_Method

Name of the method wherein the failure occured.

12.21.2 Constructor & Destructor Documentation

12.21.2.1 Exception::Exception (const OGPS_ExceptionId *id*, const OGPS_ExceptionChar * *text*, const OGPS_ExceptionChar * *details*, const OGPS_ExceptionChar * *method*) throw ()

Creates a new instance.

Parameters:

id Type of the exception object.

text Brief description of the exception.

details Detailed description of the exception. May be set to NULL.

method The name of the method wherein the failure condition occured. May be set to NULL.

References _ASSERT, details(), OpenGPS::String::FromChar(), id(), m_Details, m_Id, m_Method, and method().

Here is the call graph for this function:

12.21.2.2 Exception::Exception (const Exception & *rhs*) throw ()

Creates a new instance.

Parameters:

rhs The instance to copy from.

References m_Details, m_Id, and m_Method.

12.21.2.3 Exception::~Exception () throw () [virtual]

Destroys this instance.

12.21.3 Member Function Documentation

12.21.3.1 const OpenGPS::String & Exception::details () const throw ()

Gets a detailed description possible with hints to its avoidance.

References m_Details.

Referenced by Exception(), readonlyExample3(), readonlyExample4(), and OpenGPS::ExceptionHistory::SetLastException().

12.21.3.2 OGPS_ExceptionId Exception::id () const throw ()

Gets an identifier for the current type of excpetion.

References m_Id.

Referenced by Exception(), and OpenGPS::ExceptionHistory::SetLastException().

12.21.3.3 const OpenGPS::String & Exception::method () const throw ()

Gets the name of the method where the exception occured or NULL.

References m_Method.

Referenced by Exception(), and OpenGPS::ExceptionHistory::SetLastException().

12.21.4 Member Data Documentation

12.21.4.1 OpenGPS::String OpenGPS::Exception::m_Details [private]

Detailed description of the failure.

Referenced by details(), and Exception().

12.21.4.2 OGPS_ExceptionId OpenGPS::Exception::m_Id [private]

Describes the type of failure.

Referenced by Exception(), and id().

12.21.4.3 OpenGPS::String OpenGPS::Exception::m_Method [private]

Name of the method wherein the failure occured.

Referenced by Exception(), and method().

The documentation for this class was generated from the following files:

- exceptions.hxx
- exceptions.cxx

12.22 OpenGPS::ExceptionHistory Class Reference

```
#include <messages_c.hxx>
```

Collaboration diagram for OpenGPS::ExceptionHistory:

12.22.1 Detailed Description

Maintains a history of exceptions.

Static Public Member Functions

- static const OGPS_Character * GetLastErrorDescription ()
  Gets the detailed failure description of the last known exception or NULL.
- static OGPS_ExceptionId GetLastErrorId ()
  Gets the classifier of the last known exception or OGPS_ExNone if none is maintained so far.
- static const OGPS_Character * GetLastErrorMessage ()
  Gets the brief failure description of the last known exception or NULL.
- static void Reset ()
  Reset the history of exceptions.
- static void SetLastException ()
  Notify that an unknown type of exception occured.
- static void SetLastException (const std::exception &ex)
  Sets an exception object as the last one that occured during execution.
- static void SetLastException (const OpenGPS::Exception &ex)
  Sets an exception object as the last one that occured during execution.

Private Member Functions

- ExceptionHistory ()
  Creates a new instance.
- ~ExceptionHistory ()
  Destroys this instance.

Static Private Member Functions

- static void DumpIt ()

  Dumps a message to the error console.

Static Private Attributes

- static OpenGPS::String m_LastErrorDescription

  The detailed description of the last known failure condition or empty.

- static OGPS_ExceptionId m_LastErrorId = OGPS_ExNone

  The classifier of the last known failure condition or OGPS_ExNone.

- static OpenGPS::String m_LastErrorMessage

  The brief description of the last known failure condition or empty.

- static OpenGPS::String m_LastErrorSource

  Source of the last error condition.

12.22.2 Constructor & Destructor Documentation

12.22.2.1 OpenGPS::ExceptionHistory::ExceptionHistory () [private]

Creates a new instance.

12.22.2.2 OpenGPS::ExceptionHistory::~ExceptionHistory () [private]

Destroys this instance.

12.22.3 Member Function Documentation

12.22.3.1 void OpenGPS::ExceptionHistory::DumpIt () [static, private]

Dumps a message to the error console.

This works in _DEBUG only.

References m_LastErrorDescription, m_LastErrorMessage, m_LastErrorSource, and OpenGPS::String::ToChar().

Referenced by SetLastException().

Here is the call graph for this function:

12.22.3.2 const OGPS_Character * OpenGPS::ExceptionHistory::GetLastErrorDescription () [static]

Gets the detailed failure description of the last known exception or NULL.

References m_LastErrorDescription, and NULL.

Referenced by ogps_GetErrorDescription().

12.22.3.3 OGPS_ExceptionId OpenGPS::ExceptionHistory::GetLastErrorId () [static]

Gets the classifier of the last known exception or OGPS_ExNone if none is maintained so far.

References m_LastErrorId.

Referenced by ogps_GetErrorId(), and ogps_HasError().

12.22.3.4 const OGPS_Character * OpenGPS::ExceptionHistory::GetLastErrorMessage () [static]

Gets the brief failure description of the last known exception or NULL.

References m_LastErrorMessage, and NULL.

Referenced by ogps_GetErrorMessage().

12.22.3.5 void OpenGPS::ExceptionHistory::Reset () [static]

Reset the history of exceptions.

References m_LastErrorDescription, m_LastErrorId, m_LastErrorMessage, m_LastErrorSource, and OGPS_ExNone.

12.22.3.6 void OpenGPS::ExceptionHistory::SetLastException () [static]

Notify that an unknown type of exception occured.

References DumpIt(), m_LastErrorDescription, m_LastErrorId, m_LastErrorMessage, m_-LastErrorSource, and OGPS_ExGeneral.

Here is the call graph for this function:

12.22.3.7 void OpenGPS::ExceptionHistory::SetLastException (const std::exception & *ex*) [static]

Sets an exception object as the last one that occured during execution.

Parameters:

ex Add this exception object to the history of exceptions.

References DumpIt(), OpenGPS::String::FromChar(), m_LastErrorDescription, m_LastErrorId, m_-LastErrorMessage, m_LastErrorSource, and OGPS_ExGeneral.

Here is the call graph for this function:

12.22.3.8 void OpenGPS::ExceptionHistory::SetLastException (const OpenGPS::Exception & *ex*) [static]

Sets an exception object as the last one that occured during execution.

Parameters:

ex Add this exception object to the history of exceptions.

References OpenGPS::Exception::details(), DumpIt(), OpenGPS::String::FromChar(), OpenGPS::Exception::id(), m_LastErrorDescription, m_LastErrorId, m_LastErrorMessage, m_-LastErrorSource, and OpenGPS::Exception::method().

Here is the call graph for this function:

12.22.4 Member Data Documentation

12.22.4.1 OpenGPS::String OpenGPS::ExceptionHistory::m_LastErrorDescription [static, private]

The detailed description of the last known failure condition or empty.

Referenced by DumpIt(), GetLastErrorDescription(), Reset(), and SetLastException().

12.22.4.2 OGPS_ExceptionId OpenGPS::ExceptionHistory::m_LastErrorId = OGPS_ExNone [static, private]

The classifier of the last known failure condition or OGPS_ExNone.

Referenced by GetLastErrorId(), Reset(), and SetLastException().

12.22.4.3 OpenGPS::String OpenGPS::ExceptionHistory::m_LastErrorMessage [static, private]

The brief description of the last known failure condition or empty.

Referenced by DumpIt(), GetLastErrorMessage(), Reset(), and SetLastException().

12.22.4.4 OpenGPS::String OpenGPS::ExceptionHistory::m_LastErrorSource [static, private]

Source of the last error condition.

Referenced by DumpIt(), Reset(), and SetLastException().

The documentation for this class was generated from the following files:

- messages_c.hxx
- messages_c.cxx

12.23 OpenGPS::Schemas::ISO5436_2::FeatureType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.23.1 Detailed Description

Class corresponding to the FeatureType schema type.

Constructors

- virtual FeatureType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- FeatureType (const FeatureType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- FeatureType (const ::std::wstring &s, const ::xercesc::DOMElement *e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a string fragment.

- FeatureType (const ::xercesc::DOMAttr &a,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM attribute.

- FeatureType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- FeatureType (const ::xml_schema::token &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

- FeatureType ()

  Construct an instance from initializers for required elements and attributes.

Constructors

- virtual FeatureType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- FeatureType (const FeatureType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- FeatureType (const ::std::wstring &s, const ::xercesc::DOMElement *e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a string fragment.

- FeatureType (const ::xercesc::DOMAttr &a,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM attribute.

- FeatureType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- FeatureType (const ::xml_schema::token &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

- FeatureType ()

  Construct an instance from initializers for required elements and attributes.

12.23.2 Constructor & Destructor Documentation

12.23.2.1 OpenGPS::Schemas::ISO5436_2::FeatureType::FeatureType ()

Construct an instance from initializers for required elements and attributes.

Referenced by _clone().

12.23.2.2 OpenGPS::Schemas::ISO5436_2::FeatureType::FeatureType (const ::xml_schema::token & *token*)

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.23.2.3 OpenGPS::Schemas::ISO5436_2::FeatureType::FeatureType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.23.2.4 OpenGPS::Schemas::ISO5436_2::FeatureType::FeatureType (const ::xercesc::DOMAttr & *a*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM attribute.

Parameters:

- ***a*** A DOM attribute to extract the data from.

f Flags to construct the new instance with.

c A pointer to the object that will contain the new instance.

12.23.2.5 OpenGPS::Schemas::ISO5436_2::FeatureType::FeatureType (const ::std::wstring & *s*, const ::xercesc::DOMElement * *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a string fragment.

Parameters:

s A string fragment to extract the data from.

e A DOM element containing the string fragment.

f Flags to construct the new instance with.

c A pointer to the object that will contain the new instance.

12.23.2.6 OpenGPS::Schemas::ISO5436_2::FeatureType::FeatureType (const FeatureType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

x An instance to make a copy of.

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.23.2.7 OpenGPS::Schemas::ISO5436_2::FeatureType::FeatureType ()

Construct an instance from initializers for required elements and attributes.

12.23.2.8 OpenGPS::Schemas::ISO5436_2::FeatureType::FeatureType (const ::xml_schema::token &)

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.23.2.9 OpenGPS::Schemas::ISO5436_2::FeatureType::FeatureType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

e A DOM element to extract the data from.

f Flags to construct the new instance with.

c A pointer to the object that will contain the new instance.

12.23.2.10 OpenGPS::Schemas::ISO5436_2::FeatureType::FeatureType (const ::xercesc::DOMAttr & *a*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM attribute.

Parameters:

- ***a*** A DOM attribute to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.23.2.11 OpenGPS::Schemas::ISO5436_2::FeatureType::FeatureType (const ::std::wstring & *s*, const ::xercesc::DOMElement * *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a string fragment.

Parameters:

- ***s*** A string fragment to extract the data from.
- ***e*** A DOM element containing the string fragment.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.23.2.12 OpenGPS::Schemas::ISO5436_2::FeatureType::FeatureType (const FeatureType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.23.3 Member Function Documentation

12.23.3.1 virtual FeatureType* OpenGPS::Schemas::ISO5436_2::FeatureType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.23.3.2 FeatureType * OpenGPS::Schemas::ISO5436_2::FeatureType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References FeatureType().

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.24 OpenGPS::FloatDataPointParser Class Reference

```
#include <float_data_point_parser.hxx>
```

Inheritance diagram for OpenGPS::FloatDataPointParser:

Collaboration diagram for OpenGPS::FloatDataPointParser:

12.24.1 Detailed Description

Reads/Writes instances of OpenGPS::DataPoint of OGPS_Float point data.

Public Member Functions

- FloatDataPointParser ()

  Creates a new instance.

- virtual void Read (PointVectorReaderContext &context, DataPoint &value) throw (...)

  Reads point data from a given context/media.

- virtual void Write (PointVectorWriterContext &context, const DataPoint &value) throw (...)

  Writes point data to a given context/media.

- virtual ~FloatDataPointParser ()

  Destroys this instance.

12.24.2 Constructor & Destructor Documentation

12.24.2.1 FloatDataPointParser::FloatDataPointParser ()

Creates a new instance.

12.24.2.2 FloatDataPointParser::~FloatDataPointParser () [virtual]

Destroys this instance.

12.24.3 Member Function Documentation

12.24.3.1 void FloatDataPointParser::Read (PointVectorReaderContext & *context*, DataPoint & *value*) throw (...) [virtual]

Reads point data from a given context/media.

Using the appropriate access method of the given implementation of OpenGPS::PointVectorReaderContext the current point value read gets stored in a OpenGPS::DataPoint instance. This operation is typesafe.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

context Provides read access to the media.

value Buffer where the point data read gets stored.

Implements OpenGPS::DataPointParser.

12.24.3.2 void FloatDataPointParser::Write (PointVectorWriterContext & *context*, const DataPoint & *value*) throw (...) [virtual]

Writes point data to a given context/media.

Using the appropriate access method of the given implementation of OpenGPS::PointVectorWriterContext the point value currently stored in the OpenGPS::DataPoint instance gets written to the media. This operation is typesafe and will fail if the type of point data stored in the value object does not exacly match the current implementation of the OpenGPS::DataPointParser interface.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

context Provides write access to the media.

value Buffer where the point data to be written is stored.

Implements OpenGPS::DataPointParser.

The documentation for this class was generated from the following files:

- float_data_point_parser.hxx
- float_data_point_parser.cxx

12.25 OpenGPS::FloatInlineValidity Class Reference

```
#include <inline_validity.hxx>
```

Inheritance diagram for OpenGPS::FloatInlineValidity:

Collaboration diagram for OpenGPS::FloatInlineValidity:

12.25.1 Detailed Description

Implements OpenGPS::PointValidityProvider as a lookup of a special IEEE754 value.

If the value of the point data stored for the Z component of a point vector equals the special IEEE754 value of infinity, then the point measurement is identified as invalid.

Public Member Functions

- FloatInlineValidity (PointBuffer *const value)

  Creates a new instance.

- virtual OGPS_Boolean IsValid (const unsigned int index) const throw (...)

Gets the validity of a point vector at a given location.

- virtual void SetValid (const unsigned int index, const OGPS_Boolean value) throw (...)

  Sets the validity of a point vector at a given location.

- ~FloatInlineValidity ()

  Destroys this instance.

Protected Member Functions

- PointBuffer * GetPointBuffer ()

  Returns the point buffer of the Z axis.

- const PointBuffer * GetPointBuffer () const

  Returns the point buffer of the Z axis.

12.25.2 Constructor & Destructor Documentation

12.25.2.1 FloatInlineValidity::FloatInlineValidity (PointBuffer *const *value*)

Creates a new instance.

Parameters:

value The point buffer of the Z axis.

References _ASSERT, OpenGPS::PointBuffer::GetPointType(), and OGPS_FloatPointType.

Here is the call graph for this function:

12.25.2.2 FloatInlineValidity::~FloatInlineValidity ()

Destroys this instance.

12.25.3 Member Function Documentation

12.25.3.1 PointBuffer * PointValidityProvider::GetPointBuffer () [protected, inherited]

Returns the point buffer of the Z axis.

References OpenGPS::PointValidityProvider::m_PointBuffer.

12.25.3.2 const PointBuffer * PointValidityProvider::GetPointBuffer () const [protected, inherited]

Returns the point buffer of the Z axis.

References OpenGPS::PointValidityProvider::m_PointBuffer.

Referenced by OpenGPS::ValidBuffer::Allocate(), OpenGPS::ValidBuffer::HasInvalidMarks(), OpenGPS::ValidBuffer::IsValid(), OpenGPS::DoubleInlineValidity::IsValid(), IsValid(), OpenGPS::Int32ValidBuffer::SetValid(), OpenGPS::Int16ValidBuffer::SetValid(), OpenGPS::ValidBuffer::SetValid(), OpenGPS::DoubleInlineValidity::SetValid(), and SetValid().

12.25.3.3 OGPS_Boolean FloatInlineValidity::IsValid (const unsigned int *index*) const throw (...) [virtual]

Gets the validity of a point vector at a given location.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

index The location of the point data the validity is checked.

Returns:

Returns TRUE if valid, FALSE otherwise.

Implements OpenGPS::PointValidityProvider.

References _ASSERT, OpenGPS::PointBuffer::Get(), and OpenGPS::PointValidityProvider::GetPointBuffer().

Here is the call graph for this function:

12.25.3.4 void FloatInlineValidity::SetValid (const unsigned int *index*, const OGPS_Boolean *value*) throw (...) [virtual]

Sets the validity of a point vector at a given location.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

index The location of the point data the validity is updated.

value The value of the current validity of the point vector at the given position.

Implements OpenGPS::PointValidityProvider.

References _ASSERT, OpenGPS::PointValidityProvider::GetPointBuffer(), and OpenGPS::PointBuffer::Set().

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- inline_validity.hxx
- inline_validity.cxx

12.26 OpenGPS::FloatPointBuffer Class Reference

`#include <float_point_buffer.hxx>`

Inheritance diagram for OpenGPS::FloatPointBuffer:

Collaboration diagram for OpenGPS::FloatPointBuffer:

12.26.1 Detailed Description

Manages static memory and typesafe access.

Allocates an internal memory buffer to store point data of type OGPS_Float.

Public Member Functions

- virtual void Allocate (const OGPS_ULong size) throw (...)

  Allocates internal memory.

- FloatPointBuffer ()

  Create a new instance.

- virtual void Get (const OGPS_ULong index, OGPS_Double &value) const throw (...)

  Gets the value of the inernal memory at the given position.

- virtual void Get (const OGPS_ULong index, OGPS_Int32 &value) const throw (...)

Gets the value of the inernal memory at the given position.

- virtual void Get (const OGPS_ULong index, OGPS_Int16 &value) const throw (...)
  Gets the value of the inernal memory at the given position.
- virtual void Get (const OGPS_ULong index, OGPS_Float &value) const throw (...)
  Gets the value of the inernal memory at the given position.
- virtual OGPS_DataPointType GetPointType () const
  Gets the type of point data that can be stored within this instance.
- virtual OGPS_ULong GetSize () const
  Gets the maximum amount of point data taht can be stored.
- virtual void Set (const OGPS_ULong index, const OGPS_Double value) throw (...)
  Sets the value of the inernal memory at the given position.
- virtual void Set (const OGPS_ULong index, const OGPS_Int32 value) throw (...)
  Sets the value of the inernal memory at the given position.
- virtual void Set (const OGPS_ULong index, const OGPS_Int16 value) throw (...)
  Sets the value of the inernal memory at the given position.
- virtual void Set (const OGPS_ULong index, const OGPS_Float value) throw (...)
  Sets the value of the inernal memory at the given position.
- virtual ~FloatPointBuffer ()
  Destroys this instance.

Protected Member Functions

- OpenGPS::UnsignedBytePtr Allocate (const OGPS_ULong size, const size_t typeSize) throw (...)
  Allocates internal memory.
- void Free (OpenGPS::UnsignedBytePtr *value)
  Frees allocated memory.

Private Attributes

- OGPS_Float * m_Buffer
  Pointer to internal memory.

12.26.2 Constructor & Destructor Documentation

12.26.2.1 FloatPointBuffer::FloatPointBuffer ()

Create a new instance.

References m_Buffer, and NULL.

12.26.2.2 FloatPointBuffer::~FloatPointBuffer () [virtual]

Destroys this instance.

References OpenGPS::PointBuffer::Free(), and m_Buffer.

Here is the call graph for this function:

12.26.3 Member Function Documentation

12.26.3.1 OpenGPS::UnsignedBytePtr PointBuffer::Allocate (const OGPS_ULong *size*, const size_t *typeSize*) throw (...) [protected, inherited]

Allocates internal memory.

Remarks:

Allocated memory must be free by PointBuffer::Free manually.

Throws an exception on failure.

Parameters:

size Amount of point data to be stored.

typeSize The size of the data type of the point data stored herein in bytes.

Returns:

Returns a pointer to allocated memory or NULL.

References _ASSERT, _EX_T, OpenGPS::PointBuffer::m_Size, and OGPS_ExGeneral.

12.26.3.2 void FloatPointBuffer::Allocate (const OGPS_ULong *size*) throw (...) [virtual]

Allocates internal memory.

Throws an exception on failure.

Parameters:

size Amount of point data to be stored.

Reimplemented from OpenGPS::PointBuffer.

References _ASSERT, OpenGPS::PointBuffer::Allocate(), and m_Buffer.

Here is the call graph for this function:

12.26.3.3 void PointBuffer::Free (OpenGPS::UnsignedBytePtr * *value*) [protected, inherited]

Frees allocated memory.

See also:

PointBuffer::Allocate

Parameters:

value The pointer to the memory to be freed.

References _OPENGPS_FREE.

Referenced by OpenGPS::DoublePointBuffer::~DoublePointBuffer(), ~FloatPointBuffer(), OpenGPS::Int16PointBuffer::~Int16PointBuffer(), and OpenGPS::Int32PointBuffer::~Int32PointBuffer().

12.26.3.4 void PointBuffer::Get (const OGPS_ULong *index*, OGPS_Double & *value*) const throw (...) [virtual, inherited]

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented in OpenGPS::DoublePointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.26.3.5 void PointBuffer::Get (const OGPS_ULong *index*, OGPS_Int32 & *value*) const throw (...) [virtual, inherited]

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented in OpenGPS::Int32PointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.26.3.6 void PointBuffer::Get (const OGPS_ULong *index*, OGPS_Int16 & *value*) const throw (...) [virtual, inherited]

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented in OpenGPS::Int16PointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

Referenced by OpenGPS::PointVectorProxy::DataPointProxy::Get(), OpenGPS::DoubleInlineValidity::IsValid(), and OpenGPS::FloatInlineValidity::IsValid().

12.26.3.7 void FloatPointBuffer::Get (const OGPS_ULong *index*, OGPS_Float & *value*) const throw (...) [virtual]

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented from OpenGPS::PointBuffer.

References _ASSERT, OpenGPS::PointBuffer::GetSize(), and m_Buffer.

Here is the call graph for this function:

12.26.3.8 OGPS_DataPointType FloatPointBuffer::GetPointType () const [virtual]

Gets the type of point data that can be stored within this instance.

Reimplemented from OpenGPS::PointBuffer.

References OGPS_FloatPointType.

12.26.3.9 OGPS_ULong PointBuffer::GetSize () const [virtual, inherited]

Gets the maximum amount of point data taht can be stored.

References OpenGPS::PointBuffer::m_Size.

Referenced by OpenGPS::ValidBuffer::Allocate(), OpenGPS::Int32PointBuffer::Get(), OpenGPS::Int16PointBuffer::Get(), Get(), OpenGPS::DoublePointBuffer::Get(), OpenGPS::ValidBuffer::HasInvalidMarks(), OpenGPS::Int32PointBuffer::Set(), OpenGPS::Int16PointBuffer::Set(), Set(), and OpenGPS::DoublePointBuffer::Set().

12.26.3.10 void PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Double *value*) throw (...) [virtual, inherited]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented in OpenGPS::DoublePointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.26.3.11 void PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Int32 *value*) throw (...) [virtual, inherited]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented in OpenGPS::Int32PointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.26.3.12 void PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Int16 *value*) throw (...) [virtual, inherited]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented in OpenGPS::Int16PointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

Referenced by OpenGPS::PointVectorProxy::DataPointProxy::Set(), OpenGPS::Int32ValidBuffer::SetValid(), OpenGPS::Int16ValidBuffer::SetValid(), OpenGPS::DoubleInlineValidity::SetValid(), and OpenGPS::FloatInlineValidity::SetValid().

12.26.3.13 void FloatPointBuffer::Set (const OGPS_ULong *index*, const OGPS_Float *value*) throw (...) [virtual]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented from OpenGPS::PointBuffer.

References _ASSERT, OpenGPS::PointBuffer::GetSize(), and m_Buffer.

Here is the call graph for this function:

12.26.4 Member Data Documentation

12.26.4.1 OGPS_Float* OpenGPS::FloatPointBuffer::m_Buffer [private]

Pointer to internal memory.

Referenced by Allocate(), FloatPointBuffer(), Get(), Set(), and ~FloatPointBuffer().

The documentation for this class was generated from the following files:

- float_point_buffer.hxx
- float_point_buffer.cxx

12.27 OpenGPS::Info Class Reference

```
#include <info.hxx>
```

12.27.1 Detailed Description

Publishes license text, ownership and similar information.

Static Public Member Functions

- static void GetAbout (OpenGPS::String *const text)

  Gets a short message describing the purpose of this software library.

- static void GetCopyright (OpenGPS::String *const text)

  Gets the copyright information of this software library.

- static void GetLicense (OpenGPS::String *const text)

  Gets the license information of this software library.

- static void GetName (OpenGPS::String *const text)

  Gets a short name identifying this software library.

- static void GetVersion (OpenGPS::String *const text)

  Gets the version identifier of this software library.

- static void PrintCopyright ()

  Prints the copyright information of this software library to standard output.

- static void PrintLicense ()

Prints the license information of this software library to standard output.

- static void PrintVersion ()

  Prints the version identifier of this software library to standard output.

Private Member Functions

- Info ()
- ~Info ()

12.27.2 Constructor & Destructor Documentation

12.27.2.1 Info::Info () [private]

12.27.2.2 Info::~Info () [private]

12.27.3 Member Function Documentation

12.27.3.1 void Info::GetAbout (OpenGPS::String *const *text*) [static]

Gets a short message describing the purpose of this software library.

Parameters:

text Contains the resulting message.

References _ASSERT, and _OPENGPS_DESCRIPTION.

Referenced by _tmain(), and ogps_GetAboutInfo().

12.27.3.2 void Info::GetCopyright (OpenGPS::String *const *text*) [static]

Gets the copyright information of this software library.

Parameters:

text Contains the resulting message.

References _ASSERT.

Referenced by ogps_GetCopyrightInfo(), and PrintCopyright().

12.27.3.3 void Info::GetLicense (OpenGPS::String *const *text*) [static]

Gets the license information of this software library.

Parameters:

text Contains the resulting message.

References _ASSERT.

Referenced by ogps_GetLicenseInfo(), and PrintLicense().

12.27.3.4 void Info::GetName (OpenGPS::String *const *text*) [static]

Gets a short name identifying this software library.

Parameters:

text Contains the resulting message.

References _ASSERT, and _OPENGPS_NAME.

Referenced by _tmain(), and ogps_GetNameInfo().

12.27.3.5 void Info::GetVersion (OpenGPS::String *const *text*) [static]

Gets the version identifier of this software library.

Parameters:

text Contains the resulting message.

References _ASSERT, and _OPENGPS_VERSIONSTRING.

Referenced by _tmain(), and ogps_GetVersionInfo().

12.27.3.6 void Info::PrintCopyright () [static]

Prints the copyright information of this software library to standard output.

References GetCopyright(), and OpenGPS::String::ToChar().

Referenced by _tmain(), and ogps_PrintCopyrightInfo().

Here is the call graph for this function:

12.27.3.7 void Info::PrintLicense () [static]

Prints the license information of this software library to standard output.

References GetLicense(), and OpenGPS::String::ToChar().

Referenced by _tmain(), and ogps_PrintLicenseInfo().

Here is the call graph for this function:

12.27.3.8 void Info::PrintVersion () [static]

Prints the version identifier of this software library to standard output.

References _OPENGPS_ID, and OpenGPS::String::ToChar().

Referenced by _tmain(), and ogps_PrintVersionInfo().

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- info.hxx
- info.cxx

12.28 OpenGPS::InputBinaryFileStream Class Reference

```
#include <point_vector_iostream.hxx>
```

Inheritance diagram for OpenGPS::InputBinaryFileStream:

Collaboration diagram for OpenGPS::InputBinaryFileStream:

12.28.1 Detailed Description

A binary stream class used for reading from binary files.

Public Member Functions

- InputBinaryFileStream (const OpenGPS::String &filePath)
  Creates a new instance.
- ~InputBinaryFileStream ()
  Destroys this instance.

Private Types

- typedef std::basic_ifstream< OpenGPS::UnsignedByte > BaseType
  The type of the super class.

Private Attributes

- PointVectorInvariantLocale m_Locale
  Invariant locale.

12.28.2 Member Typedef Documentation

12.28.2.1 typedef std::basic_ifstream<OpenGPS::UnsignedByte> OpenGPS::InputBinaryFileStream::BaseType [private]

The type of the super class.

12.28.3 Constructor & Destructor Documentation

12.28.3.1 InputBinaryFileStream::InputBinaryFileStream (const OpenGPS::String & *filePath*)

Creates a new instance.

Parameters:

filePath Full path to the binary file to be read.

References m_Locale, and OpenGPS::String::ToChar().

Here is the call graph for this function:

12.28.3.2 InputBinaryFileStream::~InputBinaryFileStream ()

Destroys this instance.

12.28.4 Member Data Documentation

12.28.4.1 PointVectorInvariantLocale OpenGPS::InputBinaryFileStream::m_Locale [private]

Invariant locale.

Referenced by InputBinaryFileStream().

The documentation for this class was generated from the following files:

- point_vector_iostream.hxx
- point_vector_iostream.cxx

12.29 OpenGPS::Schemas::ISO5436_2::InstrumentType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.29.1 Detailed Description

Class corresponding to the InstrumentType schema type.

Manufacturer

Accessor and modifier functions for the Manufacturer required element.

Name of the equipment manufacturer

- typedef ::xsd::cxx::tree::traits< Manufacturer_type, wchar_t > Manufacturer_traits
  Element traits type.
- typedef ::xml_schema::token Manufacturer_type
  Element type.
- void Manufacturer (::std::auto_ptr< Manufacturer_type > p)
  Set the element value without copying.
- void Manufacturer (const Manufacturer_type &x)
  Set the element value.
- Manufacturer_type & Manufacturer ()
  Return a read-write reference to the element.
- const Manufacturer_type & Manufacturer () const
  Return a read-only (constant) reference to the element.

Manufacturer

Accessor and modifier functions for the Manufacturer required element.

Name of the equipment manufacturer

- typedef ::xsd::cxx::tree::traits< Manufacturer_type, wchar_t > Manufacturer_traits
  Element traits type.
- typedef ::xml_schema::token Manufacturer_type
  Element type.
- void Manufacturer (::std::auto_ptr< Manufacturer_type > p)
  Set the element value without copying.
- void Manufacturer (const Manufacturer_type &x)
  Set the element value.
- Manufacturer_type & Manufacturer ()
  Return a read-write reference to the element.
- const Manufacturer_type & Manufacturer () const
  Return a read-only (constant) reference to the element.

Model

Accessor and modifier functions for the Model required element.

Name of the machine model used for the measurement

- typedef ::xsd::cxx::tree::traits< Model_type, wchar_t > Model_traits

  Element traits type.

- typedef ::xml_schema::token Model_type

  Element type.

- void Model (::std::auto_ptr< Model_type > p)

  Set the element value without copying.

- void Model (const Model_type &x)

  Set the element value.

- Model_type & Model ()

  Return a read-write reference to the element.

- const Model_type & Model () const

  Return a read-only (constant) reference to the element.

Model

Accessor and modifier functions for the Model required element.

Name of the machine model used for the measurement

- typedef ::xsd::cxx::tree::traits< Model_type, wchar_t > Model_traits

  Element traits type.

- typedef ::xml_schema::token Model_type

  Element type.

- void Model (::std::auto_ptr< Model_type > p)

  Set the element value without copying.

- void Model (const Model_type &x)

  Set the element value.

- Model_type & Model ()

  Return a read-write reference to the element.

- const Model_type & Model () const

  Return a read-only (constant) reference to the element.

Serial

Accessor and modifier functions for the Serial required element.

Serial number of the machine.

- typedef ::xsd::cxx::tree::traits< Serial_type, wchar_t > Serial_traits
  Element traits type.
- typedef ::xml_schema::token Serial_type
  Element type.
- void Serial (::std::auto_ptr< Serial_type > p)
  Set the element value without copying.
- void Serial (const Serial_type &x)
  Set the element value.
- Serial_type & Serial ()
  Return a read-write reference to the element.
- const Serial_type & Serial () const
  Return a read-only (constant) reference to the element.

Serial

Accessor and modifier functions for the Serial required element.

Serial number of the machine.

- typedef ::xsd::cxx::tree::traits< Serial_type, wchar_t > Serial_traits
  Element traits type.
- typedef ::xml_schema::token Serial_type
  Element type.
- void Serial (::std::auto_ptr< Serial_type > p)
  Set the element value without copying.
- void Serial (const Serial_type &x)
  Set the element value.
- Serial_type & Serial ()
  Return a read-write reference to the element.
- const Serial_type & Serial () const
  Return a read-only (constant) reference to the element.

Version

Accessor and modifier functions for the Version required element.

Software and hardware version strings used to create this file.

- typedef ::xsd::cxx::tree::traits< Version_type, wchar_t > Version_traits
  Element traits type.
- typedef ::xml_schema::token Version_type
  Element type.
- void Version (::std::auto_ptr< Version_type > p)
  Set the element value without copying.
- void Version (const Version_type &x)
  Set the element value.
- Version_type & Version ()
  Return a read-write reference to the element.
- const Version_type & Version () const
  Return a read-only (constant) reference to the element.

Version

Accessor and modifier functions for the Version required element.

Software and hardware version strings used to create this file.

- typedef ::xsd::cxx::tree::traits< Version_type, wchar_t > Version_traits
  Element traits type.
- typedef ::xml_schema::token Version_type
  Element type.
- void Version (::std::auto_ptr< Version_type > p)
  Set the element value without copying.
- void Version (const Version_type &x)
  Set the element value.
- Version_type & Version ()
  Return a read-write reference to the element.
- const Version_type & Version () const
  Return a read-only (constant) reference to the element.

Constructors

- virtual InstrumentType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const
  Copy the object polymorphically.
- InstrumentType (const InstrumentType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)
  Copy constructor.
- InstrumentType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)
  Construct an instance from a DOM element.
- InstrumentType (const Manufacturer_type &, const Model_type &, const Serial_type &, const Version_type &)
  Construct an instance from the ultimate base and initializers for required elements and attributes.

Constructors

- virtual InstrumentType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const
  Copy the object polymorphically.
- InstrumentType (const InstrumentType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)
  Copy constructor.
- InstrumentType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)
  Construct an instance from a DOM element.
- InstrumentType (const Manufacturer_type &, const Model_type &, const Serial_type &, const Version_type &)
  Construct an instance from the ultimate base and initializers for required elements and attributes.

Protected Member Functions

- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)
- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)

Private Attributes

- ::xsd::cxx::tree::one< Manufacturer_type > Manufacturer_
- ::xsd::cxx::tree::one< Model_type > Model_
- ::xsd::cxx::tree::one< Serial_type > Serial_
- ::xsd::cxx::tree::one< Version_type > Version_

12.29.2 Member Typedef Documentation

12.29.2.1 typedef ::xsd::cxx::tree::traits< Manufacturer_type, wchar_t > OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer_traits

Element traits type.

12.29.2.2 typedef ::xsd::cxx::tree::traits< Manufacturer_type, wchar_t > OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer_traits

Element traits type.

12.29.2.3 typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_-2::InstrumentType::Manufacturer_type

Element type.

12.29.2.4 typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_-2::InstrumentType::Manufacturer_type

Element type.

12.29.2.5 typedef ::xsd::cxx::tree::traits< Model_type, wchar_t > OpenGPS::Schemas::ISO5436_2::InstrumentType::Model_traits

Element traits type.

12.29.2.6 typedef ::xsd::cxx::tree::traits< Model_type, wchar_t > OpenGPS::Schemas::ISO5436_2::InstrumentType::Model_traits

Element traits type.

12.29.2.7 typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_-2::InstrumentType::Model_type

Element type.

12.29.2.8 typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_-2::InstrumentType::Model_type

Element type.

12.29.2.9 typedef ::xsd::cxx::tree::traits< Serial_type, wchar_t > OpenGPS::Schemas::ISO5436_-2::InstrumentType::Serial_traits

Element traits type.

12.29.2.10 typedef ::xsd::cxx::tree::traits< Serial_type, wchar_t > OpenGPS::Schemas::ISO5436_2::InstrumentType::Serial_traits

Element traits type.

12.29.2.11 typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_-2::InstrumentType::Serial_type

Element type.

12.29.2.12 typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_2::InstrumentType::Serial_type

Element type.

12.29.2.13 typedef ::xsd::cxx::tree::traits< Version_type, wchar_t > OpenGPS::Schemas::ISO5436_2::InstrumentType::Version_traits

Element traits type.

12.29.2.14 typedef ::xsd::cxx::tree::traits< Version_type, wchar_t > OpenGPS::Schemas::ISO5436_2::InstrumentType::Version_traits

Element traits type.

12.29.2.15 typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_2::InstrumentType::Version_type

Element type.

12.29.2.16 typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_2::InstrumentType::Version_type

Element type.

12.29.3 Constructor & Destructor Documentation

12.29.3.1 OpenGPS::Schemas::ISO5436_2::InstrumentType::InstrumentType (const Manufacturer_type & *Manufacturer*, const Model_type & *Model*, const Serial_type & *Serial*, const Version_type & *Version*)

Construct an instance from the ultimate base and initializers for required elements and attributes.

Referenced by _clone().

12.29.3.2 OpenGPS::Schemas::ISO5436_2::InstrumentType::InstrumentType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

References parse().

Here is the call graph for this function:

12.29.3.3 OpenGPS::Schemas::ISO5436_2::InstrumentType::InstrumentType (const InstrumentType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.29.3.4 OpenGPS::Schemas::ISO5436_2::InstrumentType::InstrumentType (const Manufacturer_type &, const Model_type &, const Serial_type &, const Version_type &)

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.29.3.5 OpenGPS::Schemas::ISO5436_2::InstrumentType::InstrumentType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.29.3.6 OpenGPS::Schemas::ISO5436_2::InstrumentType::InstrumentType (const InstrumentType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.29.4 Member Function Documentation

12.29.4.1 virtual InstrumentType* OpenGPS::Schemas::ISO5436_2::InstrumentType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.29.4.2 InstrumentType * OpenGPS::Schemas::ISO5436_2::InstrumentType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References InstrumentType().

Here is the call graph for this function:

12.29.4.3 void OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer (::std::auto_ptr< Manufacturer_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.29.4.4 void OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer (const Manufacturer_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.29.4.5 Manufacturer_type& OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.29.4.6 const Manufacturer_type& OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.29.4.7 void OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer (::std::auto_ptr< Manufacturer_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.29.4.8 void OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer (const Manufacturer_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.29.4.9 Manufacturer_type& OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.29.4.10 const Manufacturer_type& OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.29.4.11 void OpenGPS::Schemas::ISO5436_2::InstrumentType::Model (::std::auto_ptr< Model_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.29.4.12 void OpenGPS::Schemas::ISO5436_2::InstrumentType::Model (const Model_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.29.4.13 Model_type& OpenGPS::Schemas::ISO5436_2::InstrumentType::Model ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.29.4.14 const Model_type& OpenGPS::Schemas::ISO5436_2::InstrumentType::Model () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.29.4.15 void OpenGPS::Schemas::ISO5436_2::InstrumentType::Model (::std::auto_ptr< Model_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.29.4.16 void OpenGPS::Schemas::ISO5436_2::InstrumentType::Model (const Model_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.29.4.17 Model_type& OpenGPS::Schemas::ISO5436_2::InstrumentType::Model ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.29.4.18 const Model_type& OpenGPS::Schemas::ISO5436_2::InstrumentType::Model () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.29.4.19 void OpenGPS::Schemas::ISO5436_2::InstrumentType::parse (::xsd::cxx::xml::dom::parser< wchar_t > &, ::xml_schema::flags) [protected]

12.29.4.20 void OpenGPS::Schemas::ISO5436_2::InstrumentType::parse (::xsd::cxx::xml::dom::parser< wchar_t > & *p*, ::xml_schema::flags *f*) [protected]

References Manufacturer(), Manufacturer_, Model(), Model_, n, Serial(), Serial_, Version(), and Version_.

Referenced by InstrumentType().

Here is the call graph for this function:

12.29.4.21 void OpenGPS::Schemas::ISO5436_2::InstrumentType::Serial (::std::auto_ptr< Serial_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.29.4.22 void OpenGPS::Schemas::ISO5436_2::InstrumentType::Serial (const Serial_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.29.4.23 Serial_type& OpenGPS::Schemas::ISO5436_2::InstrumentType::Serial ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.29.4.24 const Serial_type& OpenGPS::Schemas::ISO5436_2::InstrumentType::Serial () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.29.4.25 void OpenGPS::Schemas::ISO5436_2::InstrumentType::Serial (::std::auto_ptr< Serial_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.29.4.26 void OpenGPS::Schemas::ISO5436_2::InstrumentType::Serial (const Serial_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.29.4.27 Serial_type& OpenGPS::Schemas::ISO5436_2::InstrumentType::Serial ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.29.4.28 const Serial_type& OpenGPS::Schemas::ISO5436_2::InstrumentType::Serial () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.29.4.29 void OpenGPS::Schemas::ISO5436_2::InstrumentType::Version (::std::auto_ptr< Version_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.29.4.30 void OpenGPS::Schemas::ISO5436_2::InstrumentType::Version (const Version_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.29.4.31 Version_type& OpenGPS::Schemas::ISO5436_2::InstrumentType::Version ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.29.4.32 const Version_type& OpenGPS::Schemas::ISO5436_2::InstrumentType::Version () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.29.4.33 void OpenGPS::Schemas::ISO5436_2::InstrumentType::Version (::std::auto_ptr< Version_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.29.4.34 void OpenGPS::Schemas::ISO5436_2::InstrumentType::Version (const Version_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.29.4.35 Version_type& OpenGPS::Schemas::ISO5436_2::InstrumentType::Version ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.29.4.36 const Version_type& OpenGPS::Schemas::ISO5436_2::InstrumentType::Version () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.29.5 Member Data Documentation

12.29.5.1 xsd::cxx::tree::one< Manufacturer_type > OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer_ [private]

Referenced by parse().

12.29.5.2 xsd::cxx::tree::one< Model_type > OpenGPS::Schemas::ISO5436_2::InstrumentType::Model_ [private]

Referenced by parse().

12.29.5.3 xsd::cxx::tree::one< Serial_type > OpenGPS::Schemas::ISO5436_2::InstrumentType::Serial_ [private]

Referenced by parse().

12.29.5.4 xsd::cxx::tree::one< Version_type > OpenGPS::Schemas::ISO5436_2::InstrumentType::Version_ [private]

Referenced by parse().

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.30 OpenGPS::Int16DataPointParser Class Reference

```
#include <int16_data_point_parser.hxx>
```

Inheritance diagram for OpenGPS::Int16DataPointParser:

Collaboration diagram for OpenGPS::Int16DataPointParser:

12.30.1 Detailed Description

Reads/Writes instances of OpenGPS::DataPoint of OGPS_Int16 point data.

Public Member Functions

- Int16DataPointParser ()
  Creates a new instance.
- virtual void Read (PointVectorReaderContext &context, DataPoint &value) throw (...)
  Reads point data from a given context/media.
- virtual void Write (PointVectorWriterContext &context, const DataPoint &value) throw (...)
  Writes point data to a given context/media.
- virtual ~Int16DataPointParser ()
  Destroys this instance.

12.30.2 Constructor & Destructor Documentation

12.30.2.1 Int16DataPointParser::Int16DataPointParser ()

Creates a new instance.

12.30.2.2 Int16DataPointParser::~Int16DataPointParser () `[virtual]`

Destroys this instance.

12.30.3 Member Function Documentation

12.30.3.1 void Int16DataPointParser::Read (PointVectorReaderContext & *context*, DataPoint & *value*) throw (...) `[virtual]`

Reads point data from a given context/media.

Using the appropriate access method of the given implementation of OpenGPS::PointVectorReaderContext the current point value read gets stored in a OpenGPS::DataPoint instance. This operation is typesafe.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

context Provides read access to the media.

value Buffer where the point data read gets stored.

Implements OpenGPS::DataPointParser.

12.30.3.2 void Int16DataPointParser::Write (PointVectorWriterContext & *context*, const DataPoint & *value*) throw (...) `[virtual]`

Writes point data to a given context/media.

Using the appropriate access method of the given implementation of OpenGPS::PointVectorWriterContext the point value currently stored in the OpenGPS::DataPoint instance gets written to the media. This operation is typesafe and will fail if the type of point data stored in the value object does not exacly match the current implementation of the OpenGPS::DataPointParser interface.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

context Provides write access to the media.

value Buffer where the point data to be written is stored.

Implements OpenGPS::DataPointParser.

The documentation for this class was generated from the following files:

- int16_data_point_parser.hxx
- int16_data_point_parser.cxx

12.31 OpenGPS::Int16PointBuffer Class Reference

`#include <int16_point_buffer.hxx>`

Inheritance diagram for OpenGPS::Int16PointBuffer:

Collaboration diagram for OpenGPS::Int16PointBuffer:

12.31.1 Detailed Description

Manages static memory and typesafe access.

Allocates an internal memory buffer to store point data of type OGPS_Int16.

Public Member Functions

- virtual void Allocate (const OGPS_ULong size) throw (...)

  Allocates internal memory.

- virtual void Get (const OGPS_ULong index, OGPS_Double &value) const throw (...)

  Gets the value of the inernal memory at the given position.

- virtual void Get (const OGPS_ULong index, OGPS_Float &value) const throw (...)

  Gets the value of the inernal memory at the given position.

- virtual void Get (const OGPS_ULong index, OGPS_Int32 &value) const throw (...)

  Gets the value of the inernal memory at the given position.

- virtual void Get (const OGPS_ULong index, OGPS_Int16 &value) const throw (...)

  Gets the value of the inernal memory at the given position.

- virtual OGPS_DataPointType GetPointType () const

  Gets the type of point data that can be stored within this instance.

- virtual OGPS_ULong GetSize () const

  Gets the maximum amount of point data taht can be stored.

- Int16PointBuffer ()

  Create a new instance.

- virtual void Set (const OGPS_ULong index, const OGPS_Double value) throw (...)

  Sets the value of the inernal memory at the given position.

- virtual void Set (const OGPS_ULong index, const OGPS_Float value) throw (...)

  Sets the value of the inernal memory at the given position.

- virtual void Set (const OGPS_ULong index, const OGPS_Int32 value) throw (...)

  Sets the value of the inernal memory at the given position.

- virtual void Set (const OGPS_ULong index, const OGPS_Int16 value) throw (...)

  Sets the value of the inernal memory at the given position.

- virtual ~Int16PointBuffer ()

  Destroys this instance.

Protected Member Functions

- OpenGPS::UnsignedBytePtr Allocate (const OGPS_ULong size, const size_t typeSize) throw (...)

  Allocates internal memory.

- void Free (OpenGPS::UnsignedBytePtr *value)

  Frees allocated memory.

Private Attributes

- OGPS_Int16 * m_Buffer

  Pointer to internal memory.

12.31.2 Constructor & Destructor Documentation

12.31.2.1 Int16PointBuffer::Int16PointBuffer ()

Create a new instance.

References m_Buffer, and NULL.

12.31.2.2 Int16PointBuffer::~Int16PointBuffer () [virtual]

Destroys this instance.

References OpenGPS::PointBuffer::Free(), and m_Buffer.

Here is the call graph for this function:

12.31.3 Member Function Documentation

12.31.3.1 OpenGPS::UnsignedBytePtr PointBuffer::Allocate (const OGPS_ULong *size*, const size_t *typeSize*) throw (...) [protected, inherited]

Allocates internal memory.

Remarks:

Allocated memory must be free by PointBuffer::Free manually.

Throws an exception on failure.

Parameters:

size Amount of point data to be stored.

typeSize The size of the data type of the point data stored herein in bytes.

Returns:

Returns a pointer to allocated memory or NULL.

References _ASSERT, _EX_T, OpenGPS::PointBuffer::m_Size, and OGPS_ExGeneral.

12.31.3.2 void Int16PointBuffer::Allocate (const OGPS_ULong *size*) throw (...) [virtual]

Allocates internal memory.

Throws an exception on failure.

Parameters:

size Amount of point data to be stored.

Reimplemented from OpenGPS::PointBuffer.

References _ASSERT, OpenGPS::PointBuffer::Allocate(), and m_Buffer.

Here is the call graph for this function:

12.31.3.3 void PointBuffer::Free (OpenGPS::UnsignedBytePtr * *value*) [protected, inherited]

Frees allocated memory.

See also:

PointBuffer::Allocate

Parameters:

value The pointer to the memory to be freed.

References _OPENGPS_FREE.

Referenced by OpenGPS::DoublePointBuffer::~DoublePointBuffer(), OpenGPS::FloatPointBuffer::~FloatPointBuffer(), ~Int16PointBuffer(), and OpenGPS::Int32PointBuffer::~Int32PointBuffer().

12.31.3.4 void PointBuffer::Get (const OGPS_ULong *index*, OGPS_Double & *value*) const throw (...) [virtual, inherited]

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented in OpenGPS::DoublePointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.31.3.5 void PointBuffer::Get (const OGPS_ULong *index*, OGPS_Float & *value*) const throw (...) [virtual, inherited]

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented in OpenGPS::FloatPointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.31.3.6 void PointBuffer::Get (const OGPS_ULong *index*, OGPS_Int32 & *value*) const throw (...) `[virtual, inherited]`

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented in OpenGPS::Int32PointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.31.3.7 void Int16PointBuffer::Get (const OGPS_ULong *index*, OGPS_Int16 & *value*) const throw (...) `[virtual]`

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented from OpenGPS::PointBuffer.

References _ASSERT, OpenGPS::PointBuffer::GetSize(), and m_Buffer.

Here is the call graph for this function:

12.31.3.8 OGPS_DataPointType Int16PointBuffer::GetPointType () const `[virtual]`

Gets the type of point data that can be stored within this instance.

Reimplemented from OpenGPS::PointBuffer.

References OGPS_Int16PointType.

12.31.3.9 OGPS_ULong PointBuffer::GetSize () const `[virtual, inherited]`

Gets the maximum amount of point data taht can be stored.

References OpenGPS::PointBuffer::m_Size.

Referenced by OpenGPS::ValidBuffer::Allocate(), OpenGPS::Int32PointBuffer::Get(), Get(), OpenGPS::FloatPointBuffer::Get(), OpenGPS::DoublePointBuffer::Get(), OpenGPS::ValidBuffer::HasInvalidMarks(), OpenGPS::Int32PointBuffer::Set(), Set(), OpenGPS::FloatPointBuffer::Set(), and OpenGPS::DoublePointBuffer::Set().

12.31.3.10 void PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Double *value*) throw (...) [virtual, inherited]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented in OpenGPS::DoublePointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.31.3.11 void PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Float *value*) throw (...) [virtual, inherited]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented in OpenGPS::FloatPointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.31.3.12 void PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Int32 *value*) throw (...) [virtual, inherited]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented in OpenGPS::Int32PointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.31.3.13 void Int16PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Int16 *value*) throw (...) [virtual]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented from OpenGPS::PointBuffer.

References _ASSERT, OpenGPS::PointBuffer::GetSize(), and m_Buffer.

Here is the call graph for this function:

12.31.4 Member Data Documentation

12.31.4.1 OGPS_Int16* OpenGPS::Int16PointBuffer::m_Buffer [private]

Pointer to internal memory.

Referenced by Allocate(), Get(), Int16PointBuffer(), Set(), and ~Int16PointBuffer().

The documentation for this class was generated from the following files:

- int16_point_buffer.hxx
- int16_point_buffer.cxx

12.32 OpenGPS::Int16ValidBuffer Class Reference

```
#include <valid_buffer.hxx>
```

Inheritance diagram for OpenGPS::Int16ValidBuffer:

Collaboration diagram for OpenGPS::Int16ValidBuffer:

12.32.1 Detailed Description

Implementation of OpenGPS::ValidBuffer for Z axis of OGPS_Int16 data type.

Public Member Functions

- virtual OGPS_Boolean HasInvalidMarks () const

  Checks whether the current instance serves any point data that is marked as invalid.

- Int16ValidBuffer (PointBuffer *const value)

  Creates a new instance.

- virtual OGPS_Boolean IsAllocated () const

  Returns wheter the bit buffer has already been allocated.

- virtual OGPS_Boolean IsValid (const unsigned int index) const throw (...)

  Gets the validity of a point vector at a given location.

- virtual void Read (std::basic_istream< OpenGPS::UnsignedByte > &stream) throw (...)

  Maps the bit buffer from a binary stream.

- virtual void SetValid (const unsigned int index, const OGPS_Boolean value) throw (...)

  Sets the validity of a point vector at a given location.

- virtual void Write (std::ostream &stream) throw (...)

  Maps the bit buffer to a binary stream.

- ~Int16ValidBuffer ()

  Destroys this instance.

Protected Member Functions

- virtual void Allocate () throw (...)

  Allocates the internal bit array.

- PointBuffer * GetPointBuffer ()

  Returns the point buffer of the Z axis.

- const PointBuffer * GetPointBuffer () const

  Returns the point buffer of the Z axis.

- virtual void Reset ()

  Frees allocated resources.

12.32.2 Constructor & Destructor Documentation

12.32.2.1 Int16ValidBuffer::Int16ValidBuffer (PointBuffer *const *value*)

Creates a new instance.

Parameters:

value The point buffer of the Z axis.

References _ASSERT, OpenGPS::PointBuffer::GetPointType(), and OGPS_Int16PointType.

Here is the call graph for this function:

12.32.2.2 Int16ValidBuffer::~Int16ValidBuffer ()

Destroys this instance.

12.32.3 Member Function Documentation

12.32.3.1 void ValidBuffer::Allocate () throw (...) `[protected, virtual, inherited]`

Allocates the internal bit array.

Initially all point vectors are assumed to be valid.

References _ASSERT, OpenGPS::ValidBuffer::AllocateRaw(), OpenGPS::PointValidityProvider::GetPointBuffer(), OpenGPS::PointBuffer::GetSize(), OpenGPS::ValidBuffer::m_RawSize, and OpenGPS::ValidBuffer::m_-ValidityBuffer.

Referenced by OpenGPS::ValidBuffer::SetValid().

Here is the call graph for this function:

12.32.3.2 PointBuffer * PointValidityProvider::GetPointBuffer () [protected, inherited]

Returns the point buffer of the Z axis.

References OpenGPS::PointValidityProvider::m_PointBuffer.

12.32.3.3 const PointBuffer * PointValidityProvider::GetPointBuffer () const [protected, inherited]

Returns the point buffer of the Z axis.

References OpenGPS::PointValidityProvider::m_PointBuffer.

Referenced by OpenGPS::ValidBuffer::Allocate(), OpenGPS::ValidBuffer::HasInvalidMarks(), OpenGPS::ValidBuffer::IsValid(), OpenGPS::DoubleInlineValidity::IsValid(), OpenGPS::FloatInlineValidity::IsValid(), OpenGPS::Int32ValidBuffer::SetValid(), SetValid(), OpenGPS::ValidBuffer::SetValid(), OpenGPS::DoubleInlineValidity::SetValid(), and OpenGPS::FloatInlineValidity::SetValid().

12.32.3.4 OGPS_Boolean ValidBuffer::HasInvalidMarks () const [virtual, inherited]

Checks whether the current instance serves any point data that is marked as invalid.

Returns:

FALSE if none of the referenced point data is marked as invalid here, otherwise TRUE.

References FALSE, OpenGPS::PointValidityProvider::GetPointBuffer(), OpenGPS::PointBuffer::GetSize(), OpenGPS::ValidBuffer::IsValid(), OpenGPS::ValidBuffer::m_ValidityBuffer, and TRUE.

Here is the call graph for this function:

12.32.3.5 OGPS_Boolean ValidBuffer::IsAllocated () const [virtual, inherited]

Returns wheter the bit buffer has already been allocated.

References OpenGPS::ValidBuffer::m_ValidityBuffer, and NULL.

Referenced by OpenGPS::ISO5436_2Container::CreatePointBuffer().

12.32.3.6 OGPS_Boolean ValidBuffer::IsValid (const unsigned int *index*) const throw (...) [virtual, inherited]

Gets the validity of a point vector at a given location.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

index The location of the point data the validity is checked.

Returns:

Returns TRUE if valid, FALSE otherwise.

Implements OpenGPS::PointValidityProvider.

References _ASSERT, OpenGPS::PointValidityProvider::GetPointBuffer(), OpenGPS::ValidBuffer::m_-ValidityBuffer, and TRUE.

Referenced by OpenGPS::ValidBuffer::HasInvalidMarks().

Here is the call graph for this function:

12.32.3.7 void ValidBuffer::Read (std::basic_istream< OpenGPS::UnsignedByte > & *stream*) throw (...) [virtual, inherited]

Maps the bit buffer from a binary stream.

Parameters:

stream The bit array gets copied from here.

References _EX_T, OpenGPS::ValidBuffer::AllocateRaw(), FALSE, OpenGPS::ValidBuffer::m_-ValidityBuffer, OGPS_ExGeneral, OpenGPS::ValidBuffer::Reset(), and TRUE.

Referenced by OpenGPS::ISO5436_2Container::CreatePointBuffer().

Here is the call graph for this function:

12.32.3.8 void ValidBuffer::Reset () [protected, virtual, inherited]

Frees allocated resources.

References _OPENGPS_FREE, OpenGPS::ValidBuffer::m_RawSize, and OpenGPS::ValidBuffer::m_-ValidityBuffer.

Referenced by OpenGPS::ValidBuffer::Read(), and OpenGPS::ValidBuffer::~ValidBuffer().

12.32.3.9 void Int16ValidBuffer::SetValid (const unsigned int *index*, const OGPS_Boolean *value*) throw (...) [virtual]

Sets the validity of a point vector at a given location.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

index The location of the point data the validity is updated.

value The value of the current validity of the point vector at the given position.

Reimplemented from OpenGPS::ValidBuffer.

References OpenGPS::PointValidityProvider::GetPointBuffer(), OpenGPS::PointBuffer::Set(), and OpenGPS::ValidBuffer::SetValid().

Here is the call graph for this function:

12.32.3.10 void ValidBuffer::Write (std::ostream & *stream*) throw (...) `[virtual, inherited]`

Maps the bit buffer to a binary stream.

Parameters:

stream The internal bit array gets written to the given stream.

Returns:

Returns TRUE on success, FALSE otherwise.

References _ASSERT, _EX_T, OpenGPS::ValidBuffer::m_RawSize, OpenGPS::ValidBuffer::m_-ValidityBuffer, and OGPS_ExGeneral.

Referenced by OpenGPS::ISO5436_2Container::SaveValidPointsLink().

The documentation for this class was generated from the following files:

- valid_buffer.hxx
- valid_buffer.cxx

12.33 OpenGPS::Int32DataPointParser Class Reference

`#include <int32_data_point_parser.hxx>`

Inheritance diagram for OpenGPS::Int32DataPointParser:

Collaboration diagram for OpenGPS::Int32DataPointParser:

12.33.1 Detailed Description

Reads/Writes instances of OpenGPS::DataPoint of OGPS_Int32 point data.

Public Member Functions

- Int32DataPointParser ()
  Creates a new instance.
- virtual void Read (PointVectorReaderContext &context, DataPoint &value) throw (...)
  Reads point data from a given context/media.
- virtual void Write (PointVectorWriterContext &context, const DataPoint &value) throw (...)
  Writes point data to a given context/media.
- virtual ~Int32DataPointParser ()
  Destroys this instance.

12.33.2 Constructor & Destructor Documentation

12.33.2.1 Int32DataPointParser::Int32DataPointParser ()

Creates a new instance.

12.33.2.2 Int32DataPointParser::~Int32DataPointParser () `[virtual]`

Destroys this instance.

12.33.3 Member Function Documentation

12.33.3.1 void Int32DataPointParser::Read (PointVectorReaderContext & *context*, DataPoint & *value*) throw (...) `[virtual]`

Reads point data from a given context/media.

Using the appropriate access method of the given implementation of OpenGPS::PointVectorReaderContext the current point value read gets stored in a OpenGPS::DataPoint instance. This operation is typesafe.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

context Provides read access to the media.

value Buffer where the point data read gets stored.

Implements OpenGPS::DataPointParser.

12.33.3.2 void Int32DataPointParser::Write (PointVectorWriterContext & *context*, const DataPoint & *value*) throw (...) `[virtual]`

Writes point data to a given context/media.

Using the appropriate access method of the given implementation of OpenGPS::PointVectorWriterContext the point value currently stored in the OpenGPS::DataPoint instance gets written to the media. This operation is typesafe and will fail if the type of point data stored in the value object does not exacly match the current implementation of the OpenGPS::DataPointParser interface.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

context Provides write access to the media.

value Buffer where the point data to be written is stored.

Implements OpenGPS::DataPointParser.

The documentation for this class was generated from the following files:

- int32_data_point_parser.hxx
- int32_data_point_parser.cxx

12.34 OpenGPS::Int32PointBuffer Class Reference

`#include <int32_point_buffer.hxx>`

Inheritance diagram for OpenGPS::Int32PointBuffer:

Collaboration diagram for OpenGPS::Int32PointBuffer:

12.34.1 Detailed Description

Manages static memory and typesafe access.

Allocates an internal memory buffer to store point data of type OGPS_Int32.

Public Member Functions

- virtual void Allocate (const OGPS_ULong size) throw (...)
  Allocates internal memory.
- virtual void Get (const OGPS_ULong index, OGPS_Double &value) const throw (...)
  Gets the value of the inernal memory at the given position.
- virtual void Get (const OGPS_ULong index, OGPS_Float &value) const throw (...)
  Gets the value of the inernal memory at the given position.
- virtual void Get (const OGPS_ULong index, OGPS_Int16 &value) const throw (...)
  Gets the value of the inernal memory at the given position.
- virtual void Get (const OGPS_ULong index, OGPS_Int32 &value) const throw (...)
  Gets the value of the inernal memory at the given position.
- virtual OGPS_DataPointType GetPointType () const
  Gets the type of point data that can be stored within this instance.
- virtual OGPS_ULong GetSize () const
  Gets the maximum amount of point data taht can be stored.
- Int32PointBuffer ()
  Create a new instance.
- virtual void Set (const OGPS_ULong index, const OGPS_Double value) throw (...)
  Sets the value of the inernal memory at the given position.
- virtual void Set (const OGPS_ULong index, const OGPS_Float value) throw (...)
  Sets the value of the inernal memory at the given position.
- virtual void Set (const OGPS_ULong index, const OGPS_Int16 value) throw (...)
  Sets the value of the inernal memory at the given position.
- virtual void Set (const OGPS_ULong index, const OGPS_Int32 value) throw (...)
  Sets the value of the inernal memory at the given position.
- virtual ~Int32PointBuffer ()
  Destroys this instance.

Protected Member Functions

- OpenGPS::UnsignedBytePtr Allocate (const OGPS_ULong size, const size_t typeSize) throw (...)
  Allocates internal memory.
- void Free (OpenGPS::UnsignedBytePtr *value)
  Frees allocated memory.

Private Attributes

- OGPS_Int32 * m_Buffer

  Pointer to internal memory.

12.34.2 Constructor & Destructor Documentation

12.34.2.1 Int32PointBuffer::Int32PointBuffer ()

Create a new instance.

References m_Buffer, and NULL.

12.34.2.2 Int32PointBuffer::~Int32PointBuffer () [virtual]

Destroys this instance.

References OpenGPS::PointBuffer::Free(), and m_Buffer.

Here is the call graph for this function:

12.34.3 Member Function Documentation

12.34.3.1 OpenGPS::UnsignedBytePtr PointBuffer::Allocate (const OGPS_ULong *size*, const size_t *typeSize*) throw (...) [protected, inherited]

Allocates internal memory.

Remarks:

Allocated memory must be free by PointBuffer::Free manually.

Throws an exception on failure.

Parameters:

size Amount of point data to be stored.

typeSize The size of the data type of the point data stored herein in bytes.

Returns:

Returns a pointer to allocated memory or NULL.

References _ASSERT, _EX_T, OpenGPS::PointBuffer::m_Size, and OGPS_ExGeneral.

12.34.3.2 void Int32PointBuffer::Allocate (const OGPS_ULong *size*) throw (...) [virtual]

Allocates internal memory.

Throws an exception on failure.

Parameters:

size Amount of point data to be stored.

Reimplemented from OpenGPS::PointBuffer.

References _ASSERT, OpenGPS::PointBuffer::Allocate(), and m_Buffer.

Here is the call graph for this function:

12.34.3.3 void PointBuffer::Free (OpenGPS::UnsignedBytePtr * *value*) [protected, inherited]

Frees allocated memory.

See also:

PointBuffer::Allocate

Parameters:

value The pointer to the memory to be freed.

References _OPENGPS_FREE.

Referenced by OpenGPS::DoublePointBuffer::~DoublePointBuffer(), OpenGPS::FloatPointBuffer::~FloatPointBuffer(), OpenGPS::Int16PointBuffer::~Int16PointBuffer(), and ~Int32PointBuffer().

12.34.3.4 void PointBuffer::Get (const OGPS_ULong *index*, OGPS_Double & *value*) const throw (...) [virtual, inherited]

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented in OpenGPS::DoublePointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.34.3.5 void PointBuffer::Get (const OGPS_ULong *index*, OGPS_Float & *value*) const throw (...) [virtual, inherited]

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented in OpenGPS::FloatPointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.34.3.6 void PointBuffer::Get (const OGPS_ULong *index*, OGPS_Int16 & *value*) const throw (...) [virtual, inherited]

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented in OpenGPS::Int16PointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

Referenced by OpenGPS::PointVectorProxy::DataPointProxy::Get(), OpenGPS::DoubleInlineValidity::IsValid(), and OpenGPS::FloatInlineValidity::IsValid().

12.34.3.7 void Int32PointBuffer::Get (const OGPS_ULong *index*, OGPS_Int32 & *value*) const throw (...) [virtual]

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented from OpenGPS::PointBuffer.

References _ASSERT, OpenGPS::PointBuffer::GetSize(), and m_Buffer.

Here is the call graph for this function:

12.34.3.8 OGPS_DataPointType Int32PointBuffer::GetPointType () const [virtual]

Gets the type of point data that can be stored within this instance.

Reimplemented from OpenGPS::PointBuffer.

References OGPS_Int32PointType.

12.34.3.9 OGPS_ULong PointBuffer::GetSize () const [virtual, inherited]

Gets the maximum amount of point data taht can be stored.

References OpenGPS::PointBuffer::m_Size.

Referenced by OpenGPS::ValidBuffer::Allocate(), Get(), OpenGPS::Int16PointBuffer::Get(), OpenGPS::FloatPointBuffer::Get(), OpenGPS::DoublePointBuffer::Get(), OpenGPS::ValidBuffer::HasInvalidMarks(), Set(), OpenGPS::Int16PointBuffer::Set(), OpenGPS::FloatPointBuffer::Set(), and OpenGPS::DoublePointBuffer::Set().

12.34.3.10 void PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Double *value*) throw (...) [virtual, inherited]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented in OpenGPS::DoublePointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.34.3.11 void PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Float *value*) throw (...) [virtual, inherited]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented in OpenGPS::FloatPointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.34.3.12 void PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Int16 *value*) throw (...) [virtual, inherited]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented in OpenGPS::Int16PointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

Referenced by OpenGPS::PointVectorProxy::DataPointProxy::Set(), OpenGPS::Int32ValidBuffer::SetValid(), OpenGPS::Int16ValidBuffer::SetValid(), OpenGPS::DoubleInlineValidity::SetValid(), and OpenGPS::FloatInlineValidity::SetValid().

12.34.3.13 void Int32PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Int32 *value*) throw (...) [virtual]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented from OpenGPS::PointBuffer.

References _ASSERT, OpenGPS::PointBuffer::GetSize(), and m_Buffer.

Here is the call graph for this function:

12.34.4 Member Data Documentation

12.34.4.1 OGPS_Int32* OpenGPS::Int32PointBuffer::m_Buffer [private]

Pointer to internal memory.

Referenced by Allocate(), Get(), Int32PointBuffer(), Set(), and ~Int32PointBuffer().

The documentation for this class was generated from the following files:

- int32_point_buffer.hxx
- int32_point_buffer.cxx

12.35 OpenGPS::Int32ValidBuffer Class Reference

```
#include <valid_buffer.hxx>
```

Inheritance diagram for OpenGPS::Int32ValidBuffer:

Collaboration diagram for OpenGPS::Int32ValidBuffer:

12.35.1 Detailed Description

Implementation of OpenGPS::ValidBuffer for Z axis of OGPS_Int32 data type.

Public Member Functions

- virtual OGPS_Boolean HasInvalidMarks () const

  Checks whether the current instance serves any point data that is marked as invalid.

- Int32ValidBuffer (PointBuffer *const value)

  Creates a new instance.

- virtual OGPS_Boolean IsAllocated () const

  Returns wheter the bit buffer has already been allocated.

- virtual OGPS_Boolean IsValid (const unsigned int index) const throw (...)

  Gets the validity of a point vector at a given location.

- virtual void Read (std::basic_istream< OpenGPS::UnsignedByte > &stream) throw (...)

  Maps the bit buffer from a binary stream.

- virtual void SetValid (const unsigned int index, const OGPS_Boolean value) throw (...)

  Sets the validity of a point vector at a given location.

- virtual void Write (std::ostream &stream) throw (...)

  Maps the bit buffer to a binary stream.

- ~Int32ValidBuffer ()

  Destroys this instance.

Protected Member Functions

- virtual void Allocate () throw (...)

  Allocates the internal bit array.

- PointBuffer * GetPointBuffer ()

  Returns the point buffer of the Z axis.

- const PointBuffer * GetPointBuffer () const

  Returns the point buffer of the Z axis.

- virtual void Reset ()

  Frees allocated resources.

12.35.2 Constructor & Destructor Documentation

12.35.2.1 Int32ValidBuffer::Int32ValidBuffer (PointBuffer *const *value*)

Creates a new instance.

Parameters:

value The point buffer of the Z axis.

References _ASSERT, OpenGPS::PointBuffer::GetPointType(), and OGPS_Int32PointType.

Here is the call graph for this function:

12.35.2.2 Int32ValidBuffer::~Int32ValidBuffer ()

Destroys this instance.

12.35.3 Member Function Documentation

12.35.3.1 void ValidBuffer::Allocate () throw (...) `[protected, virtual, inherited]`

Allocates the internal bit array.

Initially all point vectors are assumed to be valid.

References _ASSERT, OpenGPS::ValidBuffer::AllocateRaw(), OpenGPS::PointValidityProvider::GetPointBuffer(), OpenGPS::PointBuffer::GetSize(), OpenGPS::ValidBuffer::m_RawSize, and OpenGPS::ValidBuffer::m_-ValidityBuffer.

Referenced by OpenGPS::ValidBuffer::SetValid().

Here is the call graph for this function:

12.35.3.2 PointBuffer * PointValidityProvider::GetPointBuffer () [protected, inherited]

Returns the point buffer of the Z axis.

References OpenGPS::PointValidityProvider::m_PointBuffer.

12.35.3.3 const PointBuffer * PointValidityProvider::GetPointBuffer () const [protected, inherited]

Returns the point buffer of the Z axis.

References OpenGPS::PointValidityProvider::m_PointBuffer.

Referenced by OpenGPS::ValidBuffer::Allocate(), OpenGPS::ValidBuffer::HasInvalidMarks(), OpenGPS::ValidBuffer::IsValid(), OpenGPS::DoubleInlineValidity::IsValid(), OpenGPS::FloatInlineValidity::IsValid(), SetValid(), OpenGPS::Int16ValidBuffer::SetValid(), OpenGPS::ValidBuffer::SetValid(), OpenGPS::DoubleInlineValidity::SetValid(), and OpenGPS::FloatInlineValidity::SetValid().

12.35.3.4 OGPS_Boolean ValidBuffer::HasInvalidMarks () const [virtual, inherited]

Checks whether the current instance serves any point data that is marked as invalid.

Returns:

FALSE if none of the referenced point data is marked as invalid here, otherwise TRUE.

References FALSE, OpenGPS::PointValidityProvider::GetPointBuffer(), OpenGPS::PointBuffer::GetSize(), OpenGPS::ValidBuffer::IsValid(), OpenGPS::ValidBuffer::m_-ValidityBuffer, and TRUE.

Here is the call graph for this function:

12.35.3.5 OGPS_Boolean ValidBuffer::IsAllocated () const [virtual, inherited]

Returns wheter the bit buffer has already been allocated.

References OpenGPS::ValidBuffer::m_ValidityBuffer, and NULL.

Referenced by OpenGPS::ISO5436_2Container::CreatePointBuffer().

12.35.3.6 OGPS_Boolean ValidBuffer::IsValid (const unsigned int *index*) const throw (...) [virtual, inherited]

Gets the validity of a point vector at a given location.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

index The location of the point data the validity is checked.

Returns:

Returns TRUE if valid, FALSE otherwise.

Implements OpenGPS::PointValidityProvider.

References _ASSERT, OpenGPS::PointValidityProvider::GetPointBuffer(), OpenGPS::ValidBuffer::m_-ValidityBuffer, and TRUE.

Referenced by OpenGPS::ValidBuffer::HasInvalidMarks().

Here is the call graph for this function:

12.35.3.7 void ValidBuffer::Read (std::basic_istream< OpenGPS::UnsignedByte > & *stream*) throw (...) [virtual, inherited]

Maps the bit buffer from a binary stream.

Parameters:

stream The bit array gets copied from here.

References _EX_T, OpenGPS::ValidBuffer::AllocateRaw(), FALSE, OpenGPS::ValidBuffer::m_-ValidityBuffer, OGPS_ExGeneral, OpenGPS::ValidBuffer::Reset(), and TRUE.

Referenced by OpenGPS::ISO5436_2Container::CreatePointBuffer().

Here is the call graph for this function:

12.35.3.8 void ValidBuffer::Reset () [protected, virtual, inherited]

Frees allocated resources.

References _OPENGPS_FREE, OpenGPS::ValidBuffer::m_RawSize, and OpenGPS::ValidBuffer::m_-ValidityBuffer.

Referenced by OpenGPS::ValidBuffer::Read(), and OpenGPS::ValidBuffer::~ValidBuffer().

12.35.3.9 void Int32ValidBuffer::SetValid (const unsigned int *index*, const OGPS_Boolean *value*) throw (...) [virtual]

Sets the validity of a point vector at a given location.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

index The location of the point data the validity is updated.

value The value of the current validity of the point vector at the given position.

Reimplemented from OpenGPS::ValidBuffer.

References OpenGPS::PointValidityProvider::GetPointBuffer(), OpenGPS::PointBuffer::Set(), and OpenGPS::ValidBuffer::SetValid().

Here is the call graph for this function:

12.35.3.10 void ValidBuffer::Write (std::ostream & *stream*) throw (...) `[virtual, inherited]`

Maps the bit buffer to a binary stream.

Parameters:

stream The internal bit array gets written to the given stream.

Returns:

Returns TRUE on success, FALSE otherwise.

References _ASSERT, _EX_T, OpenGPS::ValidBuffer::m_RawSize, OpenGPS::ValidBuffer::m_-ValidityBuffer, and OGPS_ExGeneral.

Referenced by OpenGPS::ISO5436_2Container::SaveValidPointsLink().

The documentation for this class was generated from the following files:

- valid_buffer.hxx
- valid_buffer.cxx

12.36 OpenGPS::ISO5436_2 Class Reference

`#include <iso5436_2.hxx>`

Inheritance diagram for OpenGPS::ISO5436_2:

Collaboration diagram for OpenGPS::ISO5436_2:

12.36.1 Detailed Description

Represents the ISO5436-2 XML X3P file format container.

Provides access to the content of an already existing or newly created X3P file container.

Public Member Functions

- virtual void AppendVendorSpecific (const OpenGPS::String &vendorURI, const OpenGPS::String &filePath)

  Add vendorspecific file content to the X3P archive.

- virtual void Close ()

  Closes an open file handle and frees its resources.

- virtual void Create (const Schemas::ISO5436_2::Record1Type &record1, const Schemas::ISO5436_2::Record2Type *record2, const OGPS_ULong listDimension, const OGPS_-Boolean useBinaryData=TRUE) throw (...)

  Creates a new ISO5436-2 XML X3P file.

- virtual void Create (const Schemas::ISO5436_2::Record1Type &record1, const Schemas::ISO5436_2::Record2Type *record2, const Schemas::ISO5436_2::MatrixDimensionType &matrixDimension, const OGPS_Boolean useBinaryData=TRUE) throw (...)

  Creates a new ISO5436-2 XML X3P file.

- virtual PointIteratorAutoPtr CreateNextPointIterator () throw (...)

  Creates an iterator to access point data contained in an ISO5436-2 X3P file.

- virtual PointIteratorAutoPtr CreatePrevPointIterator () throw (...)

  Creates an iterator to access point data contained in an ISO5436-2 X3P file.

- virtual OpenGPS::Schemas::ISO5436_2::ISO5436_2Type *const GetDocument ()

  Gets access to the ISO5436_2 XML document.

- virtual void GetListCoord (const OGPS_ULong index, OGPS_Double *const x, OGPS_Double *const y, OGPS_Double *const z) throw (...)

  Gets the fully transformed value of a data point vector at a given index position.

- virtual OGPS_ULong GetListDimension () const throw (...)

  Gets information on the list dimensions of the current data structure.

- virtual void GetListPoint (const OGPS_ULong index, PointVector &vector) throw (...)

  Gets the raw value of a data point vector at a given index position.

- virtual void GetMatrixCoord (const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w, OGPS_Double *const x, OGPS_Double *const y, OGPS_Double *const z) throw (...)

  Gets the fully transformed value of a data point vector at a given surface position.

- virtual void GetMatrixDimensions (OGPS_ULong *const size_u, OGPS_ULong *const size_v, OGPS_ULong *const size_w) const throw (...)

  Gets information on the matrix dimensions of the current data structure The dimensions are written to the arguments size_u, size_v, size_w.

- virtual void GetMatrixPoint (const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w, PointVector &vector) throw (...)

  Gets the raw value of a data point vector at a given surface position.

- virtual OGPS_Boolean GetVendorSpecific (const OpenGPS::String &vendorURI, const OpenGPS::String &fileName, const OpenGPS::String &targetPath)

  Extracts vendorspecific data from the current archive to a given file location.

- virtual OGPS_Boolean IsMatrix () const throw (...)

  Gets information on the structure with which the point measurement data is stored.

- virtual OGPS_Boolean IsMatrixCoordValid (OGPS_ULong u, OGPS_ULong v, OGPS_ULong w) throw (...)

  Asks if there is point vector data stored at the given matrix position.

- ISO5436_2 (const OpenGPS::String &file)

  Creates a new instance.

- ISO5436_2 (const OpenGPS::String &file, const OpenGPS::String &temp)

  Creates a new instance.

- virtual void Open (const OGPS_Boolean readOnly=FALSE) throw (...)

  Opens an existing ISO5436-2 XML X3P file.

- virtual void SetListPoint (const OGPS_ULong index, const PointVector &vector) throw (...)

  Sets the value of a three-dimensional data point vector at a given index position.

- virtual void SetMatrixPoint (const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w, const PointVector *const vector) throw (...)

  Sets the value of a three-dimensional data point vector at a given surface position.

- virtual void Write (const int compressionLevel=-1) throw (...)

  Writes any changes back to the X3P file.

- virtual ~ISO5436_2 ()

  Destructs this object.

Protected Member Functions

- ISO5436_2 (ISO5436_2 *instance=NULL)

  Creates a new instance.

Private Member Functions

- ISO5436_2 (const ISO5436_2 &src)

  The copy-ctor is not implemented.

- ISO5436_2 & operator= (const ISO5436_2 &src)

  The assignment-operator is not implemented.

Private Attributes

- ISO5436_2 * m_Instance

  Internal object instance.

- const OGPS_Boolean m_IsProtected

  When TRUE: do not allow any modifications to the data points.

12.36.2 Constructor & Destructor Documentation

12.36.2.1 ISO5436_2::ISO5436_2 (ISO5436_2 * *instance* = `NULL`) `[protected]`

Creates a new instance.

Parameters:

instance Any method calls on the newly created object will be routed to the given instance.

References m_Instance.

12.36.2.2 ISO5436_2::ISO5436_2 (const OpenGPS::String & *file*, const OpenGPS::String & *temp*)

Creates a new instance.

Remarks:

Use either ISO5436_2::Open or ISO5436_2::Create to work on the ISO5436-2 XML X3P file container.

Parameters:

file Full path to the ISO5436-2 XML X3P to operate on. This file does not need to exist.

temp Specifies a new absolute path to the directory where unpacked X3P data gets stored temporarily.

References m_Instance.

12.36.2.3 ISO5436_2::ISO5436_2 (const OpenGPS::String & *file*)

Creates a new instance.

Remarks:

Use either ISO5436_2::Open or ISO5436_2::Create to work on the ISO5436-2 XML X3P file container.

Parameters:

file Full path to the ISO5436-2 XML X3P to operate on. This file does not need to exist.

References m_Instance.

12.36.2.4 ISO5436_2::~ISO5436_2 () [virtual]

Destructs this object.

References _ASSERT, _OPENGPS_DELETE, Close(), m_Instance, and m_IsProtected.

Here is the call graph for this function:

12.36.2.5 OpenGPS::ISO5436_2::ISO5436_2 (const ISO5436_2 & *src*) [private]

The copy-ctor is not implemented.

This prevents its usage.

12.36.3 Member Function Documentation

12.36.3.1 void ISO5436_2::AppendVendorSpecific (const OpenGPS::String & *vendorURI*, const OpenGPS::String & *filePath*) [virtual]

Add vendorspecific file content to the X3P archive.

Call this before ISO5436_2::Write and the content of a file will be added to the X3P file when written. This can be called multiple times to add more than one file of your choice. These files must exist at the time when ISO5436_2::Write is being executed. The file will be added to the root of the archive with the given file name from the full path specified.

See also:

ISO5436_2::GetVendorSpecific

Parameters:

vendorURI Your very own vendor specifier in a URI conformant format.

filePath The absolute path to the file to be added to the document container.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, AppendVendorSpecific(), and m_Instance.

Referenced by AppendVendorSpecific().

Here is the call graph for this function:

12.36.3.2 void ISO5436_2::Close () [virtual]

Closes an open file handle and frees its resources.

An explicit call of this method is optional. It is executed implicitly when the object of the current instance gets out of scope.

Remarks:

This does not save any changes you made! You must call ISO5436_2::Write before if your changes should be saved.

See also:

ISO5436_2::Create, ISO5436_2::Open, ISO5436_2::Write

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, Close(), and m_Instance.

Referenced by Close(), readonlyExample3(), readonlyExample4(), and ~ISO5436_2().

Here is the call graph for this function:

12.36.3.3 void ISO5436_2::Create (const Schemas::ISO5436_2::Record1Type & *record1*, const Schemas::ISO5436_2::Record2Type * *record2*, const OGPS_ULong *listDimension*, const OGPS_Boolean *useBinaryData* = TRUE) throw (...) [virtual]

Creates a new ISO5436-2 XML X3P file.

Remarks:

The Record3 object defined in the ISO5436_2 XML specification will be created automatically.

Specific implementations may raise an exception.

Parameters:

record1 The Record1 object defined in the ISO5436_2 XML specification. The given object instance must be valid.

record2 The Record2 object defined in the ISO5436_2 XML specification. This is optional, so the parameter can be NULL. But if set, it must point to a valid instance.

listDimension Specifies the size of point measurement data that will be processed.

useBinaryData Defines whether point measurement data will be directly stored into the xml document as tag elements or if it is separately stored in a binary file within the X3P container.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, Create(), and m_Instance.

Here is the call graph for this function:

12.36.3.4 void ISO5436_2::Create (const Schemas::ISO5436_2::Record1Type & *record1*, const Schemas::ISO5436_2::Record2Type * *record2*, const Schemas::ISO5436_2::MatrixDimensionType & *matrixDimension*, const OGPS_Boolean *useBinaryData* = `TRUE`) throw (...) `[virtual]`

Creates a new ISO5436-2 XML X3P file.

Remarks:

The Record3 object defined in the ISO5436_2 XML specification will be created automatically.

Specific implementations may raise an exception.

Parameters:

record1 The Record1 object defined in the ISO5436_2 XML specification. The given object instance must be valid.

record2 The Record2 object defined in the ISO5436_2 XML specification. This is optional, so the parameter can be NULL. But if set, it must point to a valid instance.

matrixDimension Specifies the topology for which point measurement data will be processed.

useBinaryData Defines whether point measurement data will be directly stored into the xml document as tag elements or if it is separately stored in a binary file within the X3P container.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, Create(), and m_Instance.

Referenced by Create().

Here is the call graph for this function:

12.36.3.5 PointIteratorAutoPtr ISO5436_2::CreateNextPointIterator () throw (...) `[virtual]`

Creates an iterator to access point data contained in an ISO5436-2 X3P file.

Iterates the point data in forward direction.

Returns:

Returns an iterator handle on success.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, CreateNextPointIterator(), and m_Instance.

Referenced by CreateNextPointIterator(), and readonlyExample3().

Here is the call graph for this function:

12.36.3.6 PointIteratorAutoPtr ISO5436_2::CreatePrevPointIterator () throw (...) [virtual]

Creates an iterator to access point data contained in an ISO5436-2 X3P file.

Iterates the point data in backward direction.

Returns:

Returns an iterator handle on success.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, CreatePrevPointIterator(), and m_Instance.

Referenced by CreatePrevPointIterator().

Here is the call graph for this function:

12.36.3.7 OpenGPS::Schemas::ISO5436_2::ISO5436_2Type *const ISO5436_2::GetDocument () [virtual]

Gets access to the ISO5436_2 XML document.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, GetDocument(), and m_Instance.

Referenced by GetDocument(), readonlyExample3(), and readonlyExample4().

Here is the call graph for this function:

12.36.3.8 void ISO5436_2::GetListCoord (const OGPS_ULong *index*, OGPS_Double *const *x*, OGPS_Double *const *y*, OGPS_Double *const *z*) throw (...) [virtual]

Gets the fully transformed value of a data point vector at a given index position.

Other than with ogps_GetListPoint this function also applies the axes transformation specified in the axes definition area of the ISO 5436-2 XML document.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

ISO5436_2::GetListCoord is the valid access method only if point vectors are stored in list format. If this is not the case and point vectors are stored in matrix format with encoded topology information, then you must use ISO5436_2::GetMatrixCoord instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ISO5436_2::GetDocument.

See also:

ISO5436_2::GetListPoint, ISO5436_2::GetMatrixCoord

Parameters:

index The index of the surface position.

x Returns the fully transformed x component of the point value at the given index position. If this parameter is set to NULL, the x axis component will be safely ignored.

y Returns the fully transformed y component of the point value at the given index position. If this parameter is set to NULL, the y axis component will be safely ignored.

z Returns the fully transformed z component of the point value at the given index position. If this parameter is set to NULL, the z axis component will be safely ignored.

Returns:

Returns TRUE on success, FALSE otherwise.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, GetListCoord(), and m_Instance.

Referenced by GetListCoord().

Here is the call graph for this function:

12.36.3.9 OGPS_ULong ISO5436_2::GetListDimension () const throw (...) `[virtual]`

Gets information on the list dimensions of the current data structure.

Be sure that the current handle points to a list structure.

Parameters:

handle Operate on this handle object.

Returns:

The dimension of the current list structure.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, GetListDimension(), and m_Instance.

Referenced by GetListDimension().

Here is the call graph for this function:

12.36.3.10 void ISO5436_2::GetListPoint (const OGPS_ULong *index*, PointVector & *vector*) throw (...) [virtual]

Gets the raw value of a data point vector at a given index position.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

ISO5436_2::GetListPoint is the valid access method only if point vectors are stored in list format. If this is not the case and point vectors are stored in matrix format with encoded topology information, then you must use ISO5436_2::GetMatrixPoint instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ISO5436_2::GetDocument.

See also:

ISO5436_2::GetListCoord, ISO5436_2::GetMatrixPoint

Parameters:

index The index of the surface position.

vector Returns the raw point value at the given position.

Returns:

Returns TRUE on success, FALSE otherwise.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, GetListPoint(), and m_Instance.

Referenced by GetListPoint(), and readonlyExample4().

Here is the call graph for this function:

12.36.3.11 void ISO5436_2::GetMatrixCoord (const OGPS_ULong *u*, const OGPS_ULong *v*, const OGPS_ULong *w*, OGPS_Double *const *x*, OGPS_Double *const *y*, OGPS_Double *const *z*) throw (...) [virtual]

Gets the fully transformed value of a data point vector at a given surface position.

Other than with ISO5436_2::GetMatrixPoint this function also applies the axes transformation specified in the axes definition area of the ISO 5436-2 XML document.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

ISO5436_2::GetMatrixCoord is the valid access method only if point vectors are stored in matrix format with encoded topology information. If this is not the case point vectors are stored in list format. Then you must use ISO5436_2::GetListCoord instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ISO5436_2::GetDocument.

See also:

ISO5436_2::GetMatrixPoint, ISO5436_2::GetListCoord

Parameters:

u The u-direction of the surface position.

v The v-direction of the surface position.

w The w-direction of the surface position.

x Returns the fully transformed x component of the point value at the given u,v,w position. If this parameter is set to NULL, the x axis component will be safely ignored.

y Returns the fully transformed y component of the point value at the given u,v,w position. If this parameter is set to NULL, the y axis component will be safely ignored.

z Returns the fully transformed z component of the point value at the given u,v,w position. If this parameter is set to NULL, the z axis component will be safely ignored.

Returns:

Returns TRUE on success, FALSE otherwise.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, GetMatrixCoord(), and m_Instance.

Referenced by GetMatrixCoord().

Here is the call graph for this function:

12.36.3.12 void ISO5436_2::GetMatrixDimensions (OGPS_ULong *const *size_u*, OGPS_ULong *const *size_v*, OGPS_ULong *const *size_w*) const throw (...) [virtual]

Gets information on the matrix dimensions of the current data structure The dimensions are written to the arguments size_u, size_v, size_w.

Be sure that the current handle points to a matrix structure.

Parameters:

handle Operate on this handle object.

size_u Pointer to the return value of the matrix dimension in u-direction

size_v Pointer to the return value of the matrix dimension in v-direction

size_w Pointer to the return value of the matrix dimension in w-direction

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, GetMatrixDimensions(), and m_Instance.

Referenced by GetMatrixDimensions().

Here is the call graph for this function:

12.36.3.13 void ISO5436_2::GetMatrixPoint (const OGPS_ULong *u*, const OGPS_ULong *v*, const OGPS_ULong *w*, PointVector & *vector*) throw (...) `[virtual]`

Gets the raw value of a data point vector at a given surface position.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

ISO5436_2::GetMatrixPoint is the valid access method only if point vectors are stored in matrix format with encoded topology information. If this is not the case point vectors are stored in list format. Then you must use ISO5436_2::GetListPoint instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ISO5436_2::GetDocument.

See also:

ISO5436_2::GetMatrixCoord, ISO5436_2::GetListPoint

Parameters:

u The u-direction of the surface position.

v The v-direction of the surface position.

w The w-direction of the surface position.

vector Returns the raw point value at the given u,v,w position.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, GetMatrixPoint(), and m_Instance.

Referenced by GetMatrixPoint().

Here is the call graph for this function:

12.36.3.14 OGPS_Boolean ISO5436_2::GetVendorSpecific (const OpenGPS::String & *vendorURI*, const OpenGPS::String & *fileName*, const OpenGPS::String & *targetPath*) [virtual]

Extracts vendorspecific data from the current archive to a given file location.

If the current X3P archive contains vendorspecific data registered for a vendorURI under the given filename in the root directory of the archive, the compressed file will be extracted to the given location.

See also:

ISO5436_2::AppendVendorSpecific

Parameters:

vendorURI Your very own vendor specifier in a URI conformant format.

fileName The name of the file to be expected in the root of the archive which is to be decompressed.

targetPath The file in the archive will get extracted here.

Return values:

FALSE if there is no file registered for the given vendorURI within the archive, TRUE if the file has been found and extracted.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, GetVendorSpecific(), and m_Instance.

Referenced by GetVendorSpecific().

Here is the call graph for this function:

12.36.3.15 OGPS_Boolean ISO5436_2::IsMatrix () const throw (...) [virtual]

Gets information on the structure with which the point measurement data is stored.

Returns:

Returns TRUE if point vectors are stored as a matrix structure preserving topology information, or FALSE when point vectors are stored sequentially within a simple list.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, IsMatrix(), and m_Instance.

Referenced by IsMatrix().

Here is the call graph for this function:

12.36.3.16 OGPS_Boolean ISO5436_2::IsMatrixCoordValid (OGPS_ULong *u*, OGPS_ULong *v*, OGPS_ULong *w*) throw (...) [virtual]

Asks if there is point vector data stored at the given matrix position.

Since the matrix storage format encodes topology information there may not exist valid point vector data for every u,v,w position because there was no measurement data available.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

See also:

ISO5436_2::GetMatrixPoint, ISO5436_2::GetMatrixCoord, ISO5436_2::SetMatrixPoint

Parameters:

u The u-direction of the surface position.

v The v-direction of the surface position.

w The w-direction of the surface position.

Returns:

Returns TRUE if the vector point data at the given position is valid, otherwise return FALSE to indicate there is no measurement data available at this particular position.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, IsMatrixCoordValid(), and m_Instance.

Referenced by IsMatrixCoordValid().

Here is the call graph for this function:

12.36.3.17 void ISO5436_2::Open (const OGPS_Boolean *readOnly* = FALSE) throw (...) [virtual]

Opens an existing ISO5436-2 XML X3P file.

See also:

ISO5436_2::Close

Specific implementations may raise an exception.

Parameters:

readOnly If set to TRUE subsequend operations on the current object assume that you will access any obtained data as read-only and won't make any changes. This may speed up some operations. If unsure set this parameter to FALSE.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, m_Instance, and Open().

Referenced by Open(), readonlyExample3(), and readonlyExample4().

Here is the call graph for this function:

12.36.3.18 ISO5436_2& OpenGPS::ISO5436_2::operator= (const ISO5436_2 & *src*) [private]

The assignment-operator is not implemented.

This prevents its usage.

12.36.3.19 void ISO5436_2::SetListPoint (const OGPS_ULong *index*, const PointVector & *vector*) throw (...) [virtual]

Sets the value of a three-dimensional data point vector at a given index position.

Manipulates the point vector data stored in an ISO5436-2 X3P file directly.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

The data types of the values stored in the given vector parameter must correspond with the data types specified in the ISO5436-2 XML document within the axes definition area. Also ISO5436_-2::SetListPoint is the valid access method only if point vectors are stored in list format. If this is not the case and point vectors are stored in matrix format with encoded topology information then you must use ISO5436_2::SetMatrixPoint instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ISO5436_2::GetDocument.

See also:

ISO5436_2::SetMatrixPoint

Parameters:

index The index position of the point vector to manipulate.

vector Set this point value at the given index position.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, m_Instance, and SetListPoint().

Referenced by readonlyExample4(), and SetListPoint().

Here is the call graph for this function:

12.36.3.20 void ISO5436_2::SetMatrixPoint (const OGPS_ULong *u*, const OGPS_ULong *v*, const OGPS_ULong *w*, const PointVector *const *vector*) throw (...) [virtual]

Sets the value of a three-dimensional data point vector at a given surface position.

Manipulates the point vector data stored in an ISO5436-2 X3P file directly.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

The data types of the values stored in the given vector parameter must correspond with the data types specified in the ISO5436-2 XML document within the axes definition area. Also ISO5436_-2::SetMatrixPoint is the valid access method only if point vectors are stored in matrix format with encoded topology information. If this is not the case point vectors are stored in list format. Then you must use ISO5436_2::SetListPoint instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ISO5436_2::GetDocument.

See also:

ISO5436_2::SetListPoint

Parameters:

u The u-direction of the surface position.

v The v-direction of the surface position.

w The w-direction of the surface position.

vector Set this point value at the given u,v,w position. If this parameter is set to NULL, this indicates there is no measurement data available for this position. This is due to the topology encoding properties of the matrix format storage.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, m_Instance, and SetMatrixPoint().

Referenced by SetMatrixPoint().

Here is the call graph for this function:

12.36.3.21 void ISO5436_2::Write (const int *compressionLevel* = -1) throw (...) [virtual]

Writes any changes back to the X3P file.

Call this function before ISO5436_2::Close if you want to store the changes you have made.

See also:

ISO5436_2::Create, ISO5436_2::Open, ISO5436_2::Close

Specific implementations may raise an exception.

Parameters:

compressionLevel Optionally specifies the compression level used when writing the X3P file which is nothing else than a simple zip file container. The default value for this parameter is (-1) which enables standard compression level as a good trade-off between processing time and compression ratio. Values between 0 and 9 are possible. A value of 0 means "no compression" and a value of 9 enables the highest level compression rate at the cost of highest computation time.

Reimplemented in OpenGPS::ISO5436_2Container.

References _ASSERT, m_Instance, and Write().

Referenced by Write().

Here is the call graph for this function:

12.36.4 Member Data Documentation

12.36.4.1 ISO5436_2* OpenGPS::ISO5436_2::m_Instance [private]

Internal object instance.

Either "this" or ISO5436_2::ISO5436_2Container instance.

Referenced by AppendVendorSpecific(), Close(), Create(), CreateNextPointIterator(), CreatePrevPointIterator(), GetDocument(), GetListCoord(), GetListDimension(), GetListPoint(), GetMatrixCoord(), GetMatrixDimensions(), GetMatrixPoint(), GetVendorSpecific(), IsMatrix(), IsMatrixCoordValid(), ISO5436_2(), Open(), SetListPoint(), SetMatrixPoint(), Write(), and ~ISO5436_2().

12.36.4.2 const OGPS_Boolean OpenGPS::ISO5436_2::m_IsProtected [private]

When TRUE: do not allow any modifications to the data points.

Referenced by ~ISO5436_2().

The documentation for this class was generated from the following files:

- iso5436_2.hxx
- iso5436_2.cxx

12.37 OpenGPS::ISO5436_2Container Class Reference

```
#include <iso5436_2_container.hxx>
```

Inheritance diagram for OpenGPS::ISO5436_2Container:

Collaboration diagram for OpenGPS::ISO5436_2Container:

12.37.1 Detailed Description

This is the main gate to this software library.

It provides all manipulation methods to handle X3P archive files.

Public Member Functions

- virtual void AppendVendorSpecific (const OpenGPS::String &vendorURI, const OpenGPS::String &filePath)

  Add vendorspecific file content to the X3P archive.

- virtual void Close ()

  Closes an open file handle and frees its resources.

- virtual void Create (const Schemas::ISO5436_2::Record1Type &record1, const Schemas::ISO5436_2::Record2Type *record2, const OGPS_ULong listDimension, const OGPS_-Boolean useBinaryData=TRUE) throw (...)

  Creates a new ISO5436-2 XML X3P file.

- virtual void Create (const Schemas::ISO5436_2::Record1Type &record1, const Schemas::ISO5436_2::Record2Type *record2, const Schemas::ISO5436_2::MatrixDimensionType &matrixDimension, const OGPS_Boolean useBinaryData=TRUE) throw (...)

  Creates a new ISO5436-2 XML X3P file.

- virtual PointIteratorAutoPtr CreateNextPointIterator () throw (...)

  Creates an iterator to access point data contained in an ISO5436-2 X3P file.

- virtual PointIteratorAutoPtr CreatePrevPointIterator () throw (...)

  Creates an iterator to access point data contained in an ISO5436-2 X3P file.

- virtual OpenGPS::Schemas::ISO5436_2::ISO5436_2Type *const GetDocument ()

  Gets access to the ISO5436_2 XML document.

- virtual void GetListCoord (const OGPS_ULong index, OGPS_Double *const x, OGPS_Double *const y, OGPS_Double *const z) throw (...)

  Gets the fully transformed value of a data point vector at a given index position.

- virtual OGPS_ULong GetListDimension () const throw (...)

Gets information on the list dimensions of the current data structure.

- virtual void GetListPoint (const OGPS_ULong index, PointVector &vector) throw (...)

  Gets the raw value of a data point vector at a given index position.

- virtual void GetMatrixCoord (const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w, OGPS_Double *const x, OGPS_Double *const y, OGPS_Double *const z) throw (...)

  Gets the fully transformed value of a data point vector at a given surface position.

- virtual void GetMatrixDimensions (OGPS_ULong *const size_u, OGPS_ULong *const size_v, OGPS_ULong *const size_w) const throw (...)

  Gets information on the matrix dimensions of the current data structure The dimensions are written to the arguments size_u, size_v, size_w.

- virtual void GetMatrixPoint (const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w, PointVector &vector) throw (...)

  Gets the raw value of a data point vector at a given surface position.

- virtual OGPS_Boolean GetVendorSpecific (const OpenGPS::String &vendorURI, const OpenGPS::String &fileName, const OpenGPS::String &targetPath)

  Extracts vendorspecific data from the current archive to a given file location.

- virtual OGPS_Boolean IsMatrix () const throw (...)

  Gets information on the structure with which the point measurement data is stored.

- virtual OGPS_Boolean IsMatrixCoordValid (OGPS_ULong u, OGPS_ULong v, OGPS_ULong w) throw (...)

  Asks if there is point vector data stored at the given matrix position.

- ISO5436_2Container (const OpenGPS::String &file, const OpenGPS::String &temp)

  Creates a new instance.

- virtual void Open (const OGPS_Boolean readOnly=TRUE) throw (...)

  Opens an existing ISO5436-2 XML X3P file.

- virtual void SetListPoint (const OGPS_ULong index, const PointVector &vector) throw (...)

  Sets the value of a three-dimensional data point vector at a given index position.

- virtual void SetMatrixPoint (const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w, const PointVector *const vector) throw (...)

  Sets the value of a three-dimensional data point vector at a given surface position.

- virtual void Write (const int compressionLevel=-1) throw (...)

  Writes any changes back to the X3P file.

- virtual ~ISO5436_2Container ()

  Destroys this instance.

Protected Member Functions

- virtual void BuildPointVectorParser (PointVectorParserBuilder &builder) const

  Assembles a new OpenGPS::PointVectorParser object using the OpenGPS::PointVectorParserBuilder.

- virtual OGPS_Boolean BuildVectorBuffer (VectorBufferBuilder &builder) const

  Assembles a new OpenGPS::VectorBuffer object using the OpenGPS::VectorBufferBuilder.

- void CheckDocumentInstance () const throw (...)

  Checks for the current document instance and raises an excpetion if it is not instantiated.

- void Compress () throw (...)

  (Over)writes the current X3P archive file with the actual content.

- OpenGPS::String CreateContainerTempFilePath () const

  Creates a unique temporary file name.

- void CreateDocument (const Schemas::ISO5436_2::Record1Type *const record1, const Schemas::ISO5436_2::Record2Type *const record2, const Schemas::ISO5436_2::MatrixDimensionType *const matrixDimension, const OGPS_ULong listDimension, const OGPS_Boolean useBinaryData) throw (...)

  Creates the internal XML document tree structure.

- void CreatePointBuffer () throw (...)

  Sets up the internal memory storage of point data.

- virtual PointVectorProxyContext * CreatePointVectorProxyContext () const throw (...)

  Creates a new instance of a vector proxy context that is used to map point data saved distinctively as one column for every axis definition to one single row vector.

- virtual PointVectorReaderContext * CreatePointVectorReaderContext ()

  Creates an instance of appropriate access methods to read point data depending on the current configuration of the already loaded main ISO5436-2 XML document.

- virtual PointVectorWriterContext * CreatePointVectorWriterContext (zipFile handle) const

  Creates an instance of appropriate access methods to write point data depending on the current configuration of the main ISO5436-2 XML document.

- OGPS_Boolean Decompress (const OpenGPS::String &src, const OpenGPS::String &dst, const OGPS_Boolean fileNotFoundAllowed=FALSE) const throw (...)

  Decompresses a single file within the zip archive.

- void Decompress () throw (...)

  Decompresses and verifies the current X3P archive.

- void DecompressChecksum () throw (...)

  Decompresses the md5 checksum file contained within the X3P archive.

- void DecompressDataBin () throw (...)

  Decompresses the binary point data file contained within the X3P archive.

- void DecompressMain () const throw (...)

  Decompresses the main xml document contained within the X3P archive.

- OGPS_DataPointType GetAxisDataType (const Schemas::ISO5436_2::AxisDescriptionType &axis, const OGPS_Boolean incremental) const

  Gets the data type of point data of an axis.

- OpenGPS::String GetChecksumArchiveName () const

  Gets the relative path of the md5 checksum file within the zip archive.

- OpenGPS::String GetChecksumFileName () const

  Gets the full path to the temporaryily decompressed md5 checksum file.

- const OpenGPS::String & GetFilePath () const

  Gets the file path to the X3P archive the current instance is an interface for.

- OpenGPS::String GetFullFilePath () const

  Gets the absolute file path to the X3P archive the current instance is an interface for.

- OpenGPS::String GetMainArchiveName () const

  Gets the relative path of the main xml document file within the zip archive.

- OpenGPS::String GetMainFileName () const

  Gets the full path to the temporaryily decompressed main xml document file.

- OGPS_ULong GetMaxU () const throw (...)

  Gets the maximum index value possible in X direction.

- OGPS_ULong GetMaxV () const throw (...)

  Gets the maximum index value possible in Y direction.

- OGPS_ULong GetMaxW () const throw (...)

  Gets the maximum index value possible in Z direction.

- OGPS_ULong GetPointCount () const throw (...)

  Gets the amount of point data that is stored in the archive.

- OpenGPS::String GetPointDataArchiveName () const

  Gets the relative path of the binary point data file within the zip archive.

- OpenGPS::String GetPointDataFileName ()

  Gets the full path to the temporaryily decompressed binary point data file.

- const OpenGPS::String & GetTempDir () const

  Gets the full path to the directory where temporary files get stored.

- OpenGPS::String GetValidPointsArchiveName () const

  Gets the relative path of the binary point validity data file within the zip archive.

- OpenGPS::String GetValidPointsFileName ()

Gets the full path to the temporaryily decompressed binary point validity data file.

- OGPS_DataPointType GetXaxisDataType () const

  Gets the data type of point data of the X component of a vector.

- OGPS_DataPointType GetYaxisDataType () const

  Gets the data type of point data of the Y component of a vector.

- OGPS_DataPointType GetZaxisDataType () const

  Gets the data type of point data of the Z component of a vector.

- OGPS_Boolean HasDocument () const

  Returns TRUE if an ISO5436-2 XML document has been loaded into memory as a tree structure, or FALSE if the internal document handle does not exist.

- OGPS_Boolean HasValidPointsLink () const

  Checks whether validity information on point vector data is provided by an extra binary file.

- OGPS_Boolean HasVectorBuffer () const

  Returns TRUE if the internal memory storage area of vector data had been allocated, or FALSE if not.

- OGPS_Boolean IsBinary () const

  Checks whether points are stored as separate binary files or directly within the main ISO5436-2 XML document.

- void ReadDocument () throw (...)

  Creates an instance of the internal ISO5436-2 XML document tree.

- void ReadXmlDocument () throw (...)

  Reads the main ISO5436-2 XML document contained in an X3P archive to the internal document handle as a tree structure.

- void Reset ()

  Releases any resources allocated to handle the xml document tree or internal storage of vectors in memory.

- void ResetValidPointsLink ()

  Rests the valid points link xml tag of the xml document handle to its default value.

- void ResetXmlPointList ()

  Removes the point list xml tag and its content from the xml document handle.

- void SaveChecksumFile (zipFile handle, const OpenGPS::UnsignedByte checksum[16]) throw (...)

  Saves the main md5 checksum to the zip archive.

- void SavePointBuffer (zipFile handle) throw (...)

  Saves the current state of internal memory storage of point data either to the zip archive as an external binary point file or directly into the actual tree structure of the internal document handle of the main ISO5436-2 XML document depending on the current settings.

- void SaveValidPointsLink (zipFile handle) throw (...)

Saves the current state of internal memory storage of point validity data to the zip archive.

- void SaveXmlDocument (zipFile handle) throw (...)

  Writes the content of the internal document handle to the main XML document present in an X3P archive.

Private Types

- typedef std::vector< OpenGPS::String > StringList

Private Member Functions

- OGPS_Boolean ConfigureNamespaceMap (xml_schema::properties &props) const

  Sets local pathes to XSD files corresponding to the global namespaces of the ISO5436-2 XML specification.

- void ConfigureNamespaceMap (xml_schema::namespace_infomap &map) const

  Sets a valid namespace mapping for writing the ISO5436-2 XML document.

- void ConvertPointToCoord (const PointVector &vector, OGPS_Double *const x, OGPS_Double *const y, OGPS_Double *const z) throw (...)

  Extracts the three components of a point vector.

- OGPS_ULong ConvertULongLongToULong (const unsigned long long value) const throw (...)

  Converts a value of unsigned long long into a smaller type.

- OGPS_Int32 ConvertULongToInt32 (const unsigned long long value) const throw (...)

  Converts a value of unsigned long long into a shorter data type.

- void CreateTempDir () throw (...)

  Creates a temporary directory in the file system and sets ISO5436_2Container::m_TempPath.

- double GetIncrementX () const

  Gets the increment of the X axis definition or 1.0 if there is no incremental step.

- double GetIncrementY () const

  Gets the increment of the Y axis definition or 1.0 if there is no incremental step.

- double GetOffsetX () const

  Gets the offset of the X axis definition or 0.0 if there is no offset at all.

- double GetOffsetY () const

  Gets the offset of the Y axis definition or 0.0 if there is no offset at all.

- double GetOffsetZ () const

  Gets the offset of the Z axis definition or 0.0 if there is no offset at all.

- VectorBuffer * GetVectorBuffer ()

  Gets a pointer to the vector buffer or NULL.

- OGPS_Boolean HasTempDir () const

Returns TRUE if a unique temporary directory has been created or FALSE if not.

- OGPS_Boolean IsIncrementalX () const

  Returns TRUE if X has an incremental axis definition, FALSE otherwise.

- OGPS_Boolean IsIncrementalY () const

  Returns TRUE if Y has an incremental axis definition, FALSE otherwise.

- OGPS_Boolean ReadMd5FromFile (const OpenGPS::String &fileName, OpenGPS::UnsignedByte checksum[16]) const

  Reads the first md5 checksum from a file that contains md5 checksums of files.

- void RemoveTempDir ()

  Removes the current temporary directory from the file system and erases all of its content.

- OGPS_ULong SafeMultipilcation (const unsigned long long value1, const unsigned long long value2) const throw (...)

  Multiplication of two values and conversion of the result to a shorter data type.

- void TestChecksums () throw (...)

  Check if all checksums were verified.

- OGPS_Boolean VerifyChecksum (const OpenGPS::String &filePath, const OpenGPS::UnsignedBytePtr checksum, const size_t size) const

  Verifies an 128bit md5 checksum.

- void VerifyDataBinChecksum ()

  Verifies the checksum of the binary point data file.

- void VerifyMainChecksum ()

  Verifies the checksum of the main document ISO5436-2 XML file.

- void VerifyValidBinChecksum ()

  Verifies the checksum of the binary point validity data file.

- OGPS_Boolean WriteVendorSpecific (zipFile handle)

  Writes vendorspecific files to the zip container if any.

Private Attributes

- int m_CompressionLevel

  The level of compression of the zip archive.

- OGPS_Boolean m_DataBinChecksum

  FALSE, if the md5 checksum could not be verified after reading.

- Schemas::ISO5436_2::ISO5436_2Type * m_Document

  The handle to a tree strucure corresponding to an ISO5436-2 XML document file.

- OpenGPS::String m_FilePath

  The path of the X3P archive handles.

- OGPS_Boolean m_IsCreating

  TRUE if an X3P archive is to be created, FALSE if an existing archive has been opened.

- OGPS_Boolean m_IsReadOnly

  TRUE if the user wants readonly access to the X3P file only.

- OGPS_Boolean m_MainChecksum

  FALSE, if the md5 checksum could not be verified after reading.

- OpenGPS::String m_PointDataFileName

  The temporary target path of the uncompressed binary point data file.

- PointVectorAutoPtr m_PointVector

  A point vector which serves as a temporary buffer.

- PointVectorProxyContextAutoPtr m_ProxyContext

  A global point vector proxy used for index based read/writes of point vectors.

- OpenGPS::String m_TempBasePath

  The path to the global directory for temporary files.

- OpenGPS::String m_TempPath

  The path to a temporary directory unique to the current instance.

- OGPS_Boolean m_ValidBinChecksum

  FALSE, if the md5 checksum could not be verified after reading.

- OpenGPS::String m_ValidPointsFileName

  The temporary target path of the uncompressed binary point validity data file.

- VectorBufferBuilderAutoPtr m_VectorBufferBuilder

  The builder used to assemble the current vector buffer.

- StringList m_VendorSpecific

  Vendorspecific file names to be added to the container registered with one single vendor id.

- OpenGPS::String m_VendorURI

  ID of vendorspecific data or empty.

Classes

- class PointIteratorImpl

  Implementation of the point iterator interface.

12.37.2 Member Typedef Documentation

12.37.2.1 typedef std::vector<OpenGPS::String> OpenGPS::ISO5436_2Container::StringList [private]

12.37.3 Constructor & Destructor Documentation

12.37.3.1 ISO5436_2Container::ISO5436_2Container (const OpenGPS::String & *file*, const OpenGPS::String & *temp*)

Creates a new instance.

Parameters:

file The path to the X3P file that is serverd by this instance. The file may not exist yet. In this case ISO5436_2Container::Create methods must be used to create a new X3P file. Otherwise use ISO5436_2Container::Create to get an internal handle to an already existing x3P file container.

temp If not empty this specifies an alternativ directory path to save temporary files. Otherwise the default directory for temporary files used by the operating system is used. Since the file data archived in the X3P file needs to be decompressed before using it, this may eventually take relatively large space from your storage media.

References FALSE, m_CompressionLevel, m_DataBinChecksum, m_Document, m_FilePath, m_IsCreating, m_IsReadOnly, m_MainChecksum, m_TempBasePath, m_ValidBinChecksum, NULL, and TRUE.

12.37.3.2 ISO5436_2Container::~ISO5436_2Container () [virtual]

Destroys this instance.

This closes all open file handles and all changes you may have made to an X3P document get lost unless you previously saved them by executing ISO5436_2Container::Write.

References _OPENGPS_DELETE, m_Document, and OpenGPS::Environment::Reset().

Here is the call graph for this function:

12.37.4 Member Function Documentation

12.37.4.1 void ISO5436_2Container::AppendVendorSpecific (const OpenGPS::String & *vendorURI*, const OpenGPS::String & *filePath*) [virtual]

Add vendorspecific file content to the X3P archive.

Call this before ISO5436_2::Write and the content of a file will be added to the X3P file when written. This can be called multiple times to add more than one file of your choice. These files must exist at the time when ISO5436_2::Write is being executed. The file will be added to the root of the archive with the given file name from the full path specified.

See also:

ISO5436_2::GetVendorSpecific

Parameters:

vendorURI Your very own vendor specifier in a URI conformant format.

filePath The absolute path to the file to be added to the document container.

Reimplemented from OpenGPS::ISO5436_2.

References _ASSERT, _EX_T, CheckDocumentInstance(), m_VendorSpecific, m_VendorURI, and OGPS_ExInvalidOperation.

Here is the call graph for this function:

12.37.4.2 void ISO5436_2Container::BuildPointVectorParser (PointVectorParserBuilder & *builder*) const `[protected, virtual]`

Assembles a new OpenGPS::PointVectorParser object using the OpenGPS::PointVectorParserBuilder.

Parameters:

builder The instance of the builder that is used to create the vector parser.

References OpenGPS::PointVectorParserBuilder::BuildParser(), OpenGPS::PointVectorParserBuilder::BuildX(), OpenGPS::PointVectorParserBuilder::BuildY(), OpenGPS::PointVectorParserBuilder::BuildZ(), GetXaxisDataType(), GetYaxisDataType(), and GetZaxisDataType().

Referenced by CreatePointBuffer(), and SavePointBuffer().

Here is the call graph for this function:

12.37.4.3 OGPS_Boolean ISO5436_2Container::BuildVectorBuffer (VectorBufferBuilder & *builder*) const `[protected, virtual]`

Assembles a new OpenGPS::VectorBuffer object using the OpenGPS::VectorBufferBuilder.

Parameters:

builder The instance of the builder that is used to create the vector buffer.

Returns:

Returns TRUE on success, FALSE otherwise.

References _ASSERT, OpenGPS::VectorBufferBuilder::BuildBuffer(), OpenGPS::VectorBufferBuilder::BuildValidityProvider(), OpenGPS::VectorBufferBuilder::BuildX(), OpenGPS::VectorBufferBuilder::BuildY(), OpenGPS::VectorBufferBuilder::BuildZ(), GetPointCount(), GetXaxisDataType(), GetYaxisDataType(), GetZaxisDataType(), and HasDocument().

Referenced by CreateDocument(), and CreatePointBuffer().

Here is the call graph for this function:

12.37.4.4 void ISO5436_2Container::CheckDocumentInstance () const throw (...) [protected]

Checks for the current document instance and raises an excpetion if it is not instantiated.

References _EX_T, m_Document, and OGPS_ExInvalidOperation.

Referenced by AppendVendorSpecific(), CreateNextPointIterator(), CreatePrevPointIterator(), GetListDimension(), GetListPoint(), GetMatrixDimensions(), GetMatrixPoint(), GetMaxU(), GetMaxV(), GetMaxW(), GetVendorSpecific(), IsMatrix(), IsMatrixCoordValid(), SetListPoint(), SetMatrixPoint(), and Write().

12.37.4.5 void ISO5436_2Container::Close () [virtual]

Closes an open file handle and frees its resources.

An explicit call of this method is optional. It is executed implicitly when the object of the current instance gets out of scope.

Remarks:

This does not save any changes you made! You must call ISO5436_2::Write before if your changes should be saved.

See also:

ISO5436_2::Create, ISO5436_2::Open, ISO5436_2::Write

Reimplemented from OpenGPS::ISO5436_2.

References RemoveTempDir(), and Reset().

Here is the call graph for this function:

12.37.4.6 void ISO5436_2Container::Compress () throw (...) [protected]

(Over)writes the current X3P archive file with the actual content.

References _EX_T, _VERIFY, CreateContainerTempFilePath(), CreateTempDir(), FALSE, GetFullFilePath(), OpenGPS::Environment::GetInstance(), HasValidPointsLink(), NULL, OGPS_ExGeneral, OGPS_ExWarning, RemoveTempDir(), OpenGPS::Environment::RenameFile(), SavePointBuffer(), SaveValidPointsLink(), SaveXmlDocument(), OpenGPS::String::ToChar(), TRUE, and WriteVendorSpecific().

Referenced by Write().

Here is the call graph for this function:

12.37.4.7 OGPS_Boolean ISO5436_2Container::ConfigureNamespaceMap (xml_schema::properties & *props*) const [private]

Sets local pathes to XSD files corresponding to the global namespaces of the ISO5436-2 XML specification.

Parameters:

props Target for the global to local namespace mapping.

Returns:

Returns TRUE on success, FALSE otherwise.

References _OPENGPS_ENV_OPENGPS_LOCATION, _OPENGPS_FILE_URI_PREF, _OPENGPS_ISO5436_LOCATION, _OPENGPS_XSD_ISO5436_NAMESPACE, OpenGPS::Environment::ConcatPathes(), FALSE, OpenGPS::Environment::GetInstance(), OpenGPS::Environment::GetPathName(), and TRUE.

Here is the call graph for this function:

12.37.4.8 void ISO5436_2Container::ConfigureNamespaceMap (xml_schema::namespace_infomap & *map*) const [private]

Sets a valid namespace mapping for writing the ISO5436-2 XML document.

Parameters:

map Target for the valid namespace mapping.

References _OPENGPS_XSD_ISO5436_LOCATION, and _OPENGPS_XSD_ISO5436_NAMESPACE.

Referenced by ReadXmlDocument(), and SaveXmlDocument().

12.37.4.9 void ISO5436_2Container::ConvertPointToCoord (const PointVector & *vector*, OGPS_Double *const *x*, OGPS_Double *const *y*, OGPS_Double *const *z*) throw (...) [private]

Extracts the three components of a point vector.

May throw an OpenGPS::Exception on failure due to type incompatibilities or overflow.

Parameters:

vector The vector which components are to be extracted.

x Target of the value of the X component of that vector. If this equals NULL, this component is ignored.

y Target of the value of the Y component of that vector. If this equals NULL, this component is ignored.

z Target of the value of the Z component of that vector. If this equals NULL, this component is ignored.

References GetIncrementX(), GetIncrementY(), GetOffsetX(), GetOffsetY(), GetOffsetZ(), and NULL.

Referenced by GetListCoord(), and GetMatrixCoord().

Here is the call graph for this function:

12.37.4.10 OGPS_ULong ISO5436_2Container::ConvertULongLongToULong (const unsigned long long *value*) const throw (...) `[private]`

Converts a value of unsigned long long into a smaller type.

Throws an exception on overflow, so this conversion is safe.

Parameters:

value The value to be converted.

Returns:

Returns the same value, but as a cast to a smaller data type.

References _EX_T, and OGPS_ExOverflow.

Referenced by CreatePointVectorProxyContext(), GetListDimension(), GetMatrixDimensions(), GetMaxU(), GetMaxV(), GetMaxW(), and GetPointCount().

12.37.4.11 OGPS_Int32 ISO5436_2Container::ConvertULongToInt32 (const unsigned long long *value*) const throw (...) `[private]`

Converts a value of unsigned long long into a shorter data type.

Throws an exception on overflow, so this conversion is safe.

Parameters:

value The value to be converted.

Returns:

Returns the same value, but casted to a shorter data type.

References _EX_T, and OGPS_ExOverflow.

Referenced by GetListPoint(), and GetMatrixPoint().

12.37.4.12 void ISO5436_2Container::Create (const Schemas::ISO5436_2::Record1Type & *record1*, const Schemas::ISO5436_2::Record2Type * *record2*, const OGPS_ULong *listDimension*, const OGPS_Boolean *useBinaryData* = TRUE) throw (...) [virtual]

Creates a new ISO5436-2 XML X3P file.

Remarks:

The Record3 object defined in the ISO5436_2 XML specification will be created automatically.

Specific implementations may raise an exception.

Parameters:

record1 The Record1 object defined in the ISO5436_2 XML specification. The given object instance must be valid.

record2 The Record2 object defined in the ISO5436_2 XML specification. This is optional, so the parameter can be NULL. But if set, it must point to a valid instance.

listDimension Specifies the size of point measurement data that will be processed.

useBinaryData Defines whether point measurement data will be directly stored into the xml document as tag elements or if it is separately stored in a binary file within the X3P container.

Reimplemented from OpenGPS::ISO5436_2.

References _EX_T, CreateDocument(), HasDocument(), NULL, and OGPS_ExInvalidOperation.

Here is the call graph for this function:

12.37.4.13 void ISO5436_2Container::Create (const Schemas::ISO5436_2::Record1Type & *record1*, const Schemas::ISO5436_2::Record2Type * *record2*, const Schemas::ISO5436_2::MatrixDimensionType & *matrixDimension*, const OGPS_Boolean *useBinaryData* = TRUE) throw (...) [virtual]

Creates a new ISO5436-2 XML X3P file.

Remarks:

The Record3 object defined in the ISO5436_2 XML specification will be created automatically.

Specific implementations may raise an exception.

Parameters:

record1 The Record1 object defined in the ISO5436_2 XML specification. The given object instance must be valid.

record2 The Record2 object defined in the ISO5436_2 XML specification. This is optional, so the parameter can be NULL. But if set, it must point to a valid instance.

matrixDimension Specifies the topology for which point measurement data will be processed.

useBinaryData Defines whether point measurement data will be directly stored into the xml document as tag elements or if it is separately stored in a binary file within the X3P container.

Reimplemented from OpenGPS::ISO5436_2.

References _EX_T, CreateDocument(), HasDocument(), and OGPS_ExInvalidOperation.

Here is the call graph for this function:

12.37.4.14 String ISO5436_2Container::CreateContainerTempFilePath () const [protected]

Creates a unique temporary file name.

References OpenGPS::Environment::ConcatPathes(), OpenGPS::Environment::GetInstance(), GetTempDir(), and OpenGPS::Environment::GetUniqueName().

Referenced by Compress().

Here is the call graph for this function:

12.37.4.15 void ISO5436_2Container::CreateDocument (const Schemas::ISO5436_2::Record1Type *const *record1*, const Schemas::ISO5436_2::Record2Type *const *record2*, const Schemas::ISO5436_2::MatrixDimensionType *const *matrixDimension*, const OGPS_ULong *listDimension*, const OGPS_Boolean *useBinaryData*) throw (...) [protected]

Creates the internal XML document tree structure.

The Record3 and Record4 structures defined in the ISO5436-2 XML specification are created automatically.

Parameters:

record1 The content of the Record1 XML structure.

record2 The content of the Record2 XML structure.

matrixDimension The dimension of the matrix structure or NULL if a list structure is used.

listDimension The size of the data list that stores the point vectors or 0 if the matrix structure is used instead.

useBinaryData Set this to TRUE to store the point data within an external binary file. If set to FALSE point vectors will get storead within the ISO5436-2 XML document directly.

References _ASSERT, _OPENGPS_XSD_ISO5436_DATALINK_PATH, _OPENGPS_XSD_ISO5436_MAIN_CHECKSUM_PATH, BuildVectorBuffer(), CreatePointVectorProxyContext(), FALSE, OpenGPS::VectorBuffer::GetPointVectorProxy(), GetVectorBuffer(), HasDocument(), m_Document, m_IsCreating, m_IsReadOnly, m_PointVector, m_ProxyContext, m_VectorBufferBuilder, NULL, OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2(), OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3(), Reset(), and TRUE.

Referenced by Create().

Here is the call graph for this function:

12.37.4.16 PointIteratorAutoPtr ISO5436_2Container::CreateNextPointIterator () throw (...) [virtual]

Creates an iterator to access point data contained in an ISO5436-2 X3P file.

Iterates the point data in forward direction.

Returns:

Returns an iterator handle on success.

Reimplemented from OpenGPS::ISO5436_2.

References CheckDocumentInstance(), IsMatrix(), and TRUE.

Here is the call graph for this function:

12.37.4.17 void ISO5436_2Container::CreatePointBuffer () throw (...) [protected]

Sets up the internal memory storage of point data.

Creates and allocates the internal vector buffer and fills in point data from either the ISO5436-2 main xml document or from an external binary file.

References _ASSERT, BuildPointVectorParser(), BuildVectorBuffer(), CreatePointVectorProxyContext(), CreatePointVectorReaderContext(), FALSE, OpenGPS::VectorBuffer::GetPointVectorProxy(), OpenGPS::VectorBuffer::GetValidityBuffer(), OpenGPS::VectorBuffer::GetValidityProvider(), GetValidPointsFileName(), GetVectorBuffer(), HasDocument(), OpenGPS::VectorBuffer::HasValidityBuffer(), HasValidPointsLink(), HasVectorBuffer(), OpenGPS::ValidBuffer::IsAllocated(), m_PointVector, m_ProxyContext, m_VectorBufferBuilder, OpenGPS::ValidBuffer::Read(), ResetXmlPointList(), and OpenGPS::PointValidityProvider::SetValid().

Referenced by Open().

Here is the call graph for this function:

12.37.4.18 PointVectorProxyContext * ISO5436_2Container::CreatePointVectorProxyContext () const throw (...) [protected, virtual]

Creates a new instance of a vector proxy context that is used to map point data saved distinctively as one column for every axis definition to one single row vector.

The context object provides the indexing information necessary for that task.

Returns:

A new instance of a vector proxy context. The caller ist responsible of releasing this resource.

References _ASSERT, ConvertULongLongToULong(), HasDocument(), IsMatrix(), m_Document, OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3(), SafeMultipilcation(), OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX(), OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY(), and OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ().

Referenced by CreateDocument(), CreatePointBuffer(), and SavePointBuffer().

Here is the call graph for this function:

12.37.4.19 PointVectorReaderContext * ISO5436_2Container::CreatePointVectorReaderContext () [protected, virtual]

Creates an instance of appropriate access methods to read point data depending on the current configuration of the already loaded main ISO5436-2 XML document.

Returns:

An instance to access raw point data for reading or NULL on failure. The pointer returned must be released by the caller.

References OpenGPS::Environment::GetInstance(), GetPointDataFileName(), IsBinary(), OpenGPS::Environment::IsLittleEndian(), m_Document, NULL, and OpenGPS::Schemas::ISO5436_-2::ISO5436_2Type::Record3().

Referenced by CreatePointBuffer().

Here is the call graph for this function:

12.37.4.20 PointVectorWriterContext * ISO5436_2Container::CreatePointVectorWriterContext (zipFile *handle*) const [protected, virtual]

Creates an instance of appropriate access methods to write point data depending on the current configuration of the main ISO5436-2 XML document.

Parameters:

handle The handle to the zip archive is needed by the special implementation of the context for writing point data to an external binary file.

Returns:

An instance to write raw point data or NULL on failure. The pointer returned must be released by the caller.

References _ASSERT, OpenGPS::Environment::GetInstance(), IsBinary(), m_Document, NULL, and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3().

Referenced by SavePointBuffer().

Here is the call graph for this function:

12.37.4.21 PointIteratorAutoPtr ISO5436_2Container::CreatePrevPointIterator () throw (...) [virtual]

Creates an iterator to access point data contained in an ISO5436-2 X3P file.

Iterates the point data in backward direction.

Returns:

Returns an iterator handle on success.

Reimplemented from OpenGPS::ISO5436_2.

References CheckDocumentInstance(), FALSE, and IsMatrix().

Here is the call graph for this function:

12.37.4.22 void ISO5436_2Container::CreateTempDir () throw (...) [private]

Creates a temporary directory in the file system and sets ISO5436_2Container::m_TempPath.

References _ASSERT, _EX_T, OpenGPS::Environment::ConcatPathes(), OpenGPS::Environment::CreateDir(), FALSE, OpenGPS::Environment::GetInstance(), OpenGPS::Environment::GetTempDir(), OpenGPS::Environment::GetUniqueName(), HasTempDir(), m_TempBasePath, m_TempPath, OGPS_ExGeneral, and OpenGPS::Environment::PathExists().

Referenced by Compress(), and Open().

Here is the call graph for this function:

12.37.4.23 OGPS_Boolean ISO5436_2Container::Decompress (const OpenGPS::String & *src*, const OpenGPS::String & *dst*, const OGPS_Boolean *fileNotFoundAllowed* = `FALSE`) const throw (...) `[protected]`

Decompresses a single file within the zip archive.

Parameters:

src The name of the file to be decompressed. This is the relative path with the archive itself set as the root element.

dst The absolute(!) target path where uncompressed data gets stored on the media.

fileNotFoundAllowed True if it is OK if a file could not be found within the archive. This method proceeds without throwing an exception then and its return value is FALSE.

Returns:

Returns FALSE if a file could not be found in the archive (see the discussion above), TRUE in all other cases.

References _ASSERT, _EX_T, _OPENGPS_FREE, _OPENGPS_ZIP_CHUNK_MAX, _VERIFY, FALSE, GetFullFilePath(), NULL, OGPS_ExGeneral, OpenGPS::String::ToChar(), and TRUE.

Here is the call graph for this function:

12.37.4.24 void ISO5436_2Container::Decompress () throw (...) `[protected]`

Decompresses and verifies the current X3P archive.

Remarks:

If this throws an exception there may exist incorrect and incomplete data. Do call ISO5436_2Container::Reset to avoid an inconsistent state.

References _ASSERT, DecompressChecksum(), DecompressDataBin(), DecompressMain(), HasTempDir(), and ReadDocument().

Referenced by DecompressChecksum(), DecompressDataBin(), DecompressMain(), GetVendorSpecific(), and Open().

Here is the call graph for this function:

12.37.4.25 void ISO5436_2Container::DecompressChecksum () throw (...) [protected]

Decompresses the md5 checksum file contained within the X3P archive.

See also:

ISO5436_2Container::Decompress, ISO5436_2Container::GetChecksumArchiveName, ISO5436_2Container::GetChecksumFileName

References _ASSERT, _VERIFY, Decompress(), GetChecksumArchiveName(), GetChecksumFileName(), HasDocument(), TRUE, and VerifyMainChecksum().

Referenced by Decompress().

Here is the call graph for this function:

12.37.4.26 void ISO5436_2Container::DecompressDataBin () throw (...) [protected]

Decompresses the binary point data file contained within the X3P archive.

See also:

ISO5436_2Container::Decompress, ISO5436_2Container::GetPointDataArchiveName, ISO5436_-2Container::GetPointDataFileName

References _VERIFY, Decompress(), GetPointDataArchiveName(), GetPointDataFileName(), GetValid-PointsArchiveName(), GetValidPointsFileName(), HasValidPointsLink(), IsBinary(), TRUE, Verify-DataBinChecksum(), and VerifyValidBinChecksum().

Referenced by Decompress().

Here is the call graph for this function:

12.37.4.27 void ISO5436_2Container::DecompressMain () const throw (...) [protected]

Decompresses the main xml document contained within the X3P archive.

See also:

ISO5436_2Container::Decompress, ISO5436_2Container::GetMainArchiveName, ISO5436_2Container::GetMainFileName

References _VERIFY, Decompress(), GetMainArchiveName(), GetMainFileName(), and TRUE.

Referenced by Decompress().

Here is the call graph for this function:

12.37.4.28 OGPS_DataPointType ISO5436_2Container::GetAxisDataType (const Schemas::ISO5436_2::AxisDescriptionType & *axis*, const OGPS_Boolean *incremental*) const [protected]

Gets the data type of point data of an axis.

Parameters:

axis The axis description to prove.

incremental TRUE if implict point values are allowed for an incremental axis. This holds true for every axis except the Z axis.

Returns:

Returns the type of data points of an axis or OGPS_MissingPointType on an incremntal axis definition.

References _ASSERT, OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType(), OpenGPS::Schemas::ISO5436_2::DataType::D, OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType(), OpenGPS::Schemas::ISO5436_2::DataType::F, FALSE, OpenGPS::Schemas::ISO5436_2::DataType::I, OpenGPS::Schemas::ISO5436_2::AxisType::I, OpenGPS::Schemas::ISO5436_2::DataType::L, OGPS_DoublePointType, OGPS_FloatPointType, OGPS_Int16PointType, OGPS_Int32PointType, and OGPS_MissingPointType.

Referenced by GetXaxisDataType(), GetYaxisDataType(), and GetZaxisDataType().

Here is the call graph for this function:

12.37.4.29 String ISO5436_2Container::GetChecksumArchiveName () const [protected]

Gets the relative path of the md5 checksum file within the zip archive.

References _ASSERT, HasDocument(), m_Document, and OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type::Record4().

Referenced by DecompressChecksum(), GetChecksumFileName(), and SaveChecksumFile().

Here is the call graph for this function:

12.37.4.30 String ISO5436_2Container::GetChecksumFileName () const [protected]

Gets the full path to the temporaryily decompressed md5 checksum file.

References OpenGPS::Environment::ConcatPathes(), GetChecksumArchiveName(), OpenGPS::Environment::GetInstance(), and GetTempDir().

Referenced by DecompressChecksum(), and VerifyMainChecksum().

Here is the call graph for this function:

12.37.4.31 OpenGPS::Schemas::ISO5436_2::ISO5436_2Type *const ISO5436_-2Container::GetDocument () [virtual]

Gets access to the ISO5436_2 XML document.

Reimplemented from OpenGPS::ISO5436_2.

References m_Document.

12.37.4.32 const OpenGPS::String & ISO5436_2Container::GetFilePath () const [protected]

Gets the file path to the X3P archive the current instance is an interface for.

References m_FilePath.

12.37.4.33 String ISO5436_2Container::GetFullFilePath () const [protected]

Gets the absolute file path to the X3P archive the current instance is an interface for.

References m_FilePath.

Referenced by Compress(), and Decompress().

12.37.4.34 double ISO5436_2Container::GetIncrementX () const [private]

Gets the increment of the X axis definition or 1.0 if there is no incremental step.

References _ASSERT, OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType(), HasDocument(), OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment(), m_Document, and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1().

Referenced by ConvertPointToCoord().

Here is the call graph for this function:

12.37.4.35 double ISO5436_2Container::GetIncrementY () const [private]

Gets the increment of the Y axis definition or 1.0 if there is no incremental step.

References _ASSERT, OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType(), HasDocument(), OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment(), m_Document, and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1().

Referenced by ConvertPointToCoord().

Here is the call graph for this function:

12.37.4.36 void ISO5436_2Container::GetListCoord (const OGPS_ULong *index*, OGPS_Double *const *x*, OGPS_Double *const *y*, OGPS_Double *const *z*) throw (...) [virtual]

Gets the fully transformed value of a data point vector at a given index position.

Other than with ogps_GetListPoint this function also applies the axes transformation specified in the axes definition area of the ISO 5436-2 XML document.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

ISO5436_2::GetListCoord is the valid access method only if point vectors are stored in list format. If this is not the case and point vectors are stored in matrix format with encoded topology information, then you must use ISO5436_2::GetMatrixCoord instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ISO5436_2::GetDocument.

See also:

ISO5436_2::GetListPoint, ISO5436_2::GetMatrixCoord

Parameters:

index The index of the surface position.

x Returns the fully transformed x component of the point value at the given index position. If this parameter is set to NULL, the x axis component will be safely ignored.

y Returns the fully transformed y component of the point value at the given index position. If this parameter is set to NULL, the y axis component will be safely ignored.

z Returns the fully transformed z component of the point value at the given index position. If this parameter is set to NULL, the z axis component will be safely ignored.

Returns:

Returns TRUE on success, FALSE otherwise.

Reimplemented from OpenGPS::ISO5436_2.

References _ASSERT, ConvertPointToCoord(), GetListPoint(), HasDocument(), and IsMatrix().

Here is the call graph for this function:

12.37.4.37 OGPS_ULong ISO5436_2Container::GetListDimension () const throw (...) [virtual]

Gets information on the list dimensions of the current data structure.

Be sure that the current handle points to a list structure.

Parameters:

handle Operate on this handle object.

Returns:

The dimension of the current list structure.

Reimplemented from OpenGPS::ISO5436_2.

References _EX_T, CheckDocumentInstance(), ConvertULongLongToULong(), IsMatrix(), m_Document, OGPS_ExInvalidOperation, and OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type::Record3().

Here is the call graph for this function:

12.37.4.38 void ISO5436_2Container::GetListPoint (const OGPS_ULong *index*, PointVector & *vector*) throw (...) [virtual]

Gets the raw value of a data point vector at a given index position.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

ISO5436_2::GetListPoint is the valid access method only if point vectors are stored in list format. If this is not the case and point vectors are stored in matrix format with encoded topology information, then you must use ISO5436_2::GetMatrixPoint instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ISO5436_2::GetDocument.

See also:

ISO5436_2::GetListCoord, ISO5436_2::GetMatrixPoint

Parameters:

index The index of the surface position.

vector Returns the raw point value at the given position.

Returns:

Returns TRUE on success, FALSE otherwise.

Reimplemented from OpenGPS::ISO5436_2.

References _ASSERT, _EX_T, CheckDocumentInstance(), ConvertULongToInt32(), IsIncrementalX(), IsIncrementalY(), IsMatrix(), m_PointVector, m_ProxyContext, OGPS_ExInvalidOperation, and OGPS_-MissingPointType.

Referenced by OpenGPS::ISO5436_2Container::PointIteratorImpl::GetCurrent(), and GetListCoord().

Here is the call graph for this function:

12.37.4.39 String ISO5436_2Container::GetMainArchiveName () const [protected]

Gets the relative path of the main xml document file within the zip archive.

References _OPENGPS_XSD_ISO5436_MAIN_PATH.

Referenced by DecompressMain(), GetMainFileName(), SaveChecksumFile(), and SaveXmlDocument().

12.37.4.40 String ISO5436_2Container::GetMainFileName () const [protected]

Gets the full path to the temporaryily decompressed main xml document file.

References OpenGPS::Environment::ConcatPathes(), OpenGPS::Environment::GetInstance(), Get-MainArchiveName(), and GetTempDir().

Referenced by DecompressMain(), ReadXmlDocument(), and VerifyMainChecksum().

Here is the call graph for this function:

12.37.4.41 void ISO5436_2Container::GetMatrixCoord (const OGPS_ULong *u*, const OGPS_-ULong *v*, const OGPS_ULong *w*, OGPS_Double *const *x*, OGPS_Double *const *y*, OGPS_Double *const *z*) throw (...) [virtual]

Gets the fully transformed value of a data point vector at a given surface position.

Other than with ISO5436_2::GetMatrixPoint this function also applies the axes transformation specified in the axes definition area of the ISO 5436-2 XML document.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

ISO5436_2::GetMatrixCoord is the valid access method only if point vectors are stored in matrix format with encoded topology information. If this is not the case point vectors are stored in list format. Then you must use ISO5436_2::GetListCoord instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ISO5436_2::GetDocument.

See also:

ISO5436_2::GetMatrixPoint, ISO5436_2::GetListCoord

Parameters:

u The u-direction of the surface position.

v The v-direction of the surface position.

w The w-direction of the surface position.

x Returns the fully transformed x component of the point value at the given u,v,w position. If this parameter is set to NULL, the x axis component will be safely ignored.

y Returns the fully transformed y component of the point value at the given u,v,w position. If this parameter is set to NULL, the y axis component will be safely ignored.

z Returns the fully transformed z component of the point value at the given u,v,w position. If this parameter is set to NULL, the z axis component will be safely ignored.

Returns:

Returns TRUE on success, FALSE otherwise.

Reimplemented from OpenGPS::ISO5436_2.

References _ASSERT, ConvertPointToCoord(), GetMatrixPoint(), HasDocument(), and IsMatrix().

Here is the call graph for this function:

12.37.4.42 void ISO5436_2Container::GetMatrixDimensions (OGPS_ULong *const *size_u*, OGPS_ULong *const *size_v*, OGPS_ULong *const *size_w*) const throw (...) [virtual]

Gets information on the matrix dimensions of the current data structure The dimensions are written to the arguments size_u, size_v, size_w.

Be sure that the current handle points to a matrix structure.

Parameters:

handle Operate on this handle object.

size_u Pointer to the return value of the matrix dimension in u-direction

size_v Pointer to the return value of the matrix dimension in v-direction

size_w Pointer to the return value of the matrix dimension in w-direction

Reimplemented from OpenGPS::ISO5436_2.

References _EX_T, CheckDocumentInstance(), ConvertULongLongToULong(), IsMatrix(), m_Document, OGPS_ExInvalidOperation, and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3().

Here is the call graph for this function:

12.37.4.43 void ISO5436_2Container::GetMatrixPoint (const OGPS_ULong *u*, const OGPS_ULong *v*, const OGPS_ULong *w*, PointVector & *vector*) throw (...) [virtual]

Gets the raw value of a data point vector at a given surface position.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

ISO5436_2::GetMatrixPoint is the valid access method only if point vectors are stored in matrix format with encoded topology information. If this is not the case point vectors are stored in list format. Then you must use ISO5436_2::GetListPoint instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ISO5436_2::GetDocument.

See also:

ISO5436_2::GetMatrixCoord, ISO5436_2::GetListPoint

Parameters:

u The u-direction of the surface position.

v The v-direction of the surface position.

w The w-direction of the surface position.

vector Returns the raw point value at the given u,v,w position.

Reimplemented from OpenGPS::ISO5436_2.

References _ASSERT, _EX_T, CheckDocumentInstance(), ConvertULongToInt32(), OpenGPS::VectorBuffer::GetValidityProvider(), GetVectorBuffer(), IsIncrementalX(), IsIncrementalY(), IsMatrix(), OpenGPS::PointValidityProvider::IsValid(), m_PointVector, m_ProxyContext, OGPS_ExInvalidOperation, and OGPS_MissingPointType.

Referenced by OpenGPS::ISO5436_2Container::PointIteratorImpl::GetCurrent(), and GetMatrixCoord().

Here is the call graph for this function:

12.37.4.44 OGPS_ULong ISO5436_2Container::GetMaxU () const throw (...) [protected]

Gets the maximum index value possible in X direction.

For vector data in matrix topology this is the value of X direction only. This is the value of the size of the data point list in the non-matrix case.

See also:

ISO5436_2Container::IsMatrix.

Returns:

Returns the maximum of an index value in X direction (of matrix topology). There is a fallback to the maximum amount of point vectors contained in the point vector list in the non-matrix case.

References CheckDocumentInstance(), ConvertULongLongToULong(), IsMatrix(), m_Document, and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3().

Referenced by OpenGPS::ISO5436_2Container::PointIteratorImpl::HasNext(), OpenGPS::ISO5436_2Container::PointIteratorImpl::HasPrev(), OpenGPS::ISO5436_2Container::PointIteratorImpl::MoveNext(), and OpenGPS::ISO5436_2Container::PointIteratorImpl::MovePrev().

Here is the call graph for this function:

12.37.4.45 OGPS_ULong ISO5436_2Container::GetMaxV () const throw (...) [protected]

Gets the maximum index value possible in Y direction.

For vector data in matrix topology this is the value of Y direction only. This is the value of the size of the data point list in the non-matrix case.

See also:

ISO5436_2Container::IsMatrix.

Returns:

Returns the maximum of an index value in Y direction (of matrix topology). There is a fallback to the maximum amount of point vectors contained in the point vector list in the non-matrix case.

References CheckDocumentInstance(), ConvertULongLongToULong(), IsMatrix(), m_Document, and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3().

Referenced by OpenGPS::ISO5436_2Container::PointIteratorImpl::HasNext(), OpenGPS::ISO5436_2Container::PointIteratorImpl::HasPrev(), OpenGPS::ISO5436_2Container::PointIteratorImpl::MoveNext(), and OpenGPS::ISO5436_2Container::PointIteratorImpl::MovePrev().

Here is the call graph for this function:

12.37.4.46 OGPS_ULong ISO5436_2Container::GetMaxW () const throw (...) [protected]

Gets the maximum index value possible in Z direction.

For vector data in matrix topology this is the value of Z direction only. This is the value of the size of the data point list in the non-matrix case.

See also:

ISO5436_2Container::IsMatrix.

Returns:

Returns the maximum of an index value in Z direction (of matrix topology). There is a fallback to the maximum amount of point vectors contained in the point vector list in the non-matrix case.

References CheckDocumentInstance(), ConvertULongLongToULong(), IsMatrix(), m_Document, and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3().

Referenced by OpenGPS::ISO5436_2Container::PointIteratorImpl::HasNext(), OpenGPS::ISO5436_2Container::PointIteratorImpl::HasPrev(), OpenGPS::ISO5436_2Container::PointIteratorImpl::MoveNext(), and OpenGPS::ISO5436_2Container::PointIteratorImpl::MovePrev().

Here is the call graph for this function:

12.37.4.47 double ISO5436_2Container::GetOffsetX () const [private]

Gets the offset of the X axis definition or 0.0 if there is no offset at all.

References _ASSERT, HasDocument(), m_Document, OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType::Offset(), and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1().

Referenced by ConvertPointToCoord().

Here is the call graph for this function:

12.37.4.48 double ISO5436_2Container::GetOffsetY () const [private]

Gets the offset of the Y axis definition or 0.0 if there is no offset at all.

References _ASSERT, HasDocument(), m_Document, OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType::Offset(), and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1().

Referenced by ConvertPointToCoord().

Here is the call graph for this function:

12.37.4.49 double ISO5436_2Container::GetOffsetZ () const [private]

Gets the offset of the Z axis definition or 0.0 if there is no offset at all.

References _ASSERT, HasDocument(), m_Document, OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType::Offset(), and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1().

Referenced by ConvertPointToCoord().

Here is the call graph for this function:

12.37.4.50 OGPS_ULong ISO5436_2Container::GetPointCount () const throw (...) [protected]

Gets the amount of point data that is stored in the archive.

References _ASSERT, ConvertULongLongToULong(), HasDocument(), IsMatrix(), m_Document, OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3(), and SafeMultipilcation().

Referenced by BuildVectorBuffer().

Here is the call graph for this function:

12.37.4.51 String ISO5436_2Container::GetPointDataArchiveName () const [protected]

Gets the relative path of the binary point data file within the zip archive.

References _ASSERT, HasDocument(), IsBinary(), m_Document, and OpenGPS::Schemas::ISO5436_-2::ISO5436_2Type::Record3().

Referenced by DecompressDataBin(), and SavePointBuffer().

Here is the call graph for this function:

12.37.4.52 String ISO5436_2Container::GetPointDataFileName () [protected]

Gets the full path to the temporaryily decompressed binary point data file.

References OpenGPS::Environment::ConcatPathes(), OpenGPS::Environment::GetInstance(), GetTempDir(), OpenGPS::Environment::GetUniqueName(), and m_PointDataFileName.

Referenced by CreatePointVectorReaderContext(), DecompressDataBin(), and VerifyDataBinChecksum().

Here is the call graph for this function:

12.37.4.53 const OpenGPS::String & ISO5436_2Container::GetTempDir () const [protected]

Gets the full path to the directory where temporary files get stored.

References _ASSERT, and m_TempPath.

Referenced by CreateContainerTempFilePath(), GetChecksumFileName(), GetMainFileName(), GetPointDataFileName(), and GetValidPointsFileName().

12.37.4.54 String ISO5436_2Container::GetValidPointsArchiveName () const [protected]

Gets the relative path of the binary point validity data file within the zip archive.

References _ASSERT, HasDocument(), HasValidPointsLink(), IsBinary(), m_Document, and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3().

Referenced by DecompressDataBin(), and SaveValidPointsLink().

Here is the call graph for this function:

12.37.4.55 String ISO5436_2Container::GetValidPointsFileName () [protected]

Gets the full path to the temporaryily decompressed binary point validity data file.

References OpenGPS::Environment::ConcatPathes(), OpenGPS::Environment::GetInstance(), GetTempDir(), OpenGPS::Environment::GetUniqueName(), and m_ValidPointsFileName.

Referenced by CreatePointBuffer(), DecompressDataBin(), and VerifyValidBinChecksum().

Here is the call graph for this function:

12.37.4.56 VectorBuffer * ISO5436_2Container::GetVectorBuffer () [private]

Gets a pointer to the vector buffer or NULL.

References _ASSERT, and m_VectorBufferBuilder.

Referenced by CreateDocument(), CreatePointBuffer(), GetMatrixPoint(), IsMatrixCoordValid(), ResetValidPointsLink(), SavePointBuffer(), SaveValidPointsLink(), and SetMatrixPoint().

12.37.4.57 OGPS_Boolean ISO5436_2Container::GetVendorSpecific (const OpenGPS::String & *vendorURI*, const OpenGPS::String & *fileName*, const OpenGPS::String & *targetPath*) [virtual]

Extracts vendorspecific data from the current archive to a given file location.

If the current X3P archive contains vendorspecific data registered for a vendorURI under the given filename in the root directory of the archive, the compressed file will be extracted to the given location.

See also:

ISO5436_2::AppendVendorSpecific

Parameters:

vendorURI Your very own vendor specifier in a URI conformant format.

fileName The name of the file to be expected in the root of the archive which is to be decompressed.

targetPath The file in the archive will get extracted here.

Return values:

FALSE if there is no file registered for the given vendorURI within the archive, TRUE if the file has been found and extracted.

Reimplemented from OpenGPS::ISO5436_2.

References _EX_T, CheckDocumentInstance(), Decompress(), FALSE, m_Document, m_IsCreating, OGPS_ExInvalidArgument, OGPS_ExInvalidOperation, TRUE, and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID().

Here is the call graph for this function:

12.37.4.58 OGPS_DataPointType ISO5436_2Container::GetXaxisDataType () const [protected]

Gets the data type of point data of the X component of a vector.

References _ASSERT, GetAxisDataType(), HasDocument(), m_Document, OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1(), and TRUE.

Referenced by BuildPointVectorParser(), and BuildVectorBuffer().

Here is the call graph for this function:

12.37.4.59 OGPS_DataPointType ISO5436_2Container::GetYaxisDataType () const [protected]

Gets the data type of point data of the Y component of a vector.

References _ASSERT, GetAxisDataType(), HasDocument(), m_Document, OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1(), and TRUE.

Referenced by BuildPointVectorParser(), and BuildVectorBuffer().

Here is the call graph for this function:

12.37.4.60 OGPS_DataPointType ISO5436_2Container::GetZaxisDataType () const [protected]

Gets the data type of point data of the Z component of a vector.

References _ASSERT, FALSE, GetAxisDataType(), HasDocument(), m_Document, and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1().

Referenced by BuildPointVectorParser(), and BuildVectorBuffer().

Here is the call graph for this function:

12.37.4.61 OGPS_Boolean ISO5436_2Container::HasDocument () const [protected]

Returns TRUE if an ISO5436-2 XML document has been loaded into memory as a tree structure, or FALSE if the internal document handle does not exist.

References m_Document, and NULL.

Referenced by BuildVectorBuffer(), Create(), CreateDocument(), CreatePointBuffer(), CreatePointVectorProxyContext(), DecompressChecksum(), GetChecksumArchiveName(), GetIncrementX(), GetIncrementY(), GetListCoord(), GetMatrixCoord(), GetOffsetX(), GetOffsetY(), GetOffsetZ(), GetPointCount(), GetPointDataArchiveName(), GetValidPointsArchiveName(), GetXaxisDataType(), GetYaxisDataType(), GetZaxisDataType(), HasValidPointsLink(), IsBinary(), IsIncrementalX(), IsIncrementalY(), Open(), ReadDocument(), ResetValidPointsLink(), ResetXmlPointList(), SavePointBuffer(), VerifyDataBinChecksum(), VerifyMainChecksum(), and VerifyValidBinChecksum().

12.37.4.62 OGPS_Boolean ISO5436_2Container::HasTempDir () const [private]

Returns TRUE if a unique temporary directory has been created or FALSE if not.

References m_TempPath.

Referenced by CreateTempDir(), Decompress(), and RemoveTempDir().

12.37.4.63 OGPS_Boolean ISO5436_2Container::HasValidPointsLink () const [protected]

Checks whether validity information on point vector data is provided by an extra binary file.

Such a binary file is needed in connection with non-IEEE754 data types (see the axis definition) and when invalid measurements are contained within point data only.

Returns:

Returns TRUE when the existance of such a file needs to be taken into account, FALSE otherwise.

References _ASSERT, HasDocument(), IsBinary(), m_Document, and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3().

Referenced by Compress(), CreatePointBuffer(), DecompressDataBin(), GetValidPointsArchiveName(), SaveValidPointsLink(), and VerifyValidBinChecksum().

Here is the call graph for this function:

12.37.4.64 OGPS_Boolean ISO5436_2Container::HasVectorBuffer () const [protected]

Returns TRUE if the internal memory storage area of vector data had been allocated, or FALSE if not.

References _ASSERT, m_VectorBufferBuilder, and NULL.

Referenced by CreatePointBuffer(), and SavePointBuffer().

12.37.4.65 OGPS_Boolean ISO5436_2Container::IsBinary () const [protected]

Checks whether points are stored as separate binary files or directly within the main ISO5436-2 XML document.

Returns:

Returns TRUE if there exist an external binary file for point data within the archive, otherwise FALSE is returned and the point vectors are saved within the ISO5436-2 XML document also contained within the X3P archive.

References _ASSERT, HasDocument(), m_Document, and OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type::Record3().

Referenced by CreatePointVectorReaderContext(), CreatePointVectorWriterContext(), DecompressDataBin(), GetPointDataArchiveName(), GetValidPointsArchiveName(), HasValidPointsLink(), ResetValidPointsLink(), SavePointBuffer(), VerifyDataBinChecksum(), and VerifyValidBinChecksum().

Here is the call graph for this function:

12.37.4.66 OGPS_Boolean ISO5436_2Container::IsIncrementalX () const [private]

Returns TRUE if X has an incremental axis definition, FALSE otherwise.

If TRUE point data of that axis is known implicitly.

References _ASSERT, OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType(), HasDocument(), m_Document, and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1().

Referenced by GetListPoint(), GetMatrixPoint(), SetListPoint(), and SetMatrixPoint().

Here is the call graph for this function:

12.37.4.67 OGPS_Boolean ISO5436_2Container::IsIncrementalY () const [private]

Returns TRUE if Y has an incremental axis definition, FALSE otherwise.

If TRUE point data of that axis is known implicitly.

References _ASSERT, OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisType(), HasDocument(), m_Document, and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1().

Referenced by GetListPoint(), GetMatrixPoint(), SetListPoint(), and SetMatrixPoint().

Here is the call graph for this function:

12.37.4.68 OGPS_Boolean ISO5436_2Container::IsMatrix () const throw (...) [virtual]

Gets information on the structure with which the point measurement data is stored.

Returns:

Returns TRUE if point vectors are stored as a matrix structure preserving topology information, or FALSE when point vectors are stored sequentially within a simple list.

Reimplemented from OpenGPS::ISO5436_2.

References _ASSERT, _EX_T, CheckDocumentInstance(), FALSE, m_Document, OGPS_-ExInvalidOperation, OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3(), and TRUE.

Referenced by CreateNextPointIterator(), CreatePointVectorProxyContext(), CreatePrevPointIterator(), OpenGPS::ISO5436_2Container::PointIteratorImpl::GetCurrent(), GetListCoord(), GetListDimension(), GetListPoint(), GetMatrixCoord(), GetMatrixDimensions(), GetMatrixPoint(), GetMaxU(), Get-MaxV(), GetMaxW(), GetPointCount(), OpenGPS::ISO5436_2Container::PointIteratorImpl::HasNext(), OpenGPS::ISO5436_2Container::PointIteratorImpl::HasPrev(), IsMatrixCo-ordValid(), OpenGPS::ISO5436_2Container::PointIteratorImpl::MoveNext(), OpenGPS::ISO5436_2Container::PointIteratorImpl::MovePrev(), OpenGPS::ISO5436_-2Container::PointIteratorImpl::SetCurrent(), SetListPoint(), and SetMatrixPoint().

Here is the call graph for this function:

12.37.4.69 OGPS_Boolean ISO5436_2Container::IsMatrixCoordValid (OGPS_ULong *u*, OGPS_-ULong *v*, OGPS_ULong *w*) throw (...) [virtual]

Asks if there is point vector data stored at the given matrix position.

Since the matrix storage format encodes topology information there may not exist valid point vector data for every u,v,w position because there was no measurement data available.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

See also:

ISO5436_2::GetMatrixPoint, ISO5436_2::GetMatrixCoord, ISO5436_2::SetMatrixPoint

Parameters:

u The u-direction of the surface position.

v The v-direction of the surface position.

w The w-direction of the surface position.

Returns:

Returns TRUE if the vector point data at the given position is valid, otherwise return FALSE to indicate there is no measurement data available at this particular position.

Reimplemented from OpenGPS::ISO5436_2.

References _ASSERT, _EX_T, CheckDocumentInstance(), OpenGPS::VectorBuffer::GetValidityProvider(), GetVectorBuffer(), IsMatrix(), OpenGPS::PointValidityProvider::IsValid(), m_ProxyContext, and OGPS_-ExInvalidOperation.

Here is the call graph for this function:

12.37.4.70 void ISO5436_2Container::Open (const OGPS_Boolean *readOnly* = `TRUE`) throw (...) `[virtual]`

Opens an existing ISO5436-2 XML X3P file.

See also:

ISO5436_2::Close

Specific implementations may raise an exception.

Parameters:

readOnly If set to TRUE subsequend operations on the current object assume that you will access any obtained data as read-only and won't make any changes. This may speed up some operations. If unsure set this parameter to FALSE.

Reimplemented from OpenGPS::ISO5436_2.

References _EX_T, CreatePointBuffer(), CreateTempDir(), Decompress(), FALSE, HasDocument(), m_-IsCreating, m_IsReadOnly, OGPS_ExInvalidOperation, RemoveTempDir(), Reset(), and TestChecksums().

Here is the call graph for this function:

12.37.4.71 void ISO5436_2Container::ReadDocument () throw (...) [protected]

Creates an instance of the internal ISO5436-2 XML document tree.

An instance is created from the decompressed main xml document file.

References _ASSERT, HasDocument(), and ReadXmlDocument().

Referenced by Decompress().

Here is the call graph for this function:

12.37.4.72 OGPS_Boolean ISO5436_2Container::ReadMd5FromFile (const OpenGPS::String & *fileName*, OpenGPS::UnsignedByte *checksum*[16]) const [private]

Reads the first md5 checksum from a file that contains md5 checksums of files.

Parameters:

fileName The path to the file that contains md5 checksums.

checksum Target of the extracted checksum.

Returns:

Returns TRUE on success, FALSE otherwise.

References md5().

Referenced by VerifyMainChecksum().

Here is the call graph for this function:

12.37.4.73 void ISO5436_2Container::ReadXmlDocument () throw (...) [protected]

Reads the main ISO5436-2 XML document contained in an X3P archive to the internal document handle as a tree structure.

References _EX_T, ConfigureNamespaceMap(), GetMainFileName(), OpenGPS::Schemas::ISO5436_-2::ISO5436_2(), m_Document, NULL, OGPS_ExGeneral, and OGPS_ExInvalidOperation.

Referenced by ReadDocument().

Here is the call graph for this function:

12.37.4.74 void ISO5436_2Container::RemoveTempDir () [private]

Removes the current temporary directory from the file system and erases all of its content.

References OpenGPS::Environment::GetInstance(), HasTempDir(), m_PointDataFileName, m_TempPath, and m_ValidPointsFileName.

Referenced by Close(), Compress(), and Open().

Here is the call graph for this function:

12.37.4.75 void ISO5436_2Container::Reset () [protected]

Releases any resources allocated to handle the xml document tree or internal storage of vectors in memory.

References _OPENGPS_DELETE, FALSE, m_DataBinChecksum, m_Document, m_IsReadOnly, m_-MainChecksum, m_PointDataFileName, m_PointVector, m_ProxyContext, m_ValidBinChecksum, m_-ValidPointsFileName, m_VectorBufferBuilder, m_VendorSpecific, m_VendorURI, and TRUE.

Referenced by Close(), CreateDocument(), and Open().

12.37.4.76 void ISO5436_2Container::ResetValidPointsLink () [protected]

Rests the valid points link xml tag of the xml document handle to its default value.

References _ASSERT, _OPENGPS_XSD_ISO5436_VALIDPOINTSLINK_PATH, GetVector-Buffer(), HasDocument(), IsBinary(), m_Document, and OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type::Record3().

Referenced by SavePointBuffer().

Here is the call graph for this function:

12.37.4.77 void ISO5436_2Container::ResetXmlPointList () [protected]

Removes the point list xml tag and its content from the xml document handle.

References _ASSERT, HasDocument(), m_Document, and OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type::Record3().

Referenced by CreatePointBuffer(), and SavePointBuffer().

Here is the call graph for this function:

12.37.4.78 OGPS_ULong ISO5436_2Container::SafeMultipilcation (const unsigned long long *value1*, const unsigned long long *value2*) const throw (...) [private]

Multiplication of two values and conversion of the result to a shorter data type.

Throws an exception on overflow, so this conversion is safe.

Parameters:

value1 The first value of the operands.

value2 The second value of the operands.

Returns:

Returns the result of their multiplication, but casted to a shorter data type.

References _EX_T, and OGPS_ExOverflow.

Referenced by CreatePointVectorProxyContext(), and GetPointCount().

12.37.4.79 void ISO5436_2Container::SaveChecksumFile (zipFile *handle*, const OpenGPS::UnsignedByte *checksum*[16]) throw (...) [protected]

Saves the main md5 checksum to the zip archive.

Parameters:

handle The handle to the zip archive where the data is to be stored.

checksum The value of the calculated 128bit md5 checksum.

References _ASSERT, _EX_T, OpenGPS::String::ConvertFromMd5(), FALSE, GetChecksumArchiveName(), GetMainArchiveName(), m_CompressionLevel, NULL, OGPS_ExGeneral, OpenGPS::String::ToChar(), TRUE, and OpenGPS::ZipOutputStream::write().

Referenced by SaveXmlDocument().

Here is the call graph for this function:

12.37.4.80 void ISO5436_2Container::SavePointBuffer (zipFile *handle*) throw (...) [protected]

Saves the current state of internal memory storage of point data either to the zip archive as an external binary point file or directly into the actual tree structure of the internal document handle of the main ISO5436-2 XML document depending on the current settings.

Parameters:

handle The handle to the zip archive where the data is to be stored.

References _ASSERT, _EX_T, _VERIFY, BuildPointVectorParser(), CreatePointVectorProxyContext(), CreatePointVectorWriterContext(), GetPointDataArchiveName(), GetVectorBuffer(), HasDocument(), HasVectorBuffer(), IsBinary(), m_CompressionLevel, m_Document, md5(), NULL, OGPS_ExGeneral, OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3(), ResetValidPointsLink(), ResetXmlPointList(), and OpenGPS::String::ToChar().

Referenced by Compress().

Here is the call graph for this function:

12.37.4.81 void ISO5436_2Container::SaveValidPointsLink (zipFile *handle*) throw (...) [protected]

Saves the current state of internal memory storage of point validity data to the zip archive.

Parameters:

handle The handle to the zip archive where the data is to be stored.

References _ASSERT, _EX_T, _VERIFY, OpenGPS::ZipStreamBuffer::GetMd5(), OpenGPS::VectorBuffer::GetValidityBuffer(), GetValidPointsArchiveName(), GetVectorBuffer(), OpenGPS::VectorBuffer::HasValidityBuffer(), HasValidPointsLink(), m_CompressionLevel, m_-Document, md5(), NULL, OGPS_ExGeneral, OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type::Record3(), OpenGPS::String::ToChar(), TRUE, and OpenGPS::ValidBuffer::Write().

Referenced by Compress().

Here is the call graph for this function:

12.37.4.82 void ISO5436_2Container::SaveXmlDocument (zipFile *handle*) throw (...) [protected]

Writes the content of the internal document handle to the main XML document present in an X3P archive.

Parameters:

handle The handle of the X3P archive.

References _ASSERT, _EX_T, _VERIFY, ConfigureNamespaceMap(), GetMainArchiveName(), OpenGPS::ZipStreamBuffer::GetMd5(), OpenGPS::Schemas::ISO5436_2::ISO5436_2(), m_-CompressionLevel, m_Document, md5(), NULL, OGPS_ExGeneral, SaveChecksumFile(), OpenGPS::String::ToChar(), and TRUE.

Referenced by Compress().

Here is the call graph for this function:

12.37.4.83 void ISO5436_2Container::SetListPoint (const OGPS_ULong *index*, const PointVector & *vector*) throw (...) [virtual]

Sets the value of a three-dimensional data point vector at a given index position.

Manipulates the point vector data stored in an ISO5436-2 X3P file directly.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

The data types of the values stored in the given vector parameter must correspond with the data types specified in the ISO5436-2 XML document within the axes definition area. Also ISO5436_-

2::SetListPoint is the valid access method only if point vectors are stored in list format. If this is not the case and point vectors are stored in matrix format with encoded topology information then you must use ISO5436_2::SetMatrixPoint instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ISO5436_2::GetDocument.

See also:

ISO5436_2::SetMatrixPoint

Parameters:

index The index position of the point vector to manipulate.

vector Set this point value at the given index position.

Reimplemented from OpenGPS::ISO5436_2.

References _ASSERT, _EX_T, CheckDocumentInstance(), IsIncrementalX(), IsIncrementalY(), IsMatrix(), m_PointVector, m_ProxyContext, OGPS_ExInvalidOperation, OGPS_Int32PointType, and OGPS_MissingPointType.

Referenced by OpenGPS::ISO5436_2Container::PointIteratorImpl::SetCurrent().

Here is the call graph for this function:

12.37.4.84 void ISO5436_2Container::SetMatrixPoint (const OGPS_ULong *u*, const OGPS_ULong *v*, const OGPS_ULong *w*, const PointVector *const *vector*) throw (...) [virtual]

Sets the value of a three-dimensional data point vector at a given surface position.

Manipulates the point vector data stored in an ISO5436-2 X3P file directly.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

The data types of the values stored in the given vector parameter must correspond with the data types specified in the ISO5436-2 XML document within the axes definition area. Also ISO5436_2::SetMatrixPoint is the valid access method only if point vectors are stored in matrix format with encoded topology information. If this is not the case point vectors are stored in list format. Then you must use ISO5436_2::SetListPoint instead. What is the correct format for access is revealed by the ISO5436-2 XML document. Access to the ISO5436-2 XML document which is part of an X3P file container can be obtained through ISO5436_2::GetDocument.

See also:

ISO5436_2::SetListPoint

Parameters:

u The u-direction of the surface position.

v The v-direction of the surface position.

w The w-direction of the surface position.

vector Set this point value at the given u,v,w position. If this parameter is set to NULL, this indicates there is no measurement data available for this position. This is due to the topology encoding properties of the matrix format storage.

Reimplemented from OpenGPS::ISO5436_2.

References _ASSERT, _EX_T, CheckDocumentInstance(), OpenGPS::VectorBuffer::GetValidityProvider(), GetVectorBuffer(), IsIncrementalX(), IsIncrementalY(), IsMatrix(), m_PointVector, m_ProxyContext, NULL, OGPS_ExInvalidOperation, OGPS_Int32PointType, OGPS_MissingPointType, and OpenGPS::PointValidityProvider::SetValid().

Referenced by OpenGPS::ISO5436_2Container::PointIteratorImpl::SetCurrent().

Here is the call graph for this function:

12.37.4.85 void ISO5436_2Container::TestChecksums () throw (...) [private]

Check if all checksums were verified.

If any one of them could not be verified this throws an OpenGPS::Exception of type OGPS_ExWarning that may be ignored.

References _EX_T, m_DataBinChecksum, m_MainChecksum, m_ValidBinChecksum, and OGPS_-ExWarning.

Referenced by Open().

12.37.4.86 OGPS_Boolean ISO5436_2Container::VerifyChecksum (const OpenGPS::String & *filePath*, const OpenGPS::UnsignedBytePtr *checksum*, const size_t *size*) const [private]

Verifies an 128bit md5 checksum.

Parameters:

filePath The file which checksum is to be verified.

checksum The expected checksum to verify.

size The size of the checksum buffer in bytes. This must be equal to 16 always as it is a 128bit md5 sum.

Returns:

Returns TRUE when the checksum could be verified, FALSE otherwise.

References _ASSERT, FALSE, md5(), md5_file(), n, OpenGPS::String::ToChar(), and TRUE.

Referenced by VerifyDataBinChecksum(), VerifyMainChecksum(), and VerifyValidBinChecksum().

Here is the call graph for this function:

12.37.4.87 void ISO5436_2Container::VerifyDataBinChecksum () [private]

Verifies the checksum of the binary point data file.

References _ASSERT, FALSE, GetPointDataFileName(), HasDocument(), IsBinary(), m_-DataBinChecksum, m_Document, md5(), OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3(), and VerifyChecksum().

Referenced by DecompressDataBin().

Here is the call graph for this function:

12.37.4.88 void ISO5436_2Container::VerifyMainChecksum () [private]

Verifies the checksum of the main document ISO5436-2 XML file.

References _ASSERT, FALSE, GetChecksumFileName(), GetMainFileName(), HasDocument(), m_-MainChecksum, ReadMd5FromFile(), and VerifyChecksum().

Referenced by DecompressChecksum().

Here is the call graph for this function:

12.37.4.89 void ISO5436_2Container::VerifyValidBinChecksum () [private]

Verifies the checksum of the binary point validity data file.

References _ASSERT, FALSE, GetValidPointsFileName(), HasDocument(), HasValidPointsLink(), IsBinary(), m_Document, m_ValidBinChecksum, md5(), OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3(), and VerifyChecksum().

Referenced by DecompressDataBin().

Here is the call graph for this function:

12.37.4.90 void ISO5436_2Container::Write (const int *compressionLevel* = -1) throw (...) [virtual]

Writes any changes back to the X3P file.

Call this function before ISO5436_2::Close if you want to store the changes you have made.

See also:

ISO5436_2::Create, ISO5436_2::Open, ISO5436_2::Close

Specific implementations may raise an exception.

Parameters:

compressionLevel Optionally specifies the compression level used when writing the X3P file which is nothing else than a simple zip file container. The default value for this parameter is (-1) which

enables standard compression level as a good trade-off between processing time and compression ratio. Values between 0 and 9 are possible. A value of 0 means "no compression" and a value of 9 enables the highest level compression rate at the cost of highest computation time.

Reimplemented from OpenGPS::ISO5436_2.

References _ASSERT, CheckDocumentInstance(), Compress(), and m_CompressionLevel.

Here is the call graph for this function:

12.37.4.91 OGPS_Boolean ISO5436_2Container::WriteVendorSpecific (zipFile *handle*) [private]

Writes vendorspecific files to the zip container if any.

Parameters:

handle Handle of the target zip archive.

Returns:

Returns FALSE if not all of the files could be written. TRUE otherwise.

References _EX_T, _OPENGPS_FREE, _OPENGPS_ZIP_CHUNK_MAX, _VERIFY, FALSE, OpenGPS::Environment::GetFileName(), OpenGPS::Environment::GetInstance(), m_CompressionLevel, m_Document, m_VendorSpecific, m_VendorURI, n, NULL, OGPS_ExGeneral, OGPS_ExInvalidOperation, OGPS_ExWarning, OpenGPS::String::ToChar(), TRUE, and OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID().

Referenced by Compress().

Here is the call graph for this function:

12.37.5 Member Data Documentation

12.37.5.1 int OpenGPS::ISO5436_2Container::m_CompressionLevel [private]

The level of compression of the zip archive.

Referenced by ISO5436_2Container(), SaveChecksumFile(), SavePointBuffer(), SaveValidPointsLink(), SaveXmlDocument(), Write(), and WriteVendorSpecific().

12.37.5.2 OGPS_Boolean OpenGPS::ISO5436_2Container::m_DataBinChecksum [private]

FALSE, if the md5 checksum could not be verified after reading.

Referenced by ISO5436_2Container(), Reset(), TestChecksums(), and VerifyDataBinChecksum().

12.37.5.3 Schemas::ISO5436_2::ISO5436_2Type* OpenGPS::ISO5436_2Container::m_Document [private]

The handle to a tree strucure corresponding to an ISO5436-2 XML document file.

Referenced by CheckDocumentInstance(), CreateDocument(), CreatePointVectorProxyContext(), CreatePointVectorReaderContext(), CreatePointVectorWriterContext(), GetChecksumArchiveName(), GetDocument(), GetIncrementX(), GetIncrementY(), GetListDimension(), GetMatrixDimensions(), GetMaxU(), GetMaxV(), GetMaxW(), GetOffsetX(), GetOffsetY(), GetOffsetZ(), GetPointCount(), GetPointDataArchiveName(), GetValidPointsArchiveName(), GetVendorSpecific(), GetXaxisDataType(), GetYaxisDataType(), GetZaxisDataType(), HasDocument(), HasValidPointsLink(), IsBinary(), IsIncrementalX(), IsIncrementalY(), IsMatrix(), ISO5436_2Container(), ReadXmlDocument(), Reset(), ResetValidPointsLink(), ResetXmlPointList(), SavePointBuffer(), SaveValidPointsLink(), SaveXmlDocument(), VerifyDataBinChecksum(), VerifyValidBinChecksum(), WriteVendorSpecific(), and ~ISO5436_2Container().

12.37.5.4 OpenGPS::String OpenGPS::ISO5436_2Container::m_FilePath [private]

The path of the X3P archive handles.

Referenced by GetFilePath(), GetFullFilePath(), and ISO5436_2Container().

12.37.5.5 OGPS_Boolean OpenGPS::ISO5436_2Container::m_IsCreating [private]

TRUE if an X3P archive is to be created, FALSE if an existing archive has been opened.

The value is undefined if nothing happened so far.

Referenced by CreateDocument(), GetVendorSpecific(), ISO5436_2Container(), and Open().

12.37.5.6 OGPS_Boolean OpenGPS::ISO5436_2Container::m_IsReadOnly [private]

TRUE if the user wants readonly access to the X3P file only.

FALSE otherwise.

Referenced by CreateDocument(), ISO5436_2Container(), Open(), and Reset().

12.37.5.7 OGPS_Boolean OpenGPS::ISO5436_2Container::m_MainChecksum [private]

FALSE, if the md5 checksum could not be verified after reading.

Referenced by ISO5436_2Container(), Reset(), TestChecksums(), and VerifyMainChecksum().

12.37.5.8 OpenGPS::String OpenGPS::ISO5436_2Container::m_PointDataFileName [private]

The temporary target path of the uncompressed binary point data file.

Referenced by GetPointDataFileName(), RemoveTempDir(), and Reset().

12.37.5.9 PointVectorAutoPtr OpenGPS::ISO5436_2Container::m_PointVector [private]

A point vector which serves as a temporary buffer.

Referenced by CreateDocument(), CreatePointBuffer(), GetListPoint(), GetMatrixPoint(), Reset(), SetListPoint(), and SetMatrixPoint().

12.37.5.10 PointVectorProxyContextAutoPtr OpenGPS::ISO5436_2Container::m_ProxyContext [private]

A global point vector proxy used for index based read/writes of point vectors.

Referenced by CreateDocument(), CreatePointBuffer(), GetListPoint(), GetMatrixPoint(), IsMatrixCoordValid(), Reset(), SetListPoint(), and SetMatrixPoint().

12.37.5.11 OpenGPS::String OpenGPS::ISO5436_2Container::m_TempBasePath [private]

The path to the global directory for temporary files.

Referenced by CreateTempDir(), and ISO5436_2Container().

12.37.5.12 OpenGPS::String OpenGPS::ISO5436_2Container::m_TempPath [private]

The path to a temporary directory unique to the current instance.

Referenced by CreateTempDir(), GetTempDir(), HasTempDir(), and RemoveTempDir().

12.37.5.13 OGPS_Boolean OpenGPS::ISO5436_2Container::m_ValidBinChecksum [private]

FALSE, if the md5 checksum could not be verified after reading.

Referenced by ISO5436_2Container(), Reset(), TestChecksums(), and VerifyValidBinChecksum().

12.37.5.14 OpenGPS::String OpenGPS::ISO5436_2Container::m_ValidPointsFileName [private]

The temporary target path of the uncompressed binary point validity data file.

Referenced by GetValidPointsFileName(), RemoveTempDir(), and Reset().

12.37.5.15 VectorBufferBuilderAutoPtr OpenGPS::ISO5436_2Container::m_VectorBufferBuilder [private]

The builder used to assemble the current vector buffer.

Referenced by CreateDocument(), CreatePointBuffer(), GetVectorBuffer(), HasVectorBuffer(), and Reset().

12.37.5.16 StringList OpenGPS::ISO5436_2Container::m_VendorSpecific [private]

Vendorspecific file names to be added to the container registered with one single vendor id.

See also:

ISO5436_2Container::m_VendorURI

Referenced by AppendVendorSpecific(), Reset(), and WriteVendorSpecific().

12.37.5.17 OpenGPS::String OpenGPS::ISO5436_2Container::m_VendorURI [private]

ID of vendorspecific data or empty.

See also:

ISO5436_2Container::m_VendorSpecific.

Referenced by AppendVendorSpecific(), Reset(), and WriteVendorSpecific().

The documentation for this class was generated from the following files:

- iso5436_2_container.hxx
- iso5436_2_container.cxx

12.38 OpenGPS::ISO5436_2Container::PointIteratorImpl Class Reference

Inheritance diagram for OpenGPS::ISO5436_2Container::PointIteratorImpl:

Collaboration diagram for OpenGPS::ISO5436_2Container::PointIteratorImpl:

12.38.1 Detailed Description

Implementation of the point iterator interface.

A point iterator can be created and initialized by an instance of OpenGPS::ISO5436_2Container only.

Public Member Functions

- virtual void GetCurrent (PointVector &vector) throw (...)

  Gets the value of the current point vector.

- virtual OGPS_Boolean GetPosition (OGPS_ULong *const u, OGPS_ULong *const v, OGPS_-ULong *const w) const

  Gets the current position of the iterator in topology coordinates.

- virtual OGPS_Boolean GetPosition (OGPS_ULong *const index) const

  Gets the current position of the iterator.

- virtual OGPS_Boolean HasNext () const throw (...)

  Asks if there is another point available to iterate.

- virtual OGPS_Boolean HasPrev () const throw (...)

  Asks if there is another point available to iterate.

- virtual OGPS_Boolean MoveNext () throw (...)

  Moves the iterator forward.

- virtual OGPS_Boolean MovePrev () throw (...)

  Moves the iterator backward.

- PointIteratorImpl (ISO5436_2Container *const handle, const OGPS_Boolean isForward, const OGPS_Boolean isMatrix)

  Creates a new instance.

- virtual void ResetNext ()

  Resets the iterator to the beginning and turns this iterator instance into a forward iterator.

- virtual void ResetPrev ()

  Resets the iterator to the beginning and turns this iterator instance into a backward iterator.

- virtual void SetCurrent (const PointVector *const vector) throw (...)

Sets the value of the current point vector.

- virtual ~PointIteratorImpl ()

  Destroys this instance.

Private Member Functions

- PointIteratorImpl & operator= (const PointIteratorImpl &src)

  The assignment-operator is not implemented.

- PointIteratorImpl (const PointIteratorImpl &src)

  The copy-ctor is not implemented.

Private Attributes

- ISO5436_2Container *const m_Handle

  Instance of the ISO5436-2 X3P to be accessed through this popint iterator.

- OGPS_Boolean m_IsForward

  This is the forward- or backward iterator.

- OGPS_Boolean m_IsMatrix

  TRUE to use matrix indexes and access methods, but FALSE for the simple list interface.

- OGPS_Boolean m_IsReset

  TRUE if in an untouched state.

- OGPS_ULong m_U

  The current index in X direction.

- OGPS_ULong m_V

  The current index in Y direction.

- OGPS_ULong m_W

  The current index in Z direction.

12.38.2 Constructor & Destructor Documentation

12.38.2.1 ISO5436_2Container::PointIteratorImpl::PointIteratorImpl (ISO5436_2Container *const *handle*, const OGPS_Boolean *isForward*, const OGPS_Boolean *isMatrix*)

Creates a new instance.

Parameters:

handle Provides access to point data based on indexes.

isForward TRUE if iterating in increasing order of indexes but FALSE when starting from the end with decreasing indexes.

isMatrix TRUE to access the matrix interface methods of the given OpenGPS::ISO5436_2Container instance. FALSE to access the list interface methods instead.

References _ASSERT, m_IsForward, m_IsMatrix, m_IsReset, m_U, m_V, m_W, and TRUE.

12.38.2.2 ISO5436_2Container::PointIteratorImpl::~PointIteratorImpl () [virtual]

Destroys this instance.

12.38.2.3 OpenGPS::ISO5436_2Container::PointIteratorImpl::PointIteratorImpl (const PointIteratorImpl & *src*) [private]

The copy-ctor is not implemented.

This prevents its usage.

12.38.3 Member Function Documentation

12.38.3.1 void ISO5436_2Container::PointIteratorImpl::GetCurrent (PointVector & *vector*) throw (...) [virtual]

Gets the value of the current point vector.

A specific instance may throw an OpenGPS::Exception on failure,

See also:

PointIterator::MoveNext

Parameters:

vector Gets a copy of the vector at the current iterator position.

Implements OpenGPS::PointIterator.

References _ASSERT, OpenGPS::ISO5436_2Container::GetListPoint(), OpenGPS::ISO5436_2Container::GetMatrixPoint(), OpenGPS::ISO5436_2Container::IsMatrix(), m_Handle, m_IsMatrix, m_U, m_V, and m_W.

Here is the call graph for this function:

12.38.3.2 OGPS_Boolean ISO5436_2Container::PointIteratorImpl::GetPosition (OGPS_ULong *const *u*, OGPS_ULong *const *v*, OGPS_ULong *const *w*) const [virtual]

Gets the current position of the iterator in topology coordinates.

See also:

PointIterator::MoveNext

Parameters:

u Gets the position index of the u component of the surface. If this parameter is set to NULL, it will be safely ignored and nothing returns here.

v Gets the position index of the v component of the surface. If this parameter is set to NULL, it will be safely ignored and nothing returns here.

w Gets the position index of the w component of the surface. If this parameter is set to NULL, it will be safely ignored and nothing returns here.

Returns:

Returns TRUE on success, FALSE otherwise.

Implements OpenGPS::PointIterator.

References FALSE, m_IsMatrix, m_U, m_V, m_W, and TRUE.

12.38.3.3 OGPS_Boolean ISO5436_2Container::PointIteratorImpl::GetPosition (OGPS_ULong *const *index*) const [virtual]

Gets the current position of the iterator.

See also:

PointIterator::MoveNext

Parameters:

index Gets the position index.

Returns:

Returns TRUE on success, FALSE otherwise.

Implements OpenGPS::PointIterator.

References _ASSERT, FALSE, m_IsMatrix, m_U, and TRUE.

12.38.3.4 OGPS_Boolean ISO5436_2Container::PointIteratorImpl::HasNext () const throw (...) [virtual]

Asks if there is another point available to iterate.

Remarks:

Use this function with an iterator handle obtained from OpenGPS::ISO5436_-2::CreateNextPointIterator.

See also:

PointIterator::MoveNext, PointIterator::HasPrev

Returns:

Returns TRUE if there is at least one more point available to iterate, FALSE otherwise.

Implements OpenGPS::PointIterator.

References _ASSERT, FALSE, OpenGPS::ISO5436_2Container::GetMaxU(), OpenGPS::ISO5436_2Container::GetMaxV(), OpenGPS::ISO5436_2Container::GetMaxW(), OpenGPS::ISO5436_2Container::IsMatrix(), m_Handle, m_IsForward, m_IsMatrix, m_IsReset, m_U, m_V, and m_W.

Referenced by MoveNext().

Here is the call graph for this function:

12.38.3.5 OGPS_Boolean ISO5436_2Container::PointIteratorImpl::HasPrev () const throw (...) [virtual]

Asks if there is another point available to iterate.

Remarks:

Use this function with an iterator handle obtained from OpenGPS::ISO5436_2::CreatePrevPointIterator.

See also:

PointIterator::MovePrev, PointIterator::HasNext

Returns:

Returns TRUE if there is at least one more point available to iterate, FALSE otherwise.

Implements OpenGPS::PointIterator.

References _ASSERT, FALSE, OpenGPS::ISO5436_2Container::GetMaxU(), OpenGPS::ISO5436_2Container::GetMaxV(), OpenGPS::ISO5436_2Container::GetMaxW(), OpenGPS::ISO5436_2Container::IsMatrix(), m_Handle, m_IsForward, m_IsMatrix, m_IsReset, m_U, m_V, and m_W.

Referenced by MovePrev().

Here is the call graph for this function:

12.38.3.6 OGPS_Boolean ISO5436_2Container::PointIteratorImpl::MoveNext () throw (...) [virtual]

Moves the iterator forward.

Remarks:

Use this function directly after initialising the iterator object with OpenGPS::ISO5436_-2::CreateNextPointIterator to move to the first point.

See also:

PointIterator::HasNext

Returns:

Returns TRUE on success, FALSE otherwise.

Implements OpenGPS::PointIterator.

References _ASSERT, FALSE, OpenGPS::ISO5436_2Container::GetMaxU(), OpenGPS::ISO5436_-2Container::GetMaxV(), OpenGPS::ISO5436_2Container::GetMaxW(), HasNext(), OpenGPS::ISO5436_2Container::IsMatrix(), m_Handle, m_IsMatrix, m_IsReset, m_U, m_V, m_W, and TRUE.

Here is the call graph for this function:

12.38.3.7 OGPS_Boolean ISO5436_2Container::PointIteratorImpl::MovePrev () throw (...) [virtual]

Moves the iterator backward.

Remarks:

Use this function directly after initialising the iterator object with OpenGPS::ISO5436_-2::CreatePrevPointIterator to move to the first point.

See also:

PointIterator::HasPrev

Returns:

Returns TRUE on success, FALSE otherwise.

Implements OpenGPS::PointIterator.

References _ASSERT, FALSE, OpenGPS::ISO5436_2Container::GetMaxU(), OpenGPS::ISO5436_2Container::GetMaxV(), OpenGPS::ISO5436_2Container::GetMaxW(), HasPrev(), OpenGPS::ISO5436_2Container::IsMatrix(), m_Handle, m_IsMatrix, m_IsReset, m_U, m_V, m_W, and TRUE.

Here is the call graph for this function:

12.38.3.8 PointIteratorImpl& OpenGPS::ISO5436_2Container::PointIteratorImpl::operator= (const PointIteratorImpl & *src*) `[private]`

The assignment-operator is not implemented.

This prevents its usage.

12.38.3.9 void ISO5436_2Container::PointIteratorImpl::ResetNext () `[virtual]`

Resets the iterator to the beginning and turns this iterator instance into a forward iterator.

Implements OpenGPS::PointIterator.

References m_IsForward, m_IsReset, m_U, m_V, m_W, and TRUE.

12.38.3.10 void ISO5436_2Container::PointIteratorImpl::ResetPrev () `[virtual]`

Resets the iterator to the beginning and turns this iterator instance into a backward iterator.

Implements OpenGPS::PointIterator.

References FALSE, m_IsForward, m_IsReset, m_U, m_V, m_W, and TRUE.

12.38.3.11 void ISO5436_2Container::PointIteratorImpl::SetCurrent (const PointVector *const *vector*) throw (...) `[virtual]`

Sets the value of the current point vector.

A specific instance may throw an OpenGPS::Exception on failure,

See also:

PointIterator::MoveNext

Parameters:

vector New value of the current point vector. May be NULL to indicate an invalid point.

Returns:

Returns TRUE on success, FALSE otherwise.

Implements OpenGPS::PointIterator.

References _ASSERT, OpenGPS::ISO5436_2Container::IsMatrix(), m_Handle, m_IsMatrix, m_U, m_V, m_W, OpenGPS::ISO5436_2Container::SetListPoint(), and OpenGPS::ISO5436_2Container::SetMatrixPoint().

Here is the call graph for this function:

12.38.4 Member Data Documentation

12.38.4.1 ISO5436_2Container* const OpenGPS::ISO5436_2Container::PointIteratorImpl::m_Handle [private]

Instance of the ISO5436-2 X3P to be accessed through this popint iterator.

Referenced by GetCurrent(), HasNext(), HasPrev(), MoveNext(), MovePrev(), and SetCurrent().

12.38.4.2 OGPS_Boolean OpenGPS::ISO5436_2Container::PointIteratorImpl::m_IsForward [private]

This is the forward- or backward iterator.

Referenced by HasNext(), HasPrev(), PointIteratorImpl(), ResetNext(), and ResetPrev().

12.38.4.3 OGPS_Boolean OpenGPS::ISO5436_2Container::PointIteratorImpl::m_IsMatrix [private]

TRUE to use matrix indexes and access methods, but FALSE for the simple list interface.

Referenced by GetCurrent(), GetPosition(), HasNext(), HasPrev(), MoveNext(), MovePrev(), PointIteratorImpl(), and SetCurrent().

12.38.4.4 OGPS_Boolean OpenGPS::ISO5436_2Container::PointIteratorImpl::m_IsReset [private]

TRUE if in an untouched state.

Referenced by HasNext(), HasPrev(), MoveNext(), MovePrev(), PointIteratorImpl(), ResetNext(), and ResetPrev().

12.38.4.5 OGPS_ULong OpenGPS::ISO5436_2Container::PointIteratorImpl::m_U [private]

The current index in X direction.

Used when iterating both matrices and lists.

Referenced by GetCurrent(), GetPosition(), HasNext(), HasPrev(), MoveNext(), MovePrev(), PointIteratorImpl(), ResetNext(), ResetPrev(), and SetCurrent().

12.38.4.6 OGPS_ULong OpenGPS::ISO5436_2Container::PointIteratorImpl::m_V [private]

The current index in Y direction.

Used with matrices only.

Referenced by GetCurrent(), GetPosition(), HasNext(), HasPrev(), MoveNext(), MovePrev(), PointIteratorImpl(), ResetNext(), ResetPrev(), and SetCurrent().

12.38.4.7 OGPS_ULong OpenGPS::ISO5436_2Container::PointIteratorImpl::m_W [private]

The current index in Z direction.

Used with matrices only.

Referenced by GetCurrent(), GetPosition(), HasNext(), HasPrev(), MoveNext(), MovePrev(), PointIteratorImpl(), ResetNext(), ResetPrev(), and SetCurrent().

The documentation for this class was generated from the following files:

- iso5436_2_container.hxx
- point_iterator.cxx

12.39 OpenGPS::Schemas::ISO5436_2::ISO5436_2Type Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.39.1 Detailed Description

Class corresponding to the ISO5436_2Type schema type.

This is the top tag of a data file

Record1

Accessor and modifier functions for the Record1 required element.

- typedef ::xsd::cxx::tree::traits< Record1_type, wchar_t > Record1_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::Record1Type Record1_type

  Element type.

- void Record1 (::std::auto_ptr< Record1_type > p)

  Set the element value without copying.

- void Record1 (const Record1_type &x)

  Set the element value.

- Record1_type & Record1 ()

  Return a read-write reference to the element.

- const Record1_type & Record1 () const

  Return a read-only (constant) reference to the element.

Record1

Accessor and modifier functions for the Record1 required element.

- typedef ::xsd::cxx::tree::traits< Record1_type, wchar_t > Record1_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::Record1Type Record1_type

  Element type.

- void Record1 (::std::auto_ptr< Record1_type > p)

  Set the element value without copying.

- void Record1 (const Record1_type &x)

  Set the element value.

- Record1_type & Record1 ()

  Return a read-write reference to the element.

- const Record1_type & Record1 () const

  Return a read-only (constant) reference to the element.

Record2

Accessor and modifier functions for the Record2 optional element.

- typedef ::xsd::cxx::tree::optional< Record2_type > Record2_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< Record2_type, wchar_t > Record2_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::Record2Type Record2_type

  Element type.

- void Record2 (::std::auto_ptr< Record2_type > p)

  Set the element value without copying.

- void Record2 (const Record2_optional &x)

  Set the element value.

- void Record2 (const Record2_type &x)

  Set the element value.

- Record2_optional & Record2 ()

  Return a read-write reference to the element container.

- const Record2_optional & Record2 () const

  Return a read-only (constant) reference to the element container.

Record2

Accessor and modifier functions for the Record2 optional element.

- typedef ::xsd::cxx::tree::optional< Record2_type > Record2_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< Record2_type, wchar_t > Record2_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::Record2Type Record2_type

  Element type.

- void Record2 (::std::auto_ptr< Record2_type > p)

  Set the element value without copying.

- void Record2 (const Record2_optional &x)

  Set the element value.

- void Record2 (const Record2_type &x)

  Set the element value.

- Record2_optional & Record2 ()

  Return a read-write reference to the element container.

- const Record2_optional & Record2 () const

  Return a read-only (constant) reference to the element container.

Record3

Accessor and modifier functions for the Record3 required element.

- typedef ::xsd::cxx::tree::traits< Record3_type, wchar_t > Record3_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::Record3Type Record3_type

  Element type.

- void Record3 (::std::auto_ptr< Record3_type > p)

  Set the element value without copying.

- void Record3 (const Record3_type &x)

  Set the element value.

- Record3_type & Record3 ()

  Return a read-write reference to the element.

- const Record3_type & Record3 () const

  Return a read-only (constant) reference to the element.

Record3

Accessor and modifier functions for the Record3 required element.

- typedef ::xsd::cxx::tree::traits< Record3_type, wchar_t > Record3_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::Record3Type Record3_type

  Element type.

- void Record3 (::std::auto_ptr< Record3_type > p)

  Set the element value without copying.

- void Record3 (const Record3_type &x)

  Set the element value.

- Record3_type & Record3 ()

  Return a read-write reference to the element.

- const Record3_type & Record3 () const

  Return a read-only (constant) reference to the element.

Record4

Accessor and modifier functions for the Record4 required element.

- typedef ::xsd::cxx::tree::traits< Record4_type, wchar_t > Record4_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::Record4Type Record4_type

  Element type.

- void Record4 (::std::auto_ptr< Record4_type > p)

  Set the element value without copying.

- void Record4 (const Record4_type &x)

  Set the element value.

- Record4_type & Record4 ()

  Return a read-write reference to the element.

- const Record4_type & Record4 () const

  Return a read-only (constant) reference to the element.

Record4

Accessor and modifier functions for the Record4 required element.

- typedef ::xsd::cxx::tree::traits< Record4_type, wchar_t > Record4_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::Record4Type Record4_type

  Element type.

- void Record4 (::std::auto_ptr< Record4_type > p)

  Set the element value without copying.

- void Record4 (const Record4_type &x)

  Set the element value.

- Record4_type & Record4 ()

  Return a read-write reference to the element.

- const Record4_type & Record4 () const

  Return a read-only (constant) reference to the element.

VendorSpecificID

Accessor and modifier functions for the VendorSpecificID optional element.

This is an extension hook for vendor specific data formats derived from ISO5436_2_XML This tag contains a vendor specific ID which is the URL of the vendor. It does not need to be valid but it must be worldwide unique! Example: `http://www.example-inc.com/myformat`

- typedef ::xsd::cxx::tree::optional< VendorSpecificID_type > VendorSpecificID_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< VendorSpecificID_type, wchar_t > VendorSpecificID_traits

  Element traits type.

- typedef ::xml_schema::uri VendorSpecificID_type

  Element type.

- void VendorSpecificID (::std::auto_ptr< VendorSpecificID_type > p)

  Set the element value without copying.

- void VendorSpecificID (const VendorSpecificID_optional &x)

  Set the element value.

- void VendorSpecificID (const VendorSpecificID_type &x)

  Set the element value.

- VendorSpecificID_optional & VendorSpecificID ()

  Return a read-write reference to the element container.

- const VendorSpecificID_optional & VendorSpecificID () const

  Return a read-only (constant) reference to the element container.

VendorSpecificID

Accessor and modifier functions for the VendorSpecificID optional element.

This is an extension hook for vendor specific data formats derived from ISO5436_2_XML This tag contains a vendor specific ID which is the URL of the vendor. It does not need to be valid but it must be worldwide unique! Example: `http://www.example-inc.com/myformat`

- typedef ::xsd::cxx::tree::optional< VendorSpecificID_type > VendorSpecificID_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< VendorSpecificID_type, wchar_t > VendorSpecificID_traits

  Element traits type.

- typedef ::xml_schema::uri VendorSpecificID_type

  Element type.

- void VendorSpecificID (::std::auto_ptr< VendorSpecificID_type > p)

Set the element value without copying.

- void VendorSpecificID (const VendorSpecificID_optional &x)
  Set the element value.
- void VendorSpecificID (const VendorSpecificID_type &x)
  Set the element value.
- VendorSpecificID_optional & VendorSpecificID ()
  Return a read-write reference to the element container.
- const VendorSpecificID_optional & VendorSpecificID () const
  Return a read-only (constant) reference to the element container.

Constructors

- virtual ISO5436_2Type * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const
  Copy the object polymorphically.
- ISO5436_2Type (const ISO5436_2Type &x,::xml_schema::flags f=0,::xml_schema::type *c=0)
  Copy constructor.
- ISO5436_2Type (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)
  Construct an instance from a DOM element.
- ISO5436_2Type (const Record1_type &, const Record3_type &, const Record4_type &)
  Construct an instance from the ultimate base and initializers for required elements and attributes.

Constructors

- virtual ISO5436_2Type * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const
  Copy the object polymorphically.
- ISO5436_2Type (const ISO5436_2Type &x,::xml_schema::flags f=0,::xml_schema::type *c=0)
  Copy constructor.
- ISO5436_2Type (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)
  Construct an instance from a DOM element.
- ISO5436_2Type (const Record1_type &, const Record3_type &, const Record4_type &)
  Construct an instance from the ultimate base and initializers for required elements and attributes.

Protected Member Functions

- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)
- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)

Private Attributes

- ::xsd::cxx::tree::one< Record1_type > Record1_
- Record2_optional Record2_
- ::xsd::cxx::tree::one< Record3_type > Record3_
- ::xsd::cxx::tree::one< Record4_type > Record4_
- VendorSpecificID_optional VendorSpecificID_

12.39.2 Member Typedef Documentation

12.39.2.1 typedef ::xsd::cxx::tree::traits< Record1_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1_traits

Element traits type.

12.39.2.2 typedef ::xsd::cxx::tree::traits< Record1_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1_traits

Element traits type.

12.39.2.3 typedef ::OpenGPS::Schemas::ISO5436_2::Record1Type OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1_type

Element type.

12.39.2.4 typedef ::OpenGPS::Schemas::ISO5436_2::Record1Type OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1_type

Element type.

12.39.2.5 typedef ::xsd::cxx::tree::optional< Record2_type > OpenGPS::Schemas::ISO5436_-2::ISO5436_2Type::Record2_optional

Element optional container type.

12.39.2.6 typedef ::xsd::cxx::tree::optional< Record2_type > OpenGPS::Schemas::ISO5436_-2::ISO5436_2Type::Record2_optional

Element optional container type.

12.39.2.7 typedef ::xsd::cxx::tree::traits< Record2_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2_traits

Element traits type.

12.39.2.8 typedef ::xsd::cxx::tree::traits< Record2_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2_traits

Element traits type.

12.39.2.9 typedef ::OpenGPS::Schemas::ISO5436_2::Record2Type OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2_type

Element type.

12.39.2.10 typedef ::OpenGPS::Schemas::ISO5436_2::Record2Type OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2_type

Element type.

12.39.2.11 typedef ::xsd::cxx::tree::traits< Record3_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3_traits

Element traits type.

12.39.2.12 typedef ::xsd::cxx::tree::traits< Record3_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3_traits

Element traits type.

12.39.2.13 typedef ::OpenGPS::Schemas::ISO5436_2::Record3Type OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3_type

Element type.

12.39.2.14 typedef ::OpenGPS::Schemas::ISO5436_2::Record3Type OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3_type

Element type.

12.39.2.15 typedef ::xsd::cxx::tree::traits< Record4_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record4_traits

Element traits type.

12.39.2.16 typedef ::xsd::cxx::tree::traits< Record4_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record4_traits

Element traits type.

12.39.2.17 typedef ::OpenGPS::Schemas::ISO5436_2::Record4Type OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record4_type

Element type.

12.39.2.18 typedef ::OpenGPS::Schemas::ISO5436_2::Record4Type OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record4_type

Element type.

12.39.2.19 typedef ::xsd::cxx::tree::optional< VendorSpecificID_type > OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID_optional

Element optional container type.

12.39.2.20 typedef ::xsd::cxx::tree::optional< VendorSpecificID_type > OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID_optional

Element optional container type.

12.39.2.21 typedef ::xsd::cxx::tree::traits< VendorSpecificID_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID_traits

Element traits type.

12.39.2.22 typedef ::xsd::cxx::tree::traits< VendorSpecificID_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID_traits

Element traits type.

12.39.2.23 typedef ::xml_schema::uri OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type::VendorSpecificID_type

Element type.

12.39.2.24 typedef ::xml_schema::uri OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type::VendorSpecificID_type

Element type.

12.39.3 Constructor & Destructor Documentation

12.39.3.1 OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::ISO5436_2Type (const Record1_-type & *Record1*, const Record3_type & *Record3*, const Record4_type & *Record4*)

Construct an instance from the ultimate base and initializers for required elements and attributes.

Referenced by _clone().

12.39.3.2 OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::ISO5436_2Type (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

References parse().

Here is the call graph for this function:

12.39.3.3 OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::ISO5436_2Type (const ISO5436_2Type & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.39.3.4 OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::ISO5436_2Type (const Record1_type &, const Record3_type &, const Record4_type &)

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.39.3.5 OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::ISO5436_2Type (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.39.3.6 OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::ISO5436_2Type (const ISO5436_2Type & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.39.4 Member Function Documentation

12.39.4.1 virtual ISO5436_2Type* OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.39.4.2 ISO5436_2Type * OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References ISO5436_2Type().

Here is the call graph for this function:

12.39.4.3 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::parse (::xsd::cxx::xml::dom::parser< wchar_t > &, ::xml_schema::flags) [protected]

12.39.4.4 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::parse (::xsd::cxx::xml::dom::parser< wchar_t > & *p*, ::xml_schema::flags *f*) [protected]

References n, Record1(), Record1_, Record2(), Record3(), Record3_, Record4(), Record4_, and VendorSpecificID().

Referenced by ISO5436_2Type().

Here is the call graph for this function:

12.39.4.5 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1 (::std::auto_ptr< Record1_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.39.4.6 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1 (const Record1_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.39.4.7 Record1_type& OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.39.4.8 const Record1_type& OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.39.4.9 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1 (::std::auto_ptr< Record1_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References Record1_.

12.39.4.10 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1 (const Record1_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References Record1_.

12.39.4.11 ISO5436_2Type::Record1_type & OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

References Record1_.

12.39.4.12 const ISO5436_2Type::Record1_type & OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

References Record1_.

Referenced by convertFormat(), OpenGPS::ISO5436_2Container::GetIncrementX(), OpenGPS::ISO5436_2Container::GetIncrementY(), OpenGPS::ISO5436_2Container::GetOffsetX(), OpenGPS::ISO5436_2Container::GetOffsetY(), OpenGPS::ISO5436_2Container::GetOffsetZ(), OpenGPS::ISO5436_2Container::GetXaxisDataType(), OpenGPS::ISO5436_2Container::GetYaxisDataType(), OpenGPS::ISO5436_2Container::GetZaxisDataType(), OpenGPS::ISO5436_2Container::IsIncrementalX(), OpenGPS::ISO5436_2Container::IsIncrementalY(), OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), parse(), and readonlyExample2().

12.39.4.13 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2 (::std::auto_ptr< Record2_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.39.4.14 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2 (const Record2_-optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.39.4.15 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2 (const Record2_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.39.4.16 Record2_optional& OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2 ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.39.4.17 const Record2_optional& OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2 () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

12.39.4.18 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2 (::std::auto_ptr< Record2_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References Record2_.

12.39.4.19 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2 (const Record2_-optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

References Record2_.

12.39.4.20 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2 (const Record2_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References Record2_.

12.39.4.21 ISO5436_2Type::Record2_optional & OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type::Record2 ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

References Record2_.

12.39.4.22 const ISO5436_2Type::Record2_optional & OpenGPS::Schemas::ISO5436_-2::ISO5436_2Type::Record2 () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

References Record2_.

Referenced by convertFormat(), OpenGPS::ISO5436_2Container::CreateDocument(), OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.39.4.23 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3 (::std::auto_ptr< Record3_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.39.4.24 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3 (const Record3_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.39.4.25 Record3_type& OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.39.4.26 const Record3_type& OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.39.4.27 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3 (::std::auto_ptr< Record3_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References Record3_.

12.39.4.28 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3 (const Record3_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References Record3_.

12.39.4.29 ISO5436_2Type::Record3_type & OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

References Record3_.

12.39.4.30 const ISO5436_2Type::Record3_type & OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

References Record3_.

Referenced by convertFormat(), OpenGPS::ISO5436_2Container::CreateDocument(), OpenGPS::ISO5436_2Container::CreatePointVectorProxyContext(), OpenGPS::ISO5436_2Container::CreatePointVectorReaderContext(), OpenGPS::ISO5436_2Container::CreatePointVectorWriterContext(), OpenGPS::ISO5436_2Container::GetListDimension(), OpenGPS::ISO5436_2Container::GetMatrixDimensions(), OpenGPS::ISO5436_2Container::GetMaxU(), OpenGPS::ISO5436_2Container::GetMaxV(), OpenGPS::ISO5436_2Container::GetMaxW(), OpenGPS::ISO5436_2Container::GetPointCount(), OpenGPS::ISO5436_2Container::GetPointDataArchiveName(), OpenGPS::ISO5436_2Container::GetValidPointsArchiveName(), OpenGPS::ISO5436_2Container::HasValidPointsLink(), OpenGPS::ISO5436_2Container::IsBinary(), OpenGPS::ISO5436_2Container::IsMatrix(), OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), parse(), readonlyExample4(), OpenGPS::ISO5436_2Container::ResetValidPointsLink(), OpenGPS::ISO5436_2Container::ResetXmlPointList(), OpenGPS::ISO5436_2Container::SavePointBuffer(), OpenGPS::ISO5436_2Container::SaveValidPointsLink(), OpenGPS::ISO5436_2Container::VerifyDataBinChecksum(), and OpenGPS::ISO5436_2Container::VerifyValidBinChecksum().

12.39.4.31 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record4 (::std::auto_ptr< Record4_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.39.4.32 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record4 (const Record4_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.39.4.33 Record4_type& OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record4 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.39.4.34 const Record4_type& OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record4 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.39.4.35 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record4 (::std::auto_ptr< Record4_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References Record4_.

12.39.4.36 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record4 (const Record4_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References Record4_.

12.39.4.37 ISO5436_2Type::Record4_type & OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record4 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

References Record4_.

12.39.4.38 const ISO5436_2Type::Record4_type & OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record4 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

References Record4_.

Referenced by OpenGPS::ISO5436_2Container::GetChecksumArchiveName(), OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.39.4.39 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID (::std::auto_ptr< VendorSpecificID_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.39.4.40 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID (const VendorSpecificID_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.39.4.41 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID (const VendorSpecificID_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.39.4.42 VendorSpecificID_optional& OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.39.4.43 const VendorSpecificID_optional& OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

12.39.4.44 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID (::std::auto_ptr< VendorSpecificID_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References VendorSpecificID_.

12.39.4.45 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID (const VendorSpecificID_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

References VendorSpecificID_.

12.39.4.46 void OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID (const VendorSpecificID_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References VendorSpecificID_.

12.39.4.47 ISO5436_2Type::VendorSpecificID_optional & OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

References VendorSpecificID_.

12.39.4.48 const ISO5436_2Type::VendorSpecificID_optional & OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

References VendorSpecificID_.

Referenced by OpenGPS::ISO5436_2Container::GetVendorSpecific(), OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), parse(), and OpenGPS::ISO5436_2Container::WriteVendorSpecific().

12.39.5 Member Data Documentation

12.39.5.1 xsd::cxx::tree::one< Record1_type > OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1_ [private]

Referenced by parse(), and Record1().

12.39.5.2 Record2_optional OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2_- [private]

Referenced by Record2().

12.39.5.3 xsd::cxx::tree::one< Record3_type > OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type::Record3_ [private]

Referenced by parse(), and Record3().

12.39.5.4 xsd::cxx::tree::one< Record4_type > OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type::Record4_ [private]

Referenced by parse(), and Record4().

12.39.5.5 VendorSpecificID_optional OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type::VendorSpecificID_ [private]

Referenced by VendorSpecificID().

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.40 OpenGPS::Schemas::ISO5436_2::MatrixDimensionType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.40.1 Detailed Description

Class corresponding to the MatrixDimensionType schema type.

Defines the size of the 3 dimensions of the data matrix.

SizeX

Accessor and modifier functions for the SizeX required element.

Define the size of the first dimension of the data matrix

- typedef ::xsd::cxx::tree::traits< SizeX_type, wchar_t > SizeX_traits

  Element traits type.

- typedef ::xml_schema::unsigned_long SizeX_type

  Element type.

- void SizeX (const SizeX_type &x)

  Set the element value.

- SizeX_type & SizeX ()

Return a read-write reference to the element.

- const SizeX_type & SizeX () const

  Return a read-only (constant) reference to the element.

SizeX

Accessor and modifier functions for the SizeX required element.

Define the size of the first dimension of the data matrix

- typedef ::xsd::cxx::tree::traits< SizeX_type, wchar_t > SizeX_traits

  Element traits type.

- typedef ::xml_schema::unsigned_long SizeX_type

  Element type.

- void SizeX (const SizeX_type &x)

  Set the element value.

- SizeX_type & SizeX ()

  Return a read-write reference to the element.

- const SizeX_type & SizeX () const

  Return a read-only (constant) reference to the element.

SizeY

Accessor and modifier functions for the SizeY required element.

Define the size of the second dimension of the data matrix

- typedef ::xsd::cxx::tree::traits< SizeY_type, wchar_t > SizeY_traits

  Element traits type.

- typedef ::xml_schema::unsigned_long SizeY_type

  Element type.

- void SizeY (const SizeY_type &x)

  Set the element value.

- SizeY_type & SizeY ()

  Return a read-write reference to the element.

- const SizeY_type & SizeY () const

  Return a read-only (constant) reference to the element.

SizeY

Accessor and modifier functions for the SizeY required element.

Define the size of the second dimension of the data matrix

- typedef ::xsd::cxx::tree::traits< SizeY_type, wchar_t > SizeY_traits

  Element traits type.

- typedef ::xml_schema::unsigned_long SizeY_type

  Element type.

- void SizeY (const SizeY_type &x)

  Set the element value.

- SizeY_type & SizeY ()

  Return a read-write reference to the element.

- const SizeY_type & SizeY () const

  Return a read-only (constant) reference to the element.

SizeZ

Accessor and modifier functions for the SizeZ required element.

Define the size of the third dimension of the data matrix

- typedef ::xsd::cxx::tree::traits< SizeZ_type, wchar_t > SizeZ_traits

  Element traits type.

- typedef ::xml_schema::unsigned_long SizeZ_type

  Element type.

- void SizeZ (const SizeZ_type &x)

  Set the element value.

- SizeZ_type & SizeZ ()

  Return a read-write reference to the element.

- const SizeZ_type & SizeZ () const

  Return a read-only (constant) reference to the element.

SizeZ

Accessor and modifier functions for the SizeZ required element.

Define the size of the third dimension of the data matrix

- typedef ::xsd::cxx::tree::traits< SizeZ_type, wchar_t > SizeZ_traits

Element traits type.

- typedef ::xml_schema::unsigned_long SizeZ_type

  Element type.

- void SizeZ (const SizeZ_type &x)

  Set the element value.

- SizeZ_type & SizeZ ()

  Return a read-write reference to the element.

- const SizeZ_type & SizeZ () const

  Return a read-only (constant) reference to the element.

Constructors

- virtual MatrixDimensionType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- MatrixDimensionType (const MatrixDimensionType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- MatrixDimensionType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- MatrixDimensionType (const SizeX_type &, const SizeY_type &, const SizeZ_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Constructors

- virtual MatrixDimensionType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- MatrixDimensionType (const MatrixDimensionType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- MatrixDimensionType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- MatrixDimensionType (const SizeX_type &, const SizeY_type &, const SizeZ_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Protected Member Functions

- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)
- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)

Private Attributes

- ::xsd::cxx::tree::one< SizeX_type > SizeX_
- ::xsd::cxx::tree::one< SizeY_type > SizeY_
- ::xsd::cxx::tree::one< SizeZ_type > SizeZ_

12.40.2 Member Typedef Documentation

12.40.2.1 typedef ::xsd::cxx::tree::traits< SizeX_type, wchar_t > OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX_traits

Element traits type.

12.40.2.2 typedef ::xsd::cxx::tree::traits< SizeX_type, wchar_t > OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX_traits

Element traits type.

12.40.2.3 typedef ::xml_schema::unsigned_long OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX_type

Element type.

12.40.2.4 typedef ::xml_schema::unsigned_long OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX_type

Element type.

12.40.2.5 typedef ::xsd::cxx::tree::traits< SizeY_type, wchar_t > OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY_traits

Element traits type.

12.40.2.6 typedef ::xsd::cxx::tree::traits< SizeY_type, wchar_t > OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY_traits

Element traits type.

12.40.2.7 typedef ::xml_schema::unsigned_long OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY_type

Element type.

12.40.2.8 typedef ::xml_schema::unsigned_long OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY_type

Element type.

12.40.2.9 typedef ::xsd::cxx::tree::traits< SizeZ_type, wchar_t > OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ_traits

Element traits type.

12.40.2.10 typedef ::xsd::cxx::tree::traits< SizeZ_type, wchar_t > OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ_traits

Element traits type.

12.40.2.11 typedef ::xml_schema::unsigned_long OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ_type

Element type.

12.40.2.12 typedef ::xml_schema::unsigned_long OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ_type

Element type.

12.40.3 Constructor & Destructor Documentation

12.40.3.1 OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::MatrixDimensionType (const SizeX_type & *SizeX*, const SizeY_type & *SizeY*, const SizeZ_type & *SizeZ*)

Construct an instance from the ultimate base and initializers for required elements and attributes.

Referenced by _clone().

12.40.3.2 OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::MatrixDimensionType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

References parse().

Here is the call graph for this function:

12.40.3.3 OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::MatrixDimensionType (const MatrixDimensionType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.40.3.4 OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::MatrixDimensionType (const SizeX_type &, const SizeY_type &, const SizeZ_type &)

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.40.3.5 OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::MatrixDimensionType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.40.3.6 OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::MatrixDimensionType (const MatrixDimensionType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.40.4 Member Function Documentation

12.40.4.1 virtual MatrixDimensionType* OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.40.4.2 MatrixDimensionType * OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References MatrixDimensionType().

Here is the call graph for this function:

12.40.4.3 void OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::parse (::xsd::cxx::xml::dom::parser< wchar_t > &, ::xml_schema::flags) [protected]

12.40.4.4 void OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::parse (::xsd::cxx::xml::dom::parser< wchar_t > & *p*, ::xml_schema::flags *f*) [protected]

References n, SizeX(), SizeX_, SizeY(), SizeY_, SizeZ(), and SizeZ_.

Referenced by MatrixDimensionType().

Here is the call graph for this function:

12.40.4.5 void OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX (const SizeX_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.40.4.6 SizeX_type& OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.40.4.7 const SizeX_type& OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.40.4.8 void OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX (const SizeX_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.40.4.9 SizeX_type& OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.40.4.10 const SizeX_type& OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::ISO5436_2Container::CreatePointVectorProxyContext(), OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.40.4.11 void OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY (const SizeY_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.40.4.12 SizeY_type& OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.40.4.13 const SizeY_type& OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.40.4.14 void OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY (const SizeY_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.40.4.15 SizeY_type& OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.40.4.16 const SizeY_type& OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::ISO5436_2Container::CreatePointVectorProxyContext(), OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.40.4.17 void OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ (const SizeZ_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.40.4.18 SizeZ_type& OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.40.4.19 const SizeZ_type& OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.40.4.20 void OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ (const SizeZ_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.40.4.21 SizeZ_type& OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.40.4.22 const SizeZ_type& OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::ISO5436_2Container::CreatePointVectorProxyContext(), OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.40.5 Member Data Documentation

12.40.5.1 xsd::cxx::tree::one< SizeX_type > OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX_ [private]

Referenced by parse().

12.40.5.2 xsd::cxx::tree::one< SizeY_type > OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY_ [private]

Referenced by parse().

12.40.5.3 xsd::cxx::tree::one< SizeZ_type > OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ_ [private]

Referenced by parse().

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.41 md5_context Struct Reference

```
#include <md5.h>
```

12.41.1 Detailed Description

MD5 context structure.

Public Attributes

- unsigned char buffer [64]

  data block being processed

- unsigned char ipad [64]

  HMAC: inner padding.

- unsigned char opad [64]
  HMAC: outer padding.
- unsigned long state [4]
  intermediate digest state
- unsigned long total [2]
  number of bytes processed

12.41.2 Member Data Documentation

12.41.2.1 unsigned char md5_context::buffer[64]

data block being processed

Referenced by md5_update().

12.41.2.2 unsigned char md5_context::ipad[64]

HMAC: inner padding.

Referenced by md5_hmac_starts().

12.41.2.3 unsigned char md5_context::opad[64]

HMAC: outer padding.

Referenced by md5_hmac_finish(), and md5_hmac_starts().

12.41.2.4 unsigned long md5_context::state[4]

intermediate digest state

Referenced by md5_finish(), md5_process(), and md5_starts().

12.41.2.5 unsigned long md5_context::total[2]

number of bytes processed

Referenced by md5_finish(), md5_starts(), and md5_update().

The documentation for this struct was generated from the following file:

- md5.h

12.42 OpenGPS::MissingDataPointParser Class Reference

```
#include <missing_data_point_parser.hxx>
```

Inheritance diagram for OpenGPS::MissingDataPointParser:

Collaboration diagram for OpenGPS::MissingDataPointParser:

12.42.1 Detailed Description

Reads/Writes instances of OpenGPS::DataPoint of missing point data.

Public Member Functions

- MissingDataPointParser ()
  Creates a new instance.
- virtual void Read (PointVectorReaderContext &context, DataPoint &value) throw (...)
  Reads point data from a given context/media.
- virtual void Write (PointVectorWriterContext &context, const DataPoint &value) throw (...)
  Writes point data to a given context/media.
- virtual ~MissingDataPointParser ()
  Destroys this instance.

12.42.2 Constructor & Destructor Documentation

12.42.2.1 MissingDataPointParser::MissingDataPointParser ()

Creates a new instance.

12.42.2.2 MissingDataPointParser::~MissingDataPointParser () `[virtual]`

Destroys this instance.

12.42.3 Member Function Documentation

12.42.3.1 void MissingDataPointParser::Read (PointVectorReaderContext & *context*, DataPoint & *value*) throw (...) `[virtual]`

Reads point data from a given context/media.

Using the appropriate access method of the given implementation of OpenGPS::PointVectorReaderContext the current point value read gets stored in a OpenGPS::DataPoint instance. This operation is typesafe.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

context Provides read access to the media.

value Buffer where the point data read gets stored.

Implements OpenGPS::DataPointParser.

12.42.3.2 void MissingDataPointParser::Write (PointVectorWriterContext & *context*, const DataPoint & *value*) throw (...) `[virtual]`

Writes point data to a given context/media.

Using the appropriate access method of the given implementation of OpenGPS::PointVectorWriterContext the point value currently stored in the OpenGPS::DataPoint instance gets written to the media. This operation is typesafe and will fail if the type of point data stored in the value object does not exacly match the current implementation of the OpenGPS::DataPointParser interface.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

context Provides write access to the media.

value Buffer where the point data to be written is stored.

Implements OpenGPS::DataPointParser.

The documentation for this class was generated from the following files:

- missing_data_point_parser.hxx
- missing_data_point_parser.cxx

12.43 OGPS_DataPoint Struct Reference

```
#include <data_point_c.hxx>
```

Collaboration diagram for OGPS_DataPoint:

12.43.1 Detailed Description

Encapsulates the C++ structure of a data point handle used internally within the C interface.

This fact is hidden from the public because OGPS_DataPointPtr is previously introduced as an incomplete data type.

Public Attributes

- OpenGPS::DataPoint * instance

  Gets/Sets the pointer to the internal C++ object acting behind the scenes.

12.43.2 Member Data Documentation

12.43.2.1 OpenGPS::DataPoint* OGPS_DataPoint::instance

Gets/Sets the pointer to the internal C++ object acting behind the scenes.

The documentation for this struct was generated from the following file:

- data_point_c.hxx

12.44 OGPS_DataPointValue Union Reference

12.44.1 Detailed Description

Typesafe storage for every possible type of point data.

Public Attributes

- OGPS_Double doubleValue

  Stores a value of type OGPS_Double.

- OGPS_Float floatValue

  Stores a value of type OGPS_Float.

- OGPS_Int16 int16Value

  Stores a value of type OGPS_Int16.

- OGPS_Int32 int32Value

  Stores a value of type OGPS_Int32.

12.44.2 Member Data Documentation

12.44.2.1 OGPS_Double OGPS_DataPointValue::doubleValue

Stores a value of type OGPS_Double.

It is undefined if DataPointImpl::m_Type does not equal OGPS_DoublePointType.

Referenced by OpenGPS::DataPointImpl::DataPointImpl(), OpenGPS::DataPointImpl::Get(), OpenGPS::DataPointImpl::Reset(), and OpenGPS::DataPointImpl::Set().

12.44.2.2 OGPS_Float OGPS_DataPointValue::floatValue

Stores a value of type OGPS_Float.

It is undefined if DataPointImpl::m_Type does not equal OGPS_FloatPointType.

Referenced by OpenGPS::DataPointImpl::Get(), and OpenGPS::DataPointImpl::Set().

12.44.2.3 OGPS_Int16 OGPS_DataPointValue::int16Value

Stores a value of type OGPS_Int16.

It is undefined if DataPointImpl::m_Type does not equal OGPS_Int16PointType.

Referenced by OpenGPS::DataPointImpl::Get(), and OpenGPS::DataPointImpl::Set().

12.44.2.4 OGPS_Int32 OGPS_DataPointValue::int32Value

Stores a value of type OGPS_Int32.

It is undefined if DataPointImpl::m_Type does not equal OGPS_Int32PointType.

Referenced by OpenGPS::DataPointImpl::Get(), and OpenGPS::DataPointImpl::Set().

The documentation for this union was generated from the following file:

- data_point_impl.hxx

12.45 OGPS_ISO5436_2 Struct Reference

```
#include <iso5436_2_handle_c.hxx>
```

Collaboration diagram for OGPS_ISO5436_2:

12.45.1 Detailed Description

Encapsulates the internal C++ structure of an ISO5436-2 handle used within the C interface.

This fact is hidden from the public because OGPS_ISO5436_2Handle is defined as an incomplete data type.

Public Attributes

- OpenGPS::ISO5436_2 * instance

  Gets/Sets the pointer to the internal C++ object behind the scenes.

12.45.2 Member Data Documentation

12.45.2.1 OpenGPS::ISO5436_2* OGPS_ISO5436_2::instance

Gets/Sets the pointer to the internal C++ object behind the scenes.

The documentation for this struct was generated from the following file:

- iso5436_2_handle_c.hxx

12.46 OGPS_PointIterator Struct Reference

```
#include <point_iterator_c.hxx>
```

Collaboration diagram for OGPS_PointIterator:

12.46.1 Detailed Description

Encapsulates the internal C++ structure of a point iterator handle used within the C interface.

This fact is hidden from the public because OGPS_PointIteratorPtr is defined as an incomplete data type.

Public Attributes

- OpenGPS::PointIterator * instance
  Gets/Sets the pointer to the internal C++ object behind the scenes.

12.46.2 Member Data Documentation

12.46.2.1 OpenGPS::PointIterator* OGPS_PointIterator::instance

Gets/Sets the pointer to the internal C++ object behind the scenes.

The documentation for this struct was generated from the following file:

- point_iterator_c.hxx

12.47 OGPS_PointVector Struct Reference

```
#include <point_vector_c.hxx>
```

Collaboration diagram for OGPS_PointVector:

12.47.1 Detailed Description

Encapsulates the internal C++ structure of a point vector handle used within the C interface.

This fact is hidden from the public because OGPS_PointVectorPtr is defined as an incomplete data type.

Public Attributes

- OpenGPS::PointVector instance

  Gets/Sets the pointer to the internal C++ object behind the scenes.

- OGPS_DataPointPtr x

  Gets/Sets the buffered C interface wrapper for typesafe access to the X member of the internal C++ OpenGPS::PointVector instance.

- OGPS_DataPointPtr y

  Gets/Sets the buffered C interface wrapper for typesafe access to the Y member of the internal C++ OpenGPS::PointVector instance.

- OGPS_DataPointPtr z

  Gets/Sets the buffered C interface wrapper for typesafe access to the Z member of the internal C++ OpenGPS::PointVector instance.

12.47.2 Member Data Documentation

12.47.2.1 OpenGPS::PointVector OGPS_PointVector::instance

Gets/Sets the pointer to the internal C++ object behind the scenes.

12.47.2.2 OGPS_DataPointPtr OGPS_PointVector::x

Gets/Sets the buffered C interface wrapper for typesafe access to the X member of the internal C++ OpenGPS::PointVector instance.

12.47.2.3 OGPS_DataPointPtr OGPS_PointVector::y

Gets/Sets the buffered C interface wrapper for typesafe access to the Y member of the internal C++ OpenGPS::PointVector instance.

12.47.2.4 OGPS_DataPointPtr OGPS_PointVector::z

Gets/Sets the buffered C interface wrapper for typesafe access to the Z member of the internal C++ OpenGPS::PointVector instance.

The documentation for this struct was generated from the following file:

- point_vector_c.hxx

12.48 OpenGPS::OutputBinaryFileStream Class Reference

```
#include <point_vector_iostream.hxx>
```

Inheritance diagram for OpenGPS::OutputBinaryFileStream:

Collaboration diagram for OpenGPS::OutputBinaryFileStream:

12.48.1 Detailed Description

A binary stream class used for writing to binary files.

Public Member Functions

- OutputBinaryFileStream (const OpenGPS::String &filePath)
  Creates a new instance.
- ~OutputBinaryFileStream ()
  Destroys this instance.

Private Types

- typedef std::basic_ofstream< OpenGPS::UnsignedByte > BaseType
  The type of the super class.

Private Attributes

- PointVectorInvariantLocale m_Locale
  Invariant locale.

12.48.2 Member Typedef Documentation

12.48.2.1 typedef std::basic_ofstream<OpenGPS::UnsignedByte> OpenGPS::OutputBinaryFileStream::BaseType [private]

The type of the super class.

12.48.3 Constructor & Destructor Documentation

12.48.3.1 OutputBinaryFileStream::OutputBinaryFileStream (const OpenGPS::String & *filePath*)

Creates a new instance.

Parameters:

filePath Full path to the binary file to be written.

References m_Locale, and OpenGPS::String::ToChar().

Here is the call graph for this function:

12.48.3.2 OutputBinaryFileStream::~OutputBinaryFileStream ()

Destroys this instance.

12.48.4 Member Data Documentation

12.48.4.1 PointVectorInvariantLocale OpenGPS::OutputBinaryFileStream::m_Locale [private]

Invariant locale.

Referenced by OutputBinaryFileStream().

The documentation for this class was generated from the following files:

- point_vector_iostream.hxx
- point_vector_iostream.cxx

12.49 OpenGPS::PointBuffer Class Reference

```
#include <point_buffer.hxx>
```

Inheritance diagram for OpenGPS::PointBuffer:

12.49.1 Detailed Description

A static memory buffer that stores all point data belonging to a single axis.

Internally point vectors are not stored as a sequence of values of their three components (as one might have naturally thought). But rather there are three memory blocks. One for each axis that stores all its data point values in a single array. The three point buffer instances managing type-safe access to their memory blocks are assembled by the OpenGPS::VectorBufferBuilder and combined within an instance of OpenGPS::VectorBuffer. To make OpenGPS::VectorBuffer accessible as an instance of OpenGPS::PointVector one needs to obtain a OpenGPS::PointVectorProxy object by using OpenGPS::VectorBuffer::GetPointVectorProxy and providing a OpenGPS::PointVectorProxyContext object which holds the array index of the row which - expanded over the three internal memory blocks - constraints OpenGPS::VectorBuffer to be seen as one single OpenGPS::PointVector.

Public Member Functions

- virtual void Allocate (const OGPS_ULong size) throw (...)

  Allocates internal memory.

- virtual void Get (const OGPS_ULong index, OGPS_Double &value) const throw (...)

  Gets the value of the inernal memory at the given position.

- virtual void Get (const OGPS_ULong index, OGPS_Float &value) const throw (...)

  Gets the value of the inernal memory at the given position.

- virtual void Get (const OGPS_ULong index, OGPS_Int32 &value) const throw (...)

  Gets the value of the inernal memory at the given position.

- virtual void Get (const OGPS_ULong index, OGPS_Int16 &value) const throw (...)

  Gets the value of the inernal memory at the given position.

- virtual OGPS_DataPointType GetPointType () const

  Gets the type of point data that can be stored within this instance.

- virtual OGPS_ULong GetSize () const

  Gets the maximum amount of point data taht can be stored.

- virtual void Set (const OGPS_ULong index, const OGPS_Double value) throw (...)

  Sets the value of the inernal memory at the given position.

- virtual void Set (const OGPS_ULong index, const OGPS_Float value) throw (...)
  Sets the value of the inernal memory at the given position.
- virtual void Set (const OGPS_ULong index, const OGPS_Int32 value) throw (...)
  Sets the value of the inernal memory at the given position.
- virtual void Set (const OGPS_ULong index, const OGPS_Int16 value) throw (...)
  Sets the value of the inernal memory at the given position.
- virtual ~PointBuffer ()
  Destroys this instance.

Protected Member Functions

- OpenGPS::UnsignedBytePtr Allocate (const OGPS_ULong size, const size_t typeSize) throw (...)
  Allocates internal memory.
- void Free (OpenGPS::UnsignedBytePtr *value)
  Frees allocated memory.
- PointBuffer ()
  Creates a new instance.

Private Attributes

- OGPS_ULong m_Size
  Logical size or amount of point data that can be stored.

12.49.2 Constructor & Destructor Documentation

12.49.2.1 PointBuffer::PointBuffer () [protected]

Creates a new instance.

References m_Size.

12.49.2.2 PointBuffer::~PointBuffer () [virtual]

Destroys this instance.

12.49.3 Member Function Documentation

12.49.3.1 OpenGPS::UnsignedBytePtr PointBuffer::Allocate (const OGPS_ULong *size*, const size_t *typeSize*) throw (...) [protected]

Allocates internal memory.

Remarks:

Allocated memory must be free by PointBuffer::Free manually.

Throws an exception on failure.

Parameters:

size Amount of point data to be stored.

typeSize The size of the data type of the point data stored herein in bytes.

Returns:

Returns a pointer to allocated memory or NULL.

References _ASSERT, _EX_T, m_Size, and OGPS_ExGeneral.

12.49.3.2 void PointBuffer::Allocate (const OGPS_ULong *size*) throw (...) `[virtual]`

Allocates internal memory.

Throws an exception on failure.

Parameters:

size Amount of point data to be stored.

Reimplemented in OpenGPS::DoublePointBuffer, OpenGPS::FloatPointBuffer, OpenGPS::Int16PointBuffer, and OpenGPS::Int32PointBuffer.

References _EX_T, and OGPS_ExNotImplemented.

Referenced by OpenGPS::Int32PointBuffer::Allocate(), OpenGPS::Int16PointBuffer::Allocate(), OpenGPS::FloatPointBuffer::Allocate(), OpenGPS::DoublePointBuffer::Allocate(), and OpenGPS::VectorBufferBuilder::CreatePointBuffer().

12.49.3.3 void PointBuffer::Free (OpenGPS::UnsignedBytePtr * *value*) `[protected]`

Frees allocated memory.

See also:

PointBuffer::Allocate

Parameters:

value The pointer to the memory to be freed.

References _OPENGPS_FREE.

Referenced by OpenGPS::DoublePointBuffer::~DoublePointBuffer(), OpenGPS::FloatPointBuffer::~FloatPointBuffer(), OpenGPS::Int16PointBuffer::~Int16PointBuffer(), and OpenGPS::Int32PointBuffer::~Int32PointBuffer().

12.49.3.4 void PointBuffer::Get (const OGPS_ULong *index*, OGPS_Double & *value*) const throw (...) [virtual]

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented in OpenGPS::DoublePointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.49.3.5 void PointBuffer::Get (const OGPS_ULong *index*, OGPS_Float & *value*) const throw (...) [virtual]

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented in OpenGPS::FloatPointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.49.3.6 void PointBuffer::Get (const OGPS_ULong *index*, OGPS_Int32 & *value*) const throw (...) [virtual]

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented in OpenGPS::Int32PointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.49.3.7 void PointBuffer::Get (const OGPS_ULong *index*, OGPS_Int16 & *value*) const throw (...) [virtual]

Gets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Get the value from this position.

value Stores the value.

Reimplemented in OpenGPS::Int16PointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

Referenced by OpenGPS::PointVectorProxy::DataPointProxy::Get(), OpenGPS::DoubleInlineValidity::IsValid(), and OpenGPS::FloatInlineValidity::IsValid().

12.49.3.8 OGPS_DataPointType PointBuffer::GetPointType () const [virtual]

Gets the type of point data that can be stored within this instance.

Reimplemented in OpenGPS::DoublePointBuffer, OpenGPS::FloatPointBuffer, OpenGPS::Int16PointBuffer, and OpenGPS::Int32PointBuffer.

References OGPS_MissingPointType.

Referenced by OpenGPS::VectorBufferBuilder::BuildValidityProvider(), OpenGPS::DoubleInlineValidity::DoubleInlineValidity(), OpenGPS::FloatInlineValidity::FloatInlineValidity(), OpenGPS::PointVectorProxy::DataPointProxy::GetPointType(), OpenGPS::Int16ValidBuffer::Int16ValidBuffer(), and OpenGPS::Int32ValidBuffer::Int32ValidBuffer().

12.49.3.9 OGPS_ULong PointBuffer::GetSize () const [virtual]

Gets the maximum amount of point data taht can be stored.

References m_Size.

Referenced by OpenGPS::ValidBuffer::Allocate(), OpenGPS::Int32PointBuffer::Get(), OpenGPS::Int16PointBuffer::Get(), OpenGPS::FloatPointBuffer::Get(), OpenGPS::DoublePointBuffer::Get(), OpenGPS::ValidBuffer::HasInvalidMarks(), OpenGPS::Int32PointBuffer::Set(), OpenGPS::Int16PointBuffer::Set(), OpenGPS::FloatPointBuffer::Set(), and OpenGPS::DoublePointBuffer::Set().

12.49.3.10 void PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Double *value*) throw (...) [virtual]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented in OpenGPS::DoublePointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.49.3.11 void PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Float *value*) throw (...) [virtual]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented in OpenGPS::FloatPointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.49.3.12 void PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Int32 *value*) throw (...) [virtual]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented in OpenGPS::Int32PointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

12.49.3.13 void PointBuffer::Set (const OGPS_ULong *index*, const OGPS_Int16 *value*) throw (...) [virtual]

Sets the value of the inernal memory at the given position.

An exception might be thrown when this typed operation is not valid for a particular instance.

Parameters:

index Where to set the new value.

value The new value to be set.

Reimplemented in OpenGPS::Int16PointBuffer.

References _EX_T, and OGPS_ExInvalidOperation.

Referenced by OpenGPS::PointVectorProxy::DataPointProxy::Set(), OpenGPS::Int32ValidBuffer::SetValid(), OpenGPS::Int16ValidBuffer::SetValid(), OpenGPS::DoubleInlineValidity::SetValid(), and OpenGPS::FloatInlineValidity::SetValid().

12.49.4 Member Data Documentation

12.49.4.1 OGPS_ULong OpenGPS::PointBuffer::m_Size [private]

Logical size or amount of point data that can be stored.

Referenced by Allocate(), GetSize(), and PointBuffer().

The documentation for this class was generated from the following files:

- point_buffer.hxx
- point_buffer.cxx

12.50 OpenGPS::PointIterator Class Reference

`#include <point_iterator.hxx>`

Inheritance diagram for OpenGPS::PointIterator:

12.50.1 Detailed Description

Interface to a point iterator.

The point iterator may be used to traverse all point vectors contained in an ISO5436-2 XML X3P file format container.

Remarks:

An instance of OpenGPS::PointIterator can be obtained from OpenGPS::ISO5436_2::CreateNextPointIterator or OpenGPS::ISO5436_2::CreatePrevPointIterator.

Public Member Functions

- virtual void GetCurrent (PointVector &vector)=0 throw (...)

  Gets the value of the current point vector.

- virtual OGPS_Boolean GetPosition (OGPS_ULong *const index) const =0

  Gets the current position of the iterator.

- virtual OGPS_Boolean GetPosition (OGPS_ULong *const u, OGPS_ULong *const v, OGPS_ULong *const w) const =0

  Gets the current position of the iterator in topology coordinates.

- virtual OGPS_Boolean HasNext () const =0 throw (...)

  Asks if there is another point available to iterate.

- virtual OGPS_Boolean HasPrev () const =0 throw (...)

  Asks if there is another point available to iterate.

- virtual OGPS_Boolean MoveNext ()=0 throw (...)

  Moves the iterator forward.

- virtual OGPS_Boolean MovePrev ()=0 throw (...)

  Moves the iterator backward.

- virtual void ResetNext ()=0

  Resets the iterator to the beginning and turns this iterator instance into a forward iterator.

- virtual void ResetPrev ()=0

Resets the iterator to the beginning and turns this iterator instance into a backward iterator.

- virtual void SetCurrent (const PointVector *const vector)=0 throw (...)

  Sets the value of the current point vector.

- virtual ~PointIterator ()

  Destructs this instance.

Protected Member Functions

- PointIterator ()

  Creates a new instance.

12.50.2 Constructor & Destructor Documentation

12.50.2.1 PointIterator::PointIterator () [protected]

Creates a new instance.

12.50.2.2 PointIterator::~PointIterator () [virtual]

Destructs this instance.

12.50.3 Member Function Documentation

12.50.3.1 virtual void OpenGPS::PointIterator::GetCurrent (PointVector & *vector*) throw (...) [pure virtual]

Gets the value of the current point vector.

A specific instance may throw an OpenGPS::Exception on failure,

See also:

PointIterator::MoveNext

Parameters:

vector Gets a copy of the vector at the current iterator position.

Implemented in OpenGPS::ISO5436_2Container::PointIteratorImpl.

12.50.3.2 virtual OGPS_Boolean OpenGPS::PointIterator::GetPosition (OGPS_ULong *const *index*) const [pure virtual]

Gets the current position of the iterator.

See also:

PointIterator::MoveNext

Parameters:

index Gets the position index.

Returns:

Returns TRUE on success, FALSE otherwise.

Implemented in OpenGPS::ISO5436_2Container::PointIteratorImpl.

12.50.3.3 virtual OGPS_Boolean OpenGPS::PointIterator::GetPosition (OGPS_ULong *const *u*, OGPS_ULong *const *v*, OGPS_ULong *const *w*) const [pure virtual]

Gets the current position of the iterator in topology coordinates.

See also:

PointIterator::MoveNext

Parameters:

u Gets the position index of the u component of the surface. If this parameter is set to NULL, it will be safely ignored and nothing returns here.

v Gets the position index of the v component of the surface. If this parameter is set to NULL, it will be safely ignored and nothing returns here.

w Gets the position index of the w component of the surface. If this parameter is set to NULL, it will be safely ignored and nothing returns here.

Returns:

Returns TRUE on success, FALSE otherwise.

Implemented in OpenGPS::ISO5436_2Container::PointIteratorImpl.

12.50.3.4 virtual OGPS_Boolean OpenGPS::PointIterator::HasNext () const throw (...) [pure virtual]

Asks if there is another point available to iterate.

Remarks:

Use this function with an iterator handle obtained from OpenGPS::ISO5436_2::CreateNextPointIterator.

See also:

PointIterator::MoveNext, PointIterator::HasPrev

Returns:

Returns TRUE if there is at least one more point available to iterate, FALSE otherwise.

Implemented in OpenGPS::ISO5436_2Container::PointIteratorImpl.

12.50.3.5 virtual OGPS_Boolean OpenGPS::PointIterator::HasPrev () const throw (...) [pure virtual]

Asks if there is another point available to iterate.

Remarks:

Use this function with an iterator handle obtained from OpenGPS::ISO5436_-2::CreatePrevPointIterator.

See also:

PointIterator::MovePrev, PointIterator::HasNext

Returns:

Returns TRUE if there is at least one more point available to iterate, FALSE otherwise.

Implemented in OpenGPS::ISO5436_2Container::PointIteratorImpl.

12.50.3.6 virtual OGPS_Boolean OpenGPS::PointIterator::MoveNext () throw (...) [pure virtual]

Moves the iterator forward.

Remarks:

Use this function directly after initialising the iterator object with OpenGPS::ISO5436_-2::CreateNextPointIterator to move to the first point.

See also:

PointIterator::HasNext

Returns:

Returns TRUE on success, FALSE otherwise.

Implemented in OpenGPS::ISO5436_2Container::PointIteratorImpl.

12.50.3.7 virtual OGPS_Boolean OpenGPS::PointIterator::MovePrev () throw (...) [pure virtual]

Moves the iterator backward.

Remarks:

Use this function directly after initialising the iterator object with OpenGPS::ISO5436_-2::CreatePrevPointIterator to move to the first point.

See also:

PointIterator::HasPrev

Returns:

Returns TRUE on success, FALSE otherwise.

Implemented in OpenGPS::ISO5436_2Container::PointIteratorImpl.

12.50.3.8 virtual void OpenGPS::PointIterator::ResetNext () [pure virtual]

Resets the iterator to the beginning and turns this iterator instance into a forward iterator.

Implemented in OpenGPS::ISO5436_2Container::PointIteratorImpl.

12.50.3.9 virtual void OpenGPS::PointIterator::ResetPrev () [pure virtual]

Resets the iterator to the beginning and turns this iterator instance into a backward iterator.

Implemented in OpenGPS::ISO5436_2Container::PointIteratorImpl.

12.50.3.10 virtual void OpenGPS::PointIterator::SetCurrent (const PointVector *const *vector*) throw (...) [pure virtual]

Sets the value of the current point vector.

A specific instance may throw an OpenGPS::Exception on failure,

See also:

PointIterator::MoveNext

Parameters:

vector New value of the current point vector. May be NULL to indicate an invalid point.

Returns:

Returns TRUE on success, FALSE otherwise.

Implemented in OpenGPS::ISO5436_2Container::PointIteratorImpl.

The documentation for this class was generated from the following files:

- point_iterator.hxx
- point_iterator.cxx

12.51 OpenGPS::PointValidityProvider Class Reference

```
#include <point_validity_provider.hxx>
```

Inheritance diagram for OpenGPS::PointValidityProvider:

Collaboration diagram for OpenGPS::PointValidityProvider:

12.51.1 Detailed Description

Interface to communicate the validity of a point vector at a given location.

Public Member Functions

- virtual OGPS_Boolean IsValid (const unsigned int index) const =0 throw (...)

  Gets the validity of a point vector at a given location.

- virtual void SetValid (const unsigned int index, const OGPS_Boolean value)=0 throw (...)

  Sets the validity of a point vector at a given location.

- virtual ~PointValidityProvider ()

  Destroys this instance.

Protected Member Functions

- PointBuffer * GetPointBuffer ()

  Returns the point buffer of the Z axis.

- const PointBuffer * GetPointBuffer () const

  Returns the point buffer of the Z axis.

- PointValidityProvider (PointBuffer *const value)

  Creates a new instance.

Private Member Functions

- PointValidityProvider ()

  Not to be publicly used ctor.

Private Attributes

- PointBuffer * m_PointBuffer

  Point buffer of the Z axis.

12.51.2 Constructor & Destructor Documentation

12.51.2.1 PointValidityProvider::~PointValidityProvider () [virtual]

Destroys this instance.

12.51.2.2 PointValidityProvider::PointValidityProvider (PointBuffer *const *value*) [protected]

Creates a new instance.

Parameters:

value The point buffer of the Z axis. The Z axis is used to communicate validity because the Z axis is the only axis which cannot have values implicitly and therefore must always provide a valid measurement.

References _ASSERT, and m_PointBuffer.

12.51.2.3 PointValidityProvider::PointValidityProvider () [private]

Not to be publicly used ctor.

References _ASSERT, FALSE, m_PointBuffer, and NULL.

12.51.3 Member Function Documentation

12.51.3.1 PointBuffer * PointValidityProvider::GetPointBuffer () [protected]

Returns the point buffer of the Z axis.

References m_PointBuffer.

12.51.3.2 const PointBuffer * PointValidityProvider::GetPointBuffer () const [protected]

Returns the point buffer of the Z axis.

References m_PointBuffer.

Referenced by OpenGPS::ValidBuffer::Allocate(), OpenGPS::ValidBuffer::HasInvalidMarks(), OpenGPS::ValidBuffer::IsValid(), OpenGPS::DoubleInlineValidity::IsValid(), OpenGPS::FloatInlineValidity::IsValid(), OpenGPS::Int32ValidBuffer::SetValid(), OpenGPS::Int16ValidBuffer::SetValid(), OpenGPS::ValidBuffer::SetValid(), OpenGPS::DoubleInlineValidity::SetValid(), and OpenGPS::FloatInlineValidity::SetValid().

12.51.3.3 virtual OGPS_Boolean OpenGPS::PointValidityProvider::IsValid (const unsigned int *index*) const throw (...) [pure virtual]

Gets the validity of a point vector at a given location.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

index The location of the point data the validity is checked.

Returns:

Returns TRUE if valid, FALSE otherwise.

Implemented in OpenGPS::FloatInlineValidity, OpenGPS::DoubleInlineValidity, and OpenGPS::ValidBuffer.

Referenced by OpenGPS::ISO5436_2Container::GetMatrixPoint(), and OpenGPS::ISO5436_-2Container::IsMatrixCoordValid().

12.51.3.4 virtual void OpenGPS::PointValidityProvider::SetValid (const unsigned int *index*, const OGPS_Boolean *value*) throw (...) `[pure virtual]`

Sets the validity of a point vector at a given location.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

index The location of the point data the validity is updated.

value The value of the current validity of the point vector at the given position.

Implemented in OpenGPS::FloatInlineValidity, OpenGPS::DoubleInlineValidity, OpenGPS::ValidBuffer, OpenGPS::Int16ValidBuffer, and OpenGPS::Int32ValidBuffer.

Referenced by OpenGPS::ISO5436_2Container::CreatePointBuffer(), and OpenGPS::ISO5436_-2Container::SetMatrixPoint().

12.51.4 Member Data Documentation

12.51.4.1 PointBuffer* OpenGPS::PointValidityProvider::m_PointBuffer `[private]`

Point buffer of the Z axis.

Referenced by GetPointBuffer(), and PointValidityProvider().

The documentation for this class was generated from the following files:

- point_validity_provider.hxx
- point_validity_provider.cxx

12.52 OpenGPS::PointVector Class Reference

```
#include <point_vector.hxx>
```

Inheritance diagram for OpenGPS::PointVector:

Collaboration diagram for OpenGPS::PointVector:

12.52.1 Detailed Description

Typesafe representation of three-dimensional point measurement data.

Public Member Functions

- virtual void Get (PointVectorBase &value) const throw (...)

  Copies values to another OpenGPS::PointVectorBase instance.

- void GetX (OGPS_Double *const value) const throw (...)

  Gets the value of the x component of the given vector.

- void GetX (OGPS_Float *const value) const throw (...)

  Gets the value of the x component of the given vector.

- void GetX (OGPS_Int32 *const value) const throw (...)

  Gets the value of the x component of the given vector.

- void GetX (OGPS_Int16 *const value) const throw (...)

  Gets the value of the x component of the given vector.

- virtual DataPoint * GetX ()

  Gets typesafe direct access to the x component.

- virtual const DataPoint * GetX () const

  Gets typesafe direct read-only access to the x component.

- void GetXYZ (OGPS_Double *const x, OGPS_Double *const y, OGPS_Double *const z) const throw (...)

  Gets the values of each component.

- void GetY (OGPS_Double *const value) const throw (...)

  Gets the value of the y component of the given vector.

- void GetY (OGPS_Float *const value) const throw (...)

  Gets the value of the y component of the given vector.

- void GetY (OGPS_Int32 *const value) const throw (...)

  Gets the value of the y component of the given vector.

- void GetY (OGPS_Int16 *const value) const throw (...)

  Gets the value of the y component of the given vector.

- virtual DataPoint * GetY ()

  Gets typesafe direct access to the y component.

- virtual const DataPoint * GetY () const

  Gets typesafe direct read-only access to the y component.

- void GetZ (OGPS_Double *const value) const throw (...)

  Gets the value of the z component of the given vector.

- void GetZ (OGPS_Float *const value) const throw (...)

  Gets the value of the z component of the given vector.

- void GetZ (OGPS_Int32 *const value) const throw (...)

  Gets the value of the z component of the given vector.

- void GetZ (OGPS_Int16 *const value) const throw (...)

  Gets the value of the z component of the given vector.

- virtual DataPoint * GetZ ()

  Gets typesafe direct access to the z component.

- virtual const DataPoint * GetZ () const

  Gets typesafe direct read-only access to the z component.

- virtual OGPS_Boolean IsValid () const throw (...)

  Asks if this point vector stores a valid data point.

- PointVector & operator= (const PointVector &src) throw (...)
- PointVector ()

  Creates a new instance.

- virtual void Set (const PointVectorBase &value) throw (...)

  Copies values from another OpenGPS::PointVector instance.

- void SetX (const OGPS_Double value) throw (...)

  Sets the new value for the x component.

- void SetX (const OGPS_Float value) throw (...)

  Sets the new value for the x component.

- void SetX (const OGPS_Int32 value) throw (...)

  Sets the new value for the x component.

- void SetX (const OGPS_Int16 value) throw (...)

  Sets the new value for the x component.

- void SetY (const OGPS_Double value) throw (...)

Sets the new value for the y component.

- void SetY (const OGPS_Float value) throw (...)

  Sets the new value for the y component.

- void SetY (const OGPS_Int32 value) throw (...)

  Sets the new value for the y component.

- void SetY (const OGPS_Int16 value) throw (...)

  Sets the new value for the y component.

- void SetZ (const OGPS_Double value) throw (...)

  Sets the new value for the z component.

- void SetZ (const OGPS_Float value) throw (...)

  Sets the new value for the z component.

- void SetZ (const OGPS_Int32 value) throw (...)

  Sets the new value for the z component.

- void SetZ (const OGPS_Int16 value) throw (...)

  Sets the new value for the z component.

- ~PointVector ()

  Destructs this object.

Private Attributes

- DataPoint * m_X
- DataPoint * m_Y
- DataPoint * m_Z

12.52.2 Constructor & Destructor Documentation

12.52.2.1 PointVector::PointVector ()

Creates a new instance.

References m_X, m_Y, and m_Z.

12.52.2.2 PointVector::~PointVector ()

Destructs this object.

References _OPENGPS_DELETE, m_X, m_Y, and m_Z.

12.52.3 Member Function Documentation

12.52.3.1 void PointVector::Get (PointVectorBase & *value*) const throw (...) [virtual]

Copies values to another OpenGPS::PointVectorBase instance.

Throws an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Retrieves values from the current instance as a copy.

Implements OpenGPS::PointVectorBase.

References _ASSERT, m_X, m_Y, and m_Z.

12.52.3.2 void PointVector::GetX (OGPS_Double *const *value*) const throw (...)

Gets the value of the x component of the given vector.

Throws an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance or conflicting types.

See also:

OGPS_DataPointType

Parameters:

value Stores the current value on success.

References _ASSERT, OpenGPS::DataPoint::Get(), and m_X.

Here is the call graph for this function:

12.52.3.3 void PointVector::GetX (OGPS_Float *const *value*) const throw (...)

Gets the value of the x component of the given vector.

Throws an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance or conflicting types.

See also:

OGPS_DataPointType

Parameters:

value Stores the current value on success.

References _ASSERT, OpenGPS::DataPoint::Get(), and m_X.

Here is the call graph for this function:

12.52.3.4 void PointVector::GetX (OGPS_Int32 *const *value*) const throw (...)

Gets the value of the x component of the given vector.

Throws an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance or conflicting types.

See also:

OGPS_DataPointType

Parameters:

value Stores the current value on success.

References _ASSERT, OpenGPS::DataPoint::Get(), and m_X.

Here is the call graph for this function:

12.52.3.5 void PointVector::GetX (OGPS_Int16 *const *value*) const throw (...)

Gets the value of the x component of the given vector.

Throws an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance or conflicting types.

See also:

OGPS_DataPointType

Parameters:

value Stores the current value on success.

References _ASSERT, OpenGPS::DataPoint::Get(), and m_X.

Here is the call graph for this function:

12.52.3.6 DataPoint * PointVector::GetX () `[virtual]`

Gets typesafe direct access to the x component.

Implements OpenGPS::PointVectorBase.

References _ASSERT, and m_X.

12.52.3.7 const DataPoint * PointVector::GetX () const [virtual]

Gets typesafe direct read-only access to the x component.

Implements OpenGPS::PointVectorBase.

References _ASSERT, and m_X.

Referenced by readonlyExample3().

12.52.3.8 void PointVector::GetXYZ (OGPS_Double *const *x*, OGPS_Double *const *y*, OGPS_Double *const *z*) const throw (...)

Gets the values of each component.

Parameters:

- ***x*** Gets the value of the x component. If this parameter is set to NULL, this does nothing.
- ***y*** Gets the value of the y component. If this parameter is set to NULL, this does nothing.
- ***z*** Gets the value of the z component. If this parameter is set to NULL, this does nothing.

References _ASSERT, OpenGPS::DataPoint::Get(), m_X, m_Y, m_Z, and NULL.

Here is the call graph for this function:

12.52.3.9 void PointVector::GetY (OGPS_Double *const *value*) const throw (...)

Gets the value of the y component of the given vector.

Throws an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance or conflicting types.

See also:

OGPS_DataPointType

Parameters:

- ***value*** Stores the current value on success.

References _ASSERT, OpenGPS::DataPoint::Get(), and m_Y.

Here is the call graph for this function:

12.52.3.10 void PointVector::GetY (OGPS_Float *const *value*) const throw (...)

Gets the value of the y component of the given vector.

Throws an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance or conflicting types.

See also:

OGPS_DataPointType

Parameters:

value Stores the current value on success.

References _ASSERT, OpenGPS::DataPoint::Get(), and m_Y.

Here is the call graph for this function:

12.52.3.11 void PointVector::GetY (OGPS_Int32 *const *value*) const throw (...)

Gets the value of the y component of the given vector.

Throws an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance or conflicting types.

See also:

OGPS_DataPointType

Parameters:

value Stores the current value on success.

References _ASSERT, OpenGPS::DataPoint::Get(), and m_Y.

Here is the call graph for this function:

12.52.3.12 void PointVector::GetY (OGPS_Int16 *const *value*) const throw (...)

Gets the value of the y component of the given vector.

Throws an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance or conflicting types.

See also:

OGPS_DataPointType

Parameters:

value Stores the current value on success.

References _ASSERT, OpenGPS::DataPoint::Get(), and m_Y.

Here is the call graph for this function:

12.52.3.13 DataPoint * PointVector::GetY () [virtual]

Gets typesafe direct access to the y component.

Implements OpenGPS::PointVectorBase.

References _ASSERT, and m_Y.

12.52.3.14 const DataPoint * PointVector::GetY () const [virtual]

Gets typesafe direct read-only access to the y component.

Implements OpenGPS::PointVectorBase.

References _ASSERT, and m_Y.

Referenced by readonlyExample3().

12.52.3.15 void PointVector::GetZ (OGPS_Double *const *value*) const throw (...)

Gets the value of the z component of the given vector.

Throws an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance or conflicting types.

See also:

OGPS_DataPointType

Parameters:

value Stores the current value on success.

References _ASSERT, OpenGPS::DataPoint::Get(), and m_Z.

Here is the call graph for this function:

12.52.3.16 void PointVector::GetZ (OGPS_Float *const *value*) const throw (...)

Gets the value of the z component of the given vector.

Throws an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance or conflicting types.

See also:

OGPS_DataPointType

Parameters:

value Stores the current value on success.

References _ASSERT, OpenGPS::DataPoint::Get(), and m_Z.

Here is the call graph for this function:

12.52.3.17 void PointVector::GetZ (OGPS_Int32 *const *value*) const throw (...)

Gets the value of the z component of the given vector.

Throws an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance or conflicting types.

See also:

OGPS_DataPointType

Parameters:

value Stores the current value on success.

References _ASSERT, OpenGPS::DataPoint::Get(), and m_Z.

Here is the call graph for this function:

12.52.3.18 void PointVector::GetZ (OGPS_Int16 *const *value*) const throw (...)

Gets the value of the z component of the given vector.

Throws an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance or conflicting types.

See also:

OGPS_DataPointType

Parameters:

value Stores the current value on success.

References _ASSERT, OpenGPS::DataPoint::Get(), and m_Z.

Here is the call graph for this function:

12.52.3.19 DataPoint * PointVector::GetZ () `[virtual]`

Gets typesafe direct access to the z component.

Implements OpenGPS::PointVectorBase.

References _ASSERT, and m_Z.

12.52.3.20 const DataPoint * PointVector::GetZ () const [virtual]

Gets typesafe direct read-only access to the z component.

Implements OpenGPS::PointVectorBase.

References _ASSERT, and m_Z.

Referenced by readonlyExample3().

12.52.3.21 OGPS_Boolean PointVector::IsValid () const throw (...) [virtual]

Asks if this point vector stores a valid data point.

A valid point vector does not have components where some or all of its values are missing. Missing or invalid points are indicated by OGPS_MissingPointType.

Returns:

Returns TRUE if this point vector contains valid point components only, FALSE otherwise.

References _ASSERT, OpenGPS::DataPoint::IsValid(), m_X, m_Y, and m_Z.

Referenced by readonlyExample3().

Here is the call graph for this function:

12.52.3.22 PointVector & PointVector::operator= (const PointVector & *src*) throw (...)

References _ASSERT, m_X, m_Y, m_Z, and Set().

Here is the call graph for this function:

12.52.3.23 void PointVector::Set (const PointVectorBase & *value*) throw (...) [virtual]

Copies values from another OpenGPS::PointVector instance.

Throws an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value OpenGPS::PointVectorBase object to copy from.

Implements OpenGPS::PointVectorBase.

References _ASSERT, m_X, m_Y, m_Z, and OpenGPS::DataPoint::Set().

Referenced by operator=().

Here is the call graph for this function:

12.52.3.24 void PointVector::SetX (const OGPS_Double *value*) throw (...)

Sets the new value for the x component.

Parameters:

value The new value.

References _ASSERT, m_X, and OpenGPS::DataPoint::Set().

Here is the call graph for this function:

12.52.3.25 void PointVector::SetX (const OGPS_Float *value*) throw (...)

Sets the new value for the x component.

Parameters:

value The new value.

References _ASSERT, m_X, and OpenGPS::DataPoint::Set().

Here is the call graph for this function:

12.52.3.26 void PointVector::SetX (const OGPS_Int32 *value*) throw (...)

Sets the new value for the x component.

Parameters:

value The new value.

References _ASSERT, m_X, and OpenGPS::DataPoint::Set().

Here is the call graph for this function:

12.52.3.27 void PointVector::SetX (const OGPS_Int16 *value*) throw (...)

Sets the new value for the x component.

Parameters:

value The new value.

References _ASSERT, m_X, and OpenGPS::DataPoint::Set().

Here is the call graph for this function:

12.52.3.28 void PointVector::SetY (const OGPS_Double *value*) throw (...)

Sets the new value for the y component.

Parameters:

value The new value.

References _ASSERT, m_Y, and OpenGPS::DataPoint::Set().

Here is the call graph for this function:

12.52.3.29 void PointVector::SetY (const OGPS_Float *value*) throw (...)

Sets the new value for the y component.

Parameters:

value The new value.

References _ASSERT, m_Y, and OpenGPS::DataPoint::Set().

Here is the call graph for this function:

12.52.3.30 void PointVector::SetY (const OGPS_Int32 *value*) throw (...)

Sets the new value for the y component.

Parameters:

value The new value.

References _ASSERT, m_Y, and OpenGPS::DataPoint::Set().

Here is the call graph for this function:

12.52.3.31 void PointVector::SetY (const OGPS_Int16 *value*) throw (...)

Sets the new value for the y component.

Parameters:

value The new value.

References _ASSERT, m_Y, and OpenGPS::DataPoint::Set().

Here is the call graph for this function:

12.52.3.32 void PointVector::SetZ (const OGPS_Double *value*) throw (...)

Sets the new value for the z component.

Parameters:

value The new value.

References _ASSERT, m_Z, and OpenGPS::DataPoint::Set().

Here is the call graph for this function:

12.52.3.33 void PointVector::SetZ (const OGPS_Float *value*) throw (...)

Sets the new value for the z component.

Parameters:

value The new value.

References _ASSERT, m_Z, and OpenGPS::DataPoint::Set().

Here is the call graph for this function:

12.52.3.34 void PointVector::SetZ (const OGPS_Int32 *value*) throw (...)

Sets the new value for the z component.

Parameters:

value The new value.

References _ASSERT, m_Z, and OpenGPS::DataPoint::Set().

Here is the call graph for this function:

12.52.3.35 void PointVector::SetZ (const OGPS_Int16 *value*) throw (...)

Sets the new value for the z component.

Parameters:

value The new value.

References _ASSERT, m_Z, and OpenGPS::DataPoint::Set().

Here is the call graph for this function:

12.52.4 Member Data Documentation

12.52.4.1 DataPoint* OpenGPS::PointVector::m_X [private]

Referenced by Get(), GetX(), GetXYZ(), IsValid(), operator=(), PointVector(), Set(), SetX(), and ~PointVector().

12.52.4.2 DataPoint* OpenGPS::PointVector::m_Y [private]

Referenced by Get(), GetXYZ(), GetY(), IsValid(), operator=(), PointVector(), Set(), SetY(), and ~PointVector().

12.52.4.3 DataPoint* OpenGPS::PointVector::m_Z [private]

Referenced by Get(), GetXYZ(), GetZ(), IsValid(), operator=(), PointVector(), Set(), SetZ(), and ~PointVector().

The documentation for this class was generated from the following files:

- point_vector.hxx
- point_vector.cxx

12.53 OpenGPS::PointVectorBase Class Reference

`#include <point_vector_base.hxx>`

Inheritance diagram for OpenGPS::PointVectorBase:

12.53.1 Detailed Description

Typesafe and very fundamental representation of three-dimensional point measurement data.

See also:

OpenGPS::PointVector

Public Member Functions

- virtual void Get (PointVectorBase &value) const =0 throw (...)

  Copies values to another OpenGPS::PointVectorBase instance.

- virtual DataPoint * GetX ()=0

  Gets typesafe direct access to the x component.

- virtual const DataPoint * GetX () const =0

  Gets typesafe direct read-only access to the x component.

- virtual DataPoint * GetY ()=0

  Gets typesafe direct access to the y component.

- virtual const DataPoint * GetY () const =0

  Gets typesafe direct read-only access to the y component.

- virtual DataPoint * GetZ ()=0

  Gets typesafe direct access to the z component.

- virtual const DataPoint * GetZ () const =0

  Gets typesafe direct read-only access to the z component.

- virtual void Set (const PointVectorBase &value)=0 throw (...)

  Copies values from another OpenGPS::PointVector instance.

- virtual ~PointVectorBase ()

  Destructs this object.

Protected Member Functions

- PointVectorBase ()

  Creates a new instance.

12.53.2 Constructor & Destructor Documentation

12.53.2.1 PointVectorBase::~PointVectorBase () [virtual]

Destructs this object.

12.53.2.2 PointVectorBase::PointVectorBase () [protected]

Creates a new instance.

12.53.3 Member Function Documentation

12.53.3.1 virtual void OpenGPS::PointVectorBase::Get (PointVectorBase & *value*) const throw (...) [pure virtual]

Copies values to another OpenGPS::PointVectorBase instance.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Retrieves values from the current instance as a copy.

Implemented in OpenGPS::PointVector, and OpenGPS::PointVectorProxy.

12.53.3.2 virtual DataPoint* OpenGPS::PointVectorBase::GetX () [pure virtual]

Gets typesafe direct access to the x component.

Implemented in OpenGPS::PointVector, and OpenGPS::PointVectorProxy.

12.53.3.3 virtual const DataPoint* OpenGPS::PointVectorBase::GetX () const [pure virtual]

Gets typesafe direct read-only access to the x component.

Implemented in OpenGPS::PointVector, and OpenGPS::PointVectorProxy.

12.53.3.4 virtual DataPoint* OpenGPS::PointVectorBase::GetY () [pure virtual]

Gets typesafe direct access to the y component.

Implemented in OpenGPS::PointVector, and OpenGPS::PointVectorProxy.

12.53.3.5 virtual const DataPoint* OpenGPS::PointVectorBase::GetY () const [pure virtual]

Gets typesafe direct read-only access to the y component.

Implemented in OpenGPS::PointVector, and OpenGPS::PointVectorProxy.

12.53.3.6 virtual DataPoint* OpenGPS::PointVectorBase::GetZ () [pure virtual]

Gets typesafe direct access to the z component.

Implemented in OpenGPS::PointVector, and OpenGPS::PointVectorProxy.

12.53.3.7 virtual const DataPoint* OpenGPS::PointVectorBase::GetZ () const [pure virtual]

Gets typesafe direct read-only access to the z component.

Implemented in OpenGPS::PointVector, and OpenGPS::PointVectorProxy.

12.53.3.8 virtual void OpenGPS::PointVectorBase::Set (const PointVectorBase & *value*) throw (...) [pure virtual]

Copies values from another OpenGPS::PointVector instance.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value OpenGPS::PointVectorBase object to copy from.

Implemented in OpenGPS::PointVector, and OpenGPS::PointVectorProxy.

The documentation for this class was generated from the following files:

- point_vector_base.hxx
- point_vector.cxx

12.54 OpenGPS::PointVectorInputStringStream Class Reference

```
#include <point_vector_iostream.hxx>
```

Inheritance diagram for OpenGPS::PointVectorInputStringStream:

Collaboration diagram for OpenGPS::PointVectorInputStringStream:

12.54.1 Detailed Description

A string stream used for parsing a point vector as defined in ISO5436-2 XML.

Public Member Functions

- PointVectorInputStringStream (const OpenGPS::String &s)

  Creates a new instance.

- PointVectorInputStringStream ()

  Creates a new instance.

- ~PointVectorInputStringStream ()

  Destroys this instance.

Private Types

- typedef std::basic_istringstream< OGPS_Character > BaseType

  The type of the super class.

Private Attributes

- PointVectorInvariantLocale m_Locale

  Invariant locale.

12.54.2 Member Typedef Documentation

12.54.2.1 typedef std::basic_istringstream<OGPS_Character> OpenGPS::PointVectorInputStringStream::BaseType [private]

The type of the super class.

12.54.3 Constructor & Destructor Documentation

12.54.3.1 PointVectorInputStringStream::PointVectorInputStringStream ()

Creates a new instance.

References m_Locale.

12.54.3.2 PointVectorInputStringStream::PointVectorInputStringStream (const OpenGPS::String & *s*)

Creates a new instance.

Parameters:

s The source which is streamed.

References m_Locale.

12.54.3.3 PointVectorInputStringStream::~PointVectorInputStringStream ()

Destroys this instance.

12.54.4 Member Data Documentation

12.54.4.1 PointVectorInvariantLocale OpenGPS::PointVectorInputStringStream::m_Locale [private]

Invariant locale.

Referenced by PointVectorInputStringStream().

The documentation for this class was generated from the following files:

- point_vector_iostream.hxx
- point_vector_iostream.cxx

12.55 OpenGPS::PointVectorInvariantLocale Class Reference

```
#include <point_vector_iostream.hxx>
```

12.55.1 Detailed Description

The culture invariant locale.

Uses an instance of OpenGPS::PointVectorWhitespaceFacet to classify white space.

Public Member Functions

- PointVectorInvariantLocale ()
  Creates a new instance.
- ~PointVectorInvariantLocale ()
  Destroys this instance.

Private Types

- typedef std::locale BaseType

  The type of the super class.

12.55.2 Member Typedef Documentation

12.55.2.1 typedef std::locale OpenGPS::PointVectorInvariantLocale::BaseType [private]

The type of the super class.

12.55.3 Constructor & Destructor Documentation

12.55.3.1 PointVectorInvariantLocale::PointVectorInvariantLocale ()

Creates a new instance.

12.55.3.2 PointVectorInvariantLocale::~PointVectorInvariantLocale ()

Destroys this instance.

The documentation for this class was generated from the following files:

- point_vector_iostream.hxx
- point_vector_iostream.cxx

12.56 OpenGPS::PointVectorOutputStringStream Class Reference

```
#include <point_vector_iostream.hxx>
```

Inheritance diagram for OpenGPS::PointVectorOutputStringStream:

Collaboration diagram for OpenGPS::PointVectorOutputStringStream:

12.56.1 Detailed Description

A locale invariant string stream used to convert a point vector object to its representation as text according to the ISO5436-2 XML specification.

Public Member Functions

- PointVectorOutputStringStream ()
  Creates a new instance.
- ~PointVectorOutputStringStream ()
  Destroys this instance.

Private Types

- typedef std::basic_ostringstream< OGPS_Character > BaseType
  The type of the super class.

Private Attributes

- PointVectorInvariantLocale m_Locale

Invariant locale.

12.56.2 Member Typedef Documentation

12.56.2.1 typedef std::basic_ostringstream<OGPS_Character> OpenGPS::PointVectorOutputStringStream::BaseTy
`[private]`

The type of the super class.

12.56.3 Constructor & Destructor Documentation

12.56.3.1 PointVectorOutputStringStream::PointVectorOutputStringStream ()

Creates a new instance.

References m_Locale.

12.56.3.2 PointVectorOutputStringStream::~PointVectorOutputStringStream ()

Destroys this instance.

12.56.4 Member Data Documentation

12.56.4.1 PointVectorInvariantLocale OpenGPS::PointVectorOutputStringStream::m_Locale
`[private]`

Invariant locale.

Referenced by PointVectorOutputStringStream().

The documentation for this class was generated from the following files:

- point_vector_iostream.hxx
- point_vector_iostream.cxx

12.57 OpenGPS::PointVectorParser Class Reference

```
#include <point_vector_parser.hxx>
```

Collaboration diagram for OpenGPS::PointVectorParser:

12.57.1 Detailed Description

Generic parser of a point vector.

Reads/Writes an instance of OpenGPS::PointVectorBase from/to any file or resource described by an instance of OpenGPS::PointVectorReaderContext/ OpenGPS::PointVectorWriterContext.

Public Member Functions

- virtual DataPointParser * CreateDataPointParser (const OGPS_DataPointType dataType) const

  Creates the appropriate point parser corresponding to an axis data type.

- PointVectorParser ()

  Creates a new instance.

- virtual void Read (PointVectorReaderContext &context, PointVectorBase &value) throw (...)

  Reads point vector data from arbitrary media.

- void SetX (DataPointParser *const value)

  Sets the interface object to be used for parsing point data of the X component.

- void SetY (DataPointParser *const value)

  Sets the interface object to be used for parsing point data of the Y component.

- void SetZ (DataPointParser *const value)

  Sets the interface object to be used for parsing point data of the Z component.

- virtual void Write (PointVectorWriterContext &context, const PointVectorBase &value) throw (...)

  Writes point vector data to arbitrary media.

- virtual ~PointVectorParser ()

  Destroys this instance.

Private Attributes

- DataPointParser * m_X

  Instance of the point parser of the X component.

- DataPointParser * m_Y

  Instance of the point parser of the Y component.

- DataPointParser * m_Z

  Instance of the point parser of the Z component.

12.57.2 Constructor & Destructor Documentation

12.57.2.1 PointVectorParser::PointVectorParser ()

Creates a new instance.

References m_X, m_Y, m_Z, and NULL.

12.57.2.2 PointVectorParser::~PointVectorParser () [virtual]

Destroys this instance.

References _OPENGPS_DELETE, m_X, m_Y, and m_Z.

12.57.3 Member Function Documentation

12.57.3.1 DataPointParser * PointVectorParser::CreateDataPointParser (const OGPS_DataPointType *dataType*) const [virtual]

Creates the appropriate point parser corresponding to an axis data type.

Remarks:

The returned point parser instance must be free explicitly unless it is directly passed to one of the PointVectorParser::SetX, PointVectorParser::SetY, PointVectorParser::SetZ methods.

Parameters:

dataType The axis data type to instantiate the point parser for.

Returns:

Returns the instance of the point parser or NULL on failure.

References _ASSERT, FALSE, NULL, OGPS_DoublePointType, OGPS_FloatPointType, OGPS_Int16PointType, OGPS_Int32PointType, and OGPS_MissingPointType.

Referenced by OpenGPS::PointVectorParserBuilder::BuildX(), OpenGPS::PointVectorParserBuilder::BuildY(), and OpenGPS::PointVectorParserBuilder::BuildZ().

12.57.3.2 void PointVectorParser::Read (PointVectorReaderContext & *context*, PointVectorBase & *value*) throw (...) [virtual]

Reads point vector data from arbitrary media.

Throws an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

context Methods to read point data from the media.

value Holds the point data read assembled as a three-vector.

References _ASSERT, m_X, m_Y, m_Z, and OpenGPS::DataPointParser::Read().

Here is the call graph for this function:

12.57.3.3 void PointVectorParser::SetX (DataPointParser *const *value*)

Sets the interface object to be used for parsing point data of the X component.

Remarks:

The pointer passed will become managed. Its resources may become inaccessible at any time.

Parameters:

value Typesafe point parser of the first vector component.

See also:

PointVectorParser::CreateDataPointParser

References _ASSERT, _OPENGPS_DELETE, and m_X.

Referenced by OpenGPS::PointVectorParserBuilder::BuildX().

12.57.3.4 void PointVectorParser::SetY (DataPointParser *const *value*)

Sets the interface object to be used for parsing point data of the Y component.

Remarks:

The pointer passed will become managed. Its resources may become inaccessible at any time.

Parameters:

value Typesafe point parser of the second vector component.

See also:

PointVectorParser::CreateDataPointParser

References _ASSERT, _OPENGPS_DELETE, and m_Y.

Referenced by OpenGPS::PointVectorParserBuilder::BuildY().

12.57.3.5 void PointVectorParser::SetZ (DataPointParser *const *value*)

Sets the interface object to be used for parsing point data of the Z component.

Remarks:

The pointer passed will become managed. Its resources may become inaccessible at any time.

Parameters:

value Typesafe point parser of the third vector component.

See also:

PointVectorParser::CreateDataPointParser

References _ASSERT, _OPENGPS_DELETE, and m_Z.

Referenced by OpenGPS::PointVectorParserBuilder::BuildZ().

12.57.3.6 void PointVectorParser::Write (PointVectorWriterContext & *context*, const PointVectorBase & *value*) throw (...) [virtual]

Writes point vector data to arbitrary media.

Throws an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

context Methods to write point data to the media.

value Holds the vector to be written.

References _ASSERT, m_X, m_Y, m_Z, and OpenGPS::DataPointParser::Write().

Here is the call graph for this function:

12.57.4 Member Data Documentation

12.57.4.1 DataPointParser* OpenGPS::PointVectorParser::m_X [private]

Instance of the point parser of the X component.

Referenced by PointVectorParser(), Read(), SetX(), Write(), and ~PointVectorParser().

12.57.4.2 DataPointParser* OpenGPS::PointVectorParser::m_Y [private]

Instance of the point parser of the Y component.

Referenced by PointVectorParser(), Read(), SetY(), Write(), and ~PointVectorParser().

12.57.4.3 DataPointParser* OpenGPS::PointVectorParser::m_Z [private]

Instance of the point parser of the Z component.

Referenced by PointVectorParser(), Read(), SetZ(), Write(), and ~PointVectorParser().

The documentation for this class was generated from the following files:

- point_vector_parser.hxx
- point_vector_parser.cxx

12.58 OpenGPS::PointVectorParserBuilder Class Reference

```
#include <point_vector_parser_builder.hxx>
```

Collaboration diagram for OpenGPS::PointVectorParserBuilder:

12.58.1 Detailed Description

Component which handles the building process of a specialized parser object for reading and writing typed point data as a three-vector.

See also:

OpenGPS::PointVectorParser, OpenGPS::DataPointParser

Public Member Functions

- void BuildParser ()

  Builds the main parser object to be assembled.

- virtual void BuildX (const OGPS_DataPointType dataType)

  Builds the object used to parse the X component of the vector.

- virtual void BuildY (const OGPS_DataPointType dataType)

  Builds the object used to parse the Y component of the vector.

- virtual void BuildZ (const OGPS_DataPointType dataType)

  Builds the object used to parse the Z component of the vector.

- virtual PointVectorParser * GetParser ()

  Gets the assembled point vector parser.

- PointVectorParserBuilder ()

  Creates a new instance.

- virtual ~PointVectorParserBuilder ()

  Destroys this instance.

Private Attributes

- PointVectorParser * m_Parser

  The object to be built.

12.58.2 Constructor & Destructor Documentation

12.58.2.1 PointVectorParserBuilder::PointVectorParserBuilder ()

Creates a new instance.

References m_Parser, and NULL.

12.58.2.2 PointVectorParserBuilder::~PointVectorParserBuilder () [virtual]

Destroys this instance.

References _OPENGPS_DELETE, and m_Parser.

12.58.3 Member Function Documentation

12.58.3.1 void PointVectorParserBuilder::BuildParser ()

Builds the main parser object to be assembled.

Remarks:

This must be called firstly when starting the build process.

References _ASSERT, and m_Parser.

Referenced by OpenGPS::ISO5436_2Container::BuildPointVectorParser().

12.58.3.2 void PointVectorParserBuilder::BuildX (const OGPS_DataPointType *dataType*) [virtual]

Builds the object used to parse the X component of the vector.

Parameters:

dataType The data type of the X axis.

See also:

PointVectorParserBuilder::BuildParser

References _ASSERT, OpenGPS::PointVectorParser::CreateDataPointParser(), m_Parser, and OpenGPS::PointVectorParser::SetX().

Referenced by OpenGPS::ISO5436_2Container::BuildPointVectorParser().

Here is the call graph for this function:

12.58.3.3 void PointVectorParserBuilder::BuildY (const OGPS_DataPointType *dataType*) [virtual]

Builds the object used to parse the Y component of the vector.

Parameters:

dataType The data type of the Y axis.

See also:

PointVectorParserBuilder::BuildParser

References _ASSERT, OpenGPS::PointVectorParser::CreateDataPointParser(), m_Parser, and OpenGPS::PointVectorParser::SetY().

Referenced by OpenGPS::ISO5436_2Container::BuildPointVectorParser().

Here is the call graph for this function:

12.58.3.4 void PointVectorParserBuilder::BuildZ (const OGPS_DataPointType *dataType*) [virtual]

Builds the object used to parse the Z component of the vector.

Parameters:

dataType The data type of the Z axis.

See also:

PointVectorParserBuilder::BuildParser

References _ASSERT, OpenGPS::PointVectorParser::CreateDataPointParser(), m_Parser, and OpenGPS::PointVectorParser::SetZ().

Referenced by OpenGPS::ISO5436_2Container::BuildPointVectorParser().

Here is the call graph for this function:

12.58.3.5 PointVectorParser * PointVectorParserBuilder::GetParser () [virtual]

Gets the assembled point vector parser.

References m_Parser.

12.58.4 Member Data Documentation

12.58.4.1 PointVectorParser* OpenGPS::PointVectorParserBuilder::m_Parser [private]

The object to be built.

Referenced by BuildParser(), BuildX(), BuildY(), BuildZ(), GetParser(), PointVectorParserBuilder(), and ~PointVectorParserBuilder().

The documentation for this class was generated from the following files:

- point_vector_parser_builder.hxx
- point_vector_parser_builder.cxx

12.59 OpenGPS::PointVectorProxy Class Reference

```
#include <point_vector_proxy.hxx>
```

Inheritance diagram for OpenGPS::PointVectorProxy:

Collaboration diagram for OpenGPS::PointVectorProxy:

12.59.1 Detailed Description

Serves one row of the three distinct buffers of data points managed by OpenGPS::VectorBuffer and raises the expression of accessing a single point vector.

Public Member Functions

- virtual void Get (PointVectorBase &value) const throw (...)

  Copies values to another OpenGPS::PointVectorBase instance.

- virtual DataPoint * GetX ()

  Gets typesafe direct access to the x component.

- virtual const DataPoint * GetX () const

  Gets typesafe direct read-only access to the x component.

- virtual DataPoint * GetY ()

  Gets typesafe direct access to the y component.

- virtual const DataPoint * GetY () const

  Gets typesafe direct read-only access to the y component.

- virtual DataPoint * GetZ ()

  Gets typesafe direct access to the z component.

- virtual const DataPoint * GetZ () const

  Gets typesafe direct read-only access to the z component.

- PointVectorProxy (const PointVectorProxyContext *const context, VectorBuffer *const buffer)

  Creates a new instance.

- virtual void Set (const PointVectorBase &value) throw (...)

  Copies values from another OpenGPS::PointVector instance.

- virtual ~PointVectorProxy ()

  Destroys this instance.

Private Member Functions

- PointVectorProxy & operator= (const PointVectorProxy &src)

  The assignment-operator is not implemented.

- PointVectorProxy (const PointVectorProxy &src)

  The copy-ctor is not implemented.

Private Attributes

- VectorBuffer * m_Buffer

  Provides access to the point data of all three axes.

- const PointVectorProxyContext *const m_Context

  Provides information which data row is currently proxied.

- DataPoint * m_X

  Provides typesafe access to the point data of the current X component.

- DataPoint * m_Y

  Provides typesafe access to the point data of the current Y component.

- DataPoint * m_Z

  Provides typesafe access to the point data of the current Z component.

Classes

- class DataPointProxy

  Proxies one single data point pointed to by the current context information.

12.59.2 Constructor & Destructor Documentation

12.59.2.1 PointVectorProxy::PointVectorProxy (const PointVectorProxyContext *const *context*, VectorBuffer *const *buffer*)

Creates a new instance.

Parameters:

context Provides information which data row is currently proxied. This is read-only within the proxy object but can be changed from the outside.

buffer Manages the whole set of all point vectors stacked together.

References _ASSERT, OpenGPS::VectorBuffer::GetX(), OpenGPS::VectorBuffer::GetY(), OpenGPS::VectorBuffer::GetZ(), m_X, m_Y, and m_Z.

Here is the call graph for this function:

12.59.2.2 PointVectorProxy::~PointVectorProxy () [virtual]

Destroys this instance.

References _OPENGPS_DELETE, m_X, m_Y, and m_Z.

12.59.2.3 OpenGPS::PointVectorProxy::PointVectorProxy (const PointVectorProxy & *src*) [private]

The copy-ctor is not implemented.

This prevents its usage.

12.59.3 Member Function Documentation

12.59.3.1 void PointVectorProxy::Get (PointVectorBase & *value*) const throw (...) [virtual]

Copies values to another OpenGPS::PointVectorBase instance.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Retrieves values from the current instance as a copy.

Implements OpenGPS::PointVectorBase.

References _ASSERT, m_X, m_Y, m_Z, and OpenGPS::DataPoint::Set().

Here is the call graph for this function:

12.59.3.2 DataPoint * PointVectorProxy::GetX () [virtual]

Gets typesafe direct access to the x component.

Implements OpenGPS::PointVectorBase.

References _ASSERT, and m_X.

12.59.3.3 const DataPoint * PointVectorProxy::GetX () const [virtual]

Gets typesafe direct read-only access to the x component.

Implements OpenGPS::PointVectorBase.

References _ASSERT, and m_X.

12.59.3.4 DataPoint * PointVectorProxy::GetY () [virtual]

Gets typesafe direct access to the y component.

Implements OpenGPS::PointVectorBase.

References _ASSERT, and m_Y.

12.59.3.5 const DataPoint * PointVectorProxy::GetY () const [virtual]

Gets typesafe direct read-only access to the y component.

Implements OpenGPS::PointVectorBase.

References _ASSERT, and m_Y.

12.59.3.6 DataPoint * PointVectorProxy::GetZ () [virtual]

Gets typesafe direct access to the z component.

Implements OpenGPS::PointVectorBase.

References _ASSERT, and m_Z.

12.59.3.7 const DataPoint * PointVectorProxy::GetZ () const [virtual]

Gets typesafe direct read-only access to the z component.

Implements OpenGPS::PointVectorBase.

References _ASSERT, and m_Z.

12.59.3.8 PointVectorProxy& OpenGPS::PointVectorProxy::operator= (const PointVectorProxy & *src*) [private]

The assignment-operator is not implemented.

This prevents its usage.

12.59.3.9 void PointVectorProxy::Set (const PointVectorBase & *value*) throw (...) [virtual]

Copies values from another OpenGPS::PointVector instance.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value OpenGPS::PointVectorBase object to copy from.

Implements OpenGPS::PointVectorBase.

References _ASSERT, OpenGPS::DataPoint::IsValid(), m_X, m_Y, m_Z, and OpenGPS::DataPoint::Set().

Here is the call graph for this function:

12.59.4 Member Data Documentation

12.59.4.1 VectorBuffer* OpenGPS::PointVectorProxy::m_Buffer [private]

Provides access to the point data of all three axes.

12.59.4.2 const PointVectorProxyContext* const OpenGPS::PointVectorProxy::m_Context [private]

Provides information which data row is currently proxied.

12.59.4.3 DataPoint* OpenGPS::PointVectorProxy::m_X [private]

Provides typesafe access to the point data of the current X component.

Referenced by Get(), GetX(), PointVectorProxy(), Set(), and ~PointVectorProxy().

12.59.4.4 DataPoint* OpenGPS::PointVectorProxy::m_Y [private]

Provides typesafe access to the point data of the current Y component.

Referenced by Get(), GetY(), PointVectorProxy(), Set(), and ~PointVectorProxy().

12.59.4.5 DataPoint* OpenGPS::PointVectorProxy::m_Z [private]

Provides typesafe access to the point data of the current Z component.

Referenced by Get(), GetZ(), PointVectorProxy(), Set(), and ~PointVectorProxy().

The documentation for this class was generated from the following files:

- point_vector_proxy.hxx
- point_vector_proxy.cxx

12.60 OpenGPS::PointVectorProxy::DataPointProxy Class Reference

Inheritance diagram for OpenGPS::PointVectorProxy::DataPointProxy:

Collaboration diagram for OpenGPS::PointVectorProxy::DataPointProxy:

12.60.1 Detailed Description

Proxies one single data point pointed to by the current context information.

Public Member Functions

- DataPointProxy (const PointVectorProxyContext ∗const context, PointBuffer ∗const buffer)

  Creates a new instance.

- virtual OGPS_Double Get () const throw (...)

  Gets the stored value.

- virtual void Get (OGPS_Double ∗const value) const throw (...)

  Gets the stored value.

- virtual void Get (OGPS_Float ∗const value) const throw (...)

  Gets the stored value.

- virtual void Get (OGPS_Int32 ∗const value) const throw (...)

  Gets the stored value.

- virtual void Get (OGPS_Int16 ∗const value) const throw (...)

  Gets the stored value.

- virtual OGPS_DataPointType GetPointType () const throw (...)

  Gets type information of the current value stored within this data point.

- virtual OGPS_Boolean IsValid () const throw (...)

  Asks whether there is a point value stored within this instance currently.

- virtual void Reset () throw (...)

  Resets this instance to its initial state.

- virtual void Set (const DataPoint &src) throw (...)

  Stores a new value.

- virtual void Set (const OGPS_Double value) throw (...)

  Stores a new value.

- virtual void Set (const OGPS_Float value) throw (...)

  Stores a new value.

- virtual void Set (const OGPS_Int32 value) throw (...)

  Stores a new value.

- virtual void Set (const OGPS_Int16 value) throw (...)

  Stores a new value.

- virtual ~DataPointProxy ()

  Destroys this instance.

Private Member Functions

- DataPointProxy (const DataPointProxy &src)

  The copy-ctor is not implemented.

- DataPointProxy & operator= (const DataPointProxy &src)

  The assignment-operator is not implemented.

Private Attributes

- PointBuffer * m_Buffer

  Provides access to the point data.

- const PointVectorProxyContext *const m_Context

  Provides information which data row is currently proxied.

12.60.2 Constructor & Destructor Documentation

12.60.2.1 PointVectorProxy::DataPointProxy::DataPointProxy (const PointVectorProxyContext *const *context*, PointBuffer *const *buffer*)

Creates a new instance.

Parameters:

context Provides the information which row of point data is currently proxied.

buffer The buffer of all point data stacked together.

References _ASSERT.

12.60.2.2 PointVectorProxy::DataPointProxy::~DataPointProxy () [virtual]

Destroys this instance.

12.60.2.3 OpenGPS::PointVectorProxy::DataPointProxy::DataPointProxy (const DataPointProxy & *src*) [private]

The copy-ctor is not implemented.

This prevents its usage.

12.60.3 Member Function Documentation

12.60.3.1 OGPS_Double PointVectorProxy::DataPointProxy::Get () const throw (...) [virtual]

Gets the stored value.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns the stored value or 0.0 if this data point is empty (DataPoint::GetPointType returns OGPS_-MissingPointType in this case).

Implements OpenGPS::DataPoint.

References _EX_T, and OGPS_ExNotImplemented.

12.60.3.2 void PointVectorProxy::DataPointProxy::Get (OGPS_Double *const *value*) const throw (...) [virtual]

Gets the stored value.

Remarks:

If the current type does not equal OGPS_Double, the behavior is undefined.

See also:

DataPoint::GetPointType

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Destination for the currently stored value.

Implements OpenGPS::DataPoint.

References _ASSERT, _CHECK_BUFFER_AND_THROW_READ_EXCEPTION, OpenGPS::PointBuffer::Get(), OpenGPS::PointVectorProxyContext::GetIndex(), m_Buffer, and m_-Context.

Here is the call graph for this function:

12.60.3.3 void PointVectorProxy::DataPointProxy::Get (OGPS_Float *const *value*) const throw (...) [virtual]

Gets the stored value.

Remarks:

If the current type does not equal OGPS_Float, the behavior is undefined.

See also:

DataPoint::GetPointType

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Destination for the currently stored value.

Implements OpenGPS::DataPoint.

References _ASSERT, _CHECK_BUFFER_AND_THROW_READ_EXCEPTION, OpenGPS::PointBuffer::Get(), OpenGPS::PointVectorProxyContext::GetIndex(), m_Buffer, and m_-Context.

Here is the call graph for this function:

12.60.3.4 void PointVectorProxy::DataPointProxy::Get (OGPS_Int32 *const *value*) const throw (...) [virtual]

Gets the stored value.

Remarks:

If the current type does not equal OGPS_Int32, the behavior is undefined.

See also:

DataPoint::GetPointType

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Destination for the currently stored value.

Implements OpenGPS::DataPoint.

References _ASSERT, _CHECK_BUFFER_AND_THROW_READ_EXCEPTION, OpenGPS::PointBuffer::Get(), OpenGPS::PointVectorProxyContext::GetIndex(), m_Buffer, and m_-Context.

Here is the call graph for this function:

12.60.3.5 void PointVectorProxy::DataPointProxy::Get (OGPS_Int16 *const *value*) const throw (...) [virtual]

Gets the stored value.

Remarks:

If the current type does not equal OGPS_Int16, the behavior is undefined.

See also:

DataPoint::GetPointType

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Destination for the currently stored value.

Implements OpenGPS::DataPoint.

References _ASSERT, _CHECK_BUFFER_AND_THROW_READ_EXCEPTION, OpenGPS::PointBuffer::Get(), OpenGPS::PointVectorProxyContext::GetIndex(), m_Buffer, and m_-Context.

Here is the call graph for this function:

12.60.3.6 OGPS_DataPointType PointVectorProxy::DataPointProxy::GetPointType () const throw (...) [virtual]

Gets type information of the current value stored within this data point.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns OGPS_MissingPointType if this instance does not store any value at all.

Implements OpenGPS::DataPoint.

References OpenGPS::PointBuffer::GetPointType(), m_Buffer, and OGPS_MissingPointType.

Referenced by Set().

Here is the call graph for this function:

12.60.3.7 OGPS_Boolean PointVectorProxy::DataPointProxy::IsValid () const throw (...) [virtual]

Asks whether there is a point value stored within this instance currently.

See also:

DataPoint::GetPointType

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns TRUE if a value is stored, FALSE otherwise.

Implements OpenGPS::DataPoint.

References m_Buffer, and NULL.

12.60.3.8 DataPointProxy& OpenGPS::PointVectorProxy::DataPointProxy::operator= (const DataPointProxy & *src*) [private]

The assignment-operator is not implemented.

This prevents its usage.

12.60.3.9 void PointVectorProxy::DataPointProxy::Reset () throw (...) [virtual]

Resets this instance to its initial state.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Implements OpenGPS::DataPoint.

References _EX_T, m_Buffer, and OGPS_ExInvalidOperation.

12.60.3.10 void PointVectorProxy::DataPointProxy::Set (const DataPoint & *src*) throw (...) [virtual]

Stores a new value.

Copies its new entry from another DataPoint instance.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

src Object to copy from.

Implements OpenGPS::DataPoint.

References _ASSERT, _EX_T, GetPointType(), m_Context, OGPS_DoublePointType, OGPS_-ExInvalidArgument, OGPS_FloatPointType, OGPS_Int16PointType, OGPS_Int32PointType, OGPS_-MissingPointType, and Set().

Here is the call graph for this function:

12.60.3.11 void PointVectorProxy::DataPointProxy::Set (const OGPS_Double *value*) throw (...) [virtual]

Stores a new value.

Also adjusts the current type information reflecting this set operation, if necessary.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value The new value to be stored.

Implements OpenGPS::DataPoint.

References _ASSERT, _CHECK_BUFFER_AND_THROW_WRITE_EXCEPTION, OpenGPS::PointVectorProxyContext::GetIndex(), m_Buffer, m_Context, and OpenGPS::PointBuffer::Set().

Here is the call graph for this function:

12.60.3.12 void PointVectorProxy::DataPointProxy::Set (const OGPS_Float *value*) throw (...) [virtual]

Stores a new value.

Also adjusts the current type information reflecting this set operation, if necessary.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value The new value to be stored.

Implements OpenGPS::DataPoint.

References _ASSERT, _CHECK_BUFFER_AND_THROW_WRITE_EXCEPTION, OpenGPS::PointVectorProxyContext::GetIndex(), m_Buffer, m_Context, and OpenGPS::PointBuffer::Set().

Here is the call graph for this function:

12.60.3.13 void PointVectorProxy::DataPointProxy::Set (const OGPS_Int32 *value*) throw (...) [virtual]

Stores a new value.

Also adjusts the current type information reflecting this set operation, if necessary.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value The new value to be stored.

Implements OpenGPS::DataPoint.

References _ASSERT, _CHECK_BUFFER_AND_THROW_WRITE_EXCEPTION, OpenGPS::PointVectorProxyContext::GetIndex(), m_Buffer, m_Context, and OpenGPS::PointBuffer::Set().

Here is the call graph for this function:

12.60.3.14 void PointVectorProxy::DataPointProxy::Set (const OGPS_Int16 *value*) throw (...) [virtual]

Stores a new value.

Also adjusts the current type information reflecting this set operation, if necessary.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value The new value to be stored.

Implements OpenGPS::DataPoint.

References _ASSERT, _CHECK_BUFFER_AND_THROW_WRITE_EXCEPTION, OpenGPS::PointVectorProxyContext::GetIndex(), m_Buffer, m_Context, and OpenGPS::PointBuffer::Set().

Referenced by Set().

Here is the call graph for this function:

12.60.4 Member Data Documentation

12.60.4.1 PointBuffer* OpenGPS::PointVectorProxy::DataPointProxy::m_Buffer [private]

Provides access to the point data.

Referenced by Get(), GetPointType(), IsValid(), Reset(), and Set().

12.60.4.2 const PointVectorProxyContext* const OpenGPS::PointVectorProxy::DataPointProxy::m_Context [private]

Provides information which data row is currently proxied.

Referenced by Get(), and Set().

The documentation for this class was generated from the following files:

- point_vector_proxy.hxx
- data_point_proxy.cxx

12.61 OpenGPS::PointVectorProxyContext Class Reference

`#include <point_vector_proxy_context.hxx>`

Inheritance diagram for OpenGPS::PointVectorProxyContext:

12.61.1 Detailed Description

Indexing of point data managed by OpenGPS::VectorBuffer.

Public Member Functions

- virtual OGPS_Boolean CanIncrementIndex () const
  Can the current index be incremented?
- virtual OGPS_ULong GetIndex () const
  Gets the current index.
- virtual OGPS_Boolean IncrementIndex ()
  Increments the current index by one.
- virtual OGPS_Boolean IsMatrix () const =0
  Asks whether this context is for points managed in matrices or lists.
- virtual ~PointVectorProxyContext ()
  Destroys this instance.

Protected Member Functions

- PointVectorProxyContext ()
  Creates a new instance.

12.61.2 Constructor & Destructor Documentation

12.61.2.1 PointVectorProxyContext::~PointVectorProxyContext () [virtual]

Destroys this instance.

12.61.2.2 PointVectorProxyContext::PointVectorProxyContext () [protected]

Creates a new instance.

12.61.3 Member Function Documentation

12.61.3.1 OGPS_Boolean PointVectorProxyContext::CanIncrementIndex () const [virtual]

Can the current index be incremented?

Reimplemented in OpenGPS::PointVectorProxyContextList, and OpenGPS::PointVectorProxyContextMatrix.

References FALSE.

12.61.3.2 OGPS_ULong PointVectorProxyContext::GetIndex () const [virtual]

Gets the current index.

Reimplemented in OpenGPS::PointVectorProxyContextList, and OpenGPS::PointVectorProxyContextMatrix.

Referenced by OpenGPS::PointVectorProxy::DataPointProxy::Get(), and OpenGPS::PointVectorProxy::DataPointProxy::Set().

12.61.3.3 OGPS_Boolean PointVectorProxyContext::IncrementIndex () [virtual]

Increments the current index by one.

Reimplemented in OpenGPS::PointVectorProxyContextList, and OpenGPS::PointVectorProxyContextMatrix.

References FALSE.

12.61.3.4 virtual OGPS_Boolean OpenGPS::PointVectorProxyContext::IsMatrix () const [pure virtual]

Asks whether this context is for points managed in matrices or lists.

Implemented in OpenGPS::PointVectorProxyContextList, and OpenGPS::PointVectorProxyContextMatrix.

The documentation for this class was generated from the following files:

- point_vector_proxy_context.hxx
- point_vector_proxy_context.cxx

12.62 OpenGPS::PointVectorProxyContextList Class Reference

```
#include <point_vector_proxy_context_list.hxx>
```

Inheritance diagram for OpenGPS::PointVectorProxyContextList:

Collaboration diagram for OpenGPS::PointVectorProxyContextList:

12.62.1 Detailed Description

Indexing of point data managed by OpenGPS::VectorBuffer.

The indexes are calculated for point measurements stored in a simple list structure.

Public Member Functions

- virtual OGPS_Boolean CanIncrementIndex () const

  Can the current index be incremented?

- virtual OGPS_ULong GetIndex () const

  Gets the current index.

- virtual OGPS_Boolean IncrementIndex ()

  Increments the current index by one.

- virtual OGPS_Boolean IsMatrix () const

  Asks whether this context is for points managed in matrices or lists.

- PointVectorProxyContextList (const OGPS_ULong maxIndex)

  Creates a new instance.

- virtual void SetIndex (const OGPS_ULong index) throw (...)

  Sets the current index.

- virtual ~PointVectorProxyContextList ()

  Destroys this instance.

Private Attributes

- OGPS_ULong m_Index

  The current index.

- OGPS_ULong m_MaxIndex

  The maximum index possible.

12.62.2 Constructor & Destructor Documentation

12.62.2.1 PointVectorProxyContextList::PointVectorProxyContextList (const OGPS_ULong *maxIndex*)

Creates a new instance.

Parameters:

maxIndex The maximum amount of indexable measurement data. In other words: the number of elements contained in the list structure.

References m_Index, and m_MaxIndex.

12.62.2.2 PointVectorProxyContextList::~PointVectorProxyContextList () [virtual]

Destroys this instance.

12.62.3 Member Function Documentation

12.62.3.1 OGPS_Boolean PointVectorProxyContextList::CanIncrementIndex () const [virtual]

Can the current index be incremented?

Reimplemented from OpenGPS::PointVectorProxyContext.

References m_Index, and m_MaxIndex.

Referenced by IncrementIndex().

12.62.3.2 OGPS_ULong PointVectorProxyContextList::GetIndex () const [virtual]

Gets the current index.

Reimplemented from OpenGPS::PointVectorProxyContext.

References m_Index.

12.62.3.3 OGPS_Boolean PointVectorProxyContextList::IncrementIndex () [virtual]

Increments the current index by one.

Reimplemented from OpenGPS::PointVectorProxyContext.

References CanIncrementIndex(), FALSE, m_Index, and TRUE.

Here is the call graph for this function:

12.62.3.4 OGPS_Boolean PointVectorProxyContextList::IsMatrix () const [virtual]

Asks whether this context is for points managed in matrices or lists.

Implements OpenGPS::PointVectorProxyContext.

References FALSE.

12.62.3.5 void PointVectorProxyContextList::SetIndex (const OGPS_ULong *index*) throw (...) [virtual]

Sets the current index.

Parameters:

index The new arbitrary index.

Returns:

Returns TRUE on success, FALSE otherwise.

References _EX_T, m_Index, m_MaxIndex, and OGPS_ExInvalidOperation.

12.62.4 Member Data Documentation

12.62.4.1 OGPS_ULong OpenGPS::PointVectorProxyContextList::m_Index [private]

The current index.

Referenced by CanIncrementIndex(), GetIndex(), IncrementIndex(), PointVectorProxyContextList(), and SetIndex().

12.62.4.2 OGPS_ULong OpenGPS::PointVectorProxyContextList::m_MaxIndex [private]

The maximum index possible.

Referenced by CanIncrementIndex(), PointVectorProxyContextList(), and SetIndex().

The documentation for this class was generated from the following files:

- point_vector_proxy_context_list.hxx
- point_vector_proxy_context_list.cxx

12.63 OpenGPS::PointVectorProxyContextMatrix Class Reference

```
#include <point_vector_proxy_context_matrix.hxx>
```

Inheritance diagram for OpenGPS::PointVectorProxyContextMatrix:

Collaboration diagram for OpenGPS::PointVectorProxyContextMatrix:

12.63.1 Detailed Description

Indexing of point data managed by OpenGPS::VectorBuffer.

The indexes are calculated for point measurements stored in a matrix structure that saves extra topology information.

Public Member Functions

- virtual OGPS_Boolean CanIncrementIndex () const
  Can the current index be incremented?
- virtual OGPS_ULong GetIndex () const
  Gets the current index.
- virtual OGPS_Boolean IncrementIndex ()
  Increments the current index by one.
- virtual OGPS_Boolean IsMatrix () const
  Asks whether this context is for points managed in matrices or lists.
- PointVectorProxyContextMatrix (const OGPS_ULong maxU, const OGPS_ULong maxV, const OGPS_ULong maxW)
  Creates a new instance.
- void SetIndex (const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w) throw (...)
  Sets the current index.
- virtual ~PointVectorProxyContextMatrix ()
  Destroys this instance.

Private Attributes

- OGPS_ULong m_MaxU
  The maximum index possible in X direction.
- OGPS_ULong m_MaxV
  The maximum index possible in Y direction.
- OGPS_ULong m_MaxW
  The maximum index possible in Z direction.
- OGPS_ULong m_U
  The current index in X direction of the topology mapping.
- OGPS_ULong m_V
  The current index in Y direction of the topology mapping.
- OGPS_ULong m_W
  The current index in Z direction of the topology mapping.

12.63.2 Constructor & Destructor Documentation

12.63.2.1 PointVectorProxyContextMatrix::PointVectorProxyContextMatrix (const OGPS_ULong *maxU*, const OGPS_ULong *maxV*, const OGPS_ULong *maxW*)

Creates a new instance.

Parameters:

maxU The maximum index possible in X direction of the topology mapping.

maxV The maximum index possible in Y direction of the topology mapping.

maxW The maximum index possible in Z direction of the topology mapping.

References m_MaxU, m_MaxV, m_MaxW, m_U, m_V, and m_W.

12.63.2.2 PointVectorProxyContextMatrix::~PointVectorProxyContextMatrix () [virtual]

Destroys this instance.

12.63.3 Member Function Documentation

12.63.3.1 OGPS_Boolean PointVectorProxyContextMatrix::CanIncrementIndex () const [virtual]

Can the current index be incremented?

Reimplemented from OpenGPS::PointVectorProxyContext.

References m_MaxU, m_MaxV, m_MaxW, m_U, m_V, and m_W.

Referenced by IncrementIndex().

12.63.3.2 OGPS_ULong PointVectorProxyContextMatrix::GetIndex () const [virtual]

Gets the current index.

Reimplemented from OpenGPS::PointVectorProxyContext.

References m_MaxU, m_MaxW, m_U, m_V, and m_W.

12.63.3.3 OGPS_Boolean PointVectorProxyContextMatrix::IncrementIndex () [virtual]

Increments the current index by one.

Reimplemented from OpenGPS::PointVectorProxyContext.

References CanIncrementIndex(), FALSE, m_MaxU, m_MaxV, m_MaxW, m_U, m_V, m_W, and TRUE.

Here is the call graph for this function:

12.63.3.4 OGPS_Boolean PointVectorProxyContextMatrix::IsMatrix () const [virtual]

Asks whether this context is for points managed in matrices or lists.

Implements OpenGPS::PointVectorProxyContext.

References TRUE.

12.63.3.5 void PointVectorProxyContextMatrix::SetIndex (const OGPS_ULong *u*, const OGPS_-ULong *v*, const OGPS_ULong *w*) throw (...)

Sets the current index.

Parameters:

- ***u*** The new arbitrary index in X direction of the topology mapping.
- ***v*** The new arbitrary index in Y direction of the topology mapping.
- ***w*** The new arbitrary index in Z direction of the topology mapping.

Returns:

Returns TRUE on success, FALSE otherwise.

References _EX_T, m_MaxU, m_MaxV, m_MaxW, m_U, m_V, m_W, and OGPS_ExInvalidOperation.

12.63.4 Member Data Documentation

12.63.4.1 OGPS_ULong OpenGPS::PointVectorProxyContextMatrix::m_MaxU [private]

The maximum index possible in X direction.

Referenced by CanIncrementIndex(), GetIndex(), IncrementIndex(), PointVectorProxyContextMatrix(), and SetIndex().

12.63.4.2 OGPS_ULong OpenGPS::PointVectorProxyContextMatrix::m_MaxV [private]

The maximum index possible in Y direction.

Referenced by CanIncrementIndex(), IncrementIndex(), PointVectorProxyContextMatrix(), and SetIndex().

12.63.4.3 OGPS_ULong OpenGPS::PointVectorProxyContextMatrix::m_MaxW [private]

The maximum index possible in Z direction.

Referenced by CanIncrementIndex(), GetIndex(), IncrementIndex(), PointVectorProxyContextMatrix(), and SetIndex().

12.63.4.4 OGPS_ULong OpenGPS::PointVectorProxyContextMatrix::m_U [private]

The current index in X direction of the topology mapping.

Referenced by CanIncrementIndex(), GetIndex(), IncrementIndex(), PointVectorProxyContextMatrix(), and SetIndex().

12.63.4.5 OGPS_ULong OpenGPS::PointVectorProxyContextMatrix::m_V [private]

The current index in Y direction of the topology mapping.

Referenced by CanIncrementIndex(), GetIndex(), IncrementIndex(), PointVectorProxyContextMatrix(), and SetIndex().

12.63.4.6 OGPS_ULong OpenGPS::PointVectorProxyContextMatrix::m_W [private]

The current index in Z direction of the topology mapping.

Referenced by CanIncrementIndex(), GetIndex(), IncrementIndex(), PointVectorProxyContextMatrix(), and SetIndex().

The documentation for this class was generated from the following files:

- point_vector_proxy_context_matrix.hxx
- point_vector_proxy_context_matrix.cxx

12.64 OpenGPS::PointVectorReaderContext Class Reference

`#include <point_vector_reader_context.hxx>`

Inheritance diagram for OpenGPS::PointVectorReaderContext:

12.64.1 Detailed Description

Encapsulates an underlying stream of point vector data.

Provides methods for reading point data from that stream. The exact data type of point data stored at the current position of the imagined stream pointer must be known a-priori though. This meta information must be extracted from elsewhere before and is not accessible from here. Three values of point data make up one point vector. To read one point vector use PointVectorReaderContext::MoveNext to start reading. Check if the current point vector is valid. Then make three subsequent calls to the PointVectorReaderContext::Read/ PointVectorReaderContext::Skip methods (you must know the data type of all values in the stream to be able to do that).

Public Member Functions

- virtual OGPS_Boolean IsValid () const =0 throw (...)

  Asks if there is readable point vector stored at the current position.

- virtual OGPS_Boolean MoveNext ()=0 throw (...)

  Move the current reading position of the stream to the next three-vector.

- virtual void Read (OGPS_Double *const value)=0 throw (...)

  Reads the currently underlying data as OGPS_Double.

- virtual void Read (OGPS_Float *const value)=0 throw (...)

  Reads the currently underlying data as OGPS_Float.

- virtual void Read (OGPS_Int32 *const value)=0 throw (...)

  Reads the currently underlying data as OGPS_Int32.

- virtual void Read (OGPS_Int16 *const value)=0 throw (...)

  Reads the currently underlying data as OGPS_Int16.

- virtual void Skip ()=0 throw (...)

  Skips reading of the currently underlying data.

- virtual ~PointVectorReaderContext ()

  Destroys this instance.

Protected Member Functions

- PointVectorReaderContext ()

  Creates a new instance.

12.64.2 Constructor & Destructor Documentation

12.64.2.1 PointVectorReaderContext::~PointVectorReaderContext () [virtual]

Destroys this instance.

12.64.2.2 PointVectorReaderContext::PointVectorReaderContext () [protected]

Creates a new instance.

12.64.3 Member Function Documentation

12.64.3.1 virtual OGPS_Boolean OpenGPS::PointVectorReaderContext::IsValid () const throw (...) [pure virtual]

Asks if there is readable point vector stored at the current position.

A point vector may be invalid if it had not been measured, but is needed for topological consistency though.

Remarks:

This must be checked before one attempts to PointVectorReaderContext::Read/ PointVectorReaderContext::Skip any of the point data of one of the three coordinates.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns TRUE if the current point is readable, FALSE otherwise.

Implemented in OpenGPS::BinaryPointVectorReaderContext, and OpenGPS::XmlPointVectorReaderContext.

12.64.3.2 virtual OGPS_Boolean OpenGPS::PointVectorReaderContext::MoveNext () throw (...) [pure virtual]

Move the current reading position of the stream to the next three-vector.

This is possible only if all three coordinates of the current point vector data had been fully read (see PointVectorReaderContext::Read and PointVectorReaderContext::Skip member functions). If PointVectorReaderContext::IsValid returns FALSE, call this method directly to move to the next point vector in storage.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns TRUE when there is more data to be parsed, FALSE otherwise.

Implemented in OpenGPS::BinaryPointVectorReaderContext, and OpenGPS::XmlPointVectorReaderContext.

12.64.3.3 virtual void OpenGPS::PointVectorReaderContext::Read (OGPS_Double *const *value*) throw (...) [pure virtual]

Reads the currently underlying data as OGPS_Double.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implemented in OpenGPS::BinaryLSBPointVectorReaderContext, OpenGPS::BinaryMSBPointVectorReaderContext, and OpenGPS::XmlPointVectorReaderContext.

12.64.3.4 virtual void OpenGPS::PointVectorReaderContext::Read (OGPS_Float *const *value*) throw (...) [pure virtual]

Reads the currently underlying data as OGPS_Float.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implemented in OpenGPS::BinaryLSBPointVectorReaderContext, OpenGPS::BinaryMSBPointVectorReaderContext, and OpenGPS::XmlPointVectorReaderContext.

12.64.3.5 virtual void OpenGPS::PointVectorReaderContext::Read (OGPS_Int32 *const *value*) throw (...) [pure virtual]

Reads the currently underlying data as OGPS_Int32.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implemented in OpenGPS::BinaryLSBPointVectorReaderContext, OpenGPS::BinaryMSBPointVectorReaderContext, and OpenGPS::XmlPointVectorReaderContext.

12.64.3.6 virtual void OpenGPS::PointVectorReaderContext::Read (OGPS_Int16 *const *value*) throw (...) [pure virtual]

Reads the currently underlying data as OGPS_Int16.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implemented in OpenGPS::BinaryLSBPointVectorReaderContext, OpenGPS::BinaryMSBPointVectorReaderContext, and OpenGPS::XmlPointVectorReaderContext.

12.64.3.7 virtual void OpenGPS::PointVectorReaderContext::Skip () throw (...) [pure virtual]

Skips reading of the currently underlying data.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

This must be called for integrity if it is expected from the reading process that there is no point data available for reading, i.e. for the current coordinate no point data was saved because the corresponding axis is of incremental type.

Implemented in OpenGPS::BinaryPointVectorReaderContext, and OpenGPS::XmlPointVectorReaderContext.

The documentation for this class was generated from the following files:

- point_vector_reader_context.hxx
- xml_point_vector_reader_context.cxx

12.65 OpenGPS::PointVectorWhitespaceFacet Class Reference

```
#include <point_vector_iostream.hxx>
```

12.65.1 Detailed Description

Redefines the white space character set to make parsing of a point vector easier.

In an ISO5436-2 XML document the three components of a point vector are separated by a semicolon. This facet classifies the semicolon as white space, too. This way the commonly known standard io operator can be used as usual to read the three separate components of the point vector from the string stream.

Public Member Functions

- PointVectorWhitespaceFacet (const size_t refs=0)
  Creates a new instance.
- ~PointVectorWhitespaceFacet ()
  Destroys this instance.

Protected Member Functions

- virtual bool do_is (mask msk, OGPS_Character ch) const
  Classifies a character.

Private Types

- typedef std::ctype< OGPS_Character > BaseType
  The type of the super class.

12.65.2 Member Typedef Documentation

12.65.2.1 typedef std::ctype<OGPS_Character> OpenGPS::PointVectorWhitespaceFacet::BaseType [private]

The type of the super class.

12.65.3 Constructor & Destructor Documentation

12.65.3.1 PointVectorWhitespaceFacet::PointVectorWhitespaceFacet (const size_t *refs* = 0)

Creates a new instance.

12.65.3.2 PointVectorWhitespaceFacet::~PointVectorWhitespaceFacet ()

Destroys this instance.

12.65.4 Member Function Documentation

12.65.4.1 bool PointVectorWhitespaceFacet::do_is (mask *msk*, OGPS_Character *ch*) const [protected, virtual]

Classifies a character.

Parameters:

msk Check whether the character belongs to the sepcified group.

ch The character to be classified.

The documentation for this class was generated from the following files:

- point_vector_iostream.hxx
- point_vector_iostream.cxx

12.66 OpenGPS::PointVectorWriterContext Class Reference

`#include <point_vector_writer_context.hxx>`

Inheritance diagram for OpenGPS::PointVectorWriterContext:

12.66.1 Detailed Description

Encapsulates an underlying stream of point vector data.

Provides methods for writing point data to that stream. Three values of point data make up one point vector. When all three components have been written (by using the PointVectorWriterContext::Write/ PointVectorWriterContext::Skip methods) PointVectorWriterContext::MoveNext must be called to complete the transaction.

Public Member Functions

- virtual void MoveNext ()=0 throw (...)

  Complete the transaction of the current point vector.

- virtual void Skip ()=0 throw (...)

  There is no point data to be written to the underlying stream for the current axis.

- virtual void Write (const OGPS_Double *const value)=0 throw (...)

  Writes a single point of type OGPS_Double to the underlying stream.

- virtual void Write (const OGPS_Float *const value)=0 throw (...)

  Writes a single point of type OGPS_Float to the underlying stream.

- virtual void Write (const OGPS_Int32 *const value)=0 throw (...)

  Writes a single point of type OGPS_Int32 to the underlying stream.

- virtual void Write (const OGPS_Int16 *const value)=0 throw (...)

  Writes a single point of type OGPS_Int16 to the underlying stream.

- virtual ~PointVectorWriterContext ()

  Destroys this instance.

Protected Member Functions

- virtual OGPS_Boolean IsGood () const =0

  Asks if the underlying data stream is still valid.

- PointVectorWriterContext ()

  Creates a new instance.

12.66.2 Constructor & Destructor Documentation

12.66.2.1 PointVectorWriterContext::~PointVectorWriterContext () [virtual]

Destroys this instance.

12.66.2.2 PointVectorWriterContext::PointVectorWriterContext () [protected]

Creates a new instance.

12.66.3 Member Function Documentation

12.66.3.1 virtual OGPS_Boolean OpenGPS::PointVectorWriterContext::IsGood () const [protected, pure virtual]

Asks if the underlying data stream is still valid.

Returns:

Returns TRUE if no previous access to the underlying data stream was harmful. Returns FALSE if any damage occured.

Implemented in OpenGPS::BinaryPointVectorWriterContext, and OpenGPS::XmlPointVectorWriterContext.

12.66.3.2 virtual void OpenGPS::PointVectorWriterContext::MoveNext () throw (...) [pure virtual]

Complete the transaction of the current point vector.

One point vector is made up of three components (data points). After three subsequent PointVectorWriterContext::Write/ PointVectorWriterContext::Skip calls, call this method to indicate that the whole point vector has been written.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Implemented in OpenGPS::BinaryPointVectorWriterContext, and OpenGPS::XmlPointVectorWriterContext.

12.66.3.3 virtual void OpenGPS::PointVectorWriterContext::Skip () throw (...) [pure virtual]

There is no point data to be written to the underlying stream for the current axis.

One point vector has three components, whereas only the Z component needs to be set explicitly in major cases because of implicit axis definitions of X or Y. Use this skip method then (with the X or Y components) to indicate that, because a successfully written point vector must comprise all three component values.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

This must be called for integrity if there is no point data to be stored because the corresponding axis is of incremental type.

Implemented in OpenGPS::BinaryPointVectorWriterContext, and OpenGPS::XmlPointVectorWriterContext.

12.66.3.4 virtual void OpenGPS::PointVectorWriterContext::Write (const OGPS_Double *const *value*) throw (...) [pure virtual]

Writes a single point of type OGPS_Double to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implemented in OpenGPS::BinaryLSBPointVectorWriterContext, OpenGPS::BinaryMSBPointVectorWriterContext, and OpenGPS::XmlPointVectorWriterContext.

12.66.3.5 virtual void OpenGPS::PointVectorWriterContext::Write (const OGPS_Float *const *value*) throw (...) [pure virtual]

Writes a single point of type OGPS_Float to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implemented in OpenGPS::BinaryLSBPointVectorWriterContext, OpenGPS::BinaryMSBPointVectorWriterContext, and OpenGPS::XmlPointVectorWriterContext.

12.66.3.6 virtual void OpenGPS::PointVectorWriterContext::Write (const OGPS_Int32 *const *value*) throw (...) [pure virtual]

Writes a single point of type OGPS_Int32 to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implemented in OpenGPS::BinaryLSBPointVectorWriterContext, OpenGPS::BinaryMSBPointVectorWriterContext, and OpenGPS::XmlPointVectorWriterContext.

12.66.3.7 virtual void OpenGPS::PointVectorWriterContext::Write (const OGPS_Int16 *const *value*) throw (...) `[pure virtual]`

Writes a single point of type OGPS_Int16 to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implemented in OpenGPS::BinaryLSBPointVectorWriterContext, OpenGPS::BinaryMSBPointVectorWriterContext, and OpenGPS::XmlPointVectorWriterContext.

The documentation for this class was generated from the following files:

- point_vector_writer_context.hxx
- xml_point_vector_writer_context.cxx

12.67 OpenGPS::Schemas::ISO5436_2::ProbingSystemType Class Reference

`#include <iso5436_2_xsd.hxx>`

12.67.1 Detailed Description

Class corresponding to the ProbingSystemType schema type.

Identification

Accessor and modifier functions for the Identification required element.

Vendor specific identification of probe tip, lens, etc...

- typedef ::xsd::cxx::tree::traits< Identification_type, wchar_t > Identification_traits

  Element traits type.

- typedef ::xml_schema::token Identification_type

  Element type.

- void Identification (::std::auto_ptr< Identification_type > p)

  Set the element value without copying.

- void Identification (const Identification_type &x)

  Set the element value.

- Identification_type & Identification ()
  Return a read-write reference to the element.
- const Identification_type & Identification () const
  Return a read-only (constant) reference to the element.

Identification

Accessor and modifier functions for the Identification required element.

Vendor specific identification of probe tip, lens, etc...

- typedef ::xsd::cxx::tree::traits< Identification_type, wchar_t > Identification_traits
  Element traits type.
- typedef ::xml_schema::token Identification_type
  Element type.
- void Identification (::std::auto_ptr< Identification_type > p)
  Set the element value without copying.
- void Identification (const Identification_type &x)
  Set the element value.
- Identification_type & Identification ()
  Return a read-write reference to the element.
- const Identification_type & Identification () const
  Return a read-only (constant) reference to the element.

Type

Accessor and modifier functions for the Type required element.

one of "NonContacting" or "Contacting"

- typedef ::xsd::cxx::tree::traits< Type_type, wchar_t > Type_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::Type Type_type
  Element type.
- void Type (::std::auto_ptr< Type_type > p)
  Set the element value without copying.
- void Type (const Type_type &x)
  Set the element value.

- Type_type & Type ()
  Return a read-write reference to the element.
- const Type_type & Type () const
  Return a read-only (constant) reference to the element.

Type

Accessor and modifier functions for the Type required element.

one of "NonContacting" or "Contacting"

- typedef ::xsd::cxx::tree::traits< Type_type, wchar_t > Type_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::Type Type_type
  Element type.
- void Type (::std::auto_ptr< Type_type > p)
  Set the element value without copying.
- void Type (const Type_type &x)
  Set the element value.
- Type_type & Type ()
  Return a read-write reference to the element.
- const Type_type & Type () const
  Return a read-only (constant) reference to the element.

Constructors

- virtual ProbingSystemType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const
  Copy the object polymorphically.
- ProbingSystemType (const ProbingSystemType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)
  Copy constructor.
- ProbingSystemType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)
  Construct an instance from a DOM element.
- ProbingSystemType (const Type_type &, const Identification_type &)
  Construct an instance from the ultimate base and initializers for required elements and attributes.

Constructors

- virtual ProbingSystemType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- ProbingSystemType (const ProbingSystemType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- ProbingSystemType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- ProbingSystemType (const Type_type &, const Identification_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Protected Member Functions

- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)
- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)

Private Attributes

- ::xsd::cxx::tree::one< Identification_type > Identification_
- ::xsd::cxx::tree::one< Type_type > Type_

12.67.2 Member Typedef Documentation

12.67.2.1 typedef ::xsd::cxx::tree::traits< Identification_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification_traits

Element traits type.

12.67.2.2 typedef ::xsd::cxx::tree::traits< Identification_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification_traits

Element traits type.

12.67.2.3 typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification_type

Element type.

12.67.2.4 typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification_type

Element type.

12.67.2.5 typedef ::xsd::cxx::tree::traits< Type_type, wchar_t > OpenGPS::Schemas::ISO5436_-2::ProbingSystemType::Type_traits

Element traits type.

12.67.2.6 typedef ::xsd::cxx::tree::traits< Type_type, wchar_t > OpenGPS::Schemas::ISO5436_-2::ProbingSystemType::Type_traits

Element traits type.

12.67.2.7 typedef ::OpenGPS::Schemas::ISO5436_2::Type OpenGPS::Schemas::ISO5436_-2::ProbingSystemType::Type_type

Element type.

12.67.2.8 typedef ::OpenGPS::Schemas::ISO5436_2::Type OpenGPS::Schemas::ISO5436_-2::ProbingSystemType::Type_type

Element type.

12.67.3 Constructor & Destructor Documentation

12.67.3.1 OpenGPS::Schemas::ISO5436_2::ProbingSystemType::ProbingSystemType (const Type_type & *Type*, const Identification_type & *Identification*)

Construct an instance from the ultimate base and initializers for required elements and attributes.

Referenced by _clone().

12.67.3.2 OpenGPS::Schemas::ISO5436_2::ProbingSystemType::ProbingSystemType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

References parse().

Here is the call graph for this function:

12.67.3.3 OpenGPS::Schemas::ISO5436_2::ProbingSystemType::ProbingSystemType (const ProbingSystemType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.67.3.4 OpenGPS::Schemas::ISO5436_2::ProbingSystemType::ProbingSystemType (const Type_type &, const Identification_type &)

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.67.3.5 OpenGPS::Schemas::ISO5436_2::ProbingSystemType::ProbingSystemType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.67.3.6 OpenGPS::Schemas::ISO5436_2::ProbingSystemType::ProbingSystemType (const ProbingSystemType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.67.4 Member Function Documentation

12.67.4.1 virtual ProbingSystemType* OpenGPS::Schemas::ISO5436_2::ProbingSystemType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.67.4.2 ProbingSystemType * OpenGPS::Schemas::ISO5436_2::ProbingSystemType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References ProbingSystemType().

Here is the call graph for this function:

12.67.4.3 void OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification (::std::auto_ptr< Identification_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.67.4.4 void OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification (const Identification_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.67.4.5 Identification_type& OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.67.4.6 const Identification_type& OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.67.4.7 void OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification (::std::auto_ptr< Identification_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.67.4.8 void OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification (const Identification_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.67.4.9 Identification_type& OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.67.4.10 const Identification_type& OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.67.4.11 void OpenGPS::Schemas::ISO5436_2::ProbingSystemType::parse (::xsd::cxx::xml::dom::parser< wchar_t > &, ::xml_schema::flags) `[protected]`

12.67.4.12 void OpenGPS::Schemas::ISO5436_2::ProbingSystemType::parse (::xsd::cxx::xml::dom::parser< wchar_t > & *p*, ::xml_schema::flags *f*) [protected]

References Identification(), Identification_, n, Type(), and Type_.

Referenced by ProbingSystemType().

Here is the call graph for this function:

12.67.4.13 void OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Type (::std::auto_ptr< Type_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.67.4.14 void OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Type (const Type_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.67.4.15 Type_type& OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Type ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.67.4.16 const Type_type& OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Type () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.67.4.17 void OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Type (::std::auto_ptr< Type_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.67.4.18 void OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Type (const Type_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.67.4.19 Type_type& OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Type ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.67.4.20 const Type_type& OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Type () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.67.5 Member Data Documentation

12.67.5.1 xsd::cxx::tree::one< Identification_type > OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification_ [private]

Referenced by parse().

12.67.5.2 xsd::cxx::tree::one< Type_type > OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Type_ [private]

Referenced by parse().

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.68 OpenGPS::Schemas::ISO5436_2::Record1Type Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.68.1 Detailed Description

Class corresponding to the Record1Type schema type.

Record1 contains the axis description

Axes

Accessor and modifier functions for the Axes required element.

Axis description

- typedef ::xsd::cxx::tree::traits< Axes_type, wchar_t > Axes_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::AxesType Axes_type

  Element type.

- void Axes (::std::auto_ptr< Axes_type > p)

  Set the element value without copying.

- void Axes (const Axes_type &x)

  Set the element value.

- Axes_type & Axes ()

  Return a read-write reference to the element.

- const Axes_type & Axes () const

  Return a read-only (constant) reference to the element.

Axes

Accessor and modifier functions for the Axes required element.

Axis description

- typedef ::xsd::cxx::tree::traits< Axes_type, wchar_t > Axes_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::AxesType Axes_type

  Element type.

- void Axes (::std::auto_ptr< Axes_type > p)

  Set the element value without copying.

- void Axes (const Axes_type &x)

  Set the element value.

- Axes_type & Axes ()

  Return a read-write reference to the element.

- const Axes_type & Axes () const

  Return a read-only (constant) reference to the element.

FeatureType

Accessor and modifier functions for the FeatureType required element.

"SUR" for surface type feature, "PRF" for profile type feature. Profile features are allways defined as a matrix of size (N,1,M) with N beeing the number of points in the profile and M the number of layers in z-direction.

- typedef ::xsd::cxx::tree::traits< FeatureType_type, wchar_t > FeatureType_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::FeatureType FeatureType_type

  Element type.

- void FeatureType (::std::auto_ptr< FeatureType_type > p)

  Set the element value without copying.

- void FeatureType (const FeatureType_type &x)

  Set the element value.

- FeatureType_type & FeatureType ()

  Return a read-write reference to the element.

- const FeatureType_type & FeatureType () const

  Return a read-only (constant) reference to the element.

FeatureType

Accessor and modifier functions for the FeatureType required element.

"SUR" for surface type feature, "PRF" for profile type feature. Profile features are allways defined as a matrix of size (N,1,M) with N beeing the number of points in the profile and M the number of layers in z-direction.

- typedef ::xsd::cxx::tree::traits< FeatureType_type, wchar_t > FeatureType_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::FeatureType FeatureType_type
  Element type.
- void FeatureType (::std::auto_ptr< FeatureType_type > p)
  Set the element value without copying.
- void FeatureType (const FeatureType_type &x)
  Set the element value.
- FeatureType_type & FeatureType ()
  Return a read-write reference to the element.
- const FeatureType_type & FeatureType () const
  Return a read-only (constant) reference to the element.

Revision

Accessor and modifier functions for the Revision required element.

Revision of file format. Currently: ISO5436 - 2000

- typedef ::xsd::cxx::tree::traits< Revision_type, wchar_t > Revision_traits
  Element traits type.
- typedef ::xml_schema::token Revision_type
  Element type.
- void Revision (::std::auto_ptr< Revision_type > p)
  Set the element value without copying.
- void Revision (const Revision_type &x)
  Set the element value.
- Revision_type & Revision ()
  Return a read-write reference to the element.
- const Revision_type & Revision () const
  Return a read-only (constant) reference to the element.

Revision

Accessor and modifier functions for the Revision required element.

Revision of file format. Currently: ISO5436 - 2000

- typedef ::xsd::cxx::tree::traits< Revision_type, wchar_t > Revision_traits
  Element traits type.

- typedef ::xml_schema::token Revision_type

  Element type.

- void Revision (::std::auto_ptr< Revision_type > p)

  Set the element value without copying.

- void Revision (const Revision_type &x)

  Set the element value.

- Revision_type & Revision ()

  Return a read-write reference to the element.

- const Revision_type & Revision () const

  Return a read-only (constant) reference to the element.

Constructors

- virtual Record1Type * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- Record1Type (const Record1Type &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- Record1Type (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- Record1Type (const Revision_type &, const FeatureType_type &, const Axes_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Constructors

- virtual Record1Type * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- Record1Type (const Record1Type &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- Record1Type (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- Record1Type (const Revision_type &, const FeatureType_type &, const Axes_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Protected Member Functions

- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)
- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)

Private Attributes

- ::xsd::cxx::tree::one< Axes_type > Axes_
- ::xsd::cxx::tree::one< FeatureType_type > FeatureType_
- ::xsd::cxx::tree::one< Revision_type > Revision_

12.68.2 Member Typedef Documentation

12.68.2.1 typedef ::xsd::cxx::tree::traits< Axes_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record1Type::Axes_traits

Element traits type.

12.68.2.2 typedef ::xsd::cxx::tree::traits< Axes_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record1Type::Axes_traits

Element traits type.

12.68.2.3 typedef ::OpenGPS::Schemas::ISO5436_2::AxesType OpenGPS::Schemas::ISO5436_2::Record1Type::Axes_type

Element type.

12.68.2.4 typedef ::OpenGPS::Schemas::ISO5436_2::AxesType OpenGPS::Schemas::ISO5436_2::Record1Type::Axes_type

Element type.

12.68.2.5 typedef ::xsd::cxx::tree::traits< FeatureType_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record1Type::FeatureType_traits

Element traits type.

12.68.2.6 typedef ::xsd::cxx::tree::traits< FeatureType_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record1Type::FeatureType_traits

Element traits type.

12.68.2.7 typedef ::OpenGPS::Schemas::ISO5436_2::FeatureType OpenGPS::Schemas::ISO5436_2::Record1Type::FeatureType_type

Element type.

12.68.2.8 typedef ::OpenGPS::Schemas::ISO5436_2::FeatureType OpenGPS::Schemas::ISO5436_2::Record1Type::FeatureType_type

Element type.

12.68.2.9 typedef ::xsd::cxx::tree::traits< Revision_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record1Type::Revision_traits

Element traits type.

12.68.2.10 typedef ::xsd::cxx::tree::traits< Revision_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record1Type::Revision_traits

Element traits type.

12.68.2.11 typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_-2::Record1Type::Revision_type

Element type.

12.68.2.12 typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_-2::Record1Type::Revision_type

Element type.

12.68.3 Constructor & Destructor Documentation

12.68.3.1 OpenGPS::Schemas::ISO5436_2::Record1Type::Record1Type (const Revision_type & *Revision*, const FeatureType_type & *FeatureType*, const Axes_type & *Axes*)

Construct an instance from the ultimate base and initializers for required elements and attributes.

Referenced by _clone().

12.68.3.2 OpenGPS::Schemas::ISO5436_2::Record1Type::Record1Type (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

References parse().

Here is the call graph for this function:

12.68.3.3 OpenGPS::Schemas::ISO5436_2::Record1Type::Record1Type (const Record1Type & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.68.3.4 OpenGPS::Schemas::ISO5436_2::Record1Type::Record1Type (const Revision_type &, const FeatureType_type &, const Axes_type &)

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.68.3.5 OpenGPS::Schemas::ISO5436_2::Record1Type::Record1Type (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.68.3.6 OpenGPS::Schemas::ISO5436_2::Record1Type::Record1Type (const Record1Type & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.68.4 Member Function Documentation

12.68.4.1 virtual Record1Type* OpenGPS::Schemas::ISO5436_2::Record1Type::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.68.4.2 Record1Type * OpenGPS::Schemas::ISO5436_2::Record1Type::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References Record1Type().

Here is the call graph for this function:

12.68.4.3 void OpenGPS::Schemas::ISO5436_2::Record1Type::Axes (::std::auto_ptr< Axes_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.68.4.4 void OpenGPS::Schemas::ISO5436_2::Record1Type::Axes (const Axes_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.68.4.5 Axes_type& OpenGPS::Schemas::ISO5436_2::Record1Type::Axes ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.68.4.6 const Axes_type& OpenGPS::Schemas::ISO5436_2::Record1Type::Axes () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.68.4.7 void OpenGPS::Schemas::ISO5436_2::Record1Type::Axes (::std::auto_ptr< Axes_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References Axes_.

12.68.4.8 void OpenGPS::Schemas::ISO5436_2::Record1Type::Axes (const Axes_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References Axes_.

12.68.4.9 Record1Type::Axes_type & OpenGPS::Schemas::ISO5436_2::Record1Type::Axes ()

Return a read-write reference to the element.

Returns:

A reference to the element.

References Axes_.

12.68.4.10 const Record1Type::Axes_type & OpenGPS::Schemas::ISO5436_2::Record1Type::Axes () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

References Axes_.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.68.4.11 void OpenGPS::Schemas::ISO5436_2::Record1Type::FeatureType (::std::auto_ptr< FeatureType_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.68.4.12 void OpenGPS::Schemas::ISO5436_2::Record1Type::FeatureType (const FeatureType_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.68.4.13 FeatureType_type& OpenGPS::Schemas::ISO5436_2::Record1Type::FeatureType ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.68.4.14 const FeatureType_type& OpenGPS::Schemas::ISO5436_2::Record1Type::FeatureType () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.68.4.15 void OpenGPS::Schemas::ISO5436_2::Record1Type::FeatureType (::std::auto_ptr< FeatureType_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References FeatureType_.

12.68.4.16 void OpenGPS::Schemas::ISO5436_2::Record1Type::FeatureType (const FeatureType_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References FeatureType_.

12.68.4.17 Record1Type::FeatureType_type & OpenGPS::Schemas::ISO5436_2::Record1Type::FeatureType ()

Return a read-write reference to the element.

Returns:

A reference to the element.

References FeatureType_.

12.68.4.18 const Record1Type::FeatureType_type & OpenGPS::Schemas::ISO5436_2::Record1Type::FeatureType () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

References FeatureType_.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.68.4.19 void OpenGPS::Schemas::ISO5436_2::Record1Type::parse (::xsd::cxx::xml::dom::parser< wchar_t > &, ::xml_schema::flags) [protected]

12.68.4.20 void OpenGPS::Schemas::ISO5436_2::Record1Type::parse (::xsd::cxx::xml::dom::parser< wchar_t > & *p*, ::xml_schema::flags *f*) [protected]

References Axes(), Axes_, FeatureType(), FeatureType_, n, Revision(), and Revision_.

Referenced by Record1Type().

Here is the call graph for this function:

12.68.4.21 void OpenGPS::Schemas::ISO5436_2::Record1Type::Revision (::std::auto_ptr< Revision_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.68.4.22 void OpenGPS::Schemas::ISO5436_2::Record1Type::Revision (const Revision_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.68.4.23 Revision_type& OpenGPS::Schemas::ISO5436_2::Record1Type::Revision ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.68.4.24 const Revision_type& OpenGPS::Schemas::ISO5436_2::Record1Type::Revision () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.68.4.25 void OpenGPS::Schemas::ISO5436_2::Record1Type::Revision (::std::auto_ptr< Revision_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References Revision_.

12.68.4.26 void OpenGPS::Schemas::ISO5436_2::Record1Type::Revision (const Revision_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References Revision_.

12.68.4.27 Record1Type::Revision_type & OpenGPS::Schemas::ISO5436_2::Record1Type::Revision ()

Return a read-write reference to the element.

Returns:

A reference to the element.

References Revision_.

12.68.4.28 const Record1Type::Revision_type & OpenGPS::Schemas::ISO5436_2::Record1Type::Revision () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

References Revision_.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.68.5 Member Data Documentation

12.68.5.1 xsd::cxx::tree::one< Axes_type > OpenGPS::Schemas::ISO5436_2::Record1Type::Axes_ [private]

Referenced by Axes(), and parse().

12.68.5.2 xsd::cxx::tree::one< FeatureType_type > OpenGPS::Schemas::ISO5436_2::Record1Type::FeatureType_ [private]

Referenced by FeatureType(), and parse().

12.68.5.3 xsd::cxx::tree::one< Revision_type > OpenGPS::Schemas::ISO5436_2::Record1Type::Revision_ [private]

Referenced by parse(), and Revision().

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.69 OpenGPS::Schemas::ISO5436_2::Record2Type Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.69.1 Detailed Description

Class corresponding to the Record2Type schema type.

Record2 is optional and contains the metadata of the data set.

CalibrationDate

Accessor and modifier functions for the CalibrationDate required element.

Date of currently used calibration

- typedef ::xsd::cxx::tree::traits< CalibrationDate_type, wchar_t > CalibrationDate_traits

  Element traits type.

- typedef ::xml_schema::date_time CalibrationDate_type

  Element type.

- void CalibrationDate (::std::auto_ptr< CalibrationDate_type > p)

  Set the element value without copying.

- void CalibrationDate (const CalibrationDate_type &x)

  Set the element value.

- CalibrationDate_type & CalibrationDate ()

  Return a read-write reference to the element.

- const CalibrationDate_type & CalibrationDate () const

  Return a read-only (constant) reference to the element.

CalibrationDate

Accessor and modifier functions for the CalibrationDate required element.

Date of currently used calibration

- typedef ::xsd::cxx::tree::traits< CalibrationDate_type, wchar_t > CalibrationDate_traits
  Element traits type.
- typedef ::xml_schema::date_time CalibrationDate_type
  Element type.
- void CalibrationDate (::std::auto_ptr< CalibrationDate_type > p)
  Set the element value without copying.
- void CalibrationDate (const CalibrationDate_type &x)
  Set the element value.
- CalibrationDate_type & CalibrationDate ()
  Return a read-write reference to the element.
- const CalibrationDate_type & CalibrationDate () const
  Return a read-only (constant) reference to the element.

Comment

Accessor and modifier functions for the Comment optional element.

User comment to this data set

- typedef ::xsd::cxx::tree::optional< Comment_type > Comment_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< Comment_type, wchar_t > Comment_traits
  Element traits type.
- typedef ::xml_schema::string Comment_type
  Element type.
- void Comment (::std::auto_ptr< Comment_type > p)
  Set the element value without copying.
- void Comment (const Comment_optional &x)
  Set the element value.
- void Comment (const Comment_type &x)
  Set the element value.
- Comment_optional & Comment ()
  Return a read-write reference to the element container.

- const Comment_optional & Comment () const

  Return a read-only (constant) reference to the element container.

Comment

Accessor and modifier functions for the Comment optional element.

User comment to this data set

- typedef ::xsd::cxx::tree::optional< Comment_type > Comment_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< Comment_type, wchar_t > Comment_traits

  Element traits type.

- typedef ::xml_schema::string Comment_type

  Element type.

- void Comment (::std::auto_ptr< Comment_type > p)

  Set the element value without copying.

- void Comment (const Comment_optional &x)

  Set the element value.

- void Comment (const Comment_type &x)

  Set the element value.

- Comment_optional & Comment ()

  Return a read-write reference to the element container.

- const Comment_optional & Comment () const

  Return a read-only (constant) reference to the element container.

Creator

Accessor and modifier functions for the Creator optional element.

Optional name of the creator of the file: Name of the measuring person.

- typedef ::xsd::cxx::tree::optional< Creator_type > Creator_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< Creator_type, wchar_t > Creator_traits

  Element traits type.

- typedef ::xml_schema::token Creator_type

  Element type.

- void Creator (::std::auto_ptr< Creator_type > p)
  Set the element value without copying.
- void Creator (const Creator_optional &x)
  Set the element value.
- void Creator (const Creator_type &x)
  Set the element value.
- Creator_optional & Creator ()
  Return a read-write reference to the element container.
- const Creator_optional & Creator () const
  Return a read-only (constant) reference to the element container.

Creator

Accessor and modifier functions for the Creator optional element.

Optional name of the creator of the file: Name of the measuring person.

- typedef ::xsd::cxx::tree::optional< Creator_type > Creator_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< Creator_type, wchar_t > Creator_traits
  Element traits type.
- typedef ::xml_schema::token Creator_type
  Element type.
- void Creator (::std::auto_ptr< Creator_type > p)
  Set the element value without copying.
- void Creator (const Creator_optional &x)
  Set the element value.
- void Creator (const Creator_type &x)
  Set the element value.
- Creator_optional & Creator ()
  Return a read-write reference to the element container.
- const Creator_optional & Creator () const
  Return a read-only (constant) reference to the element container.

Date

Accessor and modifier functions for the Date required element.

Date and time of file creation.

- typedef ::xsd::cxx::tree::traits< Date_type, wchar_t > Date_traits

  Element traits type.

- typedef ::xml_schema::date_time Date_type

  Element type.

- void Date (::std::auto_ptr< Date_type > p)

  Set the element value without copying.

- void Date (const Date_type &x)

  Set the element value.

- Date_type & Date ()

  Return a read-write reference to the element.

- const Date_type & Date () const

  Return a read-only (constant) reference to the element.

Date

Accessor and modifier functions for the Date required element.

Date and time of file creation.

- typedef ::xsd::cxx::tree::traits< Date_type, wchar_t > Date_traits

  Element traits type.

- typedef ::xml_schema::date_time Date_type

  Element type.

- void Date (::std::auto_ptr< Date_type > p)

  Set the element value without copying.

- void Date (const Date_type &x)

  Set the element value.

- Date_type & Date ()

  Return a read-write reference to the element.

- const Date_type & Date () const

  Return a read-only (constant) reference to the element.

Instrument

Accessor and modifier functions for the Instrument required element.

- typedef ::xsd::cxx::tree::traits< Instrument_type, wchar_t > Instrument_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::InstrumentType Instrument_type
  Element type.
- void Instrument (::std::auto_ptr< Instrument_type > p)
  Set the element value without copying.
- void Instrument (const Instrument_type &x)
  Set the element value.
- Instrument_type & Instrument ()
  Return a read-write reference to the element.
- const Instrument_type & Instrument () const
  Return a read-only (constant) reference to the element.

Instrument

Accessor and modifier functions for the Instrument required element.

- typedef ::xsd::cxx::tree::traits< Instrument_type, wchar_t > Instrument_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::InstrumentType Instrument_type
  Element type.
- void Instrument (::std::auto_ptr< Instrument_type > p)
  Set the element value without copying.
- void Instrument (const Instrument_type &x)
  Set the element value.
- Instrument_type & Instrument ()
  Return a read-write reference to the element.
- const Instrument_type & Instrument () const
  Return a read-only (constant) reference to the element.

ProbingSystem

Accessor and modifier functions for the ProbingSystem required element.

- typedef ::xsd::cxx::tree::traits< ProbingSystem_type, wchar_t > ProbingSystem_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::ProbingSystemType ProbingSystem_type
  Element type.
- void ProbingSystem (::std::auto_ptr< ProbingSystem_type > p)
  Set the element value without copying.
- void ProbingSystem (const ProbingSystem_type &x)
  Set the element value.
- ProbingSystem_type & ProbingSystem ()
  Return a read-write reference to the element.
- const ProbingSystem_type & ProbingSystem () const
  Return a read-only (constant) reference to the element.

ProbingSystem

Accessor and modifier functions for the ProbingSystem required element.

- typedef ::xsd::cxx::tree::traits< ProbingSystem_type, wchar_t > ProbingSystem_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::ProbingSystemType ProbingSystem_type
  Element type.
- void ProbingSystem (::std::auto_ptr< ProbingSystem_type > p)
  Set the element value without copying.
- void ProbingSystem (const ProbingSystem_type &x)
  Set the element value.
- ProbingSystem_type & ProbingSystem ()
  Return a read-write reference to the element.
- const ProbingSystem_type & ProbingSystem () const
  Return a read-only (constant) reference to the element.

Constructors

- virtual Record2Type * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- Record2Type (const Record2Type &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- Record2Type (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- Record2Type (const Date_type &, const Instrument_type &, const CalibrationDate_type &, const ProbingSystem_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Constructors

- virtual Record2Type * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- Record2Type (const Record2Type &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- Record2Type (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- Record2Type (const Date_type &, const Instrument_type &, const CalibrationDate_type &, const ProbingSystem_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Protected Member Functions

- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)
- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)

Private Attributes

- ::xsd::cxx::tree::one< CalibrationDate_type > CalibrationDate_
- Comment_optional Comment_
- Creator_optional Creator_
- ::xsd::cxx::tree::one< Date_type > Date_
- ::xsd::cxx::tree::one< Instrument_type > Instrument_
- ::xsd::cxx::tree::one< ProbingSystem_type > ProbingSystem_

12.69.2 Member Typedef Documentation

12.69.2.1 typedef ::xsd::cxx::tree::traits< CalibrationDate_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record2Type::CalibrationDate_traits

Element traits type.

12.69.2.2 typedef ::xsd::cxx::tree::traits< CalibrationDate_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record2Type::CalibrationDate_traits

Element traits type.

12.69.2.3 typedef ::xml_schema::date_time OpenGPS::Schemas::ISO5436_-2::Record2Type::CalibrationDate_type

Element type.

12.69.2.4 typedef ::xml_schema::date_time OpenGPS::Schemas::ISO5436_-2::Record2Type::CalibrationDate_type

Element type.

12.69.2.5 typedef ::xsd::cxx::tree::optional< Comment_type > OpenGPS::Schemas::ISO5436_-2::Record2Type::Comment_optional

Element optional container type.

12.69.2.6 typedef ::xsd::cxx::tree::optional< Comment_type > OpenGPS::Schemas::ISO5436_-2::Record2Type::Comment_optional

Element optional container type.

12.69.2.7 typedef ::xsd::cxx::tree::traits< Comment_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record2Type::Comment_traits

Element traits type.

12.69.2.8 typedef ::xsd::cxx::tree::traits< Comment_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record2Type::Comment_traits

Element traits type.

12.69.2.9 typedef ::xml_schema::string OpenGPS::Schemas::ISO5436_-2::Record2Type::Comment_type

Element type.

12.69.2.10 typedef ::xml_schema::string OpenGPS::Schemas::ISO5436_-2::Record2Type::Comment_type

Element type.

12.69.2.11 typedef ::xsd::cxx::tree::optional< Creator_type > OpenGPS::Schemas::ISO5436_2::Record2Type::Creator_optional

Element optional container type.

12.69.2.12 typedef ::xsd::cxx::tree::optional< Creator_type > OpenGPS::Schemas::ISO5436_2::Record2Type::Creator_optional

Element optional container type.

12.69.2.13 typedef ::xsd::cxx::tree::traits< Creator_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record2Type::Creator_traits

Element traits type.

12.69.2.14 typedef ::xsd::cxx::tree::traits< Creator_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record2Type::Creator_traits

Element traits type.

12.69.2.15 typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_2::Record2Type::Creator_type

Element type.

12.69.2.16 typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_2::Record2Type::Creator_type

Element type.

12.69.2.17 typedef ::xsd::cxx::tree::traits< Date_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record2Type::Date_traits

Element traits type.

12.69.2.18 typedef ::xsd::cxx::tree::traits< Date_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record2Type::Date_traits

Element traits type.

12.69.2.19 typedef ::xml_schema::date_time OpenGPS::Schemas::ISO5436_2::Record2Type::Date_type

Element type.

12.69.2.20 typedef ::xml_schema::date_time OpenGPS::Schemas::ISO5436_2::Record2Type::Date_type

Element type.

12.69.2.21 typedef ::xsd::cxx::tree::traits< Instrument_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record2Type::Instrument_traits

Element traits type.

12.69.2.22 typedef ::xsd::cxx::tree::traits< Instrument_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record2Type::Instrument_traits

Element traits type.

12.69.2.23 typedef ::OpenGPS::Schemas::ISO5436_2::InstrumentType OpenGPS::Schemas::ISO5436_2::Record2Type::Instrument_type

Element type.

12.69.2.24 typedef ::OpenGPS::Schemas::ISO5436_2::InstrumentType OpenGPS::Schemas::ISO5436_2::Record2Type::Instrument_type

Element type.

12.69.2.25 typedef ::xsd::cxx::tree::traits< ProbingSystem_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record2Type::ProbingSystem_traits

Element traits type.

12.69.2.26 typedef ::xsd::cxx::tree::traits< ProbingSystem_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record2Type::ProbingSystem_traits

Element traits type.

12.69.2.27 typedef ::OpenGPS::Schemas::ISO5436_2::ProbingSystemType OpenGPS::Schemas::ISO5436_2::Record2Type::ProbingSystem_type

Element type.

12.69.2.28 typedef ::OpenGPS::Schemas::ISO5436_2::ProbingSystemType OpenGPS::Schemas::ISO5436_2::Record2Type::ProbingSystem_type

Element type.

12.69.3 Constructor & Destructor Documentation

12.69.3.1 OpenGPS::Schemas::ISO5436_2::Record2Type::Record2Type (const Date_type & *Date*, const Instrument_type & *Instrument*, const CalibrationDate_type & *CalibrationDate*, const ProbingSystem_type & *ProbingSystem*)

Construct an instance from the ultimate base and initializers for required elements and attributes.

Referenced by _clone().

12.69.3.2 OpenGPS::Schemas::ISO5436_2::Record2Type::Record2Type (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

References parse().

Here is the call graph for this function:

12.69.3.3 OpenGPS::Schemas::ISO5436_2::Record2Type::Record2Type (const Record2Type & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.69.3.4 OpenGPS::Schemas::ISO5436_2::Record2Type::Record2Type (const Date_type &, const Instrument_type &, const CalibrationDate_type &, const ProbingSystem_type &)

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.69.3.5 OpenGPS::Schemas::ISO5436_2::Record2Type::Record2Type (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.69.3.6 OpenGPS::Schemas::ISO5436_2::Record2Type::Record2Type (const Record2Type & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.69.4 Member Function Documentation

12.69.4.1 virtual Record2Type* OpenGPS::Schemas::ISO5436_2::Record2Type::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.69.4.2 Record2Type * OpenGPS::Schemas::ISO5436_2::Record2Type::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References Record2Type().

Here is the call graph for this function:

12.69.4.3 void OpenGPS::Schemas::ISO5436_2::Record2Type::CalibrationDate (::std::auto_ptr< CalibrationDate_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.69.4.4 void OpenGPS::Schemas::ISO5436_2::Record2Type::CalibrationDate (const CalibrationDate_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.69.4.5 CalibrationDate_type& OpenGPS::Schemas::ISO5436_2::Record2Type::CalibrationDate ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.69.4.6 const CalibrationDate_type& OpenGPS::Schemas::ISO5436_2::Record2Type::CalibrationDate () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.69.4.7 void OpenGPS::Schemas::ISO5436_2::Record2Type::CalibrationDate (::std::auto_ptr< CalibrationDate_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References CalibrationDate_.

12.69.4.8 void OpenGPS::Schemas::ISO5436_2::Record2Type::CalibrationDate (const CalibrationDate_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References CalibrationDate_.

12.69.4.9 Record2Type::CalibrationDate_type & OpenGPS::Schemas::ISO5436_2::Record2Type::CalibrationDate ()

Return a read-write reference to the element.

Returns:

A reference to the element.

References CalibrationDate_.

12.69.4.10 const Record2Type::CalibrationDate_type & OpenGPS::Schemas::ISO5436_2::Record2Type::CalibrationDate () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

References CalibrationDate_.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.69.4.11 void OpenGPS::Schemas::ISO5436_2::Record2Type::Comment (::std::auto_ptr< Comment_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.69.4.12 void OpenGPS::Schemas::ISO5436_2::Record2Type::Comment (const Comment_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.69.4.13 void OpenGPS::Schemas::ISO5436_2::Record2Type::Comment (const Comment_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.69.4.14 Comment_optional& OpenGPS::Schemas::ISO5436_2::Record2Type::Comment ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.69.4.15 const Comment_optional& OpenGPS::Schemas::ISO5436_2::Record2Type::Comment () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

12.69.4.16 void OpenGPS::Schemas::ISO5436_2::Record2Type::Comment (::std::auto_ptr< Comment_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References Comment_.

12.69.4.17 void OpenGPS::Schemas::ISO5436_2::Record2Type::Comment (const Comment_-optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

References Comment_.

12.69.4.18 void OpenGPS::Schemas::ISO5436_2::Record2Type::Comment (const Comment_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References Comment_.

12.69.4.19 Record2Type::Comment_optional & OpenGPS::Schemas::ISO5436_2::Record2Type::Comment ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

References Comment_.

12.69.4.20 const Record2Type::Comment_optional & OpenGPS::Schemas::ISO5436_2::Record2Type::Comment () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

References Comment_.

Referenced by mediumComplexExample(), OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), parse(), performanceDouble(), performanceInt16(), and simpleExample().

12.69.4.21 void OpenGPS::Schemas::ISO5436_2::Record2Type::Creator (::std::auto_ptr< Creator_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.69.4.22 void OpenGPS::Schemas::ISO5436_2::Record2Type::Creator (const Creator_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.69.4.23 void OpenGPS::Schemas::ISO5436_2::Record2Type::Creator (const Creator_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.69.4.24 Creator_optional& OpenGPS::Schemas::ISO5436_2::Record2Type::Creator ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.69.4.25 const Creator_optional& OpenGPS::Schemas::ISO5436_2::Record2Type::Creator () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

12.69.4.26 void OpenGPS::Schemas::ISO5436_2::Record2Type::Creator (::std::auto_ptr< Creator_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References Creator_.

12.69.4.27 void OpenGPS::Schemas::ISO5436_2::Record2Type::Creator (const Creator_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

References Creator_.

12.69.4.28 void OpenGPS::Schemas::ISO5436_2::Record2Type::Creator (const Creator_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References Creator_.

12.69.4.29 Record2Type::Creator_optional & OpenGPS::Schemas::ISO5436_2::Record2Type::Creator ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

References Creator_.

12.69.4.30 const Record2Type::Creator_optional & OpenGPS::Schemas::ISO5436_2::Record2Type::Creator () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

References Creator_.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.69.4.31 void OpenGPS::Schemas::ISO5436_2::Record2Type::Date (::std::auto_ptr< Date_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.69.4.32 void OpenGPS::Schemas::ISO5436_2::Record2Type::Date (const Date_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.69.4.33 Date_type& OpenGPS::Schemas::ISO5436_2::Record2Type::Date ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.69.4.34 const Date_type& OpenGPS::Schemas::ISO5436_2::Record2Type::Date () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.69.4.35 void OpenGPS::Schemas::ISO5436_2::Record2Type::Date (::std::auto_ptr< Date_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References Date_.

12.69.4.36 void OpenGPS::Schemas::ISO5436_2::Record2Type::Date (const Date_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References Date_.

12.69.4.37 Record2Type::Date_type & OpenGPS::Schemas::ISO5436_2::Record2Type::Date ()

Return a read-write reference to the element.

Returns:

A reference to the element.

References Date_.

12.69.4.38 const Record2Type::Date_type & OpenGPS::Schemas::ISO5436_2::Record2Type::Date () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

References Date_.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.69.4.39 void OpenGPS::Schemas::ISO5436_2::Record2Type::Instrument (::std::auto_ptr< Instrument_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.69.4.40 void OpenGPS::Schemas::ISO5436_2::Record2Type::Instrument (const Instrument_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.69.4.41 Instrument_type& OpenGPS::Schemas::ISO5436_2::Record2Type::Instrument ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.69.4.42 const Instrument_type& OpenGPS::Schemas::ISO5436_2::Record2Type::Instrument () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.69.4.43 void OpenGPS::Schemas::ISO5436_2::Record2Type::Instrument (::std::auto_ptr< Instrument_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References Instrument_.

12.69.4.44 void OpenGPS::Schemas::ISO5436_2::Record2Type::Instrument (const Instrument_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References Instrument_.

12.69.4.45 Record2Type::Instrument_type & OpenGPS::Schemas::ISO5436_2::Record2Type::Instrument ()

Return a read-write reference to the element.

Returns:

A reference to the element.

References Instrument_.

12.69.4.46 const Record2Type::Instrument_type & OpenGPS::Schemas::ISO5436_2::Record2Type::Instrument () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

References Instrument_.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.69.4.47 void OpenGPS::Schemas::ISO5436_2::Record2Type::parse (::xsd::cxx::xml::dom::parser< wchar_t > &, ::xml_schema::flags) [protected]

12.69.4.48 void OpenGPS::Schemas::ISO5436_2::Record2Type::parse (::xsd::cxx::xml::dom::parser< wchar_t > & *p*, ::xml_schema::flags *f*) [protected]

References CalibrationDate(), CalibrationDate_, Comment(), Creator(), Date(), Date_, Instrument(), Instrument_, n, ProbingSystem(), and ProbingSystem_.

Referenced by Record2Type().

Here is the call graph for this function:

12.69.4.49 void OpenGPS::Schemas::ISO5436_2::Record2Type::ProbingSystem (::std::auto_ptr< ProbingSystem_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.69.4.50 void OpenGPS::Schemas::ISO5436_2::Record2Type::ProbingSystem (const ProbingSystem_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.69.4.51 ProbingSystem_type& OpenGPS::Schemas::ISO5436_2::Record2Type::ProbingSystem ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.69.4.52 const ProbingSystem_type& OpenGPS::Schemas::ISO5436_2::Record2Type::ProbingSystem () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.69.4.53 void OpenGPS::Schemas::ISO5436_2::Record2Type::ProbingSystem (::std::auto_ptr< ProbingSystem_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References ProbingSystem_.

12.69.4.54 void OpenGPS::Schemas::ISO5436_2::Record2Type::ProbingSystem (const ProbingSystem_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References ProbingSystem_.

12.69.4.55 Record2Type::ProbingSystem_type & OpenGPS::Schemas::ISO5436_2::Record2Type::ProbingSystem ()

Return a read-write reference to the element.

Returns:

A reference to the element.

References ProbingSystem_.

12.69.4.56 const Record2Type::ProbingSystem_type & OpenGPS::Schemas::ISO5436_2::Record2Type::ProbingSystem () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

References ProbingSystem_.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.69.5 Member Data Documentation

12.69.5.1 xsd::cxx::tree::one< CalibrationDate_type > OpenGPS::Schemas::ISO5436_2::Record2Type::CalibrationDate_ [private]

Referenced by CalibrationDate(), and parse().

12.69.5.2 Comment_optional OpenGPS::Schemas::ISO5436_2::Record2Type::Comment_ [private]

Referenced by Comment().

12.69.5.3 Creator_optional OpenGPS::Schemas::ISO5436_2::Record2Type::Creator_ [private]

Referenced by Creator().

12.69.5.4 xsd::cxx::tree::one< Date_type > OpenGPS::Schemas::ISO5436_2::Record2Type::Date_ [private]

Referenced by Date(), and parse().

12.69.5.5 xsd::cxx::tree::one< Instrument_type > OpenGPS::Schemas::ISO5436_2::Record2Type::Instrument_ [private]

Referenced by Instrument(), and parse().

12.69.5.6 xsd::cxx::tree::one< ProbingSystem_type > OpenGPS::Schemas::ISO5436_2::Record2Type::ProbingSystem_ [private]

Referenced by parse(), and ProbingSystem().

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.70 OpenGPS::Schemas::ISO5436_2::Record3Type Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.70.1 Detailed Description

Class corresponding to the Record3Type schema type.

Record 3 contains the measured data.

DataLink

Accessor and modifier functions for the DataLink optional element.

Link specification to an external binary data file.

- typedef ::xsd::cxx::tree::optional< DataLink_type > DataLink_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< DataLink_type, wchar_t > DataLink_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::DataLinkType DataLink_type

  Element type.

- void DataLink (::std::auto_ptr< DataLink_type > p)

  Set the element value without copying.

- void DataLink (const DataLink_optional &x)

  Set the element value.

- void DataLink (const DataLink_type &x)

  Set the element value.

- DataLink_optional & DataLink ()

  Return a read-write reference to the element container.

- const DataLink_optional & DataLink () const

  Return a read-only (constant) reference to the element container.

DataLink

Accessor and modifier functions for the DataLink optional element.

Link specification to an external binary data file.

- typedef ::xsd::cxx::tree::optional< DataLink_type > DataLink_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< DataLink_type, wchar_t > DataLink_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::DataLinkType DataLink_type

  Element type.

- void DataLink (::std::auto_ptr< DataLink_type > p)

  Set the element value without copying.

- void DataLink (const DataLink_optional &x)

  Set the element value.

- void DataLink (const DataLink_type &x)

  Set the element value.

- DataLink_optional & DataLink ()

Return a read-write reference to the element container.

- const DataLink_optional & DataLink () const

  Return a read-only (constant) reference to the element container.

DataList

Accessor and modifier functions for the DataList optional element.

Data list is ordered like specified in DataOrder: Z-Index is empty (only one sample per pixel) X is fastest index, Y is slower, Z is slowest: (x1,y1),(x2,y1),(x3,y1),(x4,y1),(x1,y2)...

- typedef ::xsd::cxx::tree::optional< DataList_type > DataList_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< DataList_type, wchar_t > DataList_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::DataListType DataList_type

  Element type.

- void DataList (::std::auto_ptr< DataList_type > p)

  Set the element value without copying.

- void DataList (const DataList_optional &x)

  Set the element value.

- void DataList (const DataList_type &x)

  Set the element value.

- DataList_optional & DataList ()

  Return a read-write reference to the element container.

- const DataList_optional & DataList () const

  Return a read-only (constant) reference to the element container.

DataList

Accessor and modifier functions for the DataList optional element.

Data list is ordered like specified in DataOrder: Z-Index is empty (only one sample per pixel) X is fastest index, Y is slower, Z is slowest: (x1,y1),(x2,y1),(x3,y1),(x4,y1),(x1,y2)...

- typedef ::xsd::cxx::tree::optional< DataList_type > DataList_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< DataList_type, wchar_t > DataList_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::DataListType DataList_type

  Element type.

- void DataList (::std::auto_ptr< DataList_type > p)

  Set the element value without copying.

- void DataList (const DataList_optional &x)

  Set the element value.

- void DataList (const DataList_type &x)

  Set the element value.

- DataList_optional & DataList ()

  Return a read-write reference to the element container.

- const DataList_optional & DataList () const

  Return a read-only (constant) reference to the element container.

ListDimension

Accessor and modifier functions for the ListDimension optional element.

A list does specify an unordered data set like a point cloud which does not contain topologic information.

- typedef ::xsd::cxx::tree::optional< ListDimension_type > ListDimension_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< ListDimension_type, wchar_t > ListDimension_traits

  Element traits type.

- typedef ::xml_schema::unsigned_long ListDimension_type

  Element type.

- void ListDimension (const ListDimension_optional &x)

  Set the element value.

- void ListDimension (const ListDimension_type &x)

  Set the element value.

- ListDimension_optional & ListDimension ()

  Return a read-write reference to the element container.

- const ListDimension_optional & ListDimension () const

  Return a read-only (constant) reference to the element container.

ListDimension

Accessor and modifier functions for the ListDimension optional element.

A list does specify an unordered data set like a point cloud which does not contain topologic information.

- typedef ::xsd::cxx::tree::optional< ListDimension_type > ListDimension_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< ListDimension_type, wchar_t > ListDimension_traits

  Element traits type.

- typedef ::xml_schema::unsigned_long ListDimension_type

  Element type.

- void ListDimension (const ListDimension_optional &x)

  Set the element value.

- void ListDimension (const ListDimension_type &x)

  Set the element value.

- ListDimension_optional & ListDimension ()

  Return a read-write reference to the element container.

- const ListDimension_optional & ListDimension () const

  Return a read-only (constant) reference to the element container.

MatrixDimension

Accessor and modifier functions for the MatrixDimension optional element.

- typedef ::xsd::cxx::tree::optional< MatrixDimension_type > MatrixDimension_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< MatrixDimension_type, wchar_t > MatrixDimension_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::MatrixDimensionType MatrixDimension_type

  Element type.

- void MatrixDimension (::std::auto_ptr< MatrixDimension_type > p)

  Set the element value without copying.

- void MatrixDimension (const MatrixDimension_optional &x)

  Set the element value.

- void MatrixDimension (const MatrixDimension_type &x)

  Set the element value.

- MatrixDimension_optional & MatrixDimension ()
  Return a read-write reference to the element container.
- const MatrixDimension_optional & MatrixDimension () const
  Return a read-only (constant) reference to the element container.

MatrixDimension

Accessor and modifier functions for the MatrixDimension optional element.

- typedef ::xsd::cxx::tree::optional< MatrixDimension_type > MatrixDimension_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< MatrixDimension_type, wchar_t > MatrixDimension_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::MatrixDimensionType MatrixDimension_type
  Element type.
- void MatrixDimension (::std::auto_ptr< MatrixDimension_type > p)
  Set the element value without copying.
- void MatrixDimension (const MatrixDimension_optional &x)
  Set the element value.
- void MatrixDimension (const MatrixDimension_type &x)
  Set the element value.
- MatrixDimension_optional & MatrixDimension ()
  Return a read-write reference to the element container.
- const MatrixDimension_optional & MatrixDimension () const
  Return a read-only (constant) reference to the element container.

Constructors

- virtual Record3Type * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const
  Copy the object polymorphically.
- Record3Type (const Record3Type &x,::xml_schema::flags f=0,::xml_schema::type *c=0)
  Copy constructor.
- Record3Type (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)
  Construct an instance from a DOM element.
- Record3Type ()
  Construct an instance from the ultimate base and initializers for required elements and attributes.

Constructors

- virtual Record3Type * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- Record3Type (const Record3Type &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- Record3Type (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- Record3Type ()

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Protected Member Functions

- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)
- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)

Private Attributes

- DataLink_optional DataLink_
- DataList_optional DataList_
- ListDimension_optional ListDimension_
- MatrixDimension_optional MatrixDimension_

12.70.2 Member Typedef Documentation

12.70.2.1 typedef ::xsd::cxx::tree::optional< DataLink_type > OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink_optional

Element optional container type.

12.70.2.2 typedef ::xsd::cxx::tree::optional< DataLink_type > OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink_optional

Element optional container type.

12.70.2.3 typedef ::xsd::cxx::tree::traits< DataLink_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink_traits

Element traits type.

12.70.2.4 typedef ::xsd::cxx::tree::traits< DataLink_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink_traits

Element traits type.

12.70.2.5 typedef ::OpenGPS::Schemas::ISO5436_2::DataLinkType OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink_type

Element type.

12.70.2.6 typedef ::OpenGPS::Schemas::ISO5436_2::DataLinkType OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink_type

Element type.

12.70.2.7 typedef ::xsd::cxx::tree::optional< DataList_type > OpenGPS::Schemas::ISO5436_-2::Record3Type::DataList_optional

Element optional container type.

12.70.2.8 typedef ::xsd::cxx::tree::optional< DataList_type > OpenGPS::Schemas::ISO5436_-2::Record3Type::DataList_optional

Element optional container type.

12.70.2.9 typedef ::xsd::cxx::tree::traits< DataList_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record3Type::DataList_traits

Element traits type.

12.70.2.10 typedef ::xsd::cxx::tree::traits< DataList_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record3Type::DataList_traits

Element traits type.

12.70.2.11 typedef ::OpenGPS::Schemas::ISO5436_2::DataListType OpenGPS::Schemas::ISO5436_2::Record3Type::DataList_type

Element type.

12.70.2.12 typedef ::OpenGPS::Schemas::ISO5436_2::DataListType OpenGPS::Schemas::ISO5436_2::Record3Type::DataList_type

Element type.

12.70.2.13 typedef ::xsd::cxx::tree::optional< ListDimension_type > OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension_optional

Element optional container type.

12.70.2.14 typedef ::xsd::cxx::tree::optional< ListDimension_type > OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension_optional

Element optional container type.

12.70.2.15 typedef ::xsd::cxx::tree::traits< ListDimension_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension_traits

Element traits type.

12.70.2.16 typedef ::xsd::cxx::tree::traits< ListDimension_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension_traits

Element traits type.

12.70.2.17 typedef ::xml_schema::unsigned_long OpenGPS::Schemas::ISO5436_-2::Record3Type::ListDimension_type

Element type.

12.70.2.18 typedef ::xml_schema::unsigned_long OpenGPS::Schemas::ISO5436_-2::Record3Type::ListDimension_type

Element type.

12.70.2.19 typedef ::xsd::cxx::tree::optional< MatrixDimension_type > OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension_optional

Element optional container type.

12.70.2.20 typedef ::xsd::cxx::tree::optional< MatrixDimension_type > OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension_optional

Element optional container type.

12.70.2.21 typedef ::xsd::cxx::tree::traits< MatrixDimension_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension_traits

Element traits type.

12.70.2.22 typedef ::xsd::cxx::tree::traits< MatrixDimension_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension_traits

Element traits type.

12.70.2.23 typedef ::OpenGPS::Schemas::ISO5436_2::MatrixDimensionType OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension_type

Element type.

12.70.2.24 typedef ::OpenGPS::Schemas::ISO5436_2::MatrixDimensionType OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension_type

Element type.

12.70.3 Constructor & Destructor Documentation

12.70.3.1 OpenGPS::Schemas::ISO5436_2::Record3Type::Record3Type ()

Construct an instance from the ultimate base and initializers for required elements and attributes.

Referenced by _clone().

12.70.3.2 OpenGPS::Schemas::ISO5436_2::Record3Type::Record3Type (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

References parse().

Here is the call graph for this function:

12.70.3.3 OpenGPS::Schemas::ISO5436_2::Record3Type::Record3Type (const Record3Type & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.70.3.4 OpenGPS::Schemas::ISO5436_2::Record3Type::Record3Type ()

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.70.3.5 OpenGPS::Schemas::ISO5436_2::Record3Type::Record3Type (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.70.3.6 OpenGPS::Schemas::ISO5436_2::Record3Type::Record3Type (const Record3Type & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.70.4 Member Function Documentation

12.70.4.1 virtual Record3Type* OpenGPS::Schemas::ISO5436_2::Record3Type::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.70.4.2 Record3Type * OpenGPS::Schemas::ISO5436_2::Record3Type::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References Record3Type().

Here is the call graph for this function:

12.70.4.3 void OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink (::std::auto_ptr< DataLink_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.70.4.4 void OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink (const DataLink_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.70.4.5 void OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink (const DataLink_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.70.4.6 DataLink_optional& OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.70.4.7 const DataLink_optional& OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

12.70.4.8 void OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink (::std::auto_ptr< DataLink_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References DataLink_.

12.70.4.9 void OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink (const DataLink_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

References DataLink_.

12.70.4.10 void OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink (const DataLink_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References DataLink_.

12.70.4.11 Record3Type::DataLink_optional & OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

References DataLink_.

12.70.4.12 const Record3Type::DataLink_optional & OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

References DataLink_.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.70.4.13 void OpenGPS::Schemas::ISO5436_2::Record3Type::DataList (::std::auto_ptr< DataList_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.70.4.14 void OpenGPS::Schemas::ISO5436_2::Record3Type::DataList (const DataList_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.70.4.15 void OpenGPS::Schemas::ISO5436_2::Record3Type::DataList (const DataList_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.70.4.16 DataList_optional& OpenGPS::Schemas::ISO5436_2::Record3Type::DataList ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.70.4.17 const DataList_optional& OpenGPS::Schemas::ISO5436_2::Record3Type::DataList () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

12.70.4.18 void OpenGPS::Schemas::ISO5436_2::Record3Type::DataList (::std::auto_ptr< DataList_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References DataList_.

12.70.4.19 void OpenGPS::Schemas::ISO5436_2::Record3Type::DataList (const DataList_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

References DataList_.

12.70.4.20 void OpenGPS::Schemas::ISO5436_2::Record3Type::DataList (const DataList_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References DataList_.

12.70.4.21 Record3Type::DataList_optional & OpenGPS::Schemas::ISO5436_2::Record3Type::DataList ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

References DataList_.

12.70.4.22 const Record3Type::DataList_optional & OpenGPS::Schemas::ISO5436_2::Record3Type::DataList () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

References DataList_.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.70.4.23 void OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension (const ListDimension_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.70.4.24 void OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension (const ListDimension_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.70.4.25 ListDimension_optional& OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.70.4.26 const ListDimension_optional& OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

12.70.4.27 void OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension (const ListDimension_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

References ListDimension_.

12.70.4.28 void OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension (const ListDimension_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References ListDimension_.

12.70.4.29 Record3Type::ListDimension_optional & OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

References ListDimension_.

12.70.4.30 const Record3Type::ListDimension_optional & OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

References ListDimension_.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.70.4.31 void OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension (::std::auto_ptr< MatrixDimension_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.70.4.32 void OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension (const MatrixDimension_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

12.70.4.33 void OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension (const MatrixDimension_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.70.4.34 MatrixDimension_optional& OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

12.70.4.35 const MatrixDimension_optional& OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

12.70.4.36 void OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension (::std::auto_ptr< MatrixDimension_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References MatrixDimension_.

12.70.4.37 void OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension (const MatrixDimension_optional & *x*)

Set the element value.

Parameters:

x An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

References MatrixDimension_.

12.70.4.38 void OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension (const MatrixDimension_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References MatrixDimension_.

12.70.4.39 Record3Type::MatrixDimension_optional & OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension ()

Return a read-write reference to the element container.

Returns:

A reference to the optional container.

References MatrixDimension_.

12.70.4.40 const Record3Type::MatrixDimension_optional & OpenGPS::Schemas::ISO5436_2::Record3Type::MatrixDimension () const

Return a read-only (constant) reference to the element container.

Returns:

A constant reference to the optional container.

References MatrixDimension_.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.70.4.41 void OpenGPS::Schemas::ISO5436_2::Record3Type::parse (::xsd::cxx::xml::dom::parser< wchar_t > &, ::xml_schema::flags) [protected]

12.70.4.42 void OpenGPS::Schemas::ISO5436_2::Record3Type::parse (::xsd::cxx::xml::dom::parser< wchar_t > & *p*, ::xml_schema::flags *f*) [protected]

References DataLink(), DataList(), ListDimension(), MatrixDimension(), and n.

Referenced by Record3Type().

Here is the call graph for this function:

12.70.5 Member Data Documentation

12.70.5.1 DataLink_optional OpenGPS::Schemas::ISO5436_2::Record3Type::DataLink_- [private]

Referenced by DataLink().

12.70.5.2 DataList_optional OpenGPS::Schemas::ISO5436_2::Record3Type::DataList_- [private]

Referenced by DataList().

12.70.5.3 ListDimension_optional OpenGPS::Schemas::ISO5436_-2::Record3Type::ListDimension_ [private]

Referenced by ListDimension().

12.70.5.4 MatrixDimension_optional OpenGPS::Schemas::ISO5436_-2::Record3Type::MatrixDimension_ [private]

Referenced by MatrixDimension().

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.71 OpenGPS::Schemas::ISO5436_2::Record4Type Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.71.1 Detailed Description

Class corresponding to the Record4Type schema type.

Record4 contains only the checksum of the xml file.

ChecksumFile

Accessor and modifier functions for the ChecksumFile required element.

An URI pointing to an external ascii file containting an MD5 digest with a 32 byte hexadecimal MD5Checksum of the whole XML-file and its filename as produced by the unix command "md5sum". The checksum can be calculated by the unix command "md5sum main.xml >md5checksum.hex" and checked by the command "md5sum -c md5checksum.hex". Default name of the checksum file is "md5checksum.hex".

- typedef ::xsd::cxx::tree::traits< ChecksumFile_type, wchar_t > ChecksumFile_traits

  Element traits type.

- typedef ::xml_schema::string ChecksumFile_type

  Element type.

- void ChecksumFile (::std::auto_ptr< ChecksumFile_type > p)

  Set the element value without copying.

- void ChecksumFile (const ChecksumFile_type &x)

  Set the element value.

- ChecksumFile_type & ChecksumFile ()

  Return a read-write reference to the element.

- const ChecksumFile_type & ChecksumFile () const

  Return a read-only (constant) reference to the element.

ChecksumFile

Accessor and modifier functions for the ChecksumFile required element.

An URI pointing to an external ascii file containting an MD5 digest with a 32 byte hexadecimal MD5Checksum of the whole XML-file and its filename as produced by the unix command "md5sum". The checksum can be calculated by the unix command "md5sum main.xml >md5checksum.hex" and checked by the command "md5sum -c md5checksum.hex". Default name of the checksum file is "md5checksum.hex".

- typedef ::xsd::cxx::tree::traits< ChecksumFile_type, wchar_t > ChecksumFile_traits

  Element traits type.

- typedef ::xml_schema::string ChecksumFile_type

  Element type.

- void ChecksumFile (::std::auto_ptr< ChecksumFile_type > p)

Set the element value without copying.

- void ChecksumFile (const ChecksumFile_type &x)

  Set the element value.

- ChecksumFile_type & ChecksumFile ()

  Return a read-write reference to the element.

- const ChecksumFile_type & ChecksumFile () const

  Return a read-only (constant) reference to the element.

Constructors

- virtual Record4Type * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- Record4Type (const Record4Type &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- Record4Type (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- Record4Type (const ChecksumFile_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Constructors

- virtual Record4Type * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- Record4Type (const Record4Type &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- Record4Type (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- Record4Type (const ChecksumFile_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Protected Member Functions

- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)
- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)

Private Attributes

- ::xsd::cxx::tree::one< ChecksumFile_type > ChecksumFile_

12.71.2 Member Typedef Documentation

12.71.2.1 typedef ::xsd::cxx::tree::traits< ChecksumFile_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile_traits

Element traits type.

12.71.2.2 typedef ::xsd::cxx::tree::traits< ChecksumFile_type, wchar_t > OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile_traits

Element traits type.

12.71.2.3 typedef ::xml_schema::string OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile_type

Element type.

12.71.2.4 typedef ::xml_schema::string OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile_type

Element type.

12.71.3 Constructor & Destructor Documentation

12.71.3.1 OpenGPS::Schemas::ISO5436_2::Record4Type::Record4Type (const ChecksumFile_type & *ChecksumFile*)

Construct an instance from the ultimate base and initializers for required elements and attributes.

Referenced by _clone().

12.71.3.2 OpenGPS::Schemas::ISO5436_2::Record4Type::Record4Type (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

e A DOM element to extract the data from.

f Flags to construct the new instance with.

c A pointer to the object that will contain the new instance.

References parse().

Here is the call graph for this function:

12.71.3.3 OpenGPS::Schemas::ISO5436_2::Record4Type::Record4Type (const Record4Type & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.71.3.4 OpenGPS::Schemas::ISO5436_2::Record4Type::Record4Type (const ChecksumFile_type &)

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.71.3.5 OpenGPS::Schemas::ISO5436_2::Record4Type::Record4Type (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.71.3.6 OpenGPS::Schemas::ISO5436_2::Record4Type::Record4Type (const Record4Type & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.71.4 Member Function Documentation

12.71.4.1 virtual Record4Type* OpenGPS::Schemas::ISO5436_2::Record4Type::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.71.4.2 Record4Type * OpenGPS::Schemas::ISO5436_2::Record4Type::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References Record4Type().

Here is the call graph for this function:

12.71.4.3 void OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile (::std::auto_ptr< ChecksumFile_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.71.4.4 void OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile (const ChecksumFile_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.71.4.5 ChecksumFile_type& OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.71.4.6 const ChecksumFile_type& OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.71.4.7 void OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile (::std::auto_ptr< ChecksumFile_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

References ChecksumFile_.

12.71.4.8 void OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile (const ChecksumFile_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

References ChecksumFile_.

12.71.4.9 Record4Type::ChecksumFile_type & OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile ()

Return a read-write reference to the element.

Returns:

A reference to the element.

References ChecksumFile_.

12.71.4.10 const Record4Type::ChecksumFile_type & OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

References ChecksumFile_.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.71.4.11 void OpenGPS::Schemas::ISO5436_2::Record4Type::parse (::xsd::cxx::xml::dom::parser< wchar_t > &, ::xml_schema::flags) [protected]

12.71.4.12 void OpenGPS::Schemas::ISO5436_2::Record4Type::parse (::xsd::cxx::xml::dom::parser< wchar_t > & *p*, ::xml_schema::flags *f*) [protected]

References ChecksumFile(), ChecksumFile_, and n.

Referenced by Record4Type().

Here is the call graph for this function:

12.71.5 Member Data Documentation

12.71.5.1 xsd::cxx::tree::one< ChecksumFile_type > OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile_ [private]

Referenced by ChecksumFile(), and parse().

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.72 OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.72.1 Detailed Description

Class corresponding to the RotationMatrixElementType schema type.

An element of a pure rotation matrix is limited to a value range of [-1..1].

Constructors

- virtual RotationMatrixElementType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- RotationMatrixElementType (const RotationMatrixElementType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- RotationMatrixElementType (const ::std::wstring &s, const ::xercesc::DOMElement *e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a string fragment.

- RotationMatrixElementType (const ::xercesc::DOMAttr &a,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM attribute.

- RotationMatrixElementType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- RotationMatrixElementType (const ::xml_schema::double_ &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Constructors

- virtual RotationMatrixElementType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- RotationMatrixElementType (const RotationMatrixElementType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- RotationMatrixElementType (const ::std::wstring &s, const ::xercesc::DOMElement *e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a string fragment.

- RotationMatrixElementType (const ::xercesc::DOMAttr &a,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM attribute.

- RotationMatrixElementType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- RotationMatrixElementType (const ::xml_schema::double_ &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

12.72.2 Constructor & Destructor Documentation

12.72.2.1 OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType::RotationMatrixElementType (const ::xml_schema::double_ & *double_*)

Construct an instance from the ultimate base and initializers for required elements and attributes.

Referenced by _clone().

12.72.2.2 OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType::RotationMatrixElementType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.72.2.3 OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType::RotationMatrixElementType (const ::xercesc::DOMAttr & *a*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM attribute.

Parameters:

- ***a*** A DOM attribute to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.72.2.4 OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType::RotationMatrixElementType (const ::std::wstring & *s*, const ::xercesc::DOMElement * *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a string fragment.

Parameters:

- ***s*** A string fragment to extract the data from.
- ***e*** A DOM element containing the string fragment.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.72.2.5 OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType::RotationMatrixElementType (const RotationMatrixElementType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.72.2.6 OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType::RotationMatrixElementType (const ::xml_schema::double_ &)

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.72.2.7 OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType::RotationMatrixElementType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

e A DOM element to extract the data from.

f Flags to construct the new instance with.

c A pointer to the object that will contain the new instance.

12.72.2.8 OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType::RotationMatrixElementType (const ::xercesc::DOMAttr & *a*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM attribute.

Parameters:

a A DOM attribute to extract the data from.

f Flags to construct the new instance with.

c A pointer to the object that will contain the new instance.

12.72.2.9 OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType::RotationMatrixElementType (const ::std::wstring & *s*, const ::xercesc::DOMElement * *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a string fragment.

Parameters:

s A string fragment to extract the data from.

e A DOM element containing the string fragment.

f Flags to construct the new instance with.

c A pointer to the object that will contain the new instance.

12.72.2.10 OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType::RotationMatrixElementType (const RotationMatrixElementType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.72.3 Member Function Documentation

12.72.3.1 virtual RotationMatrixElementType* OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.72.3.2 RotationMatrixElementType * OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References RotationMatrixElementType().

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.73 OpenGPS::Schemas::ISO5436_2::RotationType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.73.1 Detailed Description

Class corresponding to the RotationType schema type.

The optional transformation contains a 3D rotation matrix R with 3 by 3 elements that is used to rotate the data points in its final orientation. The full transformation consists of a rotation and a following translation that is taken from the AxisDescriptionType.Offset elements: Q = R∗P + T With Q beeing the final point, P the coordinate as specified in Record3, R the 3 by 3 rotation matrix and T the 3-element offset vector. The ∗ denotes a matrix product. The formula for the x coordinate is: Qx = r11∗Px+r12∗Py+r13∗Pz + Tx. The formula for the y coordinate is: Qx = r21∗Px+r22∗Py+r23∗Pz + Ty. The formula for the x coordinate is: Qx = r31∗Px+r32∗Py+r33∗Pz + Tz.s

r11

Accessor and modifier functions for the r11 required element.

- typedef ::xsd::cxx::tree::traits< r11_type, wchar_t > r11_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r11_type

  Element type.

- void r11 (::std::auto_ptr< r11_type > p)

  Set the element value without copying.

- void r11 (const r11_type &x)

  Set the element value.

- r11_type & r11 ()

  Return a read-write reference to the element.

- const r11_type & r11 () const

  Return a read-only (constant) reference to the element.

r11

Accessor and modifier functions for the r11 required element.

- typedef ::xsd::cxx::tree::traits< r11_type, wchar_t > r11_traits

Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r11_type
  Element type.
- void r11 (::std::auto_ptr< r11_type > p)
  Set the element value without copying.
- void r11 (const r11_type &x)
  Set the element value.
- r11_type & r11 ()
  Return a read-write reference to the element.
- const r11_type & r11 () const
  Return a read-only (constant) reference to the element.

r12

Accessor and modifier functions for the r12 required element.

- typedef ::xsd::cxx::tree::traits< r12_type, wchar_t > r12_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r12_type
  Element type.
- void r12 (::std::auto_ptr< r12_type > p)
  Set the element value without copying.
- void r12 (const r12_type &x)
  Set the element value.
- r12_type & r12 ()
  Return a read-write reference to the element.
- const r12_type & r12 () const
  Return a read-only (constant) reference to the element.

r12

Accessor and modifier functions for the r12 required element.

- typedef ::xsd::cxx::tree::traits< r12_type, wchar_t > r12_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r12_type

Element type.

- void r12 (::std::auto_ptr< r12_type > p)

  Set the element value without copying.

- void r12 (const r12_type &x)

  Set the element value.

- r12_type & r12 ()

  Return a read-write reference to the element.

- const r12_type & r12 () const

  Return a read-only (constant) reference to the element.

r13

Accessor and modifier functions for the r13 required element.

- typedef ::xsd::cxx::tree::traits< r13_type, wchar_t > r13_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r13_type

  Element type.

- void r13 (::std::auto_ptr< r13_type > p)

  Set the element value without copying.

- void r13 (const r13_type &x)

  Set the element value.

- r13_type & r13 ()

  Return a read-write reference to the element.

- const r13_type & r13 () const

  Return a read-only (constant) reference to the element.

r13

Accessor and modifier functions for the r13 required element.

- typedef ::xsd::cxx::tree::traits< r13_type, wchar_t > r13_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r13_type

  Element type.

- void r13 (::std::auto_ptr< r13_type > p)

Set the element value without copying.

- void r13 (const r13_type &x)

  Set the element value.

- r13_type & r13 ()

  Return a read-write reference to the element.

- const r13_type & r13 () const

  Return a read-only (constant) reference to the element.

r21

Accessor and modifier functions for the r21 required element.

- typedef ::xsd::cxx::tree::traits< r21_type, wchar_t > r21_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r21_type

  Element type.

- void r21 (::std::auto_ptr< r21_type > p)

  Set the element value without copying.

- void r21 (const r21_type &x)

  Set the element value.

- r21_type & r21 ()

  Return a read-write reference to the element.

- const r21_type & r21 () const

  Return a read-only (constant) reference to the element.

r21

Accessor and modifier functions for the r21 required element.

- typedef ::xsd::cxx::tree::traits< r21_type, wchar_t > r21_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r21_type

  Element type.

- void r21 (::std::auto_ptr< r21_type > p)

  Set the element value without copying.

- void r21 (const r21_type &x)

Set the element value.

- r21_type & r21 ()

  Return a read-write reference to the element.

- const r21_type & r21 () const

  Return a read-only (constant) reference to the element.

r22

Accessor and modifier functions for the r22 required element.

- typedef ::xsd::cxx::tree::traits< r22_type, wchar_t > r22_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r22_type

  Element type.

- void r22 (::std::auto_ptr< r22_type > p)

  Set the element value without copying.

- void r22 (const r22_type &x)

  Set the element value.

- r22_type & r22 ()

  Return a read-write reference to the element.

- const r22_type & r22 () const

  Return a read-only (constant) reference to the element.

r22

Accessor and modifier functions for the r22 required element.

- typedef ::xsd::cxx::tree::traits< r22_type, wchar_t > r22_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r22_type

  Element type.

- void r22 (::std::auto_ptr< r22_type > p)

  Set the element value without copying.

- void r22 (const r22_type &x)

  Set the element value.

- r22_type & r22 ()

Return a read-write reference to the element.

- const r22_type & r22 () const
  Return a read-only (constant) reference to the element.

r23

Accessor and modifier functions for the r23 required element.

- typedef ::xsd::cxx::tree::traits< r23_type, wchar_t > r23_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r23_type
  Element type.
- void r23 (::std::auto_ptr< r23_type > p)
  Set the element value without copying.
- void r23 (const r23_type &x)
  Set the element value.
- r23_type & r23 ()
  Return a read-write reference to the element.
- const r23_type & r23 () const
  Return a read-only (constant) reference to the element.

r23

Accessor and modifier functions for the r23 required element.

- typedef ::xsd::cxx::tree::traits< r23_type, wchar_t > r23_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r23_type
  Element type.
- void r23 (::std::auto_ptr< r23_type > p)
  Set the element value without copying.
- void r23 (const r23_type &x)
  Set the element value.
- r23_type & r23 ()
  Return a read-write reference to the element.
- const r23_type & r23 () const
  Return a read-only (constant) reference to the element.

r31

Accessor and modifier functions for the r31 required element.

- typedef ::xsd::cxx::tree::traits< r31_type, wchar_t > r31_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r31_type
  Element type.
- void r31 (::std::auto_ptr< r31_type > p)
  Set the element value without copying.
- void r31 (const r31_type &x)
  Set the element value.
- r31_type & r31 ()
  Return a read-write reference to the element.
- const r31_type & r31 () const
  Return a read-only (constant) reference to the element.

r31

Accessor and modifier functions for the r31 required element.

- typedef ::xsd::cxx::tree::traits< r31_type, wchar_t > r31_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r31_type
  Element type.
- void r31 (::std::auto_ptr< r31_type > p)
  Set the element value without copying.
- void r31 (const r31_type &x)
  Set the element value.
- r31_type & r31 ()
  Return a read-write reference to the element.
- const r31_type & r31 () const
  Return a read-only (constant) reference to the element.

r32

Accessor and modifier functions for the r32 required element.

- typedef ::xsd::cxx::tree::traits< r32_type, wchar_t > r32_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r32_type

  Element type.

- void r32 (::std::auto_ptr< r32_type > p)

  Set the element value without copying.

- void r32 (const r32_type &x)

  Set the element value.

- r32_type & r32 ()

  Return a read-write reference to the element.

- const r32_type & r32 () const

  Return a read-only (constant) reference to the element.

r32

Accessor and modifier functions for the r32 required element.

- typedef ::xsd::cxx::tree::traits< r32_type, wchar_t > r32_traits

  Element traits type.

- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r32_type

  Element type.

- void r32 (::std::auto_ptr< r32_type > p)

  Set the element value without copying.

- void r32 (const r32_type &x)

  Set the element value.

- r32_type & r32 ()

  Return a read-write reference to the element.

- const r32_type & r32 () const

  Return a read-only (constant) reference to the element.

r33

Accessor and modifier functions for the r33 required element.

- typedef ::xsd::cxx::tree::traits< r33_type, wchar_t > r33_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r33_type
  Element type.
- void r33 (::std::auto_ptr< r33_type > p)
  Set the element value without copying.
- void r33 (const r33_type &x)
  Set the element value.
- r33_type & r33 ()
  Return a read-write reference to the element.
- const r33_type & r33 () const
  Return a read-only (constant) reference to the element.

r33

Accessor and modifier functions for the r33 required element.

- typedef ::xsd::cxx::tree::traits< r33_type, wchar_t > r33_traits
  Element traits type.
- typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType r33_type
  Element type.
- void r33 (::std::auto_ptr< r33_type > p)
  Set the element value without copying.
- void r33 (const r33_type &x)
  Set the element value.
- r33_type & r33 ()
  Return a read-write reference to the element.
- const r33_type & r33 () const
  Return a read-only (constant) reference to the element.

Constructors

- virtual RotationType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- RotationType (const RotationType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- RotationType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- RotationType (const r11_type &, const r12_type &, const r13_type &, const r21_type &, const r22_-type &, const r23_type &, const r31_type &, const r32_type &, const r33_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Constructors

- virtual RotationType * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- RotationType (const RotationType &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- RotationType (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- RotationType (const r11_type &, const r12_type &, const r13_type &, const r21_type &, const r22_-type &, const r23_type &, const r31_type &, const r32_type &, const r33_type &)

  Construct an instance from the ultimate base and initializers for required elements and attributes.

Protected Member Functions

- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)
- void parse (::xsd::cxx::xml::dom::parser< wchar_t > &,::xml_schema::flags)

Private Attributes

- ::xsd::cxx::tree::one< r11_type > r11_
- ::xsd::cxx::tree::one< r12_type > r12_
- ::xsd::cxx::tree::one< r13_type > r13_
- ::xsd::cxx::tree::one< r21_type > r21_
- ::xsd::cxx::tree::one< r22_type > r22_
- ::xsd::cxx::tree::one< r23_type > r23_
- ::xsd::cxx::tree::one< r31_type > r31_
- ::xsd::cxx::tree::one< r32_type > r32_
- ::xsd::cxx::tree::one< r33_type > r33_

12.73.2 Member Typedef Documentation

12.73.2.1 typedef ::xsd::cxx::tree::traits< r11_type, wchar_t > OpenGPS::Schemas::ISO5436_2::RotationType::r11_traits

Element traits type.

12.73.2.2 typedef ::xsd::cxx::tree::traits< r11_type, wchar_t > OpenGPS::Schemas::ISO5436_2::RotationType::r11_traits

Element traits type.

12.73.2.3 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r11_type

Element type.

12.73.2.4 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r11_type

Element type.

12.73.2.5 typedef ::xsd::cxx::tree::traits< r12_type, wchar_t > OpenGPS::Schemas::ISO5436_2::RotationType::r12_traits

Element traits type.

12.73.2.6 typedef ::xsd::cxx::tree::traits< r12_type, wchar_t > OpenGPS::Schemas::ISO5436_2::RotationType::r12_traits

Element traits type.

12.73.2.7 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r12_type

Element type.

12.73.2.8 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r12_type

Element type.

12.73.2.9 typedef ::xsd::cxx::tree::traits< r13_type, wchar_t > OpenGPS::Schemas::ISO5436_2::RotationType::r13_traits

Element traits type.

12.73.2.10 typedef ::xsd::cxx::tree::traits< r13_type, wchar_t > OpenGPS::Schemas::ISO5436_2::RotationType::r13_traits

Element traits type.

12.73.2.11 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r13_type

Element type.

12.73.2.12 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r13_type

Element type.

12.73.2.13 typedef ::xsd::cxx::tree::traits< r21_type, wchar_t > OpenGPS::Schemas::ISO5436_-2::RotationType::r21_traits

Element traits type.

12.73.2.14 typedef ::xsd::cxx::tree::traits< r21_type, wchar_t > OpenGPS::Schemas::ISO5436_-2::RotationType::r21_traits

Element traits type.

12.73.2.15 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r21_type

Element type.

12.73.2.16 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r21_type

Element type.

12.73.2.17 typedef ::xsd::cxx::tree::traits< r22_type, wchar_t > OpenGPS::Schemas::ISO5436_-2::RotationType::r22_traits

Element traits type.

12.73.2.18 typedef ::xsd::cxx::tree::traits< r22_type, wchar_t > OpenGPS::Schemas::ISO5436_-2::RotationType::r22_traits

Element traits type.

12.73.2.19 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r22_type

Element type.

12.73.2.20 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r22_type

Element type.

12.73.2.21 typedef ::xsd::cxx::tree::traits< r23_type, wchar_t > OpenGPS::Schemas::ISO5436_-2::RotationType::r23_traits

Element traits type.

12.73.2.22 typedef ::xsd::cxx::tree::traits< r23_type, wchar_t > OpenGPS::Schemas::ISO5436_-2::RotationType::r23_traits

Element traits type.

12.73.2.23 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r23_type

Element type.

12.73.2.24 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r23_type

Element type.

12.73.2.25 typedef ::xsd::cxx::tree::traits< r31_type, wchar_t > OpenGPS::Schemas::ISO5436_-2::RotationType::r31_traits

Element traits type.

12.73.2.26 typedef ::xsd::cxx::tree::traits< r31_type, wchar_t > OpenGPS::Schemas::ISO5436_-2::RotationType::r31_traits

Element traits type.

12.73.2.27 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r31_type

Element type.

12.73.2.28 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r31_type

Element type.

12.73.2.29 typedef ::xsd::cxx::tree::traits< r32_type, wchar_t > OpenGPS::Schemas::ISO5436_-2::RotationType::r32_traits

Element traits type.

12.73.2.30 typedef ::xsd::cxx::tree::traits< r32_type, wchar_t > OpenGPS::Schemas::ISO5436_-2::RotationType::r32_traits

Element traits type.

12.73.2.31 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r32_type

Element type.

12.73.2.32 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r32_type

Element type.

12.73.2.33 typedef ::xsd::cxx::tree::traits< r33_type, wchar_t > OpenGPS::Schemas::ISO5436_2::RotationType::r33_traits

Element traits type.

12.73.2.34 typedef ::xsd::cxx::tree::traits< r33_type, wchar_t > OpenGPS::Schemas::ISO5436_2::RotationType::r33_traits

Element traits type.

12.73.2.35 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r33_type

Element type.

12.73.2.36 typedef ::OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_2::RotationType::r33_type

Element type.

12.73.3 Constructor & Destructor Documentation

12.73.3.1 OpenGPS::Schemas::ISO5436_2::RotationType::RotationType (const r11_type & *r11*, const r12_type & *r12*, const r13_type & *r13*, const r21_type & *r21*, const r22_type & *r22*, const r23_type & *r23*, const r31_type & *r31*, const r32_type & *r32*, const r33_type & *r33*)

Construct an instance from the ultimate base and initializers for required elements and attributes.

Referenced by _clone().

12.73.3.2 OpenGPS::Schemas::ISO5436_2::RotationType::RotationType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

References parse().

Here is the call graph for this function:

12.73.3.3 OpenGPS::Schemas::ISO5436_2::RotationType::RotationType (const RotationType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.73.3.4 OpenGPS::Schemas::ISO5436_2::RotationType::RotationType (const r11_type &, const r12_type &, const r13_type &, const r21_type &, const r22_type &, const r23_type &, const r31_type &, const r32_type &, const r33_type &)

Construct an instance from the ultimate base and initializers for required elements and attributes.

12.73.3.5 OpenGPS::Schemas::ISO5436_2::RotationType::RotationType (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.73.3.6 OpenGPS::Schemas::ISO5436_2::RotationType::RotationType (const RotationType & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.73.4 Member Function Documentation

12.73.4.1 virtual RotationType* OpenGPS::Schemas::ISO5436_2::RotationType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.73.4.2 RotationType * OpenGPS::Schemas::ISO5436_2::RotationType::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References RotationType().

Here is the call graph for this function:

12.73.4.3 void OpenGPS::Schemas::ISO5436_2::RotationType::parse (::xsd::cxx::xml::dom::parser< wchar_t > &, ::xml_schema::flags) [protected]

12.73.4.4 void OpenGPS::Schemas::ISO5436_2::RotationType::parse (::xsd::cxx::xml::dom::parser< wchar_t > & *p*, ::xml_schema::flags *f*) [protected]

References n, r11(), r11_, r12(), r12_, r13(), r13_, r21(), r21_, r22(), r22_, r23(), r23_, r31(), r31_, r32(), r32_, r33(), and r33_.

Referenced by RotationType().

Here is the call graph for this function:

12.73.4.5 void OpenGPS::Schemas::ISO5436_2::RotationType::r11 (::std::auto_ptr< r11_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.6 void OpenGPS::Schemas::ISO5436_2::RotationType::r11 (const r11_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.7 r11_type& OpenGPS::Schemas::ISO5436_2::RotationType::r11 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.8 const r11_type& OpenGPS::Schemas::ISO5436_2::RotationType::r11 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.73.4.9 void OpenGPS::Schemas::ISO5436_2::RotationType::r11 (::std::auto_ptr< r11_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.10 void OpenGPS::Schemas::ISO5436_2::RotationType::r11 (const r11_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.11 r11_type& OpenGPS::Schemas::ISO5436_2::RotationType::r11 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.12 const r11_type& OpenGPS::Schemas::ISO5436_2::RotationType::r11 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.73.4.13 void OpenGPS::Schemas::ISO5436_2::RotationType::r12 (::std::auto_ptr< r12_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.14 void OpenGPS::Schemas::ISO5436_2::RotationType::r12 (const r12_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.15 r12_type& OpenGPS::Schemas::ISO5436_2::RotationType::r12 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.16 const r12_type& OpenGPS::Schemas::ISO5436_2::RotationType::r12 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.73.4.17 void OpenGPS::Schemas::ISO5436_2::RotationType::r12 (::std::auto_ptr< r12_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.18 void OpenGPS::Schemas::ISO5436_2::RotationType::r12 (const r12_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.19 r12_type& OpenGPS::Schemas::ISO5436_2::RotationType::r12 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.20 const r12_type& OpenGPS::Schemas::ISO5436_2::RotationType::r12 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_-2::operator==(), and parse().

12.73.4.21 void OpenGPS::Schemas::ISO5436_2::RotationType::r13 (::std::auto_ptr< r13_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.22 void OpenGPS::Schemas::ISO5436_2::RotationType::r13 (const r13_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.23 r13_type& OpenGPS::Schemas::ISO5436_2::RotationType::r13 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.24 const r13_type& OpenGPS::Schemas::ISO5436_2::RotationType::r13 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.73.4.25 void OpenGPS::Schemas::ISO5436_2::RotationType::r13 (::std::auto_ptr< r13_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.26 void OpenGPS::Schemas::ISO5436_2::RotationType::r13 (const r13_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.27 r13_type& OpenGPS::Schemas::ISO5436_2::RotationType::r13 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.28 const r13_type& OpenGPS::Schemas::ISO5436_2::RotationType::r13 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_-2::operator==(), and parse().

12.73.4.29 void OpenGPS::Schemas::ISO5436_2::RotationType::r21 (::std::auto_ptr< r21_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.30 void OpenGPS::Schemas::ISO5436_2::RotationType::r21 (const r21_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.31 r21_type& OpenGPS::Schemas::ISO5436_2::RotationType::r21 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.32 const r21_type& OpenGPS::Schemas::ISO5436_2::RotationType::r21 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.73.4.33 void OpenGPS::Schemas::ISO5436_2::RotationType::r21 (::std::auto_ptr< r21_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.34 void OpenGPS::Schemas::ISO5436_2::RotationType::r21 (const r21_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.35 r21_type& OpenGPS::Schemas::ISO5436_2::RotationType::r21 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.36 const r21_type& OpenGPS::Schemas::ISO5436_2::RotationType::r21 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.73.4.37 void OpenGPS::Schemas::ISO5436_2::RotationType::r22 (::std::auto_ptr< r22_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.38 void OpenGPS::Schemas::ISO5436_2::RotationType::r22 (const r22_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.39 r22_type& OpenGPS::Schemas::ISO5436_2::RotationType::r22 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.40 const r22_type& OpenGPS::Schemas::ISO5436_2::RotationType::r22 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.73.4.41 void OpenGPS::Schemas::ISO5436_2::RotationType::r22 (::std::auto_ptr< r22_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.42 void OpenGPS::Schemas::ISO5436_2::RotationType::r22 (const r22_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.43 r22_type& OpenGPS::Schemas::ISO5436_2::RotationType::r22 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.44 const r22_type& OpenGPS::Schemas::ISO5436_2::RotationType::r22 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_-2::operator==(), and parse().

12.73.4.45 void OpenGPS::Schemas::ISO5436_2::RotationType::r23 (::std::auto_ptr< r23_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.46 void OpenGPS::Schemas::ISO5436_2::RotationType::r23 (const r23_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.47 r23_type& OpenGPS::Schemas::ISO5436_2::RotationType::r23 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.48 const r23_type& OpenGPS::Schemas::ISO5436_2::RotationType::r23 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.73.4.49 void OpenGPS::Schemas::ISO5436_2::RotationType::r23 (::std::auto_ptr< r23_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.50 void OpenGPS::Schemas::ISO5436_2::RotationType::r23 (const r23_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.51 r23_type& OpenGPS::Schemas::ISO5436_2::RotationType::r23 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.52 const r23_type& OpenGPS::Schemas::ISO5436_2::RotationType::r23 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.73.4.53 void OpenGPS::Schemas::ISO5436_2::RotationType::r31 (::std::auto_ptr< r31_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.54 void OpenGPS::Schemas::ISO5436_2::RotationType::r31 (const r31_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.55 r31_type& OpenGPS::Schemas::ISO5436_2::RotationType::r31 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.56 const r31_type& OpenGPS::Schemas::ISO5436_2::RotationType::r31 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.73.4.57 void OpenGPS::Schemas::ISO5436_2::RotationType::r31 (::std::auto_ptr< r31_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.58 void OpenGPS::Schemas::ISO5436_2::RotationType::r31 (const r31_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.59 r31_type& OpenGPS::Schemas::ISO5436_2::RotationType::r31 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.60 const r31_type& OpenGPS::Schemas::ISO5436_2::RotationType::r31 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.73.4.61 void OpenGPS::Schemas::ISO5436_2::RotationType::r32 (::std::auto_ptr< r32_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.62 void OpenGPS::Schemas::ISO5436_2::RotationType::r32 (const r32_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.63 r32_type& OpenGPS::Schemas::ISO5436_2::RotationType::r32 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.64 const r32_type& OpenGPS::Schemas::ISO5436_2::RotationType::r32 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.73.4.65 void OpenGPS::Schemas::ISO5436_2::RotationType::r32 (::std::auto_ptr< r32_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.66 void OpenGPS::Schemas::ISO5436_2::RotationType::r32 (const r32_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.67 r32_type& OpenGPS::Schemas::ISO5436_2::RotationType::r32 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.68 const r32_type& OpenGPS::Schemas::ISO5436_2::RotationType::r32 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_-2::operator==(), and parse().

12.73.4.69 void OpenGPS::Schemas::ISO5436_2::RotationType::r33 (::std::auto_ptr< r33_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.70 void OpenGPS::Schemas::ISO5436_2::RotationType::r33 (const r33_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.71 r33_type& OpenGPS::Schemas::ISO5436_2::RotationType::r33 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.72 const r33_type& OpenGPS::Schemas::ISO5436_2::RotationType::r33 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

12.73.4.73 void OpenGPS::Schemas::ISO5436_2::RotationType::r33 (::std::auto_ptr< r33_type > *p*)

Set the element value without copying.

Parameters:

p A new value to use.

This function will try to use the passed value directly instead of making a copy.

12.73.4.74 void OpenGPS::Schemas::ISO5436_2::RotationType::r33 (const r33_type & *x*)

Set the element value.

Parameters:

x A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

12.73.4.75 r33_type& OpenGPS::Schemas::ISO5436_2::RotationType::r33 ()

Return a read-write reference to the element.

Returns:

A reference to the element.

12.73.4.76 const r33_type& OpenGPS::Schemas::ISO5436_2::RotationType::r33 () const

Return a read-only (constant) reference to the element.

Returns:

A constant reference to the element.

Referenced by OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::operator==(), and parse().

12.73.5 Member Data Documentation

12.73.5.1 xsd::cxx::tree::one< r11_type > OpenGPS::Schemas::ISO5436_2::RotationType::r11_ [private]

Referenced by parse().

12.73.5.2 xsd::cxx::tree::one< r12_type > OpenGPS::Schemas::ISO5436_2::RotationType::r12_- [private]

Referenced by parse().

12.73.5.3 xsd::cxx::tree::one< r13_type > OpenGPS::Schemas::ISO5436_2::RotationType::r13_- [private]

Referenced by parse().

12.73.5.4 xsd::cxx::tree::one< r21_type > OpenGPS::Schemas::ISO5436_2::RotationType::r21_- [private]

Referenced by parse().

12.73.5.5 xsd::cxx::tree::one< r22_type > OpenGPS::Schemas::ISO5436_2::RotationType::r22_- [private]

Referenced by parse().

12.73.5.6 xsd::cxx::tree::one< r23_type > OpenGPS::Schemas::ISO5436_2::RotationType::r23_- [private]

Referenced by parse().

12.73.5.7 xsd::cxx::tree::one< r31_type > OpenGPS::Schemas::ISO5436_2::RotationType::r31_- [private]

Referenced by parse().

12.73.5.8 xsd::cxx::tree::one< r32_type > OpenGPS::Schemas::ISO5436_2::RotationType::r32_- [private]

Referenced by parse().

12.73.5.9 xsd::cxx::tree::one< r33_type > OpenGPS::Schemas::ISO5436_2::RotationType::r33_- [private]

Referenced by parse().

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.74 OpenGPS::String Class Reference

```
#include <string.hxx>
```

Inheritance diagram for OpenGPS::String:

Collaboration diagram for OpenGPS::String:

12.74.1 Detailed Description

Stores an OGPS_Character sequence.

Todo

Add a streaming input and output operator

Public Types

- typedef OGPS_Character ElementType

  The OGPS_Character type.

Public Member Functions

- OGPS_Boolean ConvertFromMd5 (const OpenGPS::UnsignedByte md5[16])

  Stores an MD5 sum as a character sequence in hexadecimal format.

- OGPS_Boolean ConvertToMd5 (OpenGPS::UnsignedByte md5[16]) const

  Converts the current character squence representing a 128-Bit MD5 sum in hexadecimal format to the binary representation of that MD5 sum.

- size_t CopyTo (OGPS_Character *const target, const size_t size) const

  Copies the current character sequence to an external buffer.

- void FromChar (const char *const s)

  Stores an ANSI char squence.

- String (const OGPS_Character *const s)
  Creates a new instance.
- String (const BaseType &s)
  Creates a new instance.
- String ()
  Creates a new instance.
- const char * ToChar ()
  Converts the unicode character sequence to ANSI char.
- ~String ()
  Destructs an object.

Private Types

- typedef std::string BaseType
  The std::string<::OGPS_Character> type.

12.74.2 Member Typedef Documentation

12.74.2.1 typedef std::string OpenGPS::String::BaseType [private]

The std::string<::OGPS_Character> type.

12.74.2.2 typedef OGPS_Character OpenGPS::String::ElementType

The OGPS_Character type.

12.74.3 Constructor & Destructor Documentation

12.74.3.1 String::String ()

Creates a new instance.

References NULL.

12.74.3.2 String::String (const BaseType & *s*)

Creates a new instance.

Parameters:

s Initialize the newly created object with the given character sequence.

References NULL.

12.74.3.3 String::String (const OGPS_Character *const *s*)

Creates a new instance.

Parameters:

s Initialize the newly created object with the given character sequence.

References NULL.

12.74.3.4 String::~String ()

Destructs an object.

References _OPENGPS_DELETE_ARRAY.

12.74.4 Member Function Documentation

12.74.4.1 OGPS_Boolean String::ConvertFromMd5 (const OpenGPS::UnsignedByte *md5*[16])

Stores an MD5 sum as a character sequence in hexadecimal format.

Parameters:

md5 The 128-Bit MD5 value to be storedin hexadecimal format.

Returns:

Returns TRUE on success, FALSE otherwise.

References FALSE, and TRUE.

Referenced by OpenGPS::ISO5436_2Container::SaveChecksumFile().

12.74.4.2 OGPS_Boolean String::ConvertToMd5 (OpenGPS::UnsignedByte *md5*[16]) const

Converts the current character squence representing a 128-Bit MD5 sum in hexadecimal format to the binary representation of that MD5 sum.

Parameters:

md5 Gets the converted MD5 binary values.

Returns:

Returns TRUE on success, FALSE otherwise.

References _ASSERT, FALSE, and n.

12.74.4.3 size_t String::CopyTo (OGPS_Character *const *target*, const size_t *size*) const

Copies the current character sequence to an external buffer.

Parameters:

target The external buffer.

size The size of the external buffer in characters.

Returns:

On success returns the number of chracters copied to the target buffer without the terminating null character. On failure returns the number of characters the target buffer must be able to store.

References _ASSERT.

12.74.4.4 void String::FromChar (const char *const *s*)

Stores an ANSI char squence.

Parameters:

s An ANSI char squence to store as unicode internally.

References _OPENGPS_DELETE_ARRAY.

Referenced by OpenGPS::Exception::Exception(), and OpenGPS::ExceptionHistory::SetLastException().

12.74.4.5 const char * String::ToChar ()

Converts the unicode character sequence to ANSI char.

Returns:

An ANSI char pointer or NULL.

References _OPENGPS_DELETE_ARRAY.

Referenced by _tmain(), OpenGPS::ISO5436_2Container::Compress(), OpenGPS::ISO5436_2Container::Decompress(), OpenGPS::ExceptionHistory::DumpIt(), OpenGPS::InputBinaryFileStream::InputBinaryFileStream(), OpenGPS::OutputBinaryFileStream::OutputBinaryFileStream(), OpenGPS::Info::PrintCopyright(), OpenGPS::Info::PrintLicense(), OpenGPS::Info::PrintVersion(), readonlyExample(), readonlyExample2(), readonlyExample3(), readonlyExample4(), readonlyExampleMatrix(), OpenGPS::ISO5436_2Container::SaveChecksumFile(), OpenGPS::ISO5436_2Container::SavePointBuffer(), OpenGPS::ISO5436_2Container::SaveValidPointsLink(), OpenGPS::ISO5436_2Container::SaveXmlDocument(), simpleExample(), OpenGPS::ISO5436_2Container::VerifyChecksum(), and OpenGPS::ISO5436_2Container::WriteVendorSpecific().

The documentation for this class was generated from the following files:

- string.hxx
- string.cxx

12.75 OpenGPS::Schemas::ISO5436_2::Type Class Reference

```
#include <iso5436_2_xsd.hxx>
```

12.75.1 Detailed Description

Enumeration class corresponding to the Type schema type.

Public Types

- enum value { Contacting, NonContacting, Contacting, NonContacting }

  Underlying enum type.

- enum value { Contacting, NonContacting, Contacting, NonContacting }

  Underlying enum type.

Public Member Functions

- virtual Type * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- virtual Type * _clone (::xml_schema::flags f=0,::xml_schema::type *c=0) const

  Copy the object polymorphically.

- virtual operator value () const

  Implicit conversion operator to the underlying enum value.

- virtual operator value () const

  Implicit conversion operator to the underlying enum value.

- Type & operator= (value v)

  Assign the underlying enum value.

- Type & operator= (value v)

  Assign the underlying enum value.

- Type (const Type &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- Type (const ::std::wstring &s, const ::xercesc::DOMElement *e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a string fragment.

- Type (const ::xercesc::DOMAttr &a,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM attribute.

- Type (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- Type (const ::xml_schema::token &v)

  Construct an instance from the base value.

- Type (value v)

  Construct an instance from the underlying enum value.

- Type (const Type &x,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Copy constructor.

- Type (const ::std::wstring &s, const ::xercesc::DOMElement *e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a string fragment.

- Type (const ::xercesc::DOMAttr &a,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM attribute.

- Type (const ::xercesc::DOMElement &e,::xml_schema::flags f=0,::xml_schema::type *c=0)

  Construct an instance from a DOM element.

- Type (const ::xml_schema::token &v)

  Construct an instance from the base value.

- Type (value v)

  Construct an instance from the underlying enum value.

Static Public Attributes

- static const value _xsd_Type_indexes_ [2]
- static const wchar_t *const _xsd_Type_literals_ [2]

Protected Member Functions

- value _xsd_Type_convert () const
- value _xsd_Type_convert () const

12.75.2 Member Enumeration Documentation

12.75.2.1 enum OpenGPS::Schemas::ISO5436_2::Type::value

Underlying enum type.

Enumerator:

Contacting
NonContacting
Contacting
NonContacting

12.75.2.2 enum OpenGPS::Schemas::ISO5436_2::Type::value

Underlying enum type.

Enumerator:

Contacting
NonContacting
Contacting
NonContacting

12.75.3 Constructor & Destructor Documentation

12.75.3.1 OpenGPS::Schemas::ISO5436_2::Type::Type (value *v*)

Construct an instance from the underlying enum value.

Parameters:

v A enum value.

Referenced by _clone().

12.75.3.2 OpenGPS::Schemas::ISO5436_2::Type::Type (const ::xml_schema::token & *v*)

Construct an instance from the base value.

Parameters:

v A base value.

12.75.3.3 OpenGPS::Schemas::ISO5436_2::Type::Type (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

e A DOM element to extract the data from.
f Flags to construct the new instance with.
c A pointer to the object that will contain the new instance.

References _xsd_Type_convert().

Here is the call graph for this function:

12.75.3.4 OpenGPS::Schemas::ISO5436_2::Type::Type (const ::xercesc::DOMAttr & *a*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM attribute.

Parameters:

a A DOM attribute to extract the data from.
f Flags to construct the new instance with.
c A pointer to the object that will contain the new instance.

References _xsd_Type_convert().

Here is the call graph for this function:

12.75.3.5 OpenGPS::Schemas::ISO5436_2::Type::Type (const ::std::wstring & *s*, const ::xercesc::DOMElement ∗ *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type ∗ *c* = 0)

Construct an instance from a string fragment.

Parameters:

- ***s*** A string fragment to extract the data from.
- ***e*** A DOM element containing the string fragment.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

References _xsd_Type_convert().

Here is the call graph for this function:

12.75.3.6 OpenGPS::Schemas::ISO5436_2::Type::Type (const Type & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type ∗ *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.75.3.7 OpenGPS::Schemas::ISO5436_2::Type::Type (value *v*)

Construct an instance from the underlying enum value.

Parameters:

- ***v*** A enum value.

12.75.3.8 OpenGPS::Schemas::ISO5436_2::Type::Type (const ::xml_schema::token & *v*)

Construct an instance from the base value.

Parameters:

- ***v*** A base value.

12.75.3.9 OpenGPS::Schemas::ISO5436_2::Type::Type (const ::xercesc::DOMElement & *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM element.

Parameters:

- ***e*** A DOM element to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.75.3.10 OpenGPS::Schemas::ISO5436_2::Type::Type (const ::xercesc::DOMAttr & *a*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a DOM attribute.

Parameters:

- ***a*** A DOM attribute to extract the data from.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.75.3.11 OpenGPS::Schemas::ISO5436_2::Type::Type (const ::std::wstring & *s*, const ::xercesc::DOMElement * *e*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Construct an instance from a string fragment.

Parameters:

- ***s*** A string fragment to extract the data from.
- ***e*** A DOM element containing the string fragment.
- ***f*** Flags to construct the new instance with.
- ***c*** A pointer to the object that will contain the new instance.

12.75.3.12 OpenGPS::Schemas::ISO5436_2::Type::Type (const Type & *x*, ::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0)

Copy constructor.

Parameters:

- ***x*** An instance to make a copy of.
- ***f*** Flags to construct the copy with.
- ***c*** A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

12.75.4 Member Function Documentation

12.75.4.1 virtual Type* OpenGPS::Schemas::ISO5436_2::Type::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

12.75.4.2 Type * OpenGPS::Schemas::ISO5436_2::Type::_clone (::xml_schema::flags *f* = 0, ::xml_schema::type * *c* = 0) const [virtual]

Copy the object polymorphically.

Parameters:

f Flags to construct the copy with.

c A pointer to the object that will contain the copy.

Returns:

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of an instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

References Type().

Here is the call graph for this function:

12.75.4.3 value OpenGPS::Schemas::ISO5436_2::Type::_xsd_Type_convert () const [protected]

12.75.4.4 Type::value OpenGPS::Schemas::ISO5436_2::Type::_xsd_Type_convert () const [protected]

References _xsd_Type_indexes_, and _xsd_Type_literals_.

Referenced by Type().

12.75.4.5 virtual OpenGPS::Schemas::ISO5436_2::Type::operator value () const [inline, virtual]

Implicit conversion operator to the underlying enum value.

Returns:

A enum value.

12.75.4.6 virtual OpenGPS::Schemas::ISO5436_2::Type::operator value () const [inline, virtual]

Implicit conversion operator to the underlying enum value.

Returns:

A enum value.

12.75.4.7 Type& OpenGPS::Schemas::ISO5436_2::Type::operator= (value *v*)

Assign the underlying enum value.

Parameters:

v A enum value.

Returns:

A refernce to the instance.

12.75.4.8 Type& OpenGPS::Schemas::ISO5436_2::Type::operator= (value *v*)

Assign the underlying enum value.

Parameters:

v A enum value.

Returns:

A refernce to the instance.

12.75.5 Member Data Documentation

12.75.5.1 static const value OpenGPS::Schemas::ISO5436_2::Type::_xsd_Type_indexes_- [static]

Referenced by _xsd_Type_convert().

12.75.5.2 static const wchar_t *const OpenGPS::Schemas::ISO5436_2::Type::_xsd_Type_-literals_ [static]

Referenced by _xsd_Type_convert(), and OpenGPS::Schemas::ISO5436_2::operator<<().

The documentation for this class was generated from the following files:

- include/opengps/cxx/iso5436_2_xsd.hxx
- src/ISO5436_2_XML/iso5436_2_xsd.hxx
- iso5436_2_xsd.cxx

12.76 OpenGPS::ValidBuffer Class Reference

```
#include <valid_buffer.hxx>
```

Inheritance diagram for OpenGPS::ValidBuffer:

Collaboration diagram for OpenGPS::ValidBuffer:

12.76.1 Detailed Description

Implements the OpenGPS::PointValidityProvider as an external binary file.

All bit values of the binary file correspond to the location of the point vector of the same index. If the file bit is on (set to one) the point vector is valid, if the bit at the given index is off, then the point vector at the corresponding location has invalid data.

Public Member Functions

- virtual OGPS_Boolean HasInvalidMarks () const

  Checks whether the current instance serves any point data that is marked as invalid.

- virtual OGPS_Boolean IsAllocated () const

  Returns wheter the bit buffer has already been allocated.

- virtual OGPS_Boolean IsValid (const unsigned int index) const throw (...)

  Gets the validity of a point vector at a given location.

- virtual void Read (std::basic_istream< OpenGPS::UnsignedByte > &stream) throw (...)

  Maps the bit buffer from a binary stream.

- virtual void SetValid (const unsigned int index, const OGPS_Boolean value) throw (...)

  Sets the validity of a point vector at a given location.

- virtual void Write (std::ostream &stream) throw (...)

  Maps the bit buffer to a binary stream.

- virtual ~ValidBuffer ()

  Destroys this instance.

Protected Member Functions

- virtual void Allocate () throw (...)

  Allocates the internal bit array.

- PointBuffer * GetPointBuffer ()

  Returns the point buffer of the Z axis.

- const PointBuffer * GetPointBuffer () const

  Returns the point buffer of the Z axis.

- virtual void Reset ()

  Frees allocated resources.

- ValidBuffer (PointBuffer *const value)

  Creates a new instance.

Private Member Functions

- virtual void AllocateRaw (const unsigned int rawSize) throw (...)

  Allocates the internal bit array.

- ValidBuffer ()

  Not to be publicly used ctor.

Private Attributes

- OGPS_ULong m_RawSize

  Size of the bit array in bytes.

- OpenGPS::UnsignedBytePtr m_ValidityBuffer

  Pointer to the internal bit array.

12.76.2 Constructor & Destructor Documentation

12.76.2.1 ValidBuffer::~ValidBuffer () [virtual]

Destroys this instance.

References Reset().

Here is the call graph for this function:

12.76.2.2 ValidBuffer::ValidBuffer (PointBuffer *const *value*) [protected]

Creates a new instance.

Parameters:

value The point buffer of the Z axis.

References m_RawSize, m_ValidityBuffer, and NULL.

12.76.2.3 ValidBuffer::ValidBuffer () [private]

Not to be publicly used ctor.

References _ASSERT, FALSE, m_RawSize, m_ValidityBuffer, and NULL.

12.76.3 Member Function Documentation

12.76.3.1 void ValidBuffer::Allocate () throw (...) [protected, virtual]

Allocates the internal bit array.

Initially all point vectors are assumed to be valid.

References _ASSERT, AllocateRaw(), OpenGPS::PointValidityProvider::GetPointBuffer(), OpenGPS::PointBuffer::GetSize(), m_RawSize, and m_ValidityBuffer.

Referenced by SetValid().

Here is the call graph for this function:

12.76.3.2 void ValidBuffer::AllocateRaw (const unsigned int *rawSize*) throw (...) [private, virtual]

Allocates the internal bit array.

Initially all point vectors are assumed to be valid.

Parameters:

rawSize Amount of memory to be allocated in bytes.

Returns:

Returns TRUE on success, FALSE otherwise.

References _ASSERT, _EX_T, m_RawSize, m_ValidityBuffer, and OGPS_ExGeneral.

Referenced by Allocate(), and Read().

12.76.3.3 PointBuffer * PointValidityProvider::GetPointBuffer () [protected, inherited]

Returns the point buffer of the Z axis.

References OpenGPS::PointValidityProvider::m_PointBuffer.

12.76.3.4 const PointBuffer * PointValidityProvider::GetPointBuffer () const [protected, inherited]

Returns the point buffer of the Z axis.

References OpenGPS::PointValidityProvider::m_PointBuffer.

Referenced by Allocate(), HasInvalidMarks(), IsValid(), OpenGPS::DoubleInlineValidity::IsValid(), OpenGPS::FloatInlineValidity::IsValid(), OpenGPS::Int32ValidBuffer::SetValid(), OpenGPS::Int16ValidBuffer::SetValid(), SetValid(), OpenGPS::DoubleInlineValidity::SetValid(), and OpenGPS::FloatInlineValidity::SetValid().

12.76.3.5 OGPS_Boolean ValidBuffer::HasInvalidMarks () const [virtual]

Checks whether the current instance serves any point data that is marked as invalid.

Returns:

FALSE if none of the referenced point data is marked as invalid here, otherwise TRUE.

References FALSE, OpenGPS::PointValidityProvider::GetPointBuffer(), OpenGPS::PointBuffer::GetSize(), IsValid(), m_ValidityBuffer, and TRUE.

Here is the call graph for this function:

12.76.3.6 OGPS_Boolean ValidBuffer::IsAllocated () const [virtual]

Returns wheter the bit buffer has already been allocated.

References m_ValidityBuffer, and NULL.

Referenced by OpenGPS::ISO5436_2Container::CreatePointBuffer().

12.76.3.7 OGPS_Boolean ValidBuffer::IsValid (const unsigned int *index*) const throw (...) [virtual]

Gets the validity of a point vector at a given location.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

index The location of the point data the validity is checked.

Returns:

Returns TRUE if valid, FALSE otherwise.

Implements OpenGPS::PointValidityProvider.

References _ASSERT, OpenGPS::PointValidityProvider::GetPointBuffer(), m_ValidityBuffer, and TRUE.

Referenced by HasInvalidMarks().

Here is the call graph for this function:

12.76.3.8 void ValidBuffer::Read (std::basic_istream< OpenGPS::UnsignedByte > & *stream*) throw (...) [virtual]

Maps the bit buffer from a binary stream.

Parameters:

stream The bit array gets copied from here.

References _EX_T, AllocateRaw(), FALSE, m_ValidityBuffer, OGPS_ExGeneral, Reset(), and TRUE.

Referenced by OpenGPS::ISO5436_2Container::CreatePointBuffer().

Here is the call graph for this function:

12.76.3.9 void ValidBuffer::Reset () [protected, virtual]

Frees allocated resources.

References _OPENGPS_FREE, m_RawSize, and m_ValidityBuffer.

Referenced by Read(), and ~ValidBuffer().

12.76.3.10 void ValidBuffer::SetValid (const unsigned int *index*, const OGPS_Boolean *value*) throw (...) [virtual]

Sets the validity of a point vector at a given location.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

index The location of the point data the validity is updated.

value The value of the current validity of the point vector at the given position.

Implements OpenGPS::PointValidityProvider.

Reimplemented in OpenGPS::Int16ValidBuffer, and OpenGPS::Int32ValidBuffer.

References _ASSERT, Allocate(), OpenGPS::PointValidityProvider::GetPointBuffer(), and m_-ValidityBuffer.

Referenced by OpenGPS::Int32ValidBuffer::SetValid(), and OpenGPS::Int16ValidBuffer::SetValid().

Here is the call graph for this function:

12.76.3.11 void ValidBuffer::Write (std::ostream & *stream*) throw (...) [virtual]

Maps the bit buffer to a binary stream.

Parameters:

stream The internal bit array gets written to the given stream.

Returns:

Returns TRUE on success, FALSE otherwise.

References _ASSERT, _EX_T, m_RawSize, m_ValidityBuffer, and OGPS_ExGeneral.

Referenced by OpenGPS::ISO5436_2Container::SaveValidPointsLink().

12.76.4 Member Data Documentation

12.76.4.1 OGPS_ULong OpenGPS::ValidBuffer::m_RawSize [private]

Size of the bit array in bytes.

Referenced by Allocate(), AllocateRaw(), Reset(), ValidBuffer(), and Write().

12.76.4.2 OpenGPS::UnsignedBytePtr OpenGPS::ValidBuffer::m_ValidityBuffer [private]

Pointer to the internal bit array.

Referenced by Allocate(), AllocateRaw(), HasInvalidMarks(), IsAllocated(), IsValid(), Read(), Reset(), SetValid(), ValidBuffer(), and Write().

The documentation for this class was generated from the following files:

- valid_buffer.hxx
- valid_buffer.cxx

12.77 OpenGPS::VectorBuffer Class Reference

`#include <vector_buffer.hxx>`

Collaboration diagram for OpenGPS::VectorBuffer:

12.77.1 Detailed Description

Implements the memory structure of point measurement data.

For every three coordinate axis distinct memory blocks capable of serving all point data belonging to that single axis are managed. If no explicit measurement data is needed for a special axis because point data can be calculated implicitly by means of indexing then no OpenGPS::PointBuffer is provided and a NULL pointer is saved instead.

Direct manipulation of vector data is enabled by the usage of proxied point vectors.

Also information about the validity of point vectors is provided based on indexes.

Public Member Functions

- virtual PointVectorAutoPtr GetPointVectorProxy (const PointVectorProxyContext &context)

  Creates an object which proxies this object instance through a point vector.

- virtual ValidBuffer * GetValidityBuffer ()

  Gets the object representing an external point validity file.

- virtual const ValidBuffer * GetValidityBuffer () const

Gets the object representing an external point validity file.

- virtual PointValidityProvider ∗ GetValidityProvider ()

  Gets the object that tracks the validity of point vectors.

- virtual const PointValidityProvider ∗ GetValidityProvider () const

  Gets the object that tracks the validity of point vectors.

- virtual PointBuffer ∗ GetX ()

  Gets the buffer of point measurement data of the X components of all vectors.

- virtual const PointBuffer ∗ GetX () const

  Gets the buffer of point measurement data of the X components of all vectors for read-only access.

- virtual PointBuffer ∗ GetY ()

  Gets the buffer of point measurement data of the Y components of all vectors.

- virtual const PointBuffer ∗ GetY () const

  Gets the buffer of point measurement data of the Y components of all vectors for read-only access.

- virtual PointBuffer ∗ GetZ ()

  Gets the buffer of point measurement data of the Z components of all vectors.

- virtual const PointBuffer ∗ GetZ () const

  Gets the buffer of point measurement data of the Z components of all vectors for read-only access.

- OGPS_Boolean HasValidityBuffer () const

  Returns TRUE if point validity information is processed by usage of an external file.

- virtual void SetValidityProvider (PointValidityProvider ∗const value, ValidBuffer ∗const buffer)

  Sets the instance of the object that manages validity information for all point vectors.

- virtual void SetX (PointBuffer ∗const value)

  Sets the buffer for point measurement data of the X component of a vector.

- virtual void SetY (PointBuffer ∗const value)

  Sets the buffer for point measurement data of the Y component of a vector.

- virtual void SetZ (PointBuffer ∗const value)

  Sets the buffer for point measurement data of the Z component of a vector.

- VectorBuffer ()

  Creates a new instance.

- ∼VectorBuffer ()

  Destroys this instance.

Private Attributes

- ValidBuffer * m_ValidBuffer

  Buffer for validity information of all point vector indexes.

- PointValidityProvider * m_ValidityProvider

  Interface to retrieve/provide point validity information.

- PointBuffer * m_X

  Buffer for measurements of the X component of all point vectors.

- PointBuffer * m_Y

  Buffer for measurements of the Y component of all point vectors.

- PointBuffer * m_Z

  Buffer for measurements of the Z component of all point vectors.

12.77.2 Constructor & Destructor Documentation

12.77.2.1 VectorBuffer::VectorBuffer ()

Creates a new instance.

References m_ValidBuffer, m_ValidityProvider, m_X, m_Y, m_Z, and NULL.

12.77.2.2 VectorBuffer::~VectorBuffer ()

Destroys this instance.

References _OPENGPS_DELETE, m_ValidityProvider, m_X, m_Y, and m_Z.

12.77.3 Member Function Documentation

12.77.3.1 PointVectorAutoPtr VectorBuffer::GetPointVectorProxy (const PointVectorProxyContext & *context*) `[virtual]`

Creates an object which proxies this object instance through a point vector.

This OpenGPS::VectorBuffer instance manages static memory for all point vector data. A proxied point vector gives the impression of manipulating a single vector although every operation on that object is made on the data structure implemented by this OpenGPS::VectorBuffer instance directly.

Parameters:

context From here the point vector proxy object gets information which row index of the internally stacked point measurement data is actually processed.

Referenced by OpenGPS::ISO5436_2Container::CreateDocument(), and OpenGPS::ISO5436_2Container::CreatePointBuffer().

12.77.3.2 ValidBuffer * VectorBuffer::GetValidityBuffer () [virtual]

Gets the object representing an external point validity file.

May return NULL, if point validity information is not aquired through an external file but inline by the usage of special point values for invalid points.

See also:

VectorBuffer::HasValidityBuffer

References m_ValidBuffer.

12.77.3.3 const ValidBuffer * VectorBuffer::GetValidityBuffer () const [virtual]

Gets the object representing an external point validity file.

May return NULL, if point validity information is not aquired through an external file but inline by the usage of special point values for invalid points.

See also:

VectorBuffer::HasValidityBuffer

References m_ValidBuffer.

Referenced by OpenGPS::ISO5436_2Container::CreatePointBuffer(), and OpenGPS::ISO5436_2Container::SaveValidPointsLink().

12.77.3.4 PointValidityProvider * VectorBuffer::GetValidityProvider () [virtual]

Gets the object that tracks the validity of point vectors.

See also:

VectorBuffer::HasValidityBuffer

References m_ValidityProvider.

12.77.3.5 const PointValidityProvider * VectorBuffer::GetValidityProvider () const [virtual]

Gets the object that tracks the validity of point vectors.

See also:

VectorBuffer::HasValidityBuffer

References m_ValidityProvider.

Referenced by OpenGPS::ISO5436_2Container::CreatePointBuffer(), OpenGPS::ISO5436_2Container::GetMatrixPoint(), OpenGPS::ISO5436_2Container::IsMatrixCoordValid(), and OpenGPS::ISO5436_2Container::SetMatrixPoint().

12.77.3.6 PointBuffer * VectorBuffer::GetX () [virtual]

Gets the buffer of point measurement data of the X components of all vectors.

References m_X.

12.77.3.7 const PointBuffer * VectorBuffer::GetX () const [virtual]

Gets the buffer of point measurement data of the X components of all vectors for read-only access.

References m_X.

Referenced by OpenGPS::PointVectorProxy::PointVectorProxy().

12.77.3.8 PointBuffer * VectorBuffer::GetY () [virtual]

Gets the buffer of point measurement data of the Y components of all vectors.

References m_Y.

12.77.3.9 const PointBuffer * VectorBuffer::GetY () const [virtual]

Gets the buffer of point measurement data of the Y components of all vectors for read-only access.

References m_Y.

Referenced by OpenGPS::PointVectorProxy::PointVectorProxy().

12.77.3.10 PointBuffer * VectorBuffer::GetZ () [virtual]

Gets the buffer of point measurement data of the Z components of all vectors.

References m_Z.

12.77.3.11 const PointBuffer * VectorBuffer::GetZ () const [virtual]

Gets the buffer of point measurement data of the Z components of all vectors for read-only access.

References m_Z.

Referenced by OpenGPS::VectorBufferBuilder::BuildValidityProvider(), and OpenGPS::PointVectorProxy::PointVectorProxy().

12.77.3.12 OGPS_Boolean VectorBuffer::HasValidityBuffer () const

Returns TRUE if point validity information is processed by usage of an external file.

It must be taken responsibility to read/write this file then before/after usage of OpenGPS::PointValidityProvider. Returns FALSE if there is no such file handle instance which must be taken care of.

See also:

VectorBuffer::GetValidityBuffer, VectorBuffer::GetValidityProvider, VectorBuffer::SetValidityProvider

References m_ValidBuffer, and NULL.

Referenced by OpenGPS::ISO5436_2Container::CreatePointBuffer(), and OpenGPS::ISO5436_2Container::SaveValidPointsLink().

12.77.3.13 void VectorBuffer::SetValidityProvider (PointValidityProvider *const *value*, ValidBuffer *const *buffer*) [virtual]

Sets the instance of the object that manages validity information for all point vectors.

Parameters:

value Provides information about the validity of every point vector.

buffer Optionally specifies an object that enables direct access to the external file that may exist as a backend of the current OpenGPS::PointValidityProvider instance to serve point validity requests.

References _ASSERT, m_ValidBuffer, m_ValidityProvider, and NULL.

Referenced by OpenGPS::VectorBufferBuilder::BuildValidityProvider().

12.77.3.14 void VectorBuffer::SetX (PointBuffer *const *value*) [virtual]

Sets the buffer for point measurement data of the X component of a vector.

Remarks:

The pointer passed will become managed by this instance.

Parameters:

value The instance of an object that manages the data point measurements or NULL if this axis does not need explicit point values.

References _ASSERT, and m_X.

Referenced by OpenGPS::VectorBufferBuilder::BuildX().

12.77.3.15 void VectorBuffer::SetY (PointBuffer *const *value*) [virtual]

Sets the buffer for point measurement data of the Y component of a vector.

Remarks:

The pointer passed will become managed by this instance.

Parameters:

value The instance of an object that manages the data point measurements or NULL if this axis does not need explicit point values.

References _ASSERT, and m_Y.

Referenced by OpenGPS::VectorBufferBuilder::BuildY().

12.77.3.16 void VectorBuffer::SetZ (PointBuffer *const *value*) [virtual]

Sets the buffer for point measurement data of the Z component of a vector.

Remarks:

The pointer passed will become managed by this instance.

Parameters:

value The instance of an object that manages the data point measurements or NULL if this axis does not need explicit point values.

References _ASSERT, and m_Z.

Referenced by OpenGPS::VectorBufferBuilder::BuildZ().

12.77.4 Member Data Documentation

12.77.4.1 ValidBuffer* OpenGPS::VectorBuffer::m_ValidBuffer [private]

Buffer for validity information of all point vector indexes.

Referenced by GetValidityBuffer(), HasValidityBuffer(), SetValidityProvider(), and VectorBuffer().

12.77.4.2 PointValidityProvider* OpenGPS::VectorBuffer::m_ValidityProvider [private]

Interface to retrieve/provide point validity information.

Referenced by GetValidityProvider(), SetValidityProvider(), VectorBuffer(), and ~VectorBuffer().

12.77.4.3 PointBuffer* OpenGPS::VectorBuffer::m_X [private]

Buffer for measurements of the X component of all point vectors.

Referenced by GetX(), SetX(), VectorBuffer(), and ~VectorBuffer().

12.77.4.4 PointBuffer* OpenGPS::VectorBuffer::m_Y [private]

Buffer for measurements of the Y component of all point vectors.

Referenced by GetY(), SetY(), VectorBuffer(), and ~VectorBuffer().

12.77.4.5 PointBuffer* OpenGPS::VectorBuffer::m_Z [private]

Buffer for measurements of the Z component of all point vectors.

Referenced by GetZ(), SetZ(), VectorBuffer(), and ~VectorBuffer().

The documentation for this class was generated from the following files:

- vector_buffer.hxx
- vector_buffer.cxx

12.78 OpenGPS::VectorBufferBuilder Class Reference

```
#include <vector_buffer_builder.hxx>
```

Collaboration diagram for OpenGPS::VectorBufferBuilder:

12.78.1 Detailed Description

Creates an object which is able to assemble a OpenGPS::VectorBuffer instance.

Public Member Functions

- virtual OGPS_Boolean BuildBuffer ()

  Creates the initial OpenGPS::VectorBuffer to be assembled.

- virtual OGPS_Boolean BuildValidityProvider ()

  Connects the appropriate OpenGPS::PointValidityProvider.

- virtual OGPS_Boolean BuildX (const OGPS_DataPointType dataType, const OGPS_ULong size)

  Connects the appropriate OpenGPS::PointBuffer connected with the X axis description.

- virtual OGPS_Boolean BuildY (const OGPS_DataPointType dataType, const OGPS_ULong size)

  Connects the appropriate OpenGPS::PointBuffer connected with the Y axis description.

- virtual OGPS_Boolean BuildZ (const OGPS_DataPointType dataType, const OGPS_ULong size)

  Connects the appropriate OpenGPS::PointBuffer connected with the Z axis description.

- virtual VectorBuffer * GetBuffer ()

  Gets the object built.

- VectorBufferBuilder ()

  Creates a new instance.

- ~VectorBufferBuilder ()

  Destroys thisinstance.

Private Member Functions

- PointBuffer * CreatePointBuffer (const OGPS_DataPointType dataType, const OGPS_ULong size, OGPS_Boolean *const retval) const

  Creates the appropriate OpenGPS::PointBuffer instance depending on the type of point data to be handled.

Private Attributes

- VectorBuffer * m_Buffer

  The vector buffer object to be assembled.

12.78.2 Constructor & Destructor Documentation

12.78.2.1 VectorBufferBuilder::VectorBufferBuilder ()

Creates a new instance.

References m_Buffer, and NULL.

12.78.2.2 VectorBufferBuilder::~VectorBufferBuilder ()

Destroys thisinstance.

References _OPENGPS_DELETE, and m_Buffer.

12.78.3 Member Function Documentation

12.78.3.1 OGPS_Boolean VectorBufferBuilder::BuildBuffer () [virtual]

Creates the initial OpenGPS::VectorBuffer to be assembled.

Remarks:

This must preceed all other steps of the building process.

References _ASSERT, m_Buffer, and TRUE.

Referenced by OpenGPS::ISO5436_2Container::BuildVectorBuffer().

12.78.3.2 OGPS_Boolean VectorBufferBuilder::BuildValidityProvider () [virtual]

Connects the appropriate OpenGPS::PointValidityProvider.

Remarks:

This is the last step of the building process.

References _ASSERT, FALSE, OpenGPS::PointBuffer::GetPointType(), OpenGPS::VectorBuffer::GetZ(), m_Buffer, NULL, OGPS_DoublePointType, OGPS_FloatPointType, OGPS_Int16PointType, OGPS_Int32PointType, OGPS_MissingPointType, OpenGPS::VectorBuffer::SetValidityProvider(), and TRUE.

Referenced by OpenGPS::ISO5436_2Container::BuildVectorBuffer().

Here is the call graph for this function:

12.78.3.3 OGPS_Boolean VectorBufferBuilder::BuildX (const OGPS_DataPointType *dataType*, const OGPS_ULong *size*) [virtual]

Connects the appropriate OpenGPS::PointBuffer connected with the X axis description.

Parameters:

dataType The type of point data connected to the X axis. A value of OGPS_MissingPointType describes an incremental (non-explicit) axis for which no point data needs to be stored.

size The amount of points stored for the X component of all available vectors.

References _ASSERT, CreatePointBuffer(), FALSE, m_Buffer, and OpenGPS::VectorBuffer::SetX().

Referenced by OpenGPS::ISO5436_2Container::BuildVectorBuffer().

Here is the call graph for this function:

12.78.3.4 OGPS_Boolean VectorBufferBuilder::BuildY (const OGPS_DataPointType *dataType*, const OGPS_ULong *size*) [virtual]

Connects the appropriate OpenGPS::PointBuffer connected with the Y axis description.

Parameters:

dataType The type of point data connected to the Y axis. A value of OGPS_MissingPointType describes an incremental (non-explicit) axis for which no point data needs to be stored.

size The amount of points stored for the Y component of all available vectors.

References _ASSERT, CreatePointBuffer(), FALSE, m_Buffer, and OpenGPS::VectorBuffer::SetY().

Referenced by OpenGPS::ISO5436_2Container::BuildVectorBuffer().

Here is the call graph for this function:

12.78.3.5 OGPS_Boolean VectorBufferBuilder::BuildZ (const OGPS_DataPointType *dataType*, const OGPS_ULong *size*) [virtual]

Connects the appropriate OpenGPS::PointBuffer connected with the Z axis description.

Parameters:

dataType The type of point data connected to the X axis. Since the Z componet is forced to be saved explicitly regardless whether the axis is defined as incremental or not, a value of OGPS_-MissingPointType is invalid here.

size The amount of points stored for the Z component of all available vectors.

References _ASSERT, CreatePointBuffer(), FALSE, m_Buffer, OGPS_MissingPointType, and OpenGPS::VectorBuffer::SetZ().

Referenced by OpenGPS::ISO5436_2Container::BuildVectorBuffer().

Here is the call graph for this function:

12.78.3.6 PointBuffer * VectorBufferBuilder::CreatePointBuffer (const OGPS_DataPointType *dataType*, const OGPS_ULong *size*, OGPS_Boolean *const *retval*) const [private]

Creates the appropriate OpenGPS::PointBuffer instance depending on the type of point data to be handled.

Parameters:

dataType The type of point data that must be handled by the OpenGPS::PointBuffer instance to be created.

size The amount of point data to be handled.

retval Gets TRUE on success, FALSE otherwise.

Returns:

Returns an instance of a buffer object or NULL if the type of point data to be handled equals OGPS_-MissingPointType.

References _ASSERT, _OPENGPS_DELETE, OpenGPS::PointBuffer::Allocate(), FALSE, NULL, OGPS_DoublePointType, OGPS_FloatPointType, OGPS_Int16PointType, OGPS_Int32PointType, OGPS_MissingPointType, and TRUE.

Referenced by BuildX(), BuildY(), and BuildZ().

Here is the call graph for this function:

12.78.3.7 VectorBuffer * VectorBufferBuilder::GetBuffer () [virtual]

Gets the object built.

This object instance remains managed by the builder, though.

References m_Buffer.

12.78.4 Member Data Documentation

12.78.4.1 VectorBuffer* OpenGPS::VectorBufferBuilder::m_Buffer [private]

The vector buffer object to be assembled.

Referenced by BuildBuffer(), BuildValidityProvider(), BuildX(), BuildY(), BuildZ(), GetBuffer(), VectorBufferBuilder(), and ~VectorBufferBuilder().

The documentation for this class was generated from the following files:

- vector_buffer_builder.hxx
- vector_buffer_builder.cxx

12.79 OpenGPS::XmlPointVectorReaderContext Class Reference

```
#include <xml_point_vector_reader_context.hxx>
```

Inheritance diagram for OpenGPS::XmlPointVectorReaderContext:

Collaboration diagram for OpenGPS::XmlPointVectorReaderContext:

12.79.1 Detailed Description

Specialized OpenGPS::PointVectorReaderContext for point vectors stored as list of strings.

Each string in the list represents one point vector. The values of the three coordinates are seperated either by free space or a semicolon. If a value of a coordinate needs not to be stored in the string because its corresponding axis has an incremental axis definition the value is completely omittet, i.e. no semicolon is written either. If there is an empty string in the list, it is recognised as an invalid 3d point measurement.

Public Types

- typedef Schemas::ISO5436_2::DataListType::Datum_sequence StringList

Public Member Functions

- virtual OGPS_Boolean IsValid () const throw (...)

  Asks if there is readable point vector stored at the current position.

- virtual OGPS_Boolean MoveNext () throw (...)

  Move the current reading position of the stream to the next three-vector.

- virtual void Read (OGPS_Double *const value) throw (...)

  Reads the currently underlying data as OGPS_Double.

- virtual void Read (OGPS_Float *const value) throw (...)

  Reads the currently underlying data as OGPS_Float.

- virtual void Read (OGPS_Int32 *const value) throw (...)

  Reads the currently underlying data as OGPS_Int32.

- virtual void Read (OGPS_Int16 *const value) throw (...)

  Reads the currently underlying data as OGPS_Int16.

- virtual void Skip () throw (...)

  Skips reading of the currently underlying data.

- XmlPointVectorReaderContext (const StringList *const pointVectorList)

  Creates a new instance.

- virtual ~XmlPointVectorReaderContext ()

  Destroys this instance.

Protected Member Functions

- virtual OGPS_Boolean IsGood () const

  Asks if the underlying data stream is still valid.

Private Member Functions

- void Reset ()

  Resets the inner stream object.

- void Set (const OpenGPS::String &buf)

  Feeds the underlying stream object with one single string out of the inner string list to be parsed as the current vector.

Private Attributes

- OGPS_ULong m_Next

  The index of the next point vector to be parsed.

- const StringList * m_PointVectorList

  The inner list of all point vectors to be parsed.

- PointVectorInputStringStream * m_Stream

  The inner stream object which streams the current point vector.

12.79.2 Member Typedef Documentation

12.79.2.1 typedef Schemas::ISO5436_2::DataListType::Datum_sequence OpenGPS::XmlPointVectorReaderContext::StringList

12.79.3 Constructor & Destructor Documentation

12.79.3.1 XmlPointVectorReaderContext::XmlPointVectorReaderContext (const StringList *const *pointVectorList*)

Creates a new instance.

Parameters:

pointVectorList The list of point vectors to be streamed herein.

References _ASSERT, m_Next, m_Stream, and NULL.

12.79.3.2 XmlPointVectorReaderContext::~XmlPointVectorReaderContext () `[virtual]`

Destroys this instance.

References Reset().

Here is the call graph for this function:

12.79.4 Member Function Documentation

12.79.4.1 OGPS_Boolean XmlPointVectorReaderContext::IsGood () const `[protected, virtual]`

Asks if the underlying data stream is still valid.

Returns:

Returns TRUE if no previous access to the underlying data stream was harmful. Returns FALSE if any damage occured.

References _ASSERT, and m_Stream.

12.79.4.2 OGPS_Boolean XmlPointVectorReaderContext::IsValid () const throw (...) `[virtual]`

Asks if there is readable point vector stored at the current position.

A point vector may be invalid if it had not been measured, but is needed for topological consistency though.

Remarks:

This must be checked before one attempts to PointVectorReaderContext::Read/ PointVectorReaderContext::Skip any of the point data of one of the three coordinates.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns TRUE if the current point is readable, FALSE otherwise.

Implements OpenGPS::PointVectorReaderContext.

References m_Stream.

12.79.4.3 OGPS_Boolean XmlPointVectorReaderContext::MoveNext () throw (...) [virtual]

Move the current reading position of the stream to the next three-vector.

This is possible only if all three coordinates of the current point vector data had been fully read (see PointVectorReaderContext::Read and PointVectorReaderContext::Skip member functions). If PointVectorReaderContext::IsValid returns FALSE, call this method directly to move to the next point vector in storage.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Returns:

Returns TRUE when there is more data to be parsed, FALSE otherwise.

Implements OpenGPS::PointVectorReaderContext.

References _ASSERT, FALSE, m_Next, m_PointVectorList, m_Stream, Reset(), Set(), and TRUE.

Here is the call graph for this function:

12.79.4.4 void XmlPointVectorReaderContext::Read (OGPS_Double *const *value*) throw (...) [virtual]

Reads the currently underlying data as OGPS_Double.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorReaderContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_STREAM_AND_THROW_EXCEPTION, and m_Stream.

12.79.4.5 void XmlPointVectorReaderContext::Read (OGPS_Float *const *value*) throw (...) [virtual]

Reads the currently underlying data as OGPS_Float.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorReaderContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_STREAM_-AND_THROW_EXCEPTION, and m_Stream.

12.79.4.6 void XmlPointVectorReaderContext::Read (OGPS_Int32 *const *value*) throw (...) [virtual]

Reads the currently underlying data as OGPS_Int32.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorReaderContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_STREAM_-AND_THROW_EXCEPTION, and m_Stream.

12.79.4.7 void XmlPointVectorReaderContext::Read (OGPS_Int16 *const *value*) throw (...) [virtual]

Reads the currently underlying data as OGPS_Int16.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorReaderContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_STREAM_-AND_THROW_EXCEPTION, and m_Stream.

12.79.4.8 void XmlPointVectorReaderContext::Reset () [private]

Resets the inner stream object.

References _OPENGPS_DELETE, m_Next, and m_Stream.

Referenced by MoveNext(), and ~XmlPointVectorReaderContext().

12.79.4.9 void XmlPointVectorReaderContext::Set (const OpenGPS::String & *buf*) [private]

Feeds the underlying stream object with one single string out of the inner string list to be parsed as the current vector.

Parameters:

buf The string representation of the next point vector to be parsed.

References _ASSERT, and m_Stream.

Referenced by MoveNext().

12.79.4.10 void XmlPointVectorReaderContext::Skip () throw (...) [virtual]

Skips reading of the currently underlying data.

Also moves the current reading position of the stream to the next coordinate of the three-vector.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

This must be called for integrity if it is expected from the reading process that there is no point data available for reading, i.e. for the current coordinate no point data was saved because the corresponding axis is of incremental type.

Implements OpenGPS::PointVectorReaderContext.

References _CHECK_ISGOOD_AND_THROW_EXCEPTION, and _CHECK_STREAM_AND_-THROW_EXCEPTION.

12.79.5 Member Data Documentation

12.79.5.1 OGPS_ULong OpenGPS::XmlPointVectorReaderContext::m_Next [private]

The index of the next point vector to be parsed.

Referenced by MoveNext(), Reset(), and XmlPointVectorReaderContext().

12.79.5.2 const StringList* OpenGPS::XmlPointVectorReaderContext::m_PointVectorList [private]

The inner list of all point vectors to be parsed.

Referenced by MoveNext().

12.79.5.3 PointVectorInputStringStream* OpenGPS::XmlPointVectorReaderContext::m_Stream [private]

The inner stream object which streams the current point vector.

Referenced by IsGood(), IsValid(), MoveNext(), Read(), Reset(), Set(), and XmlPointVectorReaderContext().

The documentation for this class was generated from the following files:

- xml_point_vector_reader_context.hxx
- xml_point_vector_reader_context.cxx

12.80 OpenGPS::XmlPointVectorWriterContext Class Reference

```
#include <xml_point_vector_writer_context.hxx>
```

Inheritance diagram for OpenGPS::XmlPointVectorWriterContext:

Collaboration diagram for OpenGPS::XmlPointVectorWriterContext:

12.80.1 Detailed Description

Specialized OpenGPS::PointVectorWriterContext for point vectors stored as list of strings.

Each string in the list represents one point vector. The values of the three coordinates are seperated either by free space or a semicolon. If a value of a coordinate needs not to be stored in the string because its corresponding axis has an incremental axis definition the value is completely omittet, i.e. no semicolon is written either.

Public Types

- typedef Schemas::ISO5436_2::DataListType::Datum_sequence StringList

Public Member Functions

- virtual void MoveNext () throw (...)

  Complete the transaction of the current point vector.

- virtual void Skip () throw (...)

  There is no point data to be written to the underlying stream for the current axis.

- virtual void Write (const OGPS_Double *const value) throw (...)

  Writes a single point of type OGPS_Double to the underlying stream.

- virtual void Write (const OGPS_Float *const value) throw (...)

  Writes a single point of type OGPS_Float to the underlying stream.

- virtual void Write (const OGPS_Int32 *const value) throw (...)

  Writes a single point of type OGPS_Int32 to the underlying stream.

- virtual void Write (const OGPS_Int16 *const value) throw (...)

  Writes a single point of type OGPS_Int16 to the underlying stream.

- XmlPointVectorWriterContext (StringList *const pointVectorList)

  Creates a new instance.

- virtual ~XmlPointVectorWriterContext ()

  Destroys this instance.

Protected Member Functions

- virtual void AppendSeparator ()

  Appends the separator of two data point values.

- virtual OGPS_Boolean IsGood () const

  Asks if the underlying data stream is still valid.

Private Member Functions

- void Get (OpenGPS::String *const value) const

  Gets the content of the inner stream buffer.

- void Reset ()

  Resets/empties the inner stream buffer.

Private Attributes

- OGPS_Boolean m_NeedsSeparator

  TRUE if a separator needs to be added on the next call to XmlPointVectorWriterContext::Write, FALSE otherwise.

- StringList * m_PointVectorList

  The inner string list of point vectors written so far.

- PointVectorOutputStringStream * m_Stream

  The inner stream buffer which holds the current compilation of a point vector to be added to the string list.

12.80.2 Member Typedef Documentation

12.80.2.1 typedef Schemas::ISO5436_2::DataListType::Datum_sequence OpenGPS::XmlPointVectorWriterContext::StringList

12.80.3 Constructor & Destructor Documentation

12.80.3.1 XmlPointVectorWriterContext::XmlPointVectorWriterContext (StringList *const *pointVectorList*)

Creates a new instance.

Parameters:

pointVectorList The list of point vectors to be streamed herein.

References _ASSERT, FALSE, m_NeedsSeparator, m_PointVectorList, and m_Stream.

12.80.3.2 XmlPointVectorWriterContext::~XmlPointVectorWriterContext () `[virtual]`

Destroys this instance.

References _OPENGPS_DELETE, and m_Stream.

12.80.4 Member Function Documentation

12.80.4.1 void XmlPointVectorWriterContext::AppendSeparator () `[protected, virtual]`

Appends the separator of two data point values.

References _ASSERT, m_NeedsSeparator, m_Stream, and TRUE.

Referenced by Write().

12.80.4.2 void XmlPointVectorWriterContext::Get (OpenGPS::String *const *value*) const [private]

Gets the content of the inner stream buffer.

After all three vector components have been written this euqals the string representation of a single point vector.

Parameters:

value Returns the content of the stream buffer.

References _ASSERT, and m_Stream.

Referenced by MoveNext().

12.80.4.3 OGPS_Boolean XmlPointVectorWriterContext::IsGood () const [protected, virtual]

Asks if the underlying data stream is still valid.

Returns:

Returns TRUE if no previous access to the underlying data stream was harmful. Returns FALSE if any damage occured.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, and m_Stream.

12.80.4.4 void XmlPointVectorWriterContext::MoveNext () throw (...) [virtual]

Complete the transaction of the current point vector.

One point vector is made up of three components (data points). After three subsequent PointVectorWriterContext::Write/ PointVectorWriterContext::Skip calls, call this method to indicate that the whole point vector has been written.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, Get(), m_PointVectorList, m_Stream, and Reset().

Here is the call graph for this function:

12.80.4.5 void XmlPointVectorWriterContext::Reset () [private]

Resets/empties the inner stream buffer.

References FALSE, m_NeedsSeparator, and m_Stream.

Referenced by MoveNext().

12.80.4.6 void XmlPointVectorWriterContext::Skip () throw (...) [virtual]

There is no point data to be written to the underlying stream for the current axis.

One point vector has three components, whereas only the Z component needs to be set explicitly in major cases because of implicit axis definitions of X or Y. Use this skip method then (with the X or Y components) to indicate that, because a successfully written point vector must comprise all three component values.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Remarks:

This must be called for integrity if there is no point data to be stored because the corresponding axis is of incremental type.

Implements OpenGPS::PointVectorWriterContext.

References _CHECK_ISGOOD_AND_THROW_EXCEPTION, and _CHECK_STREAM_AND_-THROW_EXCEPTION.

12.80.4.7 void XmlPointVectorWriterContext::Write (const OGPS_Double *const *value*) throw (...) [virtual]

Writes a single point of type OGPS_Double to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_STREAM_-AND_THROW_EXCEPTION, AppendSeparator(), and m_Stream.

Here is the call graph for this function:

12.80.4.8 void XmlPointVectorWriterContext::Write (const OGPS_Float *const *value*) throw (...) [virtual]

Writes a single point of type OGPS_Float to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_STREAM_-AND_THROW_EXCEPTION, AppendSeparator(), and m_Stream.

Here is the call graph for this function:

12.80.4.9 void XmlPointVectorWriterContext::Write (const OGPS_Int32 *const *value*) throw (...) [virtual]

Writes a single point of type OGPS_Int32 to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_STREAM_-AND_THROW_EXCEPTION, AppendSeparator(), and m_Stream.

Here is the call graph for this function:

12.80.4.10 void XmlPointVectorWriterContext::Write (const OGPS_Int16 *const *value*) throw (...) [virtual]

Writes a single point of type OGPS_Int16 to the underlying stream.

A specific implementation may throw an OpenGPS::Exception if this operation is not permitted due to the current state of the object instance.

Parameters:

value Contains the point value on success.

Implements OpenGPS::PointVectorWriterContext.

References _ASSERT, _CHECK_ISGOOD_AND_THROW_EXCEPTION, _CHECK_STREAM_-AND_THROW_EXCEPTION, AppendSeparator(), and m_Stream.

Here is the call graph for this function:

12.80.5 Member Data Documentation

12.80.5.1 OGPS_Boolean OpenGPS::XmlPointVectorWriterContext::m_NeedsSeparator [private]

TRUE if a separator needs to be added on the next call to XmlPointVectorWriterContext::Write, FALSE otherwise.

Referenced by AppendSeparator(), Reset(), and XmlPointVectorWriterContext().

12.80.5.2 StringList* OpenGPS::XmlPointVectorWriterContext::m_PointVectorList [private]

The inner string list of point vectors written so far.

Referenced by MoveNext(), and XmlPointVectorWriterContext().

12.80.5.3 PointVectorOutputStringStream* OpenGPS::XmlPointVectorWriterContext::m_Stream [private]

The inner stream buffer which holds the current compilation of a point vector to be added to the string list.

Referenced by AppendSeparator(), Get(), IsGood(), MoveNext(), Reset(), Write(), XmlPointVectorWriterContext(), and ~XmlPointVectorWriterContext().

The documentation for this class was generated from the following files:

- xml_point_vector_writer_context.hxx
- xml_point_vector_writer_context.cxx

12.81 OpenGPS::ZipOutputStream Class Reference

```
#include <zip_stream_buffer.hxx>
```

Inheritance diagram for OpenGPS::ZipOutputStream:

Collaboration diagram for OpenGPS::ZipOutputStream:

12.81.1 Detailed Description

Provides an output stream interface to write binary data to Info-Zip archives.

Public Types

- typedef std::ostream BaseType
  Data type of the super class.

Public Member Functions

- ZipOutputStream::BaseType & write (const char *s)
  Appends a string to the stream.
- ZipOutputStream (ZipStreamBuffer &buffer)
  Creates a new instance.
- ~ZipOutputStream ()
  Destroys this instance.

12.81.2 Member Typedef Documentation

12.81.2.1 typedef std::ostream OpenGPS::ZipOutputStream::BaseType

Data type of the super class.

12.81.3 Constructor & Destructor Documentation

12.81.3.1 ZipOutputStream::ZipOutputStream (ZipStreamBuffer & *buffer*)

Creates a new instance.

Parameters:

buffer The buffer object that is streamed.

12.81.3.2 ZipOutputStream::~ZipOutputStream ()

Destroys this instance.

12.81.4 Member Function Documentation

12.81.4.1 ZipOutputStream::BaseType & ZipOutputStream::write (const char * *s*)

Appends a string to the stream.

Parameters:

s The string to append.

Referenced by OpenGPS::ISO5436_2Container::SaveChecksumFile().

The documentation for this class was generated from the following files:

- zip_stream_buffer.hxx
- zip_stream_buffer.cxx

12.82 OpenGPS::ZipStreamBuffer Class Reference

```
#include <zip_stream_buffer.hxx>
```

Collaboration diagram for OpenGPS::ZipStreamBuffer:

12.82.1 Detailed Description

Provides a buffer interface suitable for streaming the zipFile handle defined in the zlib/minizip package.

See also:

ZipOutputStream

Public Member Functions

- OGPS_Boolean GetMd5 (OpenGPS::UnsignedByte md5[16])

  Gets the current md5 checksum.

- ZipStreamBuffer (zipFile handle, const OGPS_Boolean enable_md5)

Creates a new instance.

- ~ZipStreamBuffer ()
  Destroys this instance.

Protected Member Functions

- virtual std::streamsize xsputn (const char *s, std::streamsize n)
  Overrides the super class.

Private Types

- typedef std::streambuf BaseType
  Data type of the super class.

Private Attributes

- zipFile m_Handle
  Handle to the zipFile where buffered data gets written to.
- md5_context * m_Md5Context
  The current state of md5 checksum processing.

12.82.2 Member Typedef Documentation

12.82.2.1 typedef std::streambuf OpenGPS::ZipStreamBuffer::BaseType [private]

Data type of the super class.

12.82.3 Constructor & Destructor Documentation

12.82.3.1 ZipStreamBuffer::ZipStreamBuffer (zipFile *handle*, const OGPS_Boolean *enable_md5*)

Creates a new instance.

Parameters:

handle The Info-Zip file handle buffered binary data is written to.

enable_md5 When set to TRUE generates md5 checksums of the buffered binary data, if FALSE no cheksum data will be generated.

References m_Handle, m_Md5Context, md5_starts(), and NULL.

Here is the call graph for this function:

12.82.3.2 ZipStreamBuffer::~ZipStreamBuffer ()

Destroys this instance.

References _OPENGPS_DELETE, and m_Md5Context.

12.82.4 Member Function Documentation

12.82.4.1 OGPS_Boolean ZipStreamBuffer::GetMd5 (OpenGPS::UnsignedByte *md5*[16])

Gets the current md5 checksum.

Also resets the computed md5 data internally.

Parameters:

md5 Gets the 128-bit md5 data.

Returns:

Returns TRUE on success, FALSE otherwise.

References FALSE, m_Md5Context, md5_finish(), md5_starts(), and TRUE.

Referenced by OpenGPS::BinaryPointVectorWriterContext::GetMd5(), OpenGPS::ISO5436_-2Container::SaveValidPointsLink(), and OpenGPS::ISO5436_2Container::SaveXmlDocument().

Here is the call graph for this function:

12.82.4.2 std::streamsize ZipStreamBuffer::xsputn (const char * *s*, std::streamsize *n*) [protected, virtual]

Overrides the super class.

References m_Handle, m_Md5Context, and md5_update().

Here is the call graph for this function:

12.82.5 Member Data Documentation

12.82.5.1 zipFile OpenGPS::ZipStreamBuffer::m_Handle [private]

Handle to the zipFile where buffered data gets written to.

Referenced by xsputn(), and ZipStreamBuffer().

12.82.5.2 md5_context* OpenGPS::ZipStreamBuffer::m_Md5Context [private]

The current state of md5 checksum processing.

Referenced by GetMd5(), xsputn(), ZipStreamBuffer(), and ~ZipStreamBuffer().

The documentation for this class was generated from the following files:

- zip_stream_buffer.hxx
- zip_stream_buffer.cxx

13 File Documentation

13.1 auto_ptr_types.hxx File Reference

13.1.1 Detailed Description

Agglomerated definition of std::auto_ptr commonly used.

```
#include <memory>
```

Include dependency graph for auto_ptr_types.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS
- namespace OpenGPS::Schemas
- namespace OpenGPS::Schemas::ISO5436_2

Typedefs

- typedef std::auto_ptr< PointVectorProxyContext > OpenGPS::PointVectorProxyContextAutoPtr

13.2 binary_lsb_point_vector_reader_context.cxx File Reference

```
#include "binary_lsb_point_vector_reader_context.hxx"
#include "point_vector_iostream.hxx"
#include <opengps/cxx/exceptions.hxx>
```

```
#include "stdafx.hxx"
```

Include dependency graph for binary_lsb_point_vector_reader_context.cxx:

Defines

- #define _CHECK_ISGOOD_AND_THROW_EXCEPTION

  Checks whether the underlying stream is valid.

- #define _CHECK_STREAM_AND_THROW_EXCEPTION

  Checks whether the underlying stream is valid.

13.2.1 Define Documentation

13.2.1.1 #define _CHECK_ISGOOD_AND_THROW_EXCEPTION

Value:

```
if(!IsGood()) \
   { \
   throw OpenGPS::Exception( \
      OGPS_ExInvalidOperation, \
      _EX_T("The underlying binary stream object became invalid."), \
      _EX_T("A read/write error occured."), \
      _EX_T("OpenGPS::BinaryLSBPointVectorReaderContext")); \
   }
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

Referenced by OpenGPS::XmlPointVectorReaderContext::Read(), OpenGPS::BinaryMSBPointVectorReaderContext::Read(), OpenGPS::BinaryLSBPointVectorReaderContext::Read(), OpenGPS::XmlPointVectorWriterContext::Skip(), OpenGPS::XmlPointVectorReaderContext::Skip(), OpenGPS::BinaryPointVectorWriterContext::Skip(), OpenGPS::XmlPointVectorWriterContext::Write(), OpenGPS::BinaryMSBPointVectorWriterContext::Write(), and OpenGPS::BinaryLSBPointVectorWriterContext::Write().

13.2.1.2 #define _CHECK_STREAM_AND_THROW_EXCEPTION

Value:

```
if(!HasStream()) \
   { \
   throw OpenGPS::Exception( \
      OGPS_ExInvalidOperation, \
      _EX_T("No binary file stream available."), \
      _EX_T("The operation on the binary file stream failed, because the stream has been closed already.")
      _EX_T("OpenGPS::BinaryLSBPointVectorReaderContext")); \
   }
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

Referenced by OpenGPS::BinaryPointVectorReaderContext::MoveNext(), OpenGPS::XmlPointVectorReaderContext::Read(), OpenGPS::BinaryMSBPointVectorReaderContext::Read(), OpenGPS::BinaryLSBPointVectorReaderContext::Read(), OpenGPS::XmlPointVectorWriterContext::Skip(), OpenGPS::XmlPointVectorReaderContext::Skip(), OpenGPS::BinaryPointVectorWriterContext::Skip(), OpenGPS::BinaryPointVectorReaderContext::Skip(), OpenGPS::XmlPointVectorWriterContext::Write(), OpenGPS::BinaryMSBPointVectorWriterContext::Write(), and OpenGPS::BinaryLSBPointVectorWriterContext::Write().

13.3 binary_lsb_point_vector_reader_context.hxx File Reference

13.3.1 Detailed Description

Implementation of access methods for reading typed point data from a binary file of point vectors.

`#include "binary_point_vector_reader_context.hxx"`

Include dependency graph for binary_lsb_point_vector_reader_context.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::BinaryLSBPointVectorReaderContext

  Implements OpenGPS::BinaryPointVectorReaderContext for binary files to be parsed on machines reading in least significant byte order.

13.4 binary_lsb_point_vector_writer_context.cxx File Reference

```
#include "binary_lsb_point_vector_writer_context.hxx"
#include "point_vector_iostream.hxx"
#include <opengps/cxx/exceptions.hxx>
#include "stdafx.hxx"
```

Include dependency graph for binary_lsb_point_vector_writer_context.cxx:

Defines

- #define _CHECK_ISGOOD_AND_THROW_EXCEPTION

  Checks whether the underlying stream is valid.

- #define _CHECK_STREAM_AND_THROW_EXCEPTION

  Checks whether the underlying stream is valid.

13.4.1 Define Documentation

13.4.1.1 #define _CHECK_ISGOOD_AND_THROW_EXCEPTION

Value:

```
if(!IsGood()) \
   { \
   throw OpenGPS::Exception( \
      OGPS_ExInvalidOperation, \
      _EX_T("The underlying binary stream object became invalid."), \
      _EX_T("A read/write error occured."), \
      _EX_T("OpenGPS::BinaryLSBPointVectorWriterContext")); \
   }
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

13.4.1.2 #define _CHECK_STREAM_AND_THROW_EXCEPTION

Value:

```
if(!HasStream()) \
   { \
   throw OpenGPS::Exception( \
      OGPS_ExInvalidOperation, \
      _EX_T("No binary file stream available."), \
      _EX_T("The operation on the binary file stream failed, because the stream has been closed already.")
      _EX_T("OpenGPS::BinaryLSBPointVectorWriterContext")); \
   }
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

13.5 binary_lsb_point_vector_writer_context.hxx File Reference

13.5.1 Detailed Description

Implementation of access methods for writing typed point data to a binary file of point vectors.

`#include "binary_point_vector_writer_context.hxx"`

Include dependency graph for binary_lsb_point_vector_writer_context.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::BinaryLSBPointVectorWriterContext

  Implements OpenGPS::BinaryPointVectorWriterContext for binary files to be written on machines operating in least significant byte order.

13.6 binary_msb_point_vector_reader_context.cxx File Reference

```
#include "binary_msb_point_vector_reader_context.hxx"
#include "point_vector_iostream.hxx"
#include "environment.hxx"
#include <opengps/cxx/exceptions.hxx>
#include "stdafx.hxx"
```

Include dependency graph for binary_msb_point_vector_reader_context.cxx:

Defines

- #define _CHECK_ISGOOD_AND_THROW_EXCEPTION

  Checks whether the underlying stream is valid.

- #define _CHECK_STREAM_AND_THROW_EXCEPTION

  Checks whether the underlying stream is valid.

13.6.1 Define Documentation

13.6.1.1 #define _CHECK_ISGOOD_AND_THROW_EXCEPTION

Value:

```
if(!IsGood()) \
   { \
   throw OpenGPS::Exception( \
   OGPS_ExInvalidOperation, \
   _EX_T("The underlying binary stream object became invalid."), \
   _EX_T("A read/write error occured."), \
   _EX_T("OpenGPS::BinaryMSBPointVectorReaderContext")); \
   }
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

13.6.1.2 #define _CHECK_STREAM_AND_THROW_EXCEPTION

Value:

```
if(!HasStream()) \
   { \
   throw OpenGPS::Exception( \
   OGPS_ExInvalidOperation, \
   _EX_T("No binary file stream available."), \
   _EX_T("The operation on the binary file stream failed, because the stream has been closed already."), \
   _EX_T("OpenGPS::BinaryMSBPointVectorReaderContext")); \
   }
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

13.7 binary_msb_point_vector_reader_context.hxx File Reference

13.7.1 Detailed Description

Implementation of access methods for reading typed point data from a binary file of point vectors.

`#include "binary_point_vector_reader_context.hxx"`

Include dependency graph for binary_msb_point_vector_reader_context.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::BinaryMSBPointVectorReaderContext

  Implements OpenGPS::BinaryPointVectorReaderContext for binary files to be parsed on machines reading in most significant byte order.

13.8 binary_msb_point_vector_writer_context.cxx File Reference

```
#include "binary_msb_point_vector_writer_context.hxx"
#include "point_vector_iostream.hxx"
#include <opengps/cxx/exceptions.hxx>
#include "environment.hxx"
#include "stdafx.hxx"
```

Include dependency graph for binary_msb_point_vector_writer_context.cxx:

Defines

- #define _CHECK_ISGOOD_AND_THROW_EXCEPTION
  Checks whether the underlying stream is valid.
- #define _CHECK_STREAM_AND_THROW_EXCEPTION
  Checks whether the underlying stream is valid.

13.8.1 Define Documentation

13.8.1.1 #define _CHECK_ISGOOD_AND_THROW_EXCEPTION

Value:

```
if(!IsGood()) \
   { \
   throw OpenGPS::Exception( \
   OGPS_ExInvalidOperation, \
   _EX_T("The underlying binary stream object became invalid."), \
   _EX_T("A read/write error occured."), \
   _EX_T("OpenGPS::BinaryMSBPointVectorWriterContext")); \
   }
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

13.8.1.2 #define _CHECK_STREAM_AND_THROW_EXCEPTION

Value:

```
if(!HasStream()) \
   { \
   throw OpenGPS::Exception( \
   OGPS_ExInvalidOperation, \
   _EX_T("No binary file stream available."), \
   _EX_T("The operation on the binary file stream failed, because the stream has been closed already."), \
   _EX_T("OpenGPS::BinaryMSBPointVectorWriterContext")); \
   }
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

13.9 binary_msb_point_vector_writer_context.hxx File Reference

13.9.1 Detailed Description

Implementation of access methods for writing typed point data to a binary file of point vectors.

`#include "binary_point_vector_writer_context.hxx"`

Include dependency graph for binary_msb_point_vector_writer_context.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::BinaryMSBPointVectorWriterContext

  Implements OpenGPS::BinaryPointVectorWriterContext for binary files to be written on machines operating in most significant byte order.

13.10 binary_point_vector_reader_context.cxx File Reference

`#include "binary_point_vector_reader_context.hxx"`

`#include "point_vector_iostream.hxx"`

```
#include <opengps/cxx/exceptions.hxx>
#include "stdafx.hxx"
```

Include dependency graph for binary_point_vector_reader_context.cxx:

Defines

- #define _CHECK_STREAM_AND_THROW_EXCEPTION

  Checks whether the underlying stream is valid.

13.10.1 Define Documentation

13.10.1.1 #define _CHECK_STREAM_AND_THROW_EXCEPTION

Value:

```
if(!m_Stream) \
   { \
   throw OpenGPS::Exception( \
      OGPS_ExInvalidOperation, \
      _EX_T("No binary file stream available."), \
      _EX_T("The operation on the binary file stream failed, because the stream has been closed already.")
      _EX_T("OpenGPS::BinaryPointVectorReaderContext")); \
   }
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

13.11 binary_point_vector_reader_context.hxx File Reference

13.11.1 Detailed Description

Interface for reading typed point data from an underlying binary file stream of point vectors.

```
#include "point_vector_reader_context.hxx"
```

Include dependency graph for binary_point_vector_reader_context.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::BinaryPointVectorReaderContext

  Specialized OpenGPS::PointVectorReaderContext for binary streams.

13.12 binary_point_vector_writer_context.cxx File Reference

```
#include "binary_point_vector_writer_context.hxx"
#include <opengps/cxx/exceptions.hxx>
#include "stdafx.hxx"
```

Include dependency graph for binary_point_vector_writer_context.cxx:

Defines

- #define _CHECK_ISGOOD_AND_THROW_EXCEPTION

  Checks whether the underlying stream is valid.

- #define _CHECK_STREAM_AND_THROW_EXCEPTION

  Checks whether the underlying stream is valid.

13.12.1 Define Documentation

13.12.1.1 #define _CHECK_ISGOOD_AND_THROW_EXCEPTION

Value:

```
if(!IsGood()) \
   { \
   throw OpenGPS::Exception( \
      OGPS_ExInvalidOperation, \
      _EX_T("The underlying binary stream object became invalid."), \
      _EX_T("A read/write error occured."), \
      _EX_T("OpenGPS::BinaryPointVectorWriterContext")); \
   }
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

13.12.1.2 #define _CHECK_STREAM_AND_THROW_EXCEPTION

Value:

```
if(!HasStream()) \
   { \
   throw OpenGPS::Exception( \
      OGPS_ExInvalidOperation, \
      _EX_T("No binary file stream available."), \
      _EX_T("The operation on the binary file stream failed, because the stream has been closed already.")
      _EX_T("OpenGPS::BinaryPointVectorWriterContext")); \
   }
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

13.13 binary_point_vector_writer_context.hxx File Reference

13.13.1 Detailed Description

Interface for writing typed point data to an underlying binary of point vectors.

```
#include "point_vector_writer_context.hxx"
#include "zip_stream_buffer.hxx"
```

Include dependency graph for binary_point_vector_writer_context.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::BinaryPointVectorWriterContext

  Implements OpenGPS::PointVectorWriterContext for writing to compressed binary streams of point vectors.

13.14 data_point.h File Reference

13.14.1 Detailed Description

The abstract data type of the typesafe representation of a single data point value.

The interface supports getting and setting an encapsulated point value of a predetermined data type.

```
#include <opengps/opengps.h>
#include <opengps/data_point_type.h>
```

Include dependency graph for data_point.h:

This graph shows which files directly or indirectly include this file:

Typedefs

- typedef struct _OGPS_DATA_POINT * OGPS_DataPointPtr

  Typesafe representation of a single data point value.

Functions

- _OPENGPS_EXPORT OGPS_Double ogps_GetDouble (const OGPS_DataPointPtr dataPoint)

  Gets the stored value of type OGPS_Double out of a given data point instance.

- _OPENGPS_EXPORT OGPS_Float ogps_GetFloat (const OGPS_DataPointPtr dataPoint)

  Gets the stored value of type OGPS_Float out of a given data point instance.

- _OPENGPS_EXPORT OGPS_Int16 ogps_GetInt16 (const OGPS_DataPointPtr dataPoint)

  Gets the stored value of type OGPS_Int16 out of a given data point instance.

- _OPENGPS_EXPORT OGPS_Int32 ogps_GetInt32 (const OGPS_DataPointPtr dataPoint)

  Gets the stored value of type OGPS_Int32 out of a given data point instance.

- _OPENGPS_EXPORT OGPS_DataPointType ogps_GetPointType (const OGPS_DataPointPtr dataPoint)

  Gets type information about the current value stored in a given data point instance.

- _OPENGPS_EXPORT void ogps_SetDouble (OGPS_DataPointPtr const dataPoint, const OGPS_-Double value)

  Stores a new value within a given data point instance.

- _OPENGPS_EXPORT void ogps_SetFloat (OGPS_DataPointPtr const dataPoint, const OGPS_-Float value)

  Stores a new value within a given data point instance.

- _OPENGPS_EXPORT void ogps_SetInt16 (OGPS_DataPointPtr const dataPoint, const OGPS_-Int16 value)

  Stores a new value within a given data point instance.

- _OPENGPS_EXPORT void ogps_SetInt32 (OGPS_DataPointPtr const dataPoint, const OGPS_-Int32 value)

  Stores a new value within a given data point instance.

13.15 data_point.hxx File Reference

13.15.1 Detailed Description

Typesafe representation of a single data point value.

The interface supports getting and setting an encapsulated data value of a predetermined type.

```
#include <opengps/cxx/opengps.hxx>
```

```
#include <opengps/data_point_type.h>
```

Include dependency graph for data_point.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::DataPoint

  Typesafe representation of a single data point value.

13.16 data_point_c.cxx File Reference

```
#include <opengps/data_point.h>
#include "data_point_c.hxx"
#include "messages_c.hxx"
#include "../cxx/data_point_impl.hxx"
#include "../cxx/stdafx.hxx"
```

Include dependency graph for data_point_c.cxx:

Functions

- double ogps_GetDouble (const OGPS_DataPointPtr dataPoint) throw ()

  Gets the stored value of type OGPS_Double out of a given data point instance.

- float ogps_GetFloat (const OGPS_DataPointPtr dataPoint) throw ()

  Gets the stored value of type OGPS_Float out of a given data point instance.

- short ogps_GetInt16 (const OGPS_DataPointPtr dataPoint) throw ()

  Gets the stored value of type OGPS_Int16 out of a given data point instance.

- int ogps_GetInt32 (const OGPS_DataPointPtr dataPoint) throw ()

  Gets the stored value of type OGPS_Int32 out of a given data point instance.

- OGPS_DataPointType ogps_GetPointType (const OGPS_DataPointPtr dataPoint) throw ()

  Gets type information about the current value stored in a given data point instance.

- void ogps_SetDouble (OGPS_DataPointPtr const dataPoint, const double value) throw ()

Stores a new value within a given data point instance.

- void ogps_SetFloat (OGPS_DataPointPtr const dataPoint, const float value) throw ()

  Stores a new value within a given data point instance.

- void ogps_SetInt16 (OGPS_DataPointPtr const dataPoint, const short value) throw ()

  Stores a new value within a given data point instance.

- void ogps_SetInt32 (OGPS_DataPointPtr const dataPoint, const int value) throw ()

  Stores a new value within a given data point instance.

13.17 data_point_c.hxx File Reference

13.17.1 Detailed Description

Handle mechanism that makes a C++ OpenGPS::DataPoint object accessible through its corresponding C interface.

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- struct OGPS_DataPoint

  Encapsulates the C++ structure of a data point handle used internally within the C interface.

Typedefs

- typedef struct OGPS_DataPoint ∗ OGPS_DataPointPtr

13.17.2 Typedef Documentation

13.17.2.1 typedef struct OGPS_DataPoint ∗ OGPS_DataPointPtr

13.18 data_point_impl.cxx File Reference

```
#include "data_point_impl.hxx"
#include <opengps/cxx/exceptions.hxx>
```

```
#include "stdafx.hxx"
```

Include dependency graph for data_point_impl.cxx:

13.19 data_point_impl.hxx File Reference

13.19.1 Detailed Description

An implementation of a data point to be used for typesafe storage and access of point data.

```
#include <opengps/cxx/data_point.hxx>
```

Include dependency graph for data_point_impl.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::DataPointImpl

A straightforward implementation of OpenGPS::DataPoint.

- union OGPS_DataPointValue

  Typesafe storage for every possible type of point data.

13.20 data_point_parser.hxx File Reference

13.20.1 Detailed Description

Generic interface of a data point parser for reading/writing point data of any supported data type from/to any supported media.

```
#include <opengps/cxx/opengps.hxx>
```

Include dependency graph for data_point_parser.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::DataPointParser

  Interface for reading/writing point data.

13.21 data_point_proxy.cxx File Reference

```
#include "point_vector_proxy.hxx"
#include "point_vector_proxy_context.hxx"
```

```
#include "point_buffer.hxx"
#include <opengps/cxx/exceptions.hxx>
#include "stdafx.hxx"
```

Include dependency graph for data_point_proxy.cxx:

Defines

- #define _CHECK_BUFFER_AND_THROW_READ_EXCEPTION

  Checks whether an instance of a point buffer does exist.

- #define _CHECK_BUFFER_AND_THROW_WRITE_EXCEPTION

  Checks whether an instance of a point buffer does exist.

13.21.1 Define Documentation

13.21.1.1 #define _CHECK_BUFFER_AND_THROW_READ_EXCEPTION

Value:

```
if(!m_Buffer) \
   { \
      throw OpenGPS::Exception( \
         OGPS_ExInvalidOperation, \
         _EX_T("The value of a data point could not be read."), \
         _EX_T("There exists no buffer of point data where to read from. Most likely this is an attempt to
         _EX_T("OpenGPS::PointVectorProxy::DataPointProxy::Get")); \
   }
```

Checks whether an instance of a point buffer does exist.

If not an exception is thrown. The message describing the cause of failure makes sense for a refused attempt to read point data.

Referenced by OpenGPS::PointVectorProxy::DataPointProxy::Get().

13.21.1.2 #define _CHECK_BUFFER_AND_THROW_WRITE_EXCEPTION

Value:

```
if(!m_Buffer) \
```

```
{ \
   throw OpenGPS::Exception( \
      OGPS_ExInvalidOperation, \
      _EX_T("The value of a data point could not be set."), \
      _EX_T("There exists no buffer of point data where to write to. Most likely this is an attempt to
      _EX_T("OpenGPS::PointVectorProxy::DataPointProxy::Set")); \
}
```

Checks whether an instance of a point buffer does exist.

If not an exception is thrown. The message describing the cause of failure makes sense for a refused attempt to write point data.

Referenced by OpenGPS::PointVectorProxy::DataPointProxy::Set().

13.22 data_point_type.h File Reference

13.22.1 Detailed Description

An enumeration type which may describe the data type of a current data point value.

Used by various components throughout this software library.

This graph shows which files directly or indirectly include this file:

Enumerations

- enum OGPS_DataPointType {
  OGPS_MissingPointType, OGPS_Int16PointType, OGPS_Int32PointType, OGPS_FloatPointType, OGPS_DoublePointType }

  Possible types of data stored in an OGPS_DataPointPtr instance.

13.23 double_data_point_parser.cxx File Reference

```
#include "double_data_point_parser.hxx"
#include "point_vector_reader_context.hxx"
#include "point_vector_writer_context.hxx"
#include <opengps/cxx/data_point.hxx>
#include "stdafx.hxx"
```

Include dependency graph for double_data_point_parser.cxx:

13.24 double_data_point_parser.hxx File Reference

13.24.1 Detailed Description

A data point parser for point data of type OGPS_DoublePointType.

```
#include "data_point_parser.hxx"
```

Include dependency graph for double_data_point_parser.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::DoubleDataPointParser

Reads/Writes instances of OpenGPS::DataPoint of OGPS_Double point data.

13.25 double_point_buffer.cxx File Reference

```
#include "double_point_buffer.hxx"
```

```
#include "stdafx.hxx"
```

Include dependency graph for double_point_buffer.cxx:

13.26 double_point_buffer.hxx File Reference

13.26.1 Detailed Description

Allocate static memory to store point data.

```
#include "point_buffer.hxx"
```

Include dependency graph for double_point_buffer.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::DoublePointBuffer

  Manages static memory and typesafe access.

13.27 Doxygen.cpp File Reference

13.27.1 Detailed Description

Title page of documentation, no source code.

Namespaces

- namespace OpenGPS
- namespace OpenGPS::Schemas
- namespace std
- namespace xml_schema

13.28 environment.cxx File Reference

```
#include "environment.hxx"
```

```
#include "stdafx.hxx"
```

Include dependency graph for environment.cxx:

Defines

- #define _OGPS_DOUBLE_SIZE 8
- #define _OGPS_FLOAT_SIZE 4
- #define _OGPS_INT_SIZE 4
- #define _OGPS_SHORT_SIZE 2

13.28.1 Define Documentation

13.28.1.1 #define _OGPS_DOUBLE_SIZE 8

Referenced by OpenGPS::Environment::ByteSwap64().

13.28.1.2 #define _OGPS_FLOAT_SIZE 4

Referenced by OpenGPS::Environment::ByteSwap32().

13.28.1.3 #define _OGPS_INT_SIZE 4

Referenced by OpenGPS::Environment::ByteSwap(), and OpenGPS::Environment::ByteSwap32().

13.28.1.4 #define _OGPS_SHORT_SIZE 2

Referenced by OpenGPS::Environment::ByteSwap16().

13.29 environment.hxx File Reference

13.29.1 Detailed Description

An interface to provide access to the environment of different architectures and operating systems.

```
#include <opengps/cxx/opengps.hxx>
```

Include dependency graph for environment.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::Environment

Interface for communicating with the operating system and related subjects.

13.30 exceptions.cxx File Reference

```
#include <opengps/cxx/exceptions.hxx>
#include <opengps/cxx/string.hxx>
#include "stdafx.hxx"
```

Include dependency graph for exceptions.cxx:

13.31 exceptions.hxx File Reference

13.31.1 Detailed Description

Exception types thrown within this software library.

```
#include <exception>
#include <opengps/cxx/opengps.hxx>
#include <opengps/messages.h>
#include <opengps/cxx/string.hxx>
```

Include dependency graph for exceptions.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::Exception
  Describes a general exception.

13.32 float_data_point_parser.cxx File Reference

```
#include "float_data_point_parser.hxx"
#include "point_vector_reader_context.hxx"
#include "point_vector_writer_context.hxx"
#include <opengps/cxx/data_point.hxx>
#include "stdafx.hxx"
```

Include dependency graph for float_data_point_parser.cxx:

13.33 float_data_point_parser.hxx File Reference

13.33.1 Detailed Description

A data point parser for point data of type OGPS_FloatPointType.

```
#include "data_point_parser.hxx"
```

Include dependency graph for float_data_point_parser.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::FloatDataPointParser
  Reads/Writes instances of OpenGPS::DataPoint of OGPS_Float point data.

13.34 float_point_buffer.cxx File Reference

```
#include "float_point_buffer.hxx"
#include "stdafx.hxx"
```

Include dependency graph for float_point_buffer.cxx:

13.35 float_point_buffer.hxx File Reference

13.35.1 Detailed Description

Allocate static memory to store point data.

`#include "point_buffer.hxx"`

Include dependency graph for float_point_buffer.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::FloatPointBuffer

  Manages static memory and typesafe access.

13.36 IncRev.c File Reference

13.36.1 Detailed Description

Tool to increment version numbers in text files.

(C) by Georg Wiora 1990/2008

Based on an old Amiga Ansi-C program by Georg Wiora from 3rd January 1990.

This program is published under the Boost Software License V1.0 (s. below)

Usage: Syntax: IncRev <File> <Pattern> [<Increment>] The specified <File> must contain a line with the given pattern.

- Example:

  "textexttextpattern 123"

  IncRev will search "pattern" (case sensitive), read the number and use the space after pattern to the last digit to replace it with the incremented number, in this example 124.

The substitution happens in place without using a temp file! The optional <Increment> parameter is a number to add to the current version number. Default is 1.

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

Include dependency graph for IncRev.c:

Defines

- #define REVISION 0

  Minor Version or Revision number of IncRev.

- #define VERSION 2

  Version number of IncRev.

Functions

- int findline (FILE *fp, const char *pat)

  Search for line with pattern in file.

- int main (const int argc, const char *argv[ ])

  Increment a version number in a text file.

13.36.2 Define Documentation

13.36.2.1 #define REVISION 0

Minor Version or Revision number of IncRev.

Referenced by main().

13.36.2.2 #define VERSION 2

Version number of IncRev.

Referenced by main().

13.36.3 Function Documentation

13.36.3.1 int findline (FILE * *fp*, const char * *pat*)

Search for line with pattern in file.

Position the file pointer at the end of pattern.

- `fp` File pointer to the file to search.
- `pat` pointer to the pattern string that is searched in file.

Returns:

the number of characters available for the incremented number. The file pointer is repostioned right after the pattern

References NULL.

Referenced by main().

13.36.3.2 int main (const int *argc*, const char * *argv*[ ])

Increment a version number in a text file.

- Usage: IncRev <File> <Pattern> [<Increment>]

  The specified <File> must contain a line with the given pattern.
- Example:

  "textexttextpattern 123"

  IncRev will search "pattern" (case sensitive), read the number and use the space after pattern to the last digit to replace it with the incremented number, in this example 124.

The substitution happens in place without using a temp file! The optional <Increment> parameter is a number to add to the current version number. Default is 1.

- `argc` Argument count.
- `argv` List of argument strings.

Returns:

0 for successful run and 1 for errors. Error messages are printed to `stderr` stream

References findline(), NULL, REVISION, and VERSION.

Here is the call graph for this function:

13.37 info.cxx File Reference

```
#include <opengps/cxx/info.hxx>
#include <opengps/cxx/string.hxx>
#include <iostream>
#include "stdafx.hxx"
```

Include dependency graph for info.cxx:

13.38 info.h File Reference

13.38.1 Detailed Description

Copyright and license information.

```
#include <opengps/opengps.h>
```

Include dependency graph for info.h:

This graph shows which files directly or indirectly include this file:

Functions

- _OPENGPS_EXPORT size_t ogps_GetAboutInfo (OGPS_Character *const text, const size_t size)

  Gets a short message describing the purpose of this software library.

- _OPENGPS_EXPORT size_t ogps_GetCopyrightInfo (OGPS_Character *const text, const size_t size)

  Gets the copyright information of this software library.

- _OPENGPS_EXPORT size_t ogps_GetLicenseInfo (OGPS_Character *const text, const size_t size)

  Gets the license information of this software library.

- _OPENGPS_EXPORT size_t ogps_GetNameInfo (OGPS_Character *const text, const size_t size)

  Gets a short name identifying this software library.

- _OPENGPS_EXPORT size_t ogps_GetVersionInfo (OGPS_Character *const text, const size_t size)

  Gets the version identifier of this software library.

- _OPENGPS_EXPORT void ogps_PrintCopyrightInfo ()

  Prints the copyright information of this software library to standard output.

- _OPENGPS_EXPORT void ogps_PrintLicenseInfo ()

  Prints the license information of this software library to standard output.

- _OPENGPS_EXPORT void ogps_PrintVersionInfo ()

  Prints the version identifier of this software library to standard output.

13.39 info.hxx File Reference

13.39.1 Detailed Description

Copyright and license information.

```
#include <opengps/cxx/opengps.hxx>
```

Include dependency graph for info.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::Info

  Publishes license text, ownership and similar information.

13.40 info_c.cxx File Reference

```
#include <opengps/info.h>
#include <opengps/cxx/info.hxx>
#include <opengps/cxx/string.hxx>
#include "messages_c.hxx"
```

Include dependency graph for info_c.cxx:

Functions

- size_t ogps_GetAboutInfo (OGPS_Character *const text, const size_t size) throw ()

  Gets a short message describing the purpose of this software library.

- size_t ogps_GetCopyrightInfo (OGPS_Character *const text, const size_t size) throw ()

  Gets the copyright information of this software library.

- size_t ogps_GetLicenseInfo (OGPS_Character *const text, const size_t size) throw ()

  Gets the license information of this software library.

- size_t ogps_GetNameInfo (OGPS_Character *const text, const size_t size) throw ()

  Gets a short name identifying this software library.

- size_t ogps_GetVersionInfo (OGPS_Character *const text, const size_t size) throw ()

Gets the version identifier of this software library.

- void ogps_PrintCopyrightInfo () throw ()

  Prints the copyright information of this software library to standard output.

- void ogps_PrintLicenseInfo () throw ()

  Prints the license information of this software library to standard output.

- void ogps_PrintVersionInfo () throw ()

  Prints the version identifier of this software library to standard output.

13.41 inline_validity.cxx File Reference

```
#include <limits>
#include "inline_validity.hxx"
#include "point_buffer.hxx"
#include "stdafx.hxx"
```

Include dependency graph for inline_validity.cxx:

13.42 inline_validity.hxx File Reference

13.42.1 Detailed Description

Communicate validity of point vectors through special IEEE754 values.

```
#include <opengps/cxx/opengps.hxx>
#include "point_validity_provider.hxx"
```

Include dependency graph for inline_validity.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::DoubleInlineValidity

  Implements OpenGPS::PointValidityProvider as a lookup of a special IEEE754 value.

- class OpenGPS::FloatInlineValidity

  Implements OpenGPS::PointValidityProvider as a lookup of a special IEEE754 value.

13.43 int16_data_point_parser.cxx File Reference

```
#include "int16_data_point_parser.hxx"
#include "point_vector_reader_context.hxx"
#include "point_vector_writer_context.hxx"
#include <opengps/cxx/data_point.hxx>
#include "stdafx.hxx"
```

Include dependency graph for int16_data_point_parser.cxx:

13.44 int16_data_point_parser.hxx File Reference

13.44.1 Detailed Description

A data point parser for point data of type OGPS_Int16PointType.

```
#include "data_point_parser.hxx"
```

Include dependency graph for int16_data_point_parser.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::Int16DataPointParser

Reads/Writes instances of OpenGPS::DataPoint of OGPS_Int16 point data.

13.45 int16_point_buffer.cxx File Reference

```
#include "int16_point_buffer.hxx"
#include "stdafx.hxx"
```

Include dependency graph for int16_point_buffer.cxx:

13.46 int16_point_buffer.hxx File Reference

13.46.1 Detailed Description

Allocate static memory to store point data.

```
#include "point_buffer.hxx"
```

Include dependency graph for int16_point_buffer.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::Int16PointBuffer

  Manages static memory and typesafe access.

13.47 int32_data_point_parser.cxx File Reference

```
#include "int32_data_point_parser.hxx"
#include "point_vector_reader_context.hxx"
#include "point_vector_writer_context.hxx"
#include <opengps/cxx/data_point.hxx>
#include "stdafx.hxx"
```

Include dependency graph for int32_data_point_parser.cxx:

13.48 int32_data_point_parser.hxx File Reference

13.48.1 Detailed Description

A data point parser for point data of type OGPS_Int32PointType.

```
#include "data_point_parser.hxx"
```

Include dependency graph for int32_data_point_parser.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::Int32DataPointParser

  Reads/Writes instances of OpenGPS::DataPoint of OGPS_Int32 point data.

13.49 int32_point_buffer.cxx File Reference

```
#include "int32_point_buffer.hxx"
#include "stdafx.hxx"
```

Include dependency graph for int32_point_buffer.cxx:

13.50 int32_point_buffer.hxx File Reference

13.50.1 Detailed Description

Allocate static memory to store point data.

`#include "point_buffer.hxx"`

Include dependency graph for int32_point_buffer.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::Int32PointBuffer

  Manages static memory and typesafe access.

13.51 ISO5436-sample1.x3p File Reference

13.51.1 Detailed Description

Simple matrix example with documented XML-main file.

This file gives a very simple example with a fully documented xml-main file. The profile values are stored in a 4 by 4 by 1 matrix. To view the contents of the file, extract it by using any unzip-tool. The following files are contained in this container:

- main.xml The main document xml-file.
- md5checksum.hex The MD5 checksum of the main document.

13.52 ISO5436-sample1_bin.x3p File Reference

13.52.1 Detailed Description

Simple binary matrix example with documented XML-main file.

This file contains the same data as ISO5436-sample1.x3p but uses binary encoding for the profile data. To view the contents of the file, extract it by using any unzip-tool. The following files are contained in this container:

- main.xml The main document xml-file.
- md5checksum.hex The MD5 checksum of the main document.
- bindata/data.bin Binary encoded profile data
- bindata/valid.bin Validity matrix for profile data

13.53 ISO5436-sample2.x3p File Reference

13.53.1 Detailed Description

Simple unordered list example with documented XML-main file.

This file gives a very simple example with a fully documented xml-main file. The 4 by 4 profile values are stored in an unordered list. To view the contents of the file, extract it by using any unzip-tool.

- main.xml The main document xml-file.
- md5checksum.hex The MD5 checksum of the main document.

13.54 ISO5436-sample3.x3p File Reference

13.54.1 Detailed Description

Matrix example mixed incremental/absolute axis.

This file gives a little more complicate example with a fully documented xml-main file. The 4 by 4 profile values are stored with an incremental x-axis an absolute integer y axis and an absolute double z-axis as a 4 by 4 by 1 matrix. To view the contents of the file, extract it by using any unzip-tool.

- main.xml The main document xml-file.
- md5checksum.hex The MD5 checksum of the main document.

13.55 ISO5436-sample4.x3p File Reference

13.55.1 Detailed Description

Matrix example with integer encoded z-values.

This file gives a little more complicate example with a fully documented xml-main file. The 4 by 4 profile values are stored with an incremental x- and y-axis. The z-axis is integer encoded. To view the contents of the file, extract it by using any unzip-tool.

- main.xml The main document xml-file.
- md5checksum.hex The MD5 checksum of the main document.

Variables

- PK b main xml X n

13.55.2 Variable Documentation

13.55.2.1 PK b main xml X n

Referenced by OpenGPS::String::ConvertToMd5(), OpenGPS::Schemas::ISO5436_2::ISO5436_2(), md5_file(), OpenGPS::Schemas::ISO5436_2::operator<<(), OpenGPS::Schemas::ISO5436_2::RotationType::parse(), OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::parse(), OpenGPS::Schemas::ISO5436_2::DataLinkType::parse(), OpenGPS::Schemas::ISO5436_2::DataListType::parse(), OpenGPS::Schemas::ISO5436_2::ProbingSystemType::parse(), OpenGPS::Schemas::ISO5436_2::InstrumentType::parse(), OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::parse(), OpenGPS::Schemas::ISO5436_2::AxesType::parse(), OpenGPS::Schemas::ISO5436_2::Record4Type::parse(), OpenGPS::Schemas::ISO5436_2::Record3Type::parse(), OpenGPS::Schemas::ISO5436_2::Record2Type::parse(), OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::parse(), OpenGPS::Schemas::ISO5436_2::Record1Type::parse(), performanceDouble(), performanceInt16(), OpenGPS::ISO5436_2Container::VerifyChecksum(), and OpenGPS::ISO5436_2Container::WriteVendorSpecific().

13.56 ISO5436-sample4_bin.x3p File Reference

13.56.1 Detailed Description

Matrix example with integer encoded z-values in binary format.

This file contains the same data as ISO5436-sample4.x3p but with binary encoded profile data. The 4 by 4 profile values are stored with an incremental x- and y-axis. The z-axis is integer encoded. To view the contents of the file, extract it by using any unzip-tool.

- main.xml The main document xml-file.
- md5checksum.hex The MD5 checksum of the main document.
- bindata/data.bin Binary encoded profile data
- bindata/valid.bin Validity matrix for profile data

Variables

- PK = I|7 bindata/PK

13.56.2 Variable Documentation

13.56.2.1 PK = I|7 bindata/PK

13.57 iso5436_2.cxx File Reference

```
#include <opengps/cxx/iso5436_2.hxx>
#include "iso5436_2_container.hxx"
#include "stdafx.hxx"
```

Include dependency graph for iso5436_2.cxx:

13.58 iso5436_2.h File Reference

13.58.1 Detailed Description

The abstract data type of the ISO 5436-2 X3P file specification.

It serves as the main interface for communicating with this software library. Methods for opening and creating X3P files, altering or parsing the contained ISO 5436-2 XML document and direct access to the point data are provided. Point data can be accessed either as raw data or geometrically transformed based on the axes descriptions. Also it can be chosen between an iterator interface and indexing techniques. This abstract data type is split into two files because some methods involve C++ types which have no representation in the C interface.

See also:

cxx/iso5436_2_c.hxx

```
#include <opengps/opengps.h>
#include <opengps/point_vector.h>
#include <opengps/point_iterator.h>
```

Include dependency graph for iso5436_2.h:

This graph shows which files directly or indirectly include this file:

Typedefs

- typedef struct _OGPS_ISO5436_2_HANDLE * OGPS_ISO5436_2Handle

  Represents the ISO5436-2 XML X3P file format container.

Functions

- _OPENGPS_EXPORT void ogps_AppendVendorSpecific (const OGPS_ISO5436_2Handle handle, const OGPS_Character *vendorURI, const OGPS_Character *filePath)

  Add vendorspecific file content to the X3P archive.

- _OPENGPS_EXPORT void ogps_CloseISO5436_2 (OGPS_ISO5436_2Handle *handle)

  Closes an OGPS_ISO5436_2Handle file handle and releases its resources.

- _OPENGPS_EXPORT OGPS_PointIteratorPtr ogps_CreateNextPointIterator (const OGPS_-ISO5436_2Handle handle)

  Creates an iterator to access point data contained in an ISO5436-2 X3P file.

- _OPENGPS_EXPORT OGPS_PointIteratorPtr ogps_CreatePrevPointIterator (const OGPS_-ISO5436_2Handle handle)

  Creates an iterator to access point data contained in an ISO5436-2 X3P file.

- _OPENGPS_EXPORT void ogps_GetListCoord (const OGPS_ISO5436_2Handle handle, const OGPS_ULong index, OGPS_Double *const x, OGPS_Double *const y, OGPS_Double *const z)

  Gets the fully transformed value of a data point vector at a given index position.

- _OPENGPS_EXPORT OGPS_ULong ogps_GetListDimension (const OGPS_ISO5436_2Handle handle)

  Gets information on the list dimension of the current data structure.

- _OPENGPS_EXPORT void ogps_GetListPoint (const OGPS_ISO5436_2Handle handle, const OGPS_ULong index, OGPS_PointVectorPtr const vector)

  Gets the raw value of a data point vector at a given index position.

- _OPENGPS_EXPORT void ogps_GetMatrixCoord (const OGPS_ISO5436_2Handle handle, const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w, OGPS_Double *const x, OGPS_-Double *const y, OGPS_Double *const z)

  Gets the fully transformed value of a data point vector at a given surface position.

- _OPENGPS_EXPORT void ogps_GetMatrixDimensions (const OGPS_ISO5436_2Handle handle, OGPS_ULong *const size_u, OGPS_ULong *const size_v, OGPS_ULong *const size_w)

Gets information on the matrix dimensions of the current data structure The dimensions are written to the arguments size_u, size_v, size_w.

- _OPENGPS_EXPORT void ogps_GetMatrixPoint (const OGPS_ISO5436_2Handle handle, const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w, OGPS_PointVectorPtr const vector)

  Gets the raw value of a data point vector at a given surface position.

- _OPENGPS_EXPORT OGPS_Boolean ogps_GetVendorSpecific (const OGPS_ISO5436_2Handle handle, const OGPS_Character ∗vendorURI, const OGPS_Character ∗fileName, const OGPS_Character ∗targetPath)

  Extracts vendorspecific data from the current archive to a given file location.

- _OPENGPS_EXPORT OGPS_Boolean ogps_IsMatrix (const OGPS_ISO5436_2Handle handle)

  Gets information on the structure with which the point measurement data is stored.

- _OPENGPS_EXPORT OGPS_Boolean ogps_IsMatrixCoordValid (const OGPS_ISO5436_2Handle handle, const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w)

  Asks if there is point vector data stored at the given matrix position.

- _OPENGPS_EXPORT OGPS_ISO5436_2Handle ogps_OpenISO5436_2 (const OGPS_Character ∗const file, const OGPS_Character ∗const temp=NULL, const OGPS_Boolean readOnly=FALSE)

  Opens an existing ISO5436-2 XML X3P file.

- _OPENGPS_EXPORT void ogps_SetListPoint (const OGPS_ISO5436_2Handle handle, const OGPS_ULong index, const OGPS_PointVectorPtr vector)

  Sets the value of a three-dimensional data point vector at a given index position.

- _OPENGPS_EXPORT void ogps_SetMatrixPoint (const OGPS_ISO5436_2Handle handle, const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w, const OGPS_PointVectorPtr vector)

  Sets the value of a three-dimensional data point vector at a given surface position.

- _OPENGPS_EXPORT void ogps_WriteISO5436_2 (const OGPS_ISO5436_2Handle handle, const int compressionLevel=-1)

  Writes any changes back to the X3P file.

13.59 iso5436_2.hxx File Reference

13.59.1 Detailed Description

Represents the ISO5436-2 XML X3P file format container.

It serves as the main interface for communication with this software library. Interfaces for opening and creating X3P files, altering or parsing the contained ISO 5436-2 XML documents and direct access to the point data are provided. Point data can be accessed as raw data or geometrically transformed based on the axes descriptions. Also it can be chosen between an iterator interface and indexing techniques.

```
#include <opengps/cxx/opengps.hxx>
```

```
#include <opengps/cxx/exceptions.hxx>
```

```
#include <memory>
```

Include dependency graph for iso5436_2.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS
- namespace OpenGPS::Schemas
- namespace OpenGPS::Schemas::ISO5436_2

Classes

- class OpenGPS::ISO5436_2

  Represents the ISO5436-2 XML X3P file format container.

Typedefs

- typedef std::auto_ptr< PointBuffer > OpenGPS::PointBufferAutoPtr

  std::auto_ptr for usage with OpenGPS::PointBuffer type.

- typedef std::auto_ptr< PointIterator > OpenGPS::PointIteratorAutoPtr

  std::auto_ptr for usage with OpenGPS::PointIterator type.

- typedef std::auto_ptr< PointVectorBase > OpenGPS::PointVectorAutoPtr

  std::auto_ptr for usage with OpenGPS::PointVectorBase type.

- typedef std::auto_ptr< PointVectorParserBuilder > OpenGPS::PointVectorParserBuilderAutoPtr

  std::auto_ptr for usage with OpenGPS::PointVectorParserBuilder type.

- typedef std::auto_ptr< VectorBuffer > OpenGPS::VectorBufferAutoPtr

  std::auto_ptr for usage with OpenGPS::VectorBuffer type.

- typedef std::auto_ptr< VectorBufferBuilder > OpenGPS::VectorBufferBuilderAutoPtr

  std::auto_ptr for usage with OpenGPS::VectorBufferBuilder type.

13.60 iso5436_2_c.cxx File Reference

```
#include <opengps/iso5436_2.h>
#include <opengps/cxx/iso5436_2.hxx>
#include <opengps/cxx/iso5436_2_handle.hxx>
#include "iso5436_2_handle_c.hxx"
#include "point_iterator_c.hxx"
#include "point_vector_c.hxx"
#include "messages_c.hxx"
#include "../cxx/iso5436_2_container.hxx"
#include "../cxx/stdafx.hxx"
```

Include dependency graph for iso5436_2_c.cxx:

Functions

- void ogps_AppendVendorSpecific (const OGPS_ISO5436_2Handle handle, const OGPS_Character *vendorURI, const OGPS_Character *filePath) throw ()

  Add vendorspecific file content to the X3P archive.

- void ogps_CloseISO5436_2 (OGPS_ISO5436_2Handle *handle) throw ()

  Closes an OGPS_ISO5436_2Handle file handle and releases its resources.

- OGPS_ISO5436_2Handle ogps_CreateListISO5436_2 (const OGPS_Character *file, const OGPS_Character *temp, const Schemas::ISO5436_2::Record1Type &record1, const Schemas::ISO5436_2::Record2Type *record2, const OGPS_ULong listDimension, const OGPS_Boolean useBinaryData) throw ()

  Creates a new ISO5436-2 XML X3P file.

- OGPS_ISO5436_2Handle ogps_CreateMatrixISO5436_2 (const OGPS_Character *const file, const OGPS_Character *const temp, const OpenGPS::Schemas::ISO5436_2::Record1Type &record1, const OpenGPS::Schemas::ISO5436_2::Record2Type *record2,

const OpenGPS::Schemas::ISO5436_2::MatrixDimensionType &matrixDimension, const OGPS_-Boolean useBinaryData) throw ()

Creates a new ISO5436-2 XML X3P file.

- OGPS_PointIteratorPtr ogps_CreateNextPointIterator (const OGPS_ISO5436_2Handle handle) throw ()

  Creates an iterator to access point data contained in an ISO5436-2 X3P file.

- OGPS_PointIteratorPtr ogps_CreatePrevPointIterator (const OGPS_ISO5436_2Handle handle) throw ()

  Creates an iterator to access point data contained in an ISO5436-2 X3P file.

- Schemas::ISO5436_2::ISO5436_2Type *const ogps_GetDocument (const OGPS_ISO5436_-2Handle handle) throw ()

  Provides access to the ISO5436_2 XML document.

- void ogps_GetListCoord (const OGPS_ISO5436_2Handle handle, const OGPS_ULong index, OGPS_Double *const x, OGPS_Double *const y, OGPS_Double *const z) throw ()

  Gets the fully transformed value of a data point vector at a given index position.

- OGPS_ULong ogps_GetListDimension (const OGPS_ISO5436_2Handle handle) throw ()

  Gets information on the list dimension of the current data structure.

- void ogps_GetListPoint (const OGPS_ISO5436_2Handle handle, const OGPS_ULong index, OGPS_PointVectorPtr const vector) throw ()

  Gets the raw value of a data point vector at a given index position.

- void ogps_GetMatrixCoord (const OGPS_ISO5436_2Handle handle, const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w, OGPS_Double *const x, OGPS_Double *const y, OGPS_-Double *const z) throw ()

  Gets the fully transformed value of a data point vector at a given surface position.

- void ogps_GetMatrixDimensions (const OGPS_ISO5436_2Handle handle, OGPS_ULong *const size_u, OGPS_ULong *const size_v, OGPS_ULong *const size_w) throw ()

  Gets information on the matrix dimensions of the current data structure The dimensions are written to the arguments size_u, size_v, size_w.

- void ogps_GetMatrixPoint (const OGPS_ISO5436_2Handle handle, const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w, OGPS_PointVectorPtr const vector) throw ()

  Gets the raw value of a data point vector at a given surface position.

- OGPS_Boolean ogps_GetVendorSpecific (const OGPS_ISO5436_2Handle handle, const OGPS_-Character *vendorURI, const OGPS_Character *fileName, const OGPS_Character *targetPath) throw ()

  Extracts vendorspecific data from the current archive to a given file location.

- OGPS_Boolean ogps_IsMatrix (const OGPS_ISO5436_2Handle handle) throw ()

  Gets information on the structure with which the point measurement data is stored.

- OGPS_Boolean ogps_IsMatrixCoordValid (const OGPS_ISO5436_2Handle handle, const OGPS_-ULong u, const OGPS_ULong v, const OGPS_ULong w) throw ()

Asks if there is point vector data stored at the given matrix position.

- OGPS_ISO5436_2Handle ogps_OpenISO5436_2 (const OGPS_Character *const file, const OGPS_Character *const temp, const OGPS_Boolean readOnly) throw ()

  Opens an existing ISO5436-2 XML X3P file.

- void ogps_SetListPoint (const OGPS_ISO5436_2Handle handle, const OGPS_ULong index, const OGPS_PointVectorPtr vector) throw ()

  Sets the value of a three-dimensional data point vector at a given index position.

- void ogps_SetMatrixPoint (const OGPS_ISO5436_2Handle handle, const OGPS_ULong u, const OGPS_ULong v, const OGPS_ULong w, const OGPS_PointVectorPtr vector) throw ()

  Sets the value of a three-dimensional data point vector at a given surface position.

- void ogps_WriteISO5436_2 (const OGPS_ISO5436_2Handle handle, const int compressionLevel) throw ()

  Writes any changes back to the X3P file.

13.61 iso5436_2_container.cxx File Reference

```
#include "iso5436_2_container.hxx"
#include <opengps/cxx/point_vector.hxx>
#include <opengps/cxx/data_point.hxx>
#include "point_vector_parser_builder.hxx"
#include "point_vector_parser.hxx"
#include "xml_point_vector_reader_context.hxx"
#include "xml_point_vector_writer_context.hxx"
#include "binary_lsb_point_vector_reader_context.hxx"
#include "binary_msb_point_vector_reader_context.hxx"
#include "binary_lsb_point_vector_writer_context.hxx"
#include "binary_msb_point_vector_writer_context.hxx"
#include "vector_buffer_builder.hxx"
#include "vector_buffer.hxx"
#include "point_vector_proxy_context_matrix.hxx"
#include "point_vector_proxy_context_list.hxx"
#include "environment.hxx"
#include "point_vector_iostream.hxx"
#include "zip_stream_buffer.hxx"
#include <limits>
#include <iostream>
#include <fstream>
```

```
#include <memory>
#include <iomanip>
#include <sstream>
#include <unzip.h>
#include <zip.h>
#include "../xyssl/md5.h"
#include "stdafx.hxx"
```

Include dependency graph for iso5436_2_container.cxx:

Defines

- #define _OPENGPS_FILE_URI_PREF _T("file:///")
- #define _OPENGPS_ZIP_CHUNK_MAX (256∗1024)

Typedefs

- typedef std::auto_ptr< PointVectorReaderContext > PointVectorReaderContextAutoPtr
- typedef std::auto_ptr< PointVectorWriterContext > PointVectorWriterContextAutoPtr

13.61.1 Define Documentation

13.61.1.1 #define _OPENGPS_FILE_URI_PREF _T("file:///")

Referenced by OpenGPS::ISO5436_2Container::ConfigureNamespaceMap().

13.61.1.2 #define _OPENGPS_ZIP_CHUNK_MAX (256∗1024)

Referenced by OpenGPS::ISO5436_2Container::Decompress(), and OpenGPS::ISO5436_-2Container::WriteVendorSpecific().

13.61.2 Typedef Documentation

13.61.2.1 typedef std::auto_ptr<PointVectorReaderContext> PointVectorReaderContextAutoPtr

13.61.2.2 typedef std::auto_ptr<PointVectorWriterContext> PointVectorWriterContextAutoPtr

13.62 iso5436_2_container.hxx File Reference

13.62.1 Detailed Description

Concrete implementation of the interface of an X3P container.

```
#include <opengps/cxx/iso5436_2.hxx>
#include <opengps/cxx/exceptions.hxx>
#include <opengps/data_point_type.h>
#include "point_vector_proxy_context.hxx"
#include <opengps/cxx/point_iterator.hxx>
#include <opengps/cxx/string.hxx>
#include <opengps/cxx/iso5436_2_xsd.hxx>
#include <zip.h>
#include "auto_ptr_types.hxx"
```

Include dependency graph for iso5436_2_container.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::ISO5436_2Container
  This is the main gate to this software library.
- class OpenGPS::ISO5436_2Container::PointIteratorImpl
  Implementation of the point iterator interface.

13.63 iso5436_2_handle.hxx File Reference

13.63.1 Detailed Description

Enhancing C++ part of the C interface of the abstract data type of the ISO 5436-2 X3P file format.

Since the underlying ISO 5436-2 XML document can be created using a C++ interface only, the C and C++ part of the abstract data type coded mainly in C are split up into two files.

See also:

iso5436_2.h

```
#include <opengps/cxx/opengps.hxx>
```

Include dependency graph for iso5436_2_handle.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS
- namespace OpenGPS::Schemas
- namespace OpenGPS::Schemas::ISO5436_2

Functions

- _OPENGPS_EXPORT OGPS_ISO5436_2Handle ogps_CreateListISO5436_2 (const OGPS_Character *const file, const OGPS_Character *const temp, const OpenGPS::Schemas::ISO5436_2::Record1Type &record1, const OpenGPS::Schemas::ISO5436_2::Record2Type *record2, const OGPS_ULong listDimension, const OGPS_Boolean useBinaryData=TRUE) throw ()

  Creates a new ISO5436-2 XML X3P file.

- _OPENGPS_EXPORT OGPS_ISO5436_2Handle ogps_CreateMatrixISO5436_2 (const OGPS_Character *const file, const OGPS_Character *const temp, const OpenGPS::Schemas::ISO5436_2::Record1Type &record1, const OpenGPS::Schemas::ISO5436_2::Record2Type *record2, const OpenGPS::Schemas::ISO5436_2::MatrixDimensionType &matrixDimension, const OGPS_Boolean useBinaryData=TRUE) throw ()

  Creates a new ISO5436-2 XML X3P file.

- _OPENGPS_EXPORT OpenGPS::Schemas::ISO5436_2::ISO5436_2Type *const ogps_GetDocument (const OGPS_ISO5436_2Handle handle) throw ()

  Provides access to the ISO5436_2 XML document.

13.64 iso5436_2_handle_c.hxx File Reference

13.64.1 Detailed Description

Handle mechanism which makes a C++ OpenGPS::ISO5436_2 object accessible through the corresponding C interface of an ISO5436-2 X3P file container.

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- struct OGPS_ISO5436_2

  Encapsulates the internal C++ structure of an ISO5436-2 handle used within the C interface.

Typedefs

- typedef struct OGPS_ISO5436_2 ∗ OGPS_ISO5436_2Handle

13.65 ISO5436_2_XML_Demo.cxx File Reference

13.65.1 Detailed Description

Example programm demonstrating some of the features of the X3P-data format.

```
#include <opengps/cxx/opengps.hxx>
#include <opengps/iso5436_2.h>
#include <opengps/cxx/iso5436_2.hxx>
#include <opengps/cxx/iso5436_2_handle.hxx>
#include <opengps/cxx/iso5436_2_xsd.hxx>
#include <opengps/cxx/point_iterator.hxx>
#include <opengps/cxx/point_vector.hxx>
#include <opengps/cxx/data_point.hxx>
#include <opengps/cxx/string.hxx>
#include <opengps/cxx/info.hxx>
#include <string>
```

```
#include <iostream>
#include <ostream>
#include <fstream>
#include <cstdlib>
#include <ctime>
#include <limits>
#include <tchar.h>
```

Include dependency graph for ISO5436_2_XML_Demo.cxx:

Functions

- int _cdecl _tmain (int argc, _TCHAR *argv[ ])
- void convertFormat (OpenGPS::String srcFileName, OpenGPS::String dstFileName, const OGPS_Boolean dstFormatIsBinary)
- void mediumComplexExample (OpenGPS::String fileName)
- void performanceDouble (OpenGPS::String fileName, OGPS_ULong dimension, OGPS_Boolean binary)
- void performanceInt16 (OpenGPS::String fileName, OGPS_ULong dimension, OGPS_Boolean binary)

  Performance test using Int16 z-Data.

- OGPS_Boolean PrintDimensions (const OGPS_ISO5436_2Handle handle)

  Prints list or matrix dimension of a given x3p handle to cout.

- void readonlyExample (OpenGPS::String fileName)
- void readonlyExample2 (OpenGPS::String fileName)
- void readonlyExample3 (OpenGPS::String fileName)
- void readonlyExample4 (OpenGPS::String fileName)
- void readonlyExampleMatrix (OpenGPS::String fileName)
- void simpleExample (OpenGPS::String fileName)

  Creates and writes a simple ISO5436-2 XML X3P file with two incremental and one absolute axis (the z-axis).

13.65.2 Function Documentation

13.65.2.1 int _cdecl _tmain (int *argc*, _TCHAR * *argv*[ ])

References FALSE, OpenGPS::Info::GetAbout(), OpenGPS::Info::GetName(), OpenGPS::Info::GetVersion(), mediumComplexExample(), performanceDouble(), performanceInt16(), OpenGPS::Info::PrintCopyright(), OpenGPS::Info::PrintLicense(), OpenGPS::Info::PrintVersion(), readonlyExample(), readonlyExample2(), readonlyExample3(), readonlyExample4(), simpleExample(), OpenGPS::String::ToChar(), and TRUE.

Here is the call graph for this function:

13.65.2.2 void convertFormat (OpenGPS::String *srcFileName*, OpenGPS::String *dstFileName*, const OGPS_Boolean *dstFormatIsBinary*)

References NULL, ogps_CloseISO5436_2(), ogps_CreateListISO5436_2(), ogps_-CreateMatrixISO5436_2(), ogps_CreateNextPointIterator(), ogps_CreatePointVector(), ogps_-FreePointIterator(), ogps_FreePointVector(), ogps_GetCurrentPoint(), ogps_GetDocument(), ogps_-HasError(), ogps_HasNextPoint(), ogps_IsMatrix(), ogps_IsValidPoint(), ogps_MoveNextPoint(), ogps_OpenISO5436_2(), ogps_SetCurrentPoint(), ogps_WriteISO5436_2(), PrintDimensions(), OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1(), OpenGPS::Schemas::ISO5436_-2::ISO5436_2Type::Record2(), OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3(), and TRUE.

Here is the call graph for this function:

13.65.2.3 void mediumComplexExample (OpenGPS::String *fileName*)

References OpenGPS::Schemas::ISO5436_2::Record2Type::Comment(), OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType(), F, NULL, ogps_CloseISO5436_-

2(), ogps_CreateMatrixISO5436_2(), ogps_CreateNextPointIterator(), ogps_CreatePointVector(), OGPS_FEATURE_TYPE_SURFACE_NAME, ogps_FreePointIterator(), ogps_FreePointVector(), ogps_HasError(), OGPS_ISO5436_2000_REVISION_NAME, ogps_MoveNextPoint(), ogps_-SetCurrentPoint(), ogps_SetDoubleZ(), ogps_SetFloatY(), ogps_SetInt32X(), ogps_WriteISO5436_2(), PrintDimensions(), and TRUE.

Referenced by _tmain().

Here is the call graph for this function:

13.65.2.4 void performanceDouble (OpenGPS::String *fileName*, OGPS_ULong *dimension*, OGPS_Boolean *binary*)

References OpenGPS::Schemas::ISO5436_2::Record2Type::Comment(), OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType(), n, NULL, ogps_CloseISO5436_2(), ogps_CreateListISO5436_2(), ogps_CreatePointVector(), OGPS_FEATURE_TYPE_PROFILE_NAME, ogps_FreePointVector(), OGPS_ISO5436_2000_REVISION_NAME, ogps_SetDoubleZ(), ogps_-SetListPoint(), ogps_WriteISO5436_2(), and PrintDimensions().

Referenced by _tmain().

Here is the call graph for this function:

13.65.2.5 void performanceInt16 (OpenGPS::String *fileName*, OGPS_ULong *dimension*, OGPS_-Boolean *binary*)

Performance test using Int16 z-Data.

References OpenGPS::Schemas::ISO5436_2::Record2Type::Comment(), OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType(), n, NULL, ogps_CloseISO5436_-2(), ogps_CreateListISO5436_2(), ogps_CreatePointVector(), OGPS_FEATURE_TYPE_PROFILE_-NAME, ogps_FreePointVector(), OGPS_ISO5436_2000_REVISION_NAME, ogps_SetInt16Z(), ogps_SetListPoint(), ogps_WriteISO5436_2(), and PrintDimensions().

Referenced by _tmain().

Here is the call graph for this function:

13.65.2.6 OGPS_Boolean PrintDimensions (const OGPS_ISO5436_2Handle *handle*)

Prints list or matrix dimension of a given x3p handle to cout.

Parameters:

handle;

Returns:

TRUE on success, FALSE if handle is neither a list nor a matrix.

References _ASSERT, FALSE, ogps_GetListDimension(), ogps_GetMatrixDimensions(), ogps_-HasError(), ogps_IsMatrix(), and TRUE.

Referenced by convertFormat(), mediumComplexExample(), performanceDouble(), performanceInt16(), readonlyExample(), readonlyExample2(), readonlyExampleMatrix(), and simpleExample().

Here is the call graph for this function:

13.65.2.7 void readonlyExample (OpenGPS::String *fileName*)

References NULL, ogps_CloseISO5436_2(), ogps_CreateNextPointIterator(), ogps_CreatePointVector(), ogps_FreePointIterator(), ogps_FreePointVector(), ogps_GetCurrentPoint(), ogps_GetXYZ(), ogps_-HasError(), ogps_HasNextPoint(), ogps_IsValidPoint(), ogps_MoveNextPoint(), ogps_OpenISO5436_2(), PrintDimensions(), OpenGPS::String::ToChar(), and TRUE.

Referenced by _tmain().

Here is the call graph for this function:

13.65.2.8 void readonlyExample2 (OpenGPS::String *fileName*)

References _ASSERT, NULL, ogps_CloseISO5436_2(), ogps_CreateNextPointIterator(), ogps_-CreatePointVector(), OGPS_DoublePointType, OGPS_FloatPointType, ogps_FreePointIterator(), ogps_FreePointVector(), ogps_GetCurrentPoint(), ogps_GetDocument(), ogps_GetDoubleX(), ogps_-GetDoubleY(), ogps_GetDoubleZ(), ogps_GetFloatX(), ogps_GetFloatY(), ogps_GetFloatZ(), ogps_GetInt16X(), ogps_GetInt16Y(), ogps_GetInt16Z(), ogps_GetInt32X(), ogps_GetInt32Y(), ogps_GetInt32Z(), ogps_GetPointTypeX(), ogps_GetPointTypeY(), ogps_GetPointTypeZ(), ogps_-HasError(), ogps_HasNextPoint(), OGPS_Int16PointType, OGPS_Int32PointType, ogps_IsValidPoint(), ogps_MoveNextPoint(), ogps_OpenISO5436_2(), PrintDimensions(), OpenGPS::Schemas::ISO5436_-2::ISO5436_2Type::Record1(), OpenGPS::String::ToChar(), and TRUE.

Referenced by _tmain().

Here is the call graph for this function:

13.65.2.9 void readonlyExample3 (OpenGPS::String *fileName*)

References OpenGPS::ISO5436_2::Close(), OpenGPS::ISO5436_2::CreateNextPointIterator(), OpenGPS::Exception::details(), OpenGPS::ISO5436_2::GetDocument(), OpenGPS::PointVector::GetX(), OpenGPS::PointVector::GetY(), OpenGPS::PointVector::GetZ(), OpenGPS::PointVector::IsValid(), OGPS_DoublePointType, OGPS_FloatPointType, ogps_HasError(), OGPS_Int16PointType, OGPS_-Int32PointType, OpenGPS::ISO5436_2::Open(), OpenGPS::String::ToChar(), and TRUE.

Referenced by _tmain().

Here is the call graph for this function:

13.65.2.10 void readonlyExample4 (OpenGPS::String *fileName*)

References _ASSERT, OpenGPS::ISO5436_2::Close(), OpenGPS::Exception::details(), OpenGPS::ISO5436_2::GetDocument(), OpenGPS::ISO5436_2::GetListPoint(), ogps_HasError(), OpenGPS::ISO5436_2::Open(), OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3(), OpenGPS::ISO5436_2::SetListPoint(), OpenGPS::String::ToChar(), and TRUE.

Referenced by _tmain().

Here is the call graph for this function:

13.65.2.11 void readonlyExampleMatrix (OpenGPS::String *fileName*)

References NULL, ogps_CloseISO5436_2(), ogps_CreateNextPointIterator(), ogps_CreatePointVector(), ogps_FreePointIterator(), ogps_FreePointVector(), ogps_GetCurrentPoint(), ogps_GetMatrixPosition(), ogps_GetXYZ(), ogps_HasError(), ogps_HasNextPoint(), ogps_IsValidPoint(), ogps_MoveNextPoint(), ogps_OpenISO5436_2(), PrintDimensions(), OpenGPS::String::ToChar(), and TRUE.

Here is the call graph for this function:

13.65.2.12 void simpleExample (OpenGPS::String *fileName*)

Creates and writes a simple ISO5436-2 XML X3P file with two incremental and one absolute axis (the z-axis).

Parameters:

fileName Full path to the ISO5436-2 XML X3P to write.

References OpenGPS::Schemas::ISO5436_2::Record2Type::Comment(), OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType(), FALSE, OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment(), NULL, OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Offset(), ogps_AppendVendorSpecific(), ogps_CloseISO5436_2(), ogps_CreateMatrixISO5436_2(), ogps_CreatePointVector(), OGPS_-FEATURE_TYPE_SURFACE_NAME, ogps_FreePointVector(), OGPS_ISO5436_2000_REVISION_-NAME, ogps_SetDoubleZ(), ogps_SetMatrixPoint(), ogps_WriteISO5436_2(), PrintDimensions(), and OpenGPS::String::ToChar().

Referenced by _tmain().

Here is the call graph for this function:

13.66 iso5436_2_xsd.cxx File Reference

```
#include <opengps/opengps.h>
#include <xsd/cxx/pre.hxx>
#include "iso5436_2_xsd.hxx"
#include <xsd/cxx/xml/dom/parsing-source.hxx>
#include <ostream>
#include <istream>
#include <xercesc/framework/Wrapper4InputSource.hpp>
#include <xsd/cxx/xml/sax/std-input-source.hxx>
#include <xsd/cxx/tree/error-handler.hxx>
#include <xsd/cxx/xml/dom/serialization-source.hxx>
#include <xsd/cxx/post.hxx>
```

Include dependency graph for iso5436_2_xsd.cxx:

Namespaces

- namespace OpenGPS
- namespace OpenGPS::Schemas
- namespace OpenGPS::Schemas::ISO5436_2

Functions

- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const RotationType &, const RotationType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const MatrixDimensionType &, const MatrixDimensionType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const DataLinkType &, const DataLinkType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const DataListType &, const DataListType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const ProbingSystemType &, const ProbingSystemType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const InstrumentType &, const InstrumentType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const AxisDescriptionType &, const AxisDescriptionType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const AxesType &, const AxesType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record4Type &, const Record4Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record3Type &, const Record3Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record2Type &, const Record2Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const ISO5436_2Type &, const ISO5436_2Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record1Type &, const Record1Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const Datum &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const Datum &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Datum &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const DataType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const DataType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const DataType &)

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const AxisType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const AxisType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const AxisType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const FeatureType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const FeatureType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const FeatureType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const RotationMatrixElementType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const RotationMatrixElementType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const RotationMatrixElementType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const RotationType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const MatrixDimensionType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const DataLinkType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const DataListType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const ProbingSystemType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const InstrumentType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const AxisDescriptionType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const AxesType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Record4Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Record3Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Record2Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const ISO5436_2Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Record1Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Datum &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, Type::value)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const DataType &)

- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, DataType::value)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const AxisType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, AxisType::value)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const FeatureType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const RotationMatrixElementType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const RotationType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const MatrixDimensionType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const DataLinkType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const DataListType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const ProbingSystemType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const InstrumentType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const AxisDescriptionType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const AxesType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Record4Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Record3Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Record2Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const ISO5436_2Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Record1Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const RotationType &, const RotationType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const MatrixDimensionType &, const MatrixDimensionType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const DataLinkType &, const DataLinkType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const DataListType &, const DataListType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const ProbingSystemType &, const ProbingSystemType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const InstrumentType &, const InstrumentType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const AxisDescriptionType &, const AxisDescriptionType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const AxesType &, const AxesType &)

- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Record4Type &, const Record4Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Record3Type &, const Record3Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Record2Type &, const Record2Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const ISO5436_2Type &, const ISO5436_2Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Record1Type &, const Record1Type &)

Serialization functions for the ISO5436_2 document root.

The only global element: The root node

- _OPENGPS_EXPORT::xsd::cxx::xml::dom::auto_ptr< ::xercesc::DOMDocument > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xml_schema::flags f=0)

  Serialize to a new Xerces-C++ DOM document.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::DOMDocument &d, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x,::xml_schema::flags f=0)

  Serialize to an existing Xerces-C++ DOM document.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::XMLFormatTarget &ft, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xercesc::DOMErrorHandler &eh, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a Xerces-C++ XML format target with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::XMLFormatTarget &ft, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xml_schema::error_handler &eh, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a Xerces-C++ XML format target with an error handler.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::XMLFormatTarget &ft, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a Xerces-C++ XML format target.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::ostream &os, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xercesc::DOMErrorHandler &eh, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a standard output stream with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::ostream &os, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xml_schema::error_handler &eh, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a standard output stream with an error handler.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::ostream &os, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a standard output stream.

Parsing functions for the ISO5436_2 document root.

The only global element: The root node

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::DOMDocument *d,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::xercesc::DOMDocument &d,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::xercesc::DOMInputSource &is,::xercesc::DOMErrorHandler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM input source with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::xercesc::DOMInputSource &is,::xml_schema::error_handler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM input source with an error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::xercesc::DOMInputSource &is,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM input source.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is, const ::std::wstring &id,::xercesc::DOMErrorHandler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is, const ::std::wstring &id,::xml_schema::error_handler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is, const ::std::wstring &id,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is,::xercesc::DOMErrorHandler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is,::xml_schema::error_handler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::std::wstring &uri,::xercesc::DOMErrorHandler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::std::wstring &uri,::xml_schema::error_handler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::std::wstring &uri,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.

13.67 iso5436_2_xsd.hxx File Reference

13.67.1 Detailed Description

Generated from iso5436_2.xsd.

```
#include <opengps/opengps.h>
#include <xsd/cxx/version.hxx>
#include <xsd/cxx/pre.hxx>
#include <xsd/cxx/tree/exceptions.hxx>
#include <xsd/cxx/tree/elements.hxx>
#include <xsd/cxx/tree/types.hxx>
#include <xsd/cxx/xml/error-handler.hxx>
#include <xsd/cxx/tree/parsing.hxx>
#include <xsd/cxx/tree/serialization.hxx>
#include <xsd/cxx/xml/dom/serialization-header.hxx>
```

```
#include <xsd/cxx/tree/std-ostream-operators.hxx>
#include <memory>
#include <algorithm>
#include <xsd/cxx/tree/containers.hxx>
#include <xsd/cxx/tree/list.hxx>
#include <xsd/cxx/xml/dom/parsing-header.hxx>
#include <iosfwd>
#include <xercesc/dom/DOMDocument.hpp>
#include <xercesc/dom/DOMInputSource.hpp>
#include <xercesc/dom/DOMErrorHandler.hpp>
#include <xercesc/framework/XMLFormatter.hpp>
#include <xsd/cxx/xml/dom/auto-ptr.hxx>
#include <xsd/cxx/post.hxx>
```

Include dependency graph for include/opengps/cxx/iso5436_2_xsd.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS
- namespace OpenGPS::Schemas
- namespace OpenGPS::Schemas::ISO5436_2
- namespace xml_schema

Classes

- class OpenGPS::Schemas::ISO5436_2::AxesType

  Class corresponding to the AxesType schema type.

- class OpenGPS::Schemas::ISO5436_2::AxisDescriptionType

  Class corresponding to the AxisDescriptionType schema type.

- class OpenGPS::Schemas::ISO5436_2::AxisType

  Enumeration class corresponding to the AxisType schema type.

- class OpenGPS::Schemas::ISO5436_2::DataLinkType
  Class corresponding to the DataLinkType schema type.
- class OpenGPS::Schemas::ISO5436_2::DataListType
  Class corresponding to the DataListType schema type.
- class OpenGPS::Schemas::ISO5436_2::DataType
  Enumeration class corresponding to the DataType schema type.
- class OpenGPS::Schemas::ISO5436_2::Datum
  Class corresponding to the Datum schema type.
- class OpenGPS::Schemas::ISO5436_2::FeatureType
  Class corresponding to the FeatureType schema type.
- class OpenGPS::Schemas::ISO5436_2::InstrumentType
  Class corresponding to the InstrumentType schema type.
- class OpenGPS::Schemas::ISO5436_2::ISO5436_2Type
  Class corresponding to the ISO5436_2Type schema type.
- class OpenGPS::Schemas::ISO5436_2::MatrixDimensionType
  Class corresponding to the MatrixDimensionType schema type.
- class OpenGPS::Schemas::ISO5436_2::ProbingSystemType
  Class corresponding to the ProbingSystemType schema type.
- class OpenGPS::Schemas::ISO5436_2::Record1Type
  Class corresponding to the Record1Type schema type.
- class OpenGPS::Schemas::ISO5436_2::Record2Type
  Class corresponding to the Record2Type schema type.
- class OpenGPS::Schemas::ISO5436_2::Record3Type
  Class corresponding to the Record3Type schema type.
- class OpenGPS::Schemas::ISO5436_2::Record4Type
  Class corresponding to the Record4Type schema type.
- class OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType
  Class corresponding to the RotationMatrixElementType schema type.
- class OpenGPS::Schemas::ISO5436_2::RotationType
  Class corresponding to the RotationType schema type.
- class OpenGPS::Schemas::ISO5436_2::Type
  Enumeration class corresponding to the Type schema type.

Typedefs

- typedef ::xsd::cxx::tree::base64_binary< wchar_t, simple_type > xml_schema::base64_binary
- typedef bool xml_schema::boolean
- typedef ::xsd::cxx::tree::bounds< wchar_t > xml_schema::bounds
- typedef ::xsd::cxx::tree::buffer< wchar_t > xml_schema::buffer
- typedef signed char xml_schema::byte
- typedef ::xsd::cxx::tree::date< wchar_t, simple_type > xml_schema::date
- typedef ::xsd::cxx::tree::date_time< wchar_t, simple_type > xml_schema::date_time
- typedef ::xsd::cxx::tree::day< wchar_t, simple_type > xml_schema::day
- typedef double xml_schema::decimal
- typedef ::xsd::cxx::tree::diagnostics< wchar_t > xml_schema::diagnostics
- typedef double xml_schema::double_
- typedef ::xsd::cxx::tree::duplicate_id< wchar_t > xml_schema::duplicate_id
- typedef ::xsd::cxx::tree::duration< wchar_t, simple_type > xml_schema::duration
- typedef ::xsd::cxx::tree::entities< wchar_t, simple_type, entity > xml_schema::entities
- typedef ::xsd::cxx::tree::entity< wchar_t, ncname > xml_schema::entity
- typedef ::xsd::cxx::tree::error< wchar_t > xml_schema::error
- typedef ::xsd::cxx::xml::error_handler< wchar_t > xml_schema::error_handler
- typedef ::xsd::cxx::tree::exception< wchar_t > xml_schema::exception
- typedef ::xsd::cxx::tree::expected_attribute< wchar_t > xml_schema::expected_attribute
- typedef ::xsd::cxx::tree::expected_element< wchar_t > xml_schema::expected_element
- typedef ::xsd::cxx::tree::expected_text_content< wchar_t > xml_schema::expected_text_content
- typedef ::xsd::cxx::tree::flags xml_schema::flags
- typedef float xml_schema::float_
- typedef ::xsd::cxx::tree::hex_binary< wchar_t, simple_type > xml_schema::hex_binary
- typedef ::xsd::cxx::tree::id< wchar_t, ncname > xml_schema::id
- typedef ::xsd::cxx::tree::idref< type, wchar_t, ncname > xml_schema::idref
- typedef ::xsd::cxx::tree::idrefs< wchar_t, simple_type, idref > xml_schema::idrefs
- typedef int xml_schema::int_
- typedef long long xml_schema::integer
- typedef ::xsd::cxx::tree::language< wchar_t, token > xml_schema::language
- typedef long long xml_schema::long_
- typedef ::xsd::cxx::tree::month< wchar_t, simple_type > xml_schema::month
- typedef ::xsd::cxx::tree::month_day< wchar_t, simple_type > xml_schema::month_day
- typedef ::xsd::cxx::tree::name< wchar_t, token > xml_schema::name
- typedef ::xsd::cxx::xml::dom::namespace_info< wchar_t > xml_schema::namespace_info
- typedef ::xsd::cxx::xml::dom::namespace_infomap< wchar_t > xml_schema::namespace_infomap
- typedef ::xsd::cxx::tree::ncname< wchar_t, name > xml_schema::ncname
- typedef integer xml_schema::negative_integer
- typedef ::xsd::cxx::tree::nmtoken< wchar_t, token > xml_schema::nmtoken
- typedef ::xsd::cxx::tree::nmtokens< wchar_t, simple_type, nmtoken > xml_schema::nmtokens
- typedef ::xsd::cxx::tree::no_namespace_mapping< wchar_t > xml_schema::no_namespace_-mapping
- typedef ::xsd::cxx::tree::no_prefix_mapping< wchar_t > xml_schema::no_prefix_mapping
- typedef ::xsd::cxx::tree::no_type_info< wchar_t > xml_schema::no_type_info
- typedef integer xml_schema::non_negative_integer
- typedef integer xml_schema::non_positive_integer
- typedef ::xsd::cxx::tree::normalized_string< wchar_t, string > xml_schema::normalized_string
- typedef ::xsd::cxx::tree::not_derived< wchar_t > xml_schema::not_derived

- typedef ::xsd::cxx::tree::parsing< wchar_t > xml_schema::parsing
- typedef integer xml_schema::positive_integer
- typedef ::xsd::cxx::tree::properties< wchar_t > xml_schema::properties
- typedef ::xsd::cxx::tree::qname< wchar_t, simple_type, uri, ncname > xml_schema::qname
- typedef ::xsd::cxx::tree::serialization< wchar_t > xml_schema::serialization
- typedef ::xsd::cxx::tree::severity xml_schema::severity
- typedef short xml_schema::short_
- typedef ::xsd::cxx::tree::simple_type< type > xml_schema::simple_type
- typedef ::xsd::cxx::tree::string< wchar_t, simple_type > xml_schema::string
- typedef ::xsd::cxx::tree::time< wchar_t, simple_type > xml_schema::time
- typedef ::xsd::cxx::tree::token< wchar_t, normalized_string > xml_schema::token
- typedef ::xsd::cxx::tree::type xml_schema::type
- typedef ::xsd::cxx::tree::unexpected_element< wchar_t > xml_schema::unexpected_element
- typedef ::xsd::cxx::tree::unexpected_enumerator< wchar_t > xml_schema::unexpected_enumerator
- typedef unsigned char xml_schema::unsigned_byte
- typedef unsigned int xml_schema::unsigned_int
- typedef unsigned long long xml_schema::unsigned_long
- typedef unsigned short xml_schema::unsigned_short
- typedef ::xsd::cxx::tree::uri< wchar_t, simple_type > xml_schema::uri
- typedef ::xsd::cxx::tree::xsi_already_in_use< wchar_t > xml_schema::xsi_already_in_use
- typedef ::xsd::cxx::tree::year< wchar_t, simple_type > xml_schema::year
- typedef ::xsd::cxx::tree::year_month< wchar_t, simple_type > xml_schema::year_month

Functions

- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const RotationType &, const RotationType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const MatrixDimensionType &, const MatrixDimensionType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const DataLinkType &, const DataLinkType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const DataListType &, const DataListType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const ProbingSystemType &, const ProbingSystemType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const InstrumentType &, const InstrumentType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const AxisDescriptionType &, const AxisDescriptionType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const AxesType &, const AxesType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record4Type &, const Record4Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record3Type &, const Record3Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record2Type &, const Record2Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const ISO5436_2Type &, const ISO5436_2Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record1Type &, const Record1Type &)

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const Datum &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const Datum &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Datum &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const DataType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const DataType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const DataType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const AxisType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const AxisType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const AxisType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const FeatureType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const FeatureType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const FeatureType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const RotationMatrixElementType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const RotationMatrixElementType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const RotationMatrixElementType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const RotationType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const MatrixDimensionType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const DataLinkType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const DataListType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const ProbingSystemType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const InstrumentType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const AxisDescriptionType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const AxesType &)

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Record4Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Record3Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Record2Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const ISO5436_2Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Record1Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Datum &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, Type::value)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const DataType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, DataType::value)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const AxisType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, AxisType::value)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const FeatureType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const RotationMatrixElementType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const RotationType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const MatrixDimensionType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const DataLinkType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const DataListType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const ProbingSystemType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const InstrumentType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const AxisDescriptionType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const AxesType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Record4Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Record3Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Record2Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const ISO5436_2Type &)

- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Record1Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const RotationType &, const RotationType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const MatrixDimensionType &, const MatrixDimensionType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const DataLinkType &, const DataLinkType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const DataListType &, const DataListType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const ProbingSystemType &, const ProbingSystemType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const InstrumentType &, const InstrumentType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const AxisDescriptionType &, const AxisDescriptionType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const AxesType &, const AxesType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Record4Type &, const Record4Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Record3Type &, const Record3Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Record2Type &, const Record2Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const ISO5436_2Type &, const ISO5436_2Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Record1Type &, const Record1Type &)

Serialization functions for the ISO5436_2 document root.

The only global element: The root node

- _OPENGPS_EXPORT::xsd::cxx::xml::dom::auto_ptr< ::xercesc::DOMDocument > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xml_schema::flags f=0)

  Serialize to a new Xerces-C++ DOM document.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::DOMDocument &d, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x,::xml_schema::flags f=0)

  Serialize to an existing Xerces-C++ DOM document.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::XMLFormatTarget &ft, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xercesc::DOMErrorHandler &eh, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a Xerces-C++ XML format target with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::XMLFormatTarget &ft, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xml_schema::error_handler &eh, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

Serialize to a Xerces-C++ XML format target with an error handler.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::XMLFormatTarget &ft, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a Xerces-C++ XML format target.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::ostream &os, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xercesc::DOMErrorHandler &eh, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a standard output stream with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::ostream &os, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xml_schema::error_handler &eh, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a standard output stream with an error handler.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::ostream &os, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a standard output stream.

Parsing functions for the ISO5436_2 document root.

The only global element: The root node

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::DOMDocument *d,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::xercesc::DOMDocument &d,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::xercesc::DOMInputSource &is,::xercesc::DOMErrorHandler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM input source with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::xercesc::DOMInputSource &is,::xml_schema::error_handler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM input source with an error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::xercesc::DOMInputSource &is,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

Parse a Xerces-C++ DOM input source.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is, const ::std::wstring &id,::xercesc::DOMErrorHandler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is, const ::std::wstring &id,::xml_schema::error_handler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is, const ::std::wstring &id,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is,::xercesc::DOMErrorHandler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is,::xml_schema::error_handler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::std::wstring &uri,::xercesc::DOMErrorHandler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::std::wstring &uri,::xml_schema::error_handler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::std::wstring &uri,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.

Variables

- const XMLCh *const xml_schema::tree_node_key = ::xsd::cxx::tree::user_data_keys::node

13.68 iso5436_2_xsd.hxx File Reference

13.68.1 Detailed Description

Generated from iso5436_2.xsd.

```
#include <opengps/opengps.h>
#include <xsd/cxx/version.hxx>
#include <xsd/cxx/pre.hxx>
#include <xsd/cxx/tree/exceptions.hxx>
#include <xsd/cxx/tree/elements.hxx>
#include <xsd/cxx/tree/types.hxx>
#include <xsd/cxx/xml/error-handler.hxx>
#include <xsd/cxx/tree/parsing.hxx>
#include <xsd/cxx/tree/serialization.hxx>
#include <xsd/cxx/xml/dom/serialization-header.hxx>
#include <xsd/cxx/tree/std-ostream-operators.hxx>
#include <memory>
#include <algorithm>
#include <xsd/cxx/tree/containers.hxx>
#include <xsd/cxx/tree/list.hxx>
#include <xsd/cxx/xml/dom/parsing-header.hxx>
#include <iosfwd>
#include <xercesc/dom/DOMDocument.hpp>
#include <xercesc/dom/DOMInputSource.hpp>
#include <xercesc/dom/DOMErrorHandler.hpp>
#include <xercesc/framework/XMLFormatter.hpp>
#include <xsd/cxx/xml/dom/auto-ptr.hxx>
#include <xsd/cxx/post.hxx>
```

Include dependency graph for src/ISO5436_2_XML/iso5436_2_xsd.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS
- namespace OpenGPS::Schemas
- namespace OpenGPS::Schemas::ISO5436_2
- namespace xml_schema

Classes

- class OpenGPS::Schemas::ISO5436_2::AxesType
  Class corresponding to the AxesType schema type.
- class OpenGPS::Schemas::ISO5436_2::AxisDescriptionType
  Class corresponding to the AxisDescriptionType schema type.
- class OpenGPS::Schemas::ISO5436_2::AxisType
  Enumeration class corresponding to the AxisType schema type.
- class OpenGPS::Schemas::ISO5436_2::DataLinkType
  Class corresponding to the DataLinkType schema type.
- class OpenGPS::Schemas::ISO5436_2::DataListType
  Class corresponding to the DataListType schema type.
- class OpenGPS::Schemas::ISO5436_2::DataType
  Enumeration class corresponding to the DataType schema type.
- class OpenGPS::Schemas::ISO5436_2::Datum
  Class corresponding to the Datum schema type.
- class OpenGPS::Schemas::ISO5436_2::FeatureType
  Class corresponding to the FeatureType schema type.
- class OpenGPS::Schemas::ISO5436_2::InstrumentType
  Class corresponding to the InstrumentType schema type.
- class OpenGPS::Schemas::ISO5436_2::ISO5436_2Type
  Class corresponding to the ISO5436_2Type schema type.
- class OpenGPS::Schemas::ISO5436_2::MatrixDimensionType
  Class corresponding to the MatrixDimensionType schema type.
- class OpenGPS::Schemas::ISO5436_2::ProbingSystemType
  Class corresponding to the ProbingSystemType schema type.
- class OpenGPS::Schemas::ISO5436_2::Record1Type
  Class corresponding to the Record1Type schema type.
- class OpenGPS::Schemas::ISO5436_2::Record2Type
  Class corresponding to the Record2Type schema type.

- class OpenGPS::Schemas::ISO5436_2::Record3Type

  Class corresponding to the Record3Type schema type.

- class OpenGPS::Schemas::ISO5436_2::Record4Type

  Class corresponding to the Record4Type schema type.

- class OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType

  Class corresponding to the RotationMatrixElementType schema type.

- class OpenGPS::Schemas::ISO5436_2::RotationType

  Class corresponding to the RotationType schema type.

- class OpenGPS::Schemas::ISO5436_2::Type

  Enumeration class corresponding to the Type schema type.

Functions

- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const RotationType &, const RotationType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const MatrixDimensionType &, const MatrixDimensionType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const DataLinkType &, const DataLinkType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const DataListType &, const DataListType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const ProbingSystemType &, const ProbingSystemType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const InstrumentType &, const InstrumentType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const AxisDescriptionType &, const AxisDescriptionType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const AxesType &, const AxesType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record4Type &, const Record4Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record3Type &, const Record3Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record2Type &, const Record2Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const ISO5436_2Type &, const ISO5436_2Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record1Type &, const Record1Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const Datum &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const Datum &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Datum &)

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const DataType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const DataType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const DataType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const AxisType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const AxisType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const AxisType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const FeatureType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const FeatureType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const FeatureType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &, const RotationMatrixElementType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &, const RotationMatrixElementType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const RotationMatrixElementType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const RotationType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const MatrixDimensionType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const DataLinkType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const DataListType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const ProbingSystemType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const InstrumentType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const AxisDescriptionType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const AxesType &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Record4Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Record3Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Record2Type &)

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const ISO5436_2Type &)
- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &, const Record1Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Datum &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, Type::value)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const DataType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, DataType::value)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const AxisType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, AxisType::value)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const FeatureType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const RotationMatrixElementType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const RotationType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const MatrixDimensionType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const DataLinkType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const DataListType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const ProbingSystemType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const InstrumentType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const AxisDescriptionType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const AxesType &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Record4Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Record3Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Record2Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const ISO5436_2Type &)
- _OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &, const Record1Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const RotationType &, const RotationType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const MatrixDimensionType &, const MatrixDimensionType &)

- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const DataLinkType &, const DataLinkType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const DataListType &, const DataListType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const ProbingSystemType &, const ProbingSystemType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const InstrumentType &, const InstrumentType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const AxisDescriptionType &, const AxisDescriptionType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const AxesType &, const AxesType &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Record4Type &, const Record4Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Record3Type &, const Record3Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Record2Type &, const Record2Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const ISO5436_2Type &, const ISO5436_2Type &)
- _OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator== (const Record1Type &, const Record1Type &)

Serialization functions for the ISO5436_2 document root.

The only global element: The root node

- _OPENGPS_EXPORT::xsd::cxx::xml::dom::auto_ptr< ::xercesc::DOMDocument > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xml_schema::flags f=0)

  Serialize to a new Xerces-C++ DOM document.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::DOMDocument &d, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x,::xml_schema::flags f=0)

  Serialize to an existing Xerces-C++ DOM document.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::XMLFormatTarget &ft, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xercesc::DOMErrorHandler &eh, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a Xerces-C++ XML format target with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::XMLFormatTarget &ft, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xml_schema::error_handler &eh, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a Xerces-C++ XML format target with an error handler.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::XMLFormatTarget &ft, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

Serialize to a Xerces-C++ XML format target.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::ostream &os, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xercesc::DOMErrorHandler &eh, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a standard output stream with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::ostream &os, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m,::xml_schema::error_handler &eh, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a standard output stream with an error handler.

- _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::ostream &os, const ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type &x, const ::xml_schema::namespace_infomap &m, const ::std::wstring &e=L"UTF-8",::xml_schema::flags f=0)

  Serialize to a standard output stream.

Parsing functions for the ISO5436_2 document root.

The only global element: The root node

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::xercesc::DOMDocument *d,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::xercesc::DOMDocument &d,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::xercesc::DOMInputSource &is,::xercesc::DOMErrorHandler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM input source with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::xercesc::DOMInputSource &is,::xml_schema::error_handler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM input source with an error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::xercesc::DOMInputSource &is,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM input source.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is, const ::std::wstring &id,::xercesc::DOMErrorHandler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is, const ::std::wstring &id,::xml_schema::error_handler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is, const ::std::wstring &id,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is,::xercesc::DOMErrorHandler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is,::xml_schema::error_handler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (::std::istream &is,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::std::wstring &uri,::xercesc::DOMErrorHandler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::std::wstring &uri,::xml_schema::error_handler &eh,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.

- _OPENGPS_EXPORT::std::auto_ptr< ::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type > OpenGPS::Schemas::ISO5436_2::ISO5436_2 (const ::std::wstring &uri,::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.

13.69 md5.c File Reference

```
#include "md5.h"
#include <string.h>
#include <stdio.h>
```

Include dependency graph for md5.c:

Defines

- #define F(x, y, z) (y ^ (x | ~z))
- #define F(x, y, z) (x ^ y ^ z)
- #define F(x, y, z) (y ^ (z & (x ^ y)))
- #define F(x, y, z) (z ^ (x & (y ^ z)))
- #define GET_ULONG_LE(n, b, i)
- #define P(a, b, c, d, k, s, t)
- #define PUT_ULONG_LE(n, b, i)
- #define S(x, n) ((x << n) | ((x & 0xFFFFFFFF) >> (32 - n)))
- #define XYSSL_MD5_C

Functions

- void md5 (unsigned char *input, int ilen, unsigned char output[16])

  Output = MD5(input buffer).

- int md5_file (const char *path, unsigned char output[16])

  Output = MD5(file contents).

- void md5_finish (md5_context *ctx, unsigned char output[16])

  MD5 final digest.

- void md5_hmac (unsigned char *key, int keylen, unsigned char *input, int ilen, unsigned char output[16])

  Output = HMAC-MD5(hmac key, input buffer).

- void md5_hmac_finish (md5_context *ctx, unsigned char output[16])

  MD5 HMAC final digest.

- void md5_hmac_starts (md5_context *ctx, unsigned char *key, int keylen)

  MD5 HMAC context setup.

- void md5_hmac_update (md5_context *ctx, unsigned char *input, int ilen)

  MD5 HMAC process buffer.

- static void md5_process (md5_context *ctx, const unsigned char data[64])
- void md5_starts (md5_context *ctx)

  MD5 context setup.

- void md5_update (md5_context *ctx, const unsigned char *input, int ilen)

  MD5 process buffer.

Variables

- static const unsigned char md5_padding [64]

13.69.1 Define Documentation

13.69.1.1 #define F(x, y, z) (y ^ (x | ~z))

13.69.1.2 #define F(x, y, z) (x ^ y ^ z)

13.69.1.3 #define F(x, y, z) (y ^ (z & (x ^ y)))

13.69.1.4 #define F(x, y, z) (z ^ (x & (y ^ z)))

Referenced by mediumComplexExample().

13.69.1.5 #define GET_ULONG_LE(n, b, i)

Value:

```
{                                                       \
    (n) = ( (unsigned long) (b)[(i)     ]       )       \
        | ( (unsigned long) (b)[(i) + 1] <<  8 )        \
        | ( (unsigned long) (b)[(i) + 2] << 16 )        \
        | ( (unsigned long) (b)[(i) + 3] << 24 );       \
}
```

Referenced by md5_process().

13.69.1.6 #define P(a, b, c, d, k, s, t)

Value:

```
{                                                       \
    a += F(b,c,d) + X[k] + t; a = S(a,s) + b;           \
}
```

Referenced by md5_process().

13.69.1.7 #define PUT_ULONG_LE(n, b, i)

Value:

```
{                                                       \
    (b)[(i)     ] = (unsigned char) ( (n)       );      \
    (b)[(i) + 1] = (unsigned char) ( (n) >>  8 );       \
    (b)[(i) + 2] = (unsigned char) ( (n) >> 16 );       \
    (b)[(i) + 3] = (unsigned char) ( (n) >> 24 );       \
}
```

Referenced by md5_finish().

13.69.1.8 #define S(x, n) ((x << n) | ((x & 0xFFFFFFFF) >> (32 - n)))

13.69.1.9 #define XYSSL_MD5_C

13.69.2 Function Documentation

13.69.2.1 void md5 (unsigned char * *input*, int *ilen*, unsigned char *output*[16])

Output = MD5(input buffer).

Parameters:

input buffer holding the data
ilen length of the input data
output MD5 checksum result

References md5_finish(), md5_starts(), and md5_update().

Referenced by md5_hmac_starts(), OpenGPS::ISO5436_2Container::ReadMd5FromFile(), OpenGPS::ISO5436_2Container::SavePointBuffer(), OpenGPS::ISO5436_2Container::SaveValidPointsLink(), OpenGPS::ISO5436_2Container::SaveXmlDocument(), OpenGPS::ISO5436_2Container::VerifyChecksum(), OpenGPS::ISO5436_2Container::VerifyDataBinChecksum(), and OpenGPS::ISO5436_2Container::VerifyValidBinChecksum().

Here is the call graph for this function:

13.69.2.2 int md5_file (const char * *path*, unsigned char *output*[16])

Output = MD5(file contents).

Parameters:

path input file name
output MD5 checksum result

Returns:

0 if successful, 1 if fopen failed, or 2 if fread failed

References md5_finish(), md5_starts(), md5_update(), n, and NULL.

Referenced by OpenGPS::ISO5436_2Container::VerifyChecksum().

Here is the call graph for this function:

13.69.2.3 void md5_finish (md5_context * *ctx*, unsigned char *output*[16])

MD5 final digest.

Parameters:

ctx MD5 context

output MD5 checksum result

References md5_padding, md5_update(), PUT_ULONG_LE, md5_context::state, and md5_context::total.

Referenced by OpenGPS::ZipStreamBuffer::GetMd5(), md5(), md5_file(), and md5_hmac_finish().

Here is the call graph for this function:

13.69.2.4 void md5_hmac (unsigned char * *key*, int *keylen*, unsigned char * *input*, int *ilen*, unsigned char *output*[16])

Output = HMAC-MD5(hmac key, input buffer).

Parameters:

key HMAC secret key

keylen length of the HMAC key

input buffer holding the data

ilen length of the input data

output HMAC-MD5 result

References md5_hmac_finish(), md5_hmac_starts(), and md5_hmac_update().

Here is the call graph for this function:

13.69.2.5 void md5_hmac_finish (md5_context * *ctx*, unsigned char *output*[16])

MD5 HMAC final digest.

Parameters:

ctx HMAC context

output MD5 HMAC checksum result

References md5_finish(), md5_starts(), md5_update(), and md5_context::opad.

Referenced by md5_hmac().

Here is the call graph for this function:

13.69.2.6 void md5_hmac_starts (md5_context * *ctx*, unsigned char * *key*, int *keylen*)

MD5 HMAC context setup.

Parameters:

ctx HMAC context to be initialized

key HMAC secret key

keylen length of the HMAC key

References md5_context::ipad, md5(), md5_starts(), md5_update(), and md5_context::opad.

Referenced by md5_hmac().

Here is the call graph for this function:

13.69.2.7 void md5_hmac_update (md5_context * *ctx*, unsigned char * *input*, int *ilen*)

MD5 HMAC process buffer.

Parameters:

ctx HMAC context

input buffer holding the data

ilen length of the input data

References md5_update().

Referenced by md5_hmac().

Here is the call graph for this function:

13.69.2.8 static void md5_process (md5_context * *ctx*, const unsigned char *data*[64]) [static]

References GET_ULONG_LE, P, and md5_context::state.

Referenced by md5_update().

13.69.2.9 void md5_starts (md5_context * *ctx*)

MD5 context setup.

Parameters:

ctx context to be initialized

References md5_context::state, and md5_context::total.

Referenced by OpenGPS::ZipStreamBuffer::GetMd5(), md5(), md5_file(), md5_hmac_finish(), md5_-hmac_starts(), and OpenGPS::ZipStreamBuffer::ZipStreamBuffer().

13.69.2.10 void md5_update (md5_context * *ctx*, const unsigned char * *input*, int *ilen*)

MD5 process buffer.

Parameters:

ctx MD5 context

input buffer holding the data

ilen length of the input data

References md5_context::buffer, md5_process(), and md5_context::total.

Referenced by md5(), md5_file(), md5_finish(), md5_hmac_finish(), md5_hmac_starts(), md5_hmac_-update(), and OpenGPS::ZipStreamBuffer::xsputn().

Here is the call graph for this function:

13.69.3 Variable Documentation

13.69.3.1 const unsigned char md5_padding[64] [static]

Initial value:

```
{
 0x80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
}
```

Referenced by md5_finish().

13.70 md5.h File Reference

13.70.1 Detailed Description

This graph shows which files directly or indirectly include this file:

Classes

- struct md5_context

  MD5 context structure.

Functions

- void md5 (unsigned char ∗input, int ilen, unsigned char output[16])

  Output = MD5(input buffer).

- int md5_file (const char ∗path, unsigned char output[16])

  Output = MD5(file contents).

- void md5_finish (md5_context ∗ctx, unsigned char output[16])

  MD5 final digest.

- void md5_hmac (unsigned char ∗key, int keylen, unsigned char ∗input, int ilen, unsigned char output[16])

  Output = HMAC-MD5(hmac key, input buffer).

- void md5_hmac_finish (md5_context ∗ctx, unsigned char output[16])

  MD5 HMAC final digest.

- void md5_hmac_starts (md5_context ∗ctx, unsigned char ∗key, int keylen)

MD5 HMAC context setup.

- void md5_hmac_update (md5_context *ctx, unsigned char *input, int ilen)

  MD5 HMAC process buffer.

- int md5_self_test (int verbose)

  Checkup routine.

- void md5_starts (md5_context *ctx)

  MD5 context setup.

- void md5_update (md5_context *ctx, const unsigned char *input, int ilen)

  MD5 process buffer.

13.70.2 Function Documentation

13.70.2.1 void md5 (unsigned char * *input*, int *ilen*, unsigned char *output*[16])

Output = MD5(input buffer).

Parameters:

input buffer holding the data

ilen length of the input data

output MD5 checksum result

References md5_finish(), md5_starts(), and md5_update().

Referenced by md5_hmac_starts(), OpenGPS::ISO5436_2Container::ReadMd5FromFile(), OpenGPS::ISO5436_2Container::SavePointBuffer(), OpenGPS::ISO5436_2Container::SaveValidPointsLink(), OpenGPS::ISO5436_2Container::SaveXmlDocument(), OpenGPS::ISO5436_2Container::VerifyChecksum(), OpenGPS::ISO5436_2Container::VerifyDataBinChecksum(), and OpenGPS::ISO5436_2Container::VerifyValidBinChecksum().

Here is the call graph for this function:

13.70.2.2 int md5_file (const char * *path*, unsigned char *output*[16])

Output = MD5(file contents).

Parameters:

path input file name

output MD5 checksum result

Returns:

0 if successful, 1 if fopen failed, or 2 if fread failed

References md5_finish(), md5_starts(), md5_update(), n, and NULL.

Referenced by OpenGPS::ISO5436_2Container::VerifyChecksum().

Here is the call graph for this function:

13.70.2.3 void md5_finish (md5_context * *ctx*, unsigned char *output*[16])

MD5 final digest.

Parameters:

ctx MD5 context

output MD5 checksum result

References md5_padding, md5_update(), PUT_ULONG_LE, md5_context::state, and md5_context::total.

Referenced by OpenGPS::ZipStreamBuffer::GetMd5(), md5(), md5_file(), and md5_hmac_finish().

Here is the call graph for this function:

13.70.2.4 void md5_hmac (unsigned char * *key*, int *keylen*, unsigned char * *input*, int *ilen*, unsigned char *output*[16])

Output = HMAC-MD5(hmac key, input buffer).

Parameters:

key HMAC secret key

keylen length of the HMAC key

input buffer holding the data

ilen length of the input data

output HMAC-MD5 result

References md5_hmac_finish(), md5_hmac_starts(), and md5_hmac_update().

Here is the call graph for this function:

13.70.2.5 void md5_hmac_finish (md5_context * *ctx*, unsigned char *output*[16])

MD5 HMAC final digest.

Parameters:

ctx HMAC context

output MD5 HMAC checksum result

References md5_finish(), md5_starts(), md5_update(), and md5_context::opad.

Referenced by md5_hmac().

Here is the call graph for this function:

13.70.2.6 void md5_hmac_starts (md5_context * *ctx*, unsigned char * *key*, int *keylen*)

MD5 HMAC context setup.

Parameters:

ctx HMAC context to be initialized

key HMAC secret key

keylen length of the HMAC key

References md5_context::ipad, md5(), md5_starts(), md5_update(), and md5_context::opad.

Referenced by md5_hmac().

Here is the call graph for this function:

13.70.2.7 void md5_hmac_update (md5_context * *ctx*, unsigned char * *input*, int *ilen*)

MD5 HMAC process buffer.

Parameters:

ctx HMAC context

input buffer holding the data

ilen length of the input data

References md5_update().

Referenced by md5_hmac().

Here is the call graph for this function:

13.70.2.8 int md5_self_test (int *verbose*)

Checkup routine.

Returns:

0 if successful, or 1 if the test failed

13.70.2.9 void md5_starts (md5_context * *ctx*)

MD5 context setup.

Parameters:

ctx context to be initialized

References md5_context::state, and md5_context::total.

Referenced by OpenGPS::ZipStreamBuffer::GetMd5(), md5(), md5_file(), md5_hmac_finish(), md5_-hmac_starts(), and OpenGPS::ZipStreamBuffer::ZipStreamBuffer().

13.70.2.10 void md5_update (md5_context * *ctx*, const unsigned char * *input*, int *ilen*)

MD5 process buffer.

Parameters:

ctx MD5 context

input buffer holding the data

ilen length of the input data

References md5_context::buffer, md5_process(), and md5_context::total.

Referenced by md5(), md5_file(), md5_finish(), md5_hmac_finish(), md5_hmac_starts(), md5_hmac_-update(), and OpenGPS::ZipStreamBuffer::xsputn().

Here is the call graph for this function:

13.71 messages.h File Reference

13.71.1 Detailed Description

Global handling of error and warning messages.

This wrappes C++ exception handling to be usable in the C interface. See OpenGPS::Exception for details about throwing exceptions in this software library.

```
#include <opengps/opengps.h>
```

Include dependency graph for messages.h:

This graph shows which files directly or indirectly include this file:

Enumerations

- enum OGPS_ExceptionId {
  OGPS_ExNone, OGPS_ExGeneral, OGPS_ExInvalidArgument, OGPS_ExInvalidOperation, OGPS_ExNotImplemented, OGPS_ExOverflow, OGPS_ExWarning }
  Possible failure conditions.

Functions

- _OPENGPS_EXPORT const OGPS_Character * ogps_GetErrorDescription ()

  Gets the last handled detailed error description or NULL.

- _OPENGPS_EXPORT OGPS_ExceptionId ogps_GetErrorId ()

  Gets the type of the last handled error condition.

- _OPENGPS_EXPORT const OGPS_Character * ogps_GetErrorMessage ()

  Possible failure conditions.

- _OPENGPS_EXPORT OGPS_Boolean ogps_HasError ()

  Returns TRUE when the statement executed previously failed.

13.72 messages_c.cxx File Reference

```
#include <opengps/messages.h>
#include "messages_c.hxx"
#include "../cxx/stdafx.hxx"
```

Include dependency graph for messages_c.cxx:

Functions

- const OGPS_Character * ogps_GetErrorDescription ()

  Gets the last handled detailed error description or NULL.

- OGPS_ExceptionId ogps_GetErrorId ()

  Gets the type of the last handled error condition.

- const OGPS_Character * ogps_GetErrorMessage ()

  Possible failure conditions.

- OGPS_Boolean ogps_HasError ()

  Returns TRUE when the statement executed previously failed.

13.73 messages_c.hxx File Reference

13.73.1 Detailed Description

Error and warning messages during application.

```
#include <opengps/cxx/opengps.hxx>
#include <opengps/cxx/exceptions.hxx>
#include <opengps/cxx/string.hxx>
```

Include dependency graph for messages_c.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::ExceptionHistory

  Maintains a history of exceptions.

Defines

- #define _OPENGPS_GENERIC_EXCEPTION_HANDLER(STATEMENT)

Helper for handling thrown exceptions.

- #define _OPENGPS_GENERIC_EXCEPTION_HANDLER_CLEANUP(STATEMENT, CLEANUP_STATEMENT)

  Helper for handling thrown exceptions.

- #define _OPENGPS_GENERIC_EXCEPTION_HANDLER_RETVALBOOL(STATEMENT)

  Helper for handling thrown exceptions.

13.73.2 Define Documentation

13.73.2.1 #define _OPENGPS_GENERIC_EXCEPTION_HANDLER(STATEMENT)

Value:

```
OpenGPS::ExceptionHistory::Reset(); \
   try \
   { \
      STATEMENT; \
   } \
   catch(const OpenGPS::Exception& ex) \
   { \
      OpenGPS::ExceptionHistory::SetLastException(ex); \
   } \
   catch(const std::exception& sx) \
   { \
      OpenGPS::ExceptionHistory::SetLastException(sx); \
   } \
   catch(...) \
   { \
      OpenGPS::ExceptionHistory::SetLastException(); \
   }
```

Helper for handling thrown exceptions.

First resets the history of exceptions, see OpenGPS::ExceptionHistory::Reset. Then executes a statement that is surrounded by a try-catch block. If an exception is thrown within the statement it is added to the history of exceptions, see OpenGPS::ExceptionHistory::SetLastException.

See _OPENGPS_GENERIC_EXCEPTION_HANDLER_RETVALBOOL for statements of boolean return type.

Parameters:

STATEMENT A statement to be executed.

Referenced by ogps_AppendVendorSpecific(), ogps_CloseISO5436_2(), ogps_FreePointIterator(), ogps_FreePointVector(), ogps_GetAboutInfo(), ogps_GetCopyrightInfo(), ogps_GetCurrentPoint(), ogps_GetDocument(), ogps_GetDouble(), ogps_GetDoubleX(), ogps_GetDoubleY(), ogps_GetDoubleZ(), ogps_GetFloat(), ogps_GetFloatX(), ogps_GetFloatY(), ogps_GetFloatZ(), ogps_GetInt16(), ogps_GetInt16X(), ogps_GetInt16Y(), ogps_GetInt16Z(), ogps_GetInt32(), ogps_GetInt32X(), ogps_GetInt32Y(), ogps_GetInt32Z(), ogps_GetLicenseInfo(), ogps_GetListCoord(), ogps_GetListDimension(), ogps_GetListPoint(), ogps_GetMatrixCoord(), ogps_GetMatrixDimensions(), ogps_GetMatrixPoint(), ogps_GetNameInfo(), ogps_GetPointType(), ogps_GetPointTypeX(), ogps_GetPointTypeY(), ogps_GetPointTypeZ(), ogps_GetVendorSpecific(), ogps_GetVersionInfo(), ogps_GetX(), ogps_GetXYZ(), ogps_GetY(), ogps_GetZ(), ogps_IsMatrix(), ogps_IsValidPoint(), ogps_PrintCopyrightInfo(), ogps_PrintLicenseInfo(), ogps_PrintVersionInfo(), ogps_ResetNextPointIterator(),

ogps_ResetPrevPointIterator(), ogps_SetCurrentPoint(), ogps_SetDouble(), ogps_SetDoubleX(), ogps_-SetDoubleY(), ogps_SetDoubleZ(), ogps_SetFloat(), ogps_SetFloatX(), ogps_SetFloatY(), ogps_-SetFloatZ(), ogps_SetInt16(), ogps_SetInt16X(), ogps_SetInt16Y(), ogps_SetInt16Z(), ogps_SetInt32(), ogps_SetInt32X(), ogps_SetInt32Y(), ogps_SetInt32Z(), ogps_SetListPoint(), ogps_SetMatrixPoint(), and ogps_WriteISO5436_2().

13.73.2.2 #define _OPENGPS_GENERIC_EXCEPTION_HANDLER_CLEANUP(STATEMENT, CLEANUP_STATEMENT)

Value:

```
OpenGPS::ExceptionHistory::Reset(); \
   try \
   { \
      STATEMENT; \
   } \
   catch(const OpenGPS::Exception& ex) \
   { \
      CLEANUP_STATEMENT; \
      OpenGPS::ExceptionHistory::SetLastException(ex); \
   } \
   catch(const std::exception& sx) \
   { \
      CLEANUP_STATEMENT; \
      OpenGPS::ExceptionHistory::SetLastException(sx); \
   } \
   catch(...) \
   { \
      CLEANUP_STATEMENT; \
      OpenGPS::ExceptionHistory::SetLastException(); \
   }
```

Helper for handling thrown exceptions.

First resets the history of exceptions, see OpenGPS::ExceptionHistory::Reset. Then executes a statement that is surrounded by a try-catch block. If an exception is thrown within the statement it is added to the history of exceptions, see OpenGPS::ExceptionHistory::SetLastException.

Parameters:

STATEMENT The statement to be executed within the try-block.

CLEANUP_STATEMENT The statement to be executed within the catch-block.

Referenced by ogps_CreateListISO5436_2(), ogps_CreateMatrixISO5436_2(), ogps_-CreateNextPointIterator(), ogps_CreatePrevPointIterator(), and ogps_OpenISO5436_2().

13.73.2.3 #define _OPENGPS_GENERIC_EXCEPTION_HANDLER_-RETVALBOOL(STATEMENT)

Value:

```
OpenGPS::ExceptionHistory::Reset(); \
   try \
   { \
      return (STATEMENT); \
   } \
   catch(const OpenGPS::Exception& ex) \
   { \
      OpenGPS::ExceptionHistory::SetLastException(ex); \
```

```
} \
catch(const std::exception& sx) \
{ \
   OpenGPS::ExceptionHistory::SetLastException(sx); \
} \
catch(...) \
{ \
   OpenGPS::ExceptionHistory::SetLastException(); \
} \
return FALSE;
```

Helper for handling thrown exceptions.

First resets the history of exceptions, see OpenGPS::ExceptionHistory::Reset. Then executes a statement that is surrounded by a try-catch block. If an exception is thrown within the statement it is added to the history of exceptions, see OpenGPS::ExceptionHistory::SetLastException.

Parameters:

STATEMENT A statement with boolean return value to be executed.

Referenced by ogps_GetListPosition(), ogps_GetMatrixPosition(), ogps_HasNextPoint(), ogps_-HasPrevPoint(), ogps_IsMatrixCoordValid(), ogps_MoveNextPoint(), and ogps_MovePrevPoint().

13.74 missing_data_point_parser.cxx File Reference

```
#include "missing_data_point_parser.hxx"
#include "point_vector_reader_context.hxx"
#include "point_vector_writer_context.hxx"
#include <opengps/cxx/data_point.hxx>
#include "stdafx.hxx"
```

Include dependency graph for missing_data_point_parser.cxx:

13.75 missing_data_point_parser.hxx File Reference

13.75.1 Detailed Description

A data point parser for point data of type OGPS_MissingPointType.

```
#include "data_point_parser.hxx"
```

Include dependency graph for missing_data_point_parser.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::MissingDataPointParser

  Reads/Writes instances of OpenGPS::DataPoint of missing point data.

13.76 opengps.h File Reference

13.76.1 Detailed Description

Common define's and typedef's.

See also:

cxx/opengps.hxx

This graph shows which files directly or indirectly include this file:

Defines

- #define _EX_T(x) (x)

  Comprises text used to describe raised exceptions, that might be localized.

- #define _OPENGPS_EXPORT

  Manages export and import of symbols when used as shared library.

- #define FALSE 0

  The FALSE boolean value.

- #define NULL 0

  Initial value for pointers when they do not point to any memory location.

- #define OGPS_FEATURE_TYPE_PROFILE_NAME _T("PRO")

  Gives the name of the profile feature type in an ISO 5436-2 document.

- #define OGPS_FEATURE_TYPE_SURFACE_NAME _T("SUR")

  Gives the name of the surface feature type in an ISO 5436-2 document.

- #define OGPS_ISO5436_2000_REVISION_NAME _T("ISO5436 - 2000")

  Gives the revision name of an ISO 5436-2 document.

- #define TRUE 1

  The TRUE boolean value.

Typedefs

- typedef int OGPS_Boolean

  Holds a boolean value.

- typedef char OGPS_Character

  The current type of characters.

- typedef double OGPS_Double

  Represents measurement data of type double.

- typedef float OGPS_Float

  Represents measurement data of type float.

- typedef short OGPS_Int16

  Represents measurement data of type short.

- typedef int OGPS_Int32

  Represents measurement data of type int.

- typedef unsigned long OGPS_ULong

  Represents most index ranges.

13.77 opengps.hxx File Reference

13.77.1 Detailed Description

Common define's and typedef's.

This file is an enhanced version of opengps.h for C++ maintance.

See also:

opengps.h

```
#include <opengps/opengps.h>
```

Include dependency graph for opengps.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS
- namespace OpenGPS::Schemas

Typedefs

- typedef char OpenGPS::Byte

  Data type of the size of one (possibly signed) byte.

- typedef Byte * OpenGPS::BytePtr

  Pointer to a data type of the size of one (possibly signed) byte.

- typedef char OpenGPS::OGPS_ExceptionChar

  Character type used to provide more information about an exception.

- typedef unsigned char OpenGPS::UnsignedByte

  Data type of the size of one unsigned byte.

- typedef UnsignedByte * OpenGPS::UnsignedBytePtr

  Pointer to a data type of the size of one unsigned byte.

13.78 point_buffer.cxx File Reference

```
#include "point_buffer.hxx"
#include "stdafx.hxx"
#include <stdlib.h>
#include <opengps/cxx/exceptions.hxx>
```

Include dependency graph for point_buffer.cxx:

13.79 point_buffer.hxx File Reference

13.79.1 Detailed Description

Allocate static memory to store point data.

```
#include <opengps/cxx/opengps.hxx>
#include <opengps/data_point_type.h>
```

Include dependency graph for point_buffer.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::PointBuffer

  A static memory buffer that stores all point data belonging to a single axis.

13.80 point_iterator.cxx File Reference

```
#include "iso5436_2_container.hxx"
#include <opengps/cxx/point_vector.hxx>
#include "stdafx.hxx"
```

Include dependency graph for point_iterator.cxx:

13.81 point_iterator.h File Reference

13.81.1 Detailed Description

The abstract data type of a point iterator used for direct manipulation of point data stored in an ISO 5436-2 X3P file.

```
#include <opengps/opengps.h>
#include <opengps/point_vector.h>
```

Include dependency graph for point_iterator.h:

This graph shows which files directly or indirectly include this file:

Typedefs

- typedef struct _OGPS_POINT_ITERATOR * OGPS_PointIteratorPtr

  Interface to a point iterator.

Functions

- _OPENGPS_EXPORT void ogps_FreePointIterator (OGPS_PointIteratorPtr *const iterator)

  Frees the point iterator instance.

- _OPENGPS_EXPORT void ogps_GetCurrentPoint (const OGPS_PointIteratorPtr iterator, OGPS_PointVectorPtr const vector)

  Gets the value of the current point vector.

- _OPENGPS_EXPORT OGPS_Boolean ogps_GetListPosition (const OGPS_PointIteratorPtr iterator, OGPS_ULong *index)

  Gets the current position of the iterator.

- _OPENGPS_EXPORT OGPS_Boolean ogps_GetMatrixPosition (const OGPS_PointIteratorPtr iterator, OGPS_ULong *u, OGPS_ULong *v, OGPS_ULong *w)

  Gets the current position of the iterator in topology coordinates.

- _OPENGPS_EXPORT OGPS_Boolean ogps_HasNextPoint (const OGPS_PointIteratorPtr iterator)

  Asks if there is another point available to iterate.

- _OPENGPS_EXPORT OGPS_Boolean ogps_HasPrevPoint (const OGPS_PointIteratorPtr iterator)

  Asks if there is another point available to iterate.

- _OPENGPS_EXPORT OGPS_Boolean ogps_MoveNextPoint (OGPS_PointIteratorPtr const iterator)

  Moves the iterator forward.

- _OPENGPS_EXPORT OGPS_Boolean ogps_MovePrevPoint (OGPS_PointIteratorPtr const iterator)

  Moves the iterator backward.

- _OPENGPS_EXPORT void ogps_ResetNextPointIterator (OGPS_PointIteratorPtr const iterator)

Resets the iterator to the beginning and turns this iterator instance into a forward iterator.

- _OPENGPS_EXPORT void ogps_ResetPrevPointIterator (OGPS_PointIteratorPtr const iterator)

  Resets the iterator to the beginning and turns this iterator instance into a backward iterator.

- _OPENGPS_EXPORT void ogps_SetCurrentPoint (const OGPS_PointIteratorPtr iterator, const OGPS_PointVectorPtr vector)

  Sets the value of the current point vector.

13.82 point_iterator.hxx File Reference

13.82.1 Detailed Description

Interface to a point iterator used for direct manipulation of point data stored in an ISO 5436-2 X3P file.

```
#include <opengps/cxx/opengps.hxx>
```

Include dependency graph for point_iterator.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::PointIterator

  Interface to a point iterator.

13.83 point_iterator_c.cxx File Reference

```
#include <opengps/point_iterator.h>
#include "point_iterator_c.hxx"
#include "point_vector_c.hxx"
#include "messages_c.hxx"
#include <opengps/cxx/point_iterator.hxx>
#include "../cxx/stdafx.hxx"
```

Include dependency graph for point_iterator_c.cxx:

Functions

- void ogps_FreePointIterator (OGPS_PointIteratorPtr *const iterator) throw ()

  Frees the point iterator instance.

- void ogps_GetCurrentPoint (const OGPS_PointIteratorPtr iterator, OGPS_PointVectorPtr const vector) throw ()

  Gets the value of the current point vector.

- OGPS_Boolean ogps_GetListPosition (const OGPS_PointIteratorPtr iterator, OGPS_ULong *index) throw ()

  Gets the current position of the iterator.

- OGPS_Boolean ogps_GetMatrixPosition (const OGPS_PointIteratorPtr iterator, OGPS_ULong *u, OGPS_ULong *v, OGPS_ULong *w) throw ()

  Gets the current position of the iterator in topology coordinates.

- OGPS_Boolean ogps_HasNextPoint (const OGPS_PointIteratorPtr iterator) throw ()

  Asks if there is another point available to iterate.

- OGPS_Boolean ogps_HasPrevPoint (const OGPS_PointIteratorPtr iterator) throw ()

  Asks if there is another point available to iterate.

- OGPS_Boolean ogps_MoveNextPoint (OGPS_PointIteratorPtr const iterator) throw ()

  Moves the iterator forward.

- OGPS_Boolean ogps_MovePrevPoint (OGPS_PointIteratorPtr const iterator) throw ()
  Moves the iterator backward.
- void ogps_ResetNextPointIterator (OGPS_PointIteratorPtr const iterator) throw ()
  Resets the iterator to the beginning and turns this iterator instance into a forward iterator.
- void ogps_ResetPrevPointIterator (OGPS_PointIteratorPtr const iterator) throw ()
  Resets the iterator to the beginning and turns this iterator instance into a backward iterator.
- void ogps_SetCurrentPoint (const OGPS_PointIteratorPtr iterator, const OGPS_PointVectorPtr vector) throw ()
  Sets the value of the current point vector.

13.84 point_iterator_c.hxx File Reference

13.84.1 Detailed Description

Handle mechanism which makes a C++ OpenGPS::PointIterator object accessible through the corresponding C interface of a point iterator.

```
#include "../cxx/auto_ptr_types.hxx"
```

Include dependency graph for point_iterator_c.hxx:

This graph shows which files directly or indirectly include this file:

Classes

- struct OGPS_PointIterator
  Encapsulates the internal C++ structure of a point iterator handle used within the C interface.

Typedefs

- typedef struct OGPS_PointIterator * OGPS_PointIteratorPtr

13.84.2 Typedef Documentation

13.84.2.1 typedef struct OGPS_PointIterator * OGPS_PointIteratorPtr

13.85 point_validity_provider.cxx File Reference

```
#include "point_validity_provider.hxx"
#include "point_buffer.hxx"
#include "stdafx.hxx"
```

Include dependency graph for point_validity_provider.cxx:

13.86 point_validity_provider.hxx File Reference

13.86.1 Detailed Description

Interface to communicate whether the point vector at a given location contains valid data.

```
#include <opengps/cxx/opengps.hxx>
```

Include dependency graph for point_validity_provider.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::PointValidityProvider

  Interface to communicate the validity of a point vector at a given location.

13.87 point_vector.cxx File Reference

```
#include "data_point_impl.hxx"
#include <opengps/cxx/point_vector.hxx>
#include "stdafx.hxx"
```

Include dependency graph for point_vector.cxx:

13.88 point_vector.h File Reference

13.88.1 Detailed Description

The abstract data type of a point vector which holds values for all components of 3D point data stored in an ISO 5436-2 X3P file.

```
#include <opengps/opengps.h>
#include <opengps/data_point.h>
```

Include dependency graph for point_vector.h:

This graph shows which files directly or indirectly include this file:

Typedefs

- typedef struct _OGPS_POINT_VECTOR * OGPS_PointVectorPtr

  Typesafe representation of three-dimensional point measurement data.

Functions

- _OPENGPS_EXPORT OGPS_PointVectorPtr ogps_CreatePointVector (void)

  Creates an instance of type OGPS_PointVectorPtr.

- _OPENGPS_EXPORT void ogps_FreePointVector (OGPS_PointVectorPtr *const vector)

  Frees an instance of a point vector.

- _OPENGPS_EXPORT OGPS_Double ogps_GetDoubleX (const OGPS_PointVectorPtr vector)

  Gets the value of the x component of the given vector.

- _OPENGPS_EXPORT OGPS_Double ogps_GetDoubleY (const OGPS_PointVectorPtr vector)

  Gets the value of the y component of the given vector.

- _OPENGPS_EXPORT OGPS_Double ogps_GetDoubleZ (const OGPS_PointVectorPtr vector)

  Gets the value of the z component of the given vector.

- _OPENGPS_EXPORT OGPS_Float ogps_GetFloatX (const OGPS_PointVectorPtr vector)

Gets the value of the x component of the given vector.

- _OPENGPS_EXPORT OGPS_Float ogps_GetFloatY (const OGPS_PointVectorPtr vector)

  Gets the value of the y component of the given vector.

- _OPENGPS_EXPORT OGPS_Float ogps_GetFloatZ (const OGPS_PointVectorPtr vector)

  Gets the value of the z component of the given vector.

- _OPENGPS_EXPORT OGPS_Int16 ogps_GetInt16X (const OGPS_PointVectorPtr vector)

  Gets the value of the x component of the given vector.

- _OPENGPS_EXPORT OGPS_Int16 ogps_GetInt16Y (const OGPS_PointVectorPtr vector)

  Gets the value of the y component of the given vector.

- _OPENGPS_EXPORT OGPS_Int16 ogps_GetInt16Z (const OGPS_PointVectorPtr vector)

  Gets the value of the z component of the given vector.

- _OPENGPS_EXPORT OGPS_Int32 ogps_GetInt32X (const OGPS_PointVectorPtr vector)

  Gets the value of the x component of the given vector.

- _OPENGPS_EXPORT OGPS_Int32 ogps_GetInt32Y (const OGPS_PointVectorPtr vector)

  Gets the value of the y component of the given vector.

- _OPENGPS_EXPORT OGPS_Int32 ogps_GetInt32Z (const OGPS_PointVectorPtr vector)

  Gets the value of the z component of the given vector.

- _OPENGPS_EXPORT OGPS_DataPointType ogps_GetPointTypeX (const OGPS_PointVectorPtr vector)

  Gets the type of points of the x component of the given vector.

- _OPENGPS_EXPORT OGPS_DataPointType ogps_GetPointTypeY (const OGPS_PointVectorPtr vector)

  Gets the type of points of the y component of the given vector.

- _OPENGPS_EXPORT OGPS_DataPointType ogps_GetPointTypeZ (const OGPS_PointVectorPtr vector)

  Gets the type of points of the z component of the given vector.

- _OPENGPS_EXPORT OGPS_DataPointPtr const ogps_GetX (OGPS_PointVectorPtr const vector)

  Gets the x component of the given vector.

- _OPENGPS_EXPORT void ogps_GetXYZ (const OGPS_PointVectorPtr vector, OGPS_Double *const x, OGPS_Double *const y, OGPS_Double *const z)

  Gets the values of each component of the given point vector.

- _OPENGPS_EXPORT OGPS_DataPointPtr const ogps_GetY (OGPS_PointVectorPtr const vector)

  Gets the y component of the given vector.

- _OPENGPS_EXPORT OGPS_DataPointPtr const ogps_GetZ (OGPS_PointVectorPtr const vector)

  Gets the z component of the given vector.

- _OPENGPS_EXPORT OGPS_Boolean ogps_IsValidPoint (const OGPS_PointVectorPtr vector)

  Asks if the given point vector stores a valid data point.

- _OPENGPS_EXPORT void ogps_SetDoubleX (OGPS_PointVectorPtr const vector, const OGPS_-Double value)

  Sets the new value for the x component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetDoubleY (OGPS_PointVectorPtr const vector, const OGPS_-Double value)

  Sets the new value for the y component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetDoubleZ (OGPS_PointVectorPtr const vector, const OGPS_-Double value)

  Sets the new value for the z component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetFloatX (OGPS_PointVectorPtr const vector, const OGPS_-Float value)

  Sets the new value for the x component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetFloatY (OGPS_PointVectorPtr const vector, const OGPS_-Float value)

  Sets the new value for the y component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetFloatZ (OGPS_PointVectorPtr const vector, const OGPS_-Float value)

  Sets the new value for the z component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetInt16X (OGPS_PointVectorPtr const vector, const OGPS_-Int16 value)

  Sets the new value for the x component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetInt16Y (OGPS_PointVectorPtr const vector, const OGPS_-Int16 value)

  Sets the new value for the y component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetInt16Z (OGPS_PointVectorPtr const vector, const OGPS_-Int16 value)

  Sets the new value for the z component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetInt32X (OGPS_PointVectorPtr const vector, const OGPS_-Int32 value)

  Sets the new value for the x component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetInt32Y (OGPS_PointVectorPtr const vector, const OGPS_-Int32 value)

  Sets the new value for the y component of a given point vector.

- _OPENGPS_EXPORT void ogps_SetInt32Z (OGPS_PointVectorPtr const vector, const OGPS_-Int32 value)

  Sets the new value for the z component of a given point vector.

13.89 point_vector.hxx File Reference

13.89.1 Detailed Description

Typesafe representation of three-dimensional point measurement data.

A point vector holds all three components of 3D point data stored in an ISO 5436-2 X3P file and provides methods for typesafe access.

```
#include <opengps/cxx/opengps.hxx>
```

```
#include <opengps/cxx/point_vector_base.hxx>
```

Include dependency graph for point_vector.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::PointVector

  Typesafe representation of three-dimensional point measurement data.

13.90 point_vector_base.hxx File Reference

13.90.1 Detailed Description

Very fundamental representation of three-dimensional point measurement data.

A point vector holds all three components of 3D point data stored in an ISO 5436-2 X3P file and provides methods for typesafe access.

```
#include <opengps/cxx/opengps.hxx>
```

Include dependency graph for point_vector_base.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::PointVectorBase

  Typesafe and very fundamental representation of three-dimensional point measurement data.

13.91 point_vector_c.cxx File Reference

```
#include <opengps/point_vector.h>
#include <opengps/cxx/point_vector.hxx>
#include <opengps/cxx/data_point.hxx>
#include "data_point_c.hxx"
#include "point_vector_c.hxx"
#include "messages_c.hxx"
```

```
#include "../cxx/stdafx.hxx"
```

Include dependency graph for point_vector_c.cxx:

Functions

- OGPS_PointVectorPtr ogps_CreatePointVector (void) throw ()

  Creates an instance of type OGPS_PointVectorPtr.

- void ogps_FreePointVector (OGPS_PointVectorPtr *vector) throw ()

  Frees an instance of a point vector.

- OGPS_Double ogps_GetDoubleX (const OGPS_PointVectorPtr vector) throw ()

  Gets the value of the x component of the given vector.

- OGPS_Double ogps_GetDoubleY (const OGPS_PointVectorPtr vector) throw ()

  Gets the value of the y component of the given vector.

- OGPS_Double ogps_GetDoubleZ (const OGPS_PointVectorPtr vector) throw ()

  Gets the value of the z component of the given vector.

- OGPS_Float ogps_GetFloatX (const OGPS_PointVectorPtr vector) throw ()

  Gets the value of the x component of the given vector.

- OGPS_Float ogps_GetFloatY (const OGPS_PointVectorPtr vector) throw ()

  Gets the value of the y component of the given vector.

- OGPS_Float ogps_GetFloatZ (const OGPS_PointVectorPtr vector) throw ()

  Gets the value of the z component of the given vector.

- OGPS_Int16 ogps_GetInt16X (const OGPS_PointVectorPtr vector) throw ()

  Gets the value of the x component of the given vector.

- OGPS_Int16 ogps_GetInt16Y (const OGPS_PointVectorPtr vector) throw ()

  Gets the value of the y component of the given vector.

- OGPS_Int16 ogps_GetInt16Z (const OGPS_PointVectorPtr vector) throw ()

Gets the value of the z component of the given vector.

- OGPS_Int32 ogps_GetInt32X (const OGPS_PointVectorPtr vector) throw ()

  Gets the value of the x component of the given vector.

- OGPS_Int32 ogps_GetInt32Y (const OGPS_PointVectorPtr vector) throw ()

  Gets the value of the y component of the given vector.

- OGPS_Int32 ogps_GetInt32Z (const OGPS_PointVectorPtr vector) throw ()

  Gets the value of the z component of the given vector.

- OGPS_DataPointType ogps_GetPointTypeX (const OGPS_PointVectorPtr vector) throw ()

  Gets the type of points of the x component of the given vector.

- OGPS_DataPointType ogps_GetPointTypeY (const OGPS_PointVectorPtr vector) throw ()

  Gets the type of points of the y component of the given vector.

- OGPS_DataPointType ogps_GetPointTypeZ (const OGPS_PointVectorPtr vector) throw ()

  Gets the type of points of the z component of the given vector.

- OGPS_DataPointPtr const ogps_GetX (OGPS_PointVectorPtr const vector) throw ()

  Gets the x component of the given vector.

- void ogps_GetXYZ (const OGPS_PointVectorPtr vector, OGPS_Double *const x, OGPS_Double *const y, OGPS_Double *const z) throw ()

  Gets the values of each component of the given point vector.

- OGPS_DataPointPtr const ogps_GetY (OGPS_PointVectorPtr const vector) throw ()

  Gets the y component of the given vector.

- OGPS_DataPointPtr const ogps_GetZ (OGPS_PointVectorPtr const vector) throw ()

  Gets the z component of the given vector.

- OGPS_Boolean ogps_IsValidPoint (const OGPS_PointVectorPtr vector) throw ()

  Asks if the given point vector stores a valid data point.

- void ogps_SetDoubleX (OGPS_PointVectorPtr const vector, const OGPS_Double value) throw ()

  Sets the new value for the x component of a given point vector.

- void ogps_SetDoubleY (OGPS_PointVectorPtr const vector, const OGPS_Double value) throw ()

  Sets the new value for the y component of a given point vector.

- void ogps_SetDoubleZ (OGPS_PointVectorPtr const vector, const OGPS_Double value) throw ()

  Sets the new value for the z component of a given point vector.

- void ogps_SetFloatX (OGPS_PointVectorPtr const vector, const OGPS_Float value) throw ()

  Sets the new value for the x component of a given point vector.

- void ogps_SetFloatY (OGPS_PointVectorPtr const vector, const OGPS_Float value) throw ()

  Sets the new value for the y component of a given point vector.

- void ogps_SetFloatZ (OGPS_PointVectorPtr const vector, const OGPS_Float value) throw ()

  Sets the new value for the z component of a given point vector.

- void ogps_SetInt16X (OGPS_PointVectorPtr const vector, const OGPS_Int16 value) throw ()

  Sets the new value for the x component of a given point vector.

- void ogps_SetInt16Y (OGPS_PointVectorPtr const vector, const OGPS_Int16 value) throw ()

  Sets the new value for the y component of a given point vector.

- void ogps_SetInt16Z (OGPS_PointVectorPtr const vector, const OGPS_Int16 value) throw ()

  Sets the new value for the z component of a given point vector.

- void ogps_SetInt32X (OGPS_PointVectorPtr const vector, const OGPS_Int32 value) throw ()

  Sets the new value for the x component of a given point vector.

- void ogps_SetInt32Y (OGPS_PointVectorPtr const vector, const OGPS_Int32 value) throw ()

  Sets the new value for the y component of a given point vector.

- void ogps_SetInt32Z (OGPS_PointVectorPtr const vector, const OGPS_Int32 value) throw ()

  Sets the new value for the z component of a given point vector.

13.92 point_vector_c.hxx File Reference

13.92.1 Detailed Description

Handle mechanism which makes a C++ OpenGPS::PointVector object accessible through the corresponding C interface of a point vector.

```
#include <opengps/cxx/point_vector.hxx>
```

Include dependency graph for point_vector_c.hxx:

This graph shows which files directly or indirectly include this file:

Classes

- struct OGPS_PointVector

  Encapsulates the internal C++ structure of a point vector handle used within the C interface.

Typedefs

- typedef struct OGPS_PointVector * OGPS_PointVectorPtr

13.92.2 Typedef Documentation

13.92.2.1 typedef struct OGPS_PointVector * OGPS_PointVectorPtr

13.93 point_vector_iostream.cxx File Reference

```
#include "point_vector_iostream.hxx"
#include "stdafx.hxx"
#include <opengps/cxx/string.hxx>
```

Include dependency graph for point_vector_iostream.cxx:

13.94 point_vector_iostream.hxx File Reference

13.94.1 Detailed Description

Locale related stuff to support locale invariant parsing of point vector data .

```
#include <opengps/cxx/opengps.hxx>
#include <xlocale>
```

```
#include <sstream>
#include <fstream>
```

Include dependency graph for point_vector_iostream.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::InputBinaryFileStream

  A binary stream class used for reading from binary files.

- class OpenGPS::OutputBinaryFileStream

  A binary stream class used for writing to binary files.

- class OpenGPS::PointVectorInputStringStream

  A string stream used for parsing a point vector as defined in ISO5436-2 XML.

- class OpenGPS::PointVectorInvariantLocale

  The culture invariant locale.

- class OpenGPS::PointVectorOutputStringStream

  A locale invariant string stream used to convert a point vector object to its representation as text according to the ISO5436-2 XML specification.

- class OpenGPS::PointVectorWhitespaceFacet

  Redefines the white space character set to make parsing of a point vector easier.

13.95 point_vector_parser.cxx File Reference

```
#include "point_vector_parser.hxx"
#include "int16_data_point_parser.hxx"
```

```
#include "int32_data_point_parser.hxx"
#include "float_data_point_parser.hxx"
#include "double_data_point_parser.hxx"
#include "missing_data_point_parser.hxx"
#include "point_vector_reader_context.hxx"
#include "point_vector_writer_context.hxx"
#include <opengps/cxx/point_vector_base.hxx>
#include "stdafx.hxx"
```

Include dependency graph for point_vector_parser.cxx:

13.96 point_vector_parser.hxx File Reference

13.96.1 Detailed Description

Generic parser of a point vector.

```
#include <opengps/cxx/opengps.hxx>
#include <opengps/data_point_type.h>
```

Include dependency graph for point_vector_parser.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::PointVectorParser

  Generic parser of a point vector.

13.97 point_vector_parser_builder.cxx File Reference

```
#include "point_vector_parser_builder.hxx"
#include "point_vector_parser.hxx"
#include "stdafx.hxx"
```

Include dependency graph for point_vector_parser_builder.cxx:

13.98 point_vector_parser_builder.hxx File Reference

13.98.1 Detailed Description

Builder which assembles a concrete instance of the generic point vector parser.

```
#include <opengps/cxx/opengps.hxx>
#include <opengps/data_point_type.h>
```

Include dependency graph for point_vector_parser_builder.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::PointVectorParserBuilder

  Component which handles the building process of a specialized parser object for reading and writing typed point data as a three-vector.

13.99 point_vector_proxy.cxx File Reference

```
#include "point_vector_proxy.hxx"
#include "point_vector_proxy_context.hxx"
#include "vector_buffer.hxx"
#include "point_buffer.hxx"
#include "stdafx.hxx"
```

Include dependency graph for point_vector_proxy.cxx:

13.100 point_vector_proxy.hxx File Reference

13.100.1 Detailed Description

Make the internal memory structure of distinct data point buffers accessable as point vector data.

```
#include <opengps/cxx/point_vector_base.hxx>
```

```
#include <opengps/cxx/data_point.hxx>
```

Include dependency graph for point_vector_proxy.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::PointVectorProxy

  Serves one row of the three distinct buffers of data points managed by OpenGPS::VectorBuffer and raises the expression of accessing a single point vector.

- class OpenGPS::PointVectorProxy::DataPointProxy

  Proxies one single data point pointed to by the current context information.

13.101 point_vector_proxy_context.cxx File Reference

```
#include "point_vector_proxy_context.hxx"
#include "stdafx.hxx"
```

Include dependency graph for point_vector_proxy_context.cxx:

13.102 point_vector_proxy_context.hxx File Reference

13.102.1 Detailed Description

Indexing of point data managed by OpenGPS::VectorBuffer.

```
#include <opengps/cxx/opengps.hxx>
```

Include dependency graph for point_vector_proxy_context.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::PointVectorProxyContext

  Indexing of point data managed by OpenGPS::VectorBuffer.

13.103 point_vector_proxy_context_list.cxx File Reference

```
#include "point_vector_proxy_context_list.hxx"
#include <opengps/cxx/exceptions.hxx>
#include "stdafx.hxx"
```

Include dependency graph for point_vector_proxy_context_list.cxx:

13.104 point_vector_proxy_context_list.hxx File Reference

13.104.1 Detailed Description

Indexing of point data managed by OpenGPS::VectorBuffer.

```
#include <opengps/cxx/opengps.hxx>
#include "point_vector_proxy_context.hxx"
```

Include dependency graph for point_vector_proxy_context_list.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::PointVectorProxyContextList

  Indexing of point data managed by OpenGPS::VectorBuffer.

13.105 point_vector_proxy_context_matrix.cxx File Reference

```
#include "point_vector_proxy_context_matrix.hxx"
#include <opengps/cxx/exceptions.hxx>
#include "stdafx.hxx"
```

Include dependency graph for point_vector_proxy_context_matrix.cxx:

13.106 point_vector_proxy_context_matrix.hxx File Reference

13.106.1 Detailed Description

Indexing of point data managed by OpenGPS::VectorBuffer.

```
#include <opengps/cxx/opengps.hxx>
#include "point_vector_proxy_context.hxx"
```

Include dependency graph for point_vector_proxy_context_matrix.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::PointVectorProxyContextMatrix

  Indexing of point data managed by OpenGPS::VectorBuffer.

13.107 point_vector_reader_context.hxx File Reference

13.107.1 Detailed Description

Interface for reading typed point data from an underlying stream of point vectors.

```
#include <opengps/cxx/opengps.hxx>
```

Include dependency graph for point_vector_reader_context.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::PointVectorReaderContext

  Encapsulates an underlying stream of point vector data.

13.108 point_vector_writer_context.hxx File Reference

13.108.1 Detailed Description

Interface for writing typed point data to an underlying stream of point vectors.

`#include <opengps/cxx/opengps.hxx>`

Include dependency graph for point_vector_writer_context.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::PointVectorWriterContext

  Encapsulates an underlying stream of point vector data.

13.109 Sample_Files_Description_for_Doxygen.h File Reference

13.110 stdafx.hxx File Reference

13.110.1 Detailed Description

Standard includes and defines.

```
#include <tchar.h>
```

```
#include "version.h"
```

Include dependency graph for stdafx.hxx:

This graph shows which files directly or indirectly include this file:

Defines

- #define _ASSERT(x) ((void)0)
- #define _OPENGPS_BINFORMAT_DOUBLE_SIZE 8
- #define _OPENGPS_BINFORMAT_FLOAT_SIZE 4
- #define _OPENGPS_BINFORMAT_INT16_SIZE 2
- #define _OPENGPS_BINFORMAT_INT32_SIZE 4
- #define _OPENGPS_DELETE(x) if((x) != NULL) { delete (x); (x) = NULL; }
- #define _OPENGPS_DELETE_ARRAY(x) if((x) != NULL) { delete[ ] (x); (x) = NULL; }
- #define _OPENGPS_DESCRIPTION _T("openGPS class library implementing an xml-version of ISO 5436-2 file format.")

  Short description of the library.

- #define _OPENGPS_ENV_OPENGPS_LOCATION _T("OPENGPS_LOCATION")
- #define _OPENGPS_EXCEPTION_MESG(x) x
- #define _OPENGPS_EXPERIMENTAL _T(" (Experimental Version!)")
- #define _OPENGPS_FREE(x) if((x) != NULL) { free(x); (x) = NULL; }
- #define _OPENGPS_ID _OPENGPS_NAME _T(" (V") _OPENGPS_VERSIONSTRING _T(")")

  Define ID string with programm name and Version.

- #define _OPENGPS_ISO5436_LOCATION _T("iso5436_2.xsd")

- #define _OPENGPS_MSTR(x) #x

  Build a version string from version numbers.

- #define _OPENGPS_NAME _T("openGPS ISO 5436-2 XML") _OPENGPS_EXPERIMENTAL

  Name of the program.

- #define _OPENGPS_VERSIONSTRING _OPENGPS_VERSIONSTRING_M(_OPENGPS_VERSION, _OPENGPS_MINVERSION, _OPENGPS_BUILD, _OPENGPS_REVISION)

  Build a version string.

- #define _OPENGPS_VERSIONSTRING_M(ver, mver, build, rev) _OPENGPS_MSTR(ver) _T(".") _OPENGPS_MSTR(mver) _T(".") _OPENGPS_MSTR(build) _T(".") _T(rev)
- #define _OPENGPS_XSD_ISO5436_DATALINK_PATH _T("bindata/data.bin")
- #define _OPENGPS_XSD_ISO5436_LOCATION _T("http://www.opengps.eu/2008/ISO5436_2/ISO5436_2.xsd")
- #define _OPENGPS_XSD_ISO5436_MAIN_CHECKSUM_PATH _T("md5checksum.hex")
- #define _OPENGPS_XSD_ISO5436_MAIN_PATH _T("main.xml")
- #define _OPENGPS_XSD_ISO5436_NAMESPACE _T("http://www.opengps.eu/2008/ISO5436_2")
- #define _OPENGPS_XSD_ISO5436_VALIDPOINTSLINK_PATH _T("bindata/valid.bin")
- #define _VERIFY(func, retval) func

13.110.2 Define Documentation

13.110.2.1 #define _ASSERT(x) ((void)0)

Referenced by OpenGPS::ValidBuffer::Allocate(), OpenGPS::PointBuffer::Allocate(), OpenGPS::Int32PointBuffer::Allocate(), OpenGPS::Int16PointBuffer::Allocate(), OpenGPS::FloatPointBuffer::Allocate(), OpenGPS::DoublePointBuffer::Allocate(), OpenGPS::ValidBuffer::AllocateRaw(), OpenGPS::XmlPointVectorWriterContext::AppendSeparator(), OpenGPS::ISO5436_2Container::AppendVendorSpecific(), OpenGPS::ISO5436_2::AppendVendorSpecific(), OpenGPS::VectorBufferBuilder::BuildBuffer(), OpenGPS::PointVectorParserBuilder::BuildParser(), OpenGPS::VectorBufferBuilder::BuildValidityProvider(), OpenGPS::ISO5436_2Container::BuildVectorBuffer(), OpenGPS::VectorBufferBuilder::BuildX(), OpenGPS::PointVectorParserBuilder::BuildX(), OpenGPS::VectorBufferBuilder::BuildY(), OpenGPS::PointVectorParserBuilder::BuildY(), OpenGPS::VectorBufferBuilder::BuildZ(), OpenGPS::PointVectorParserBuilder::BuildZ(), OpenGPS::Environment::ByteSwap(), OpenGPS::Environment::ByteSwap16(), OpenGPS::Environment::ByteSwap32(), OpenGPS::Environment::ByteSwap64(), OpenGPS::ISO5436_2::Close(), OpenGPS::String::ConvertToMd5(), OpenGPS::String::CopyTo(), OpenGPS::ISO5436_2::Create(), OpenGPS::PointVectorParser::CreateDataPointParser(), OpenGPS::ISO5436_2Container::CreateDocument(), OpenGPS::ISO5436_2::CreateNextPointIterator(), OpenGPS::VectorBufferBuilder::CreatePointBuffer(), OpenGPS::ISO5436_2Container::CreatePointBuffer(), OpenGPS::ISO5436_2Container::CreatePointVectorProxyContext(), OpenGPS::ISO5436_2Container::CreatePointVectorWriterContext(), OpenGPS::ISO5436_2::CreatePrevPointIterator(), OpenGPS::ISO5436_2Container::CreateTempDir(), OpenGPS::PointVectorProxy::DataPointProxy::DataPointProxy(), OpenGPS::ISO5436_2Container::Decompress(), OpenGPS::ISO5436_2Container::DecompressChecksum(), OpenGPS::DoubleInlineValidity::DoubleInlineValidity(), OpenGPS::Exception::Exception(), OpenGPS::FloatInlineValidity::FloatInlineValidity(), OpenGPS::XmlPointVectorWriterContext::Get(), OpenGPS::PointVectorProxy::Get(), OpenGPS::PointVector::Get(), OpenGPS::Int32PointBuffer::Get(), OpenGPS::Int16PointBuffer::Get(), OpenGPS::FloatPointBuffer::Get(),

OpenGPS::DoublePointBuffer::Get(), OpenGPS::PointVectorProxy::DataPointProxy::Get(), OpenGPS::DataPointImpl::Get(), OpenGPS::Info::GetAbout(), OpenGPS::ISO5436_2Container::GetAxisDataType(), OpenGPS::ISO5436_2Container::GetChecksumArchiveName(), OpenGPS::Info::GetCopyright(), OpenGPS::ISO5436_2Container::PointIteratorImpl::GetCurrent(), OpenGPS::ISO5436_2::GetDocument(), OpenGPS::ISO5436_2Container::GetIncrementX(), OpenGPS::ISO5436_2Container::GetIncrementY(), OpenGPS::Info::GetLicense(), OpenGPS::ISO5436_2Container::GetListCoord(), OpenGPS::ISO5436_2::GetListCoord(), OpenGPS::ISO5436_2::GetListDimension(), OpenGPS::ISO5436_2Container::GetListPoint(), OpenGPS::ISO5436_2::GetListPoint(), OpenGPS::ISO5436_2Container::GetMatrixCoord(), OpenGPS::ISO5436_2::GetMatrixCoord(), OpenGPS::ISO5436_2::GetMatrixDimensions(), OpenGPS::ISO5436_2Container::GetMatrixPoint(), OpenGPS::ISO5436_2::GetMatrixPoint(), OpenGPS::BinaryPointVectorWriterContext::GetMd5(), OpenGPS::Info::GetName(), OpenGPS::ISO5436_2Container::GetOffsetX(), OpenGPS::ISO5436_2Container::GetOffsetY(), OpenGPS::ISO5436_2Container::GetOffsetZ(), OpenGPS::ISO5436_2Container::GetPointCount(), OpenGPS::ISO5436_2Container::GetPointDataArchiveName(), OpenGPS::ISO5436_2Container::PointIteratorImpl::GetPosition(), OpenGPS::BinaryPointVectorWriterContext::GetStream(), OpenGPS::ISO5436_2Container::GetTempDir(), OpenGPS::ISO5436_2Container::GetValidPointsArchiveName(), OpenGPS::ISO5436_2Container::GetVectorBuffer(), OpenGPS::ISO5436_2::GetVendorSpecific(), OpenGPS::Info::GetVersion(), OpenGPS::PointVectorProxy::GetX(), OpenGPS::PointVector::GetX(), OpenGPS::ISO5436_2Container::GetXaxisDataType(), OpenGPS::PointVector::GetXYZ(), OpenGPS::PointVectorProxy::GetY(), OpenGPS::PointVector::GetY(), OpenGPS::ISO5436_2Container::GetYaxisDataType(), OpenGPS::PointVectorProxy::GetZ(), OpenGPS::PointVector::GetZ(), OpenGPS::ISO5436_2Container::GetZaxisDataType(), OpenGPS::ISO5436_2Container::PointIteratorImpl::HasNext(), OpenGPS::ISO5436_2Container::PointIteratorImpl::HasPrev(), OpenGPS::BinaryPointVectorWriterContext::HasStream(), OpenGPS::ISO5436_2Container::HasValidPointsLink(), OpenGPS::ISO5436_2Container::HasVectorBuffer(), OpenGPS::Int16ValidBuffer::Int16ValidBuffer(), OpenGPS::Int32ValidBuffer::Int32ValidBuffer(), OpenGPS::ISO5436_2Container::IsBinary(), OpenGPS::XmlPointVectorWriterContext::IsGood(), OpenGPS::XmlPointVectorReaderContext::IsGood(), OpenGPS::BinaryPointVectorWriterContext::IsGood(), OpenGPS::BinaryPointVectorReaderContext::IsGood(), OpenGPS::ISO5436_2Container::IsIncrementalX(), OpenGPS::ISO5436_2Container::IsIncrementalY(), OpenGPS::Environment::IsLittleEndian(), OpenGPS::ISO5436_2Container::IsMatrix(), OpenGPS::ISO5436_2::IsMatrix(), OpenGPS::ISO5436_2Container::IsMatrixCoordValid(), OpenGPS::ISO5436_2::IsMatrixCoordValid(), OpenGPS::ValidBuffer::IsValid(), OpenGPS::PointVector::IsValid(), OpenGPS::DoubleInlineValidity::IsValid(), OpenGPS::FloatInlineValidity::IsValid(), OpenGPS::XmlPointVectorWriterContext::MoveNext(), OpenGPS::XmlPointVectorReaderContext::MoveNext(), OpenGPS::ISO5436_2Container::PointIteratorImpl::MoveNext(), OpenGPS::ISO5436_2Container::PointIteratorImpl::MovePrev(), ogps_AppendVendorSpecific(), ogps_CloseISO5436_2(), ogps_CreateListISO5436_2(), ogps_CreateMatrixISO5436_2(), ogps_CreateNextPointIterator(), ogps_CreatePrevPointIterator(), ogps_FreePointIterator(), ogps_GetCurrentPoint(), ogps_GetDocument(), ogps_GetDouble(), ogps_GetDoubleX(), ogps_GetDoubleY(), ogps_GetDoubleZ(), ogps_GetFloat(), ogps_GetFloatX(), ogps_GetFloatY(), ogps_GetFloatZ(), ogps_GetInt16(), ogps_GetInt16X(), ogps_GetInt16Y(), ogps_GetInt16Z(), ogps_GetInt32(), ogps_GetInt32X(), ogps_GetInt32Y(), ogps_GetInt32Z(), ogps_GetListCoord(), ogps_GetListDimension(), ogps_GetListPoint(), ogps_GetListPosition(), ogps_GetMatrixCoord(), ogps_GetMatrixDimensions(), ogps_GetMatrixPoint(), ogps_GetMatrixPosition(), ogps_GetPointType(), ogps_GetPointTypeX(), ogps_GetPointTypeY(), ogps_GetPointTypeZ(), ogps_GetVendorSpecific(), ogps_GetX(), ogps_GetXYZ(), ogps_GetY(), ogps_GetZ(), ogps_HasNextPoint(), ogps_HasPrevPoint(), ogps_IsMatrix(), ogps_IsMatrixCoordValid(), ogps_IsValidPoint(), ogps_MoveNextPoint(), ogps_MovePrevPoint(), ogps_OpenISO5436_2(), ogps_ResetNextPointIterator(), ogps_ResetPrevPointIterator(), ogps_SetCurrentPoint(), ogps_SetDouble(), ogps_SetDoubleX(), ogps_SetDoubleY(), ogps_SetDoubleZ(), ogps_SetFloat(), ogps_SetFloatX(), ogps_SetFloatY(), ogps_SetFloatZ(), ogps_SetInt16(), ogps_-

SetInt16X(), ogps_SetInt16Y(), ogps_SetInt16Z(), ogps_SetInt32(), ogps_SetInt32X(), ogps_-SetInt32Y(), ogps_SetInt32Z(), ogps_SetListPoint(), ogps_SetMatrixPoint(), ogps_WriteISO5436_-2(), OpenGPS::ISO5436_2::Open(), OpenGPS::PointVector::operator=(), OpenGPS::ISO5436_-2Container::PointIteratorImpl::PointIteratorImpl(), OpenGPS::PointValidityProvider::PointValidityProvider(), OpenGPS::PointVectorProxy::PointVectorProxy(), PrintDimensions(), OpenGPS::XmlPointVectorReaderContext::Read(), OpenGPS::PointVectorParser::Read(), OpenGPS::BinaryMSBPointVectorReaderContext::Read(), OpenGPS::BinaryLSBPointVectorReaderContext::Read(), OpenGPS::ISO5436_-2Container::ReadDocument(), readonlyExample2(), readonlyExample4(), OpenGPS::ISO5436_2Container::ResetValidPointsLink(), OpenGPS::ISO5436_-2Container::ResetXmlPointList(), OpenGPS::ISO5436_2Container::SaveChecksumFile(), OpenGPS::ISO5436_2Container::SavePointBuffer(), OpenGPS::ISO5436_-2Container::SaveValidPointsLink(), OpenGPS::ISO5436_2Container::SaveXmlDocument(), OpenGPS::XmlPointVectorReaderContext::Set(), OpenGPS::PointVectorProxy::Set(), OpenGPS::PointVector::Set(), OpenGPS::Int32PointBuffer::Set(), OpenGPS::Int16PointBuffer::Set(), OpenGPS::FloatPointBuffer::Set(), OpenGPS::DoublePointBuffer::Set(), OpenGPS::PointVectorProxy::DataPointProxy::Set(), OpenGPS::ISO5436_-2Container::PointIteratorImpl::SetCurrent(), OpenGPS::ISO5436_2Container::SetListPoint(), OpenGPS::ISO5436_2::SetListPoint(), OpenGPS::ISO5436_2Container::SetMatrixPoint(), OpenGPS::ISO5436_2::SetMatrixPoint(), OpenGPS::ValidBuffer::SetValid(), OpenGPS::DoubleInlineValidity::SetValid(), OpenGPS::FloatInlineValidity::SetValid(), OpenGPS::VectorBuffer::SetValidityProvider(), OpenGPS::VectorBuffer::SetX(), OpenGPS::PointVectorParser::SetX(), OpenGPS::PointVector::SetX(), OpenGPS::VectorBuffer::SetY(), OpenGPS::PointVectorParser::SetY(), OpenGPS::PointVector::SetY(), OpenGPS::VectorBuffer::SetZ(), OpenGPS::PointVectorParser::SetZ(), OpenGPS::PointVector::SetZ(), OpenGPS::ValidBuffer::ValidBuffer(), OpenGPS::ISO5436_2Container::VerifyChecksum(), OpenGPS::ISO5436_-2Container::VerifyDataBinChecksum(), OpenGPS::ISO5436_2Container::VerifyMainChecksum(), OpenGPS::ISO5436_2Container::VerifyValidBinChecksum(), OpenGPS::XmlPointVectorWriterContext::Write(), OpenGPS::ValidBuffer::Write(), OpenGPS::PointVectorParser::Write(), OpenGPS::ISO5436_2Container::Write(), OpenGPS::ISO5436_2::Write(), OpenGPS::BinaryMSBPointVectorWriterContext::Write(), OpenGPS::BinaryLSBPointVectorWriterContext::Write(), OpenGPS::XmlPointVectorReaderContext::XmlPointVectorReaderContext(), OpenGPS::XmlPointVectorWriterContext::XmlPointVectorWriterContext(), and OpenGPS::ISO5436_-2::∼ISO5436_2().

13.110.2.2 #define _OPENGPS_BINFORMAT_DOUBLE_SIZE 8

Referenced by OpenGPS::BinaryMSBPointVectorReaderContext::Read(), OpenGPS::BinaryLSBPointVectorReaderContext::Read(), OpenGPS::BinaryMSBPointVectorWriterContext::Write(), and OpenGPS::BinaryLSBPointVectorWriterContext::Write().

13.110.2.3 #define _OPENGPS_BINFORMAT_FLOAT_SIZE 4

Referenced by OpenGPS::BinaryMSBPointVectorReaderContext::Read(), OpenGPS::BinaryLSBPointVectorReaderContext::Read(), OpenGPS::BinaryMSBPointVectorWriterContext::Write(), and OpenGPS::BinaryLSBPointVectorWriterContext::Write().

13.110.2.4 #define _OPENGPS_BINFORMAT_INT16_SIZE 2

Referenced by OpenGPS::BinaryMSBPointVectorReaderContext::Read(), OpenGPS::BinaryLSBPointVectorReaderContext::Read(), OpenGPS::BinaryMSBPointVectorWriterContext::Write(), and OpenGPS::BinaryLSBPointVectorWriterContext::Write().

13.110.2.5 #define _OPENGPS_BINFORMAT_INT32_SIZE 4

Referenced by OpenGPS::BinaryMSBPointVectorReaderContext::Read(), OpenGPS::BinaryLSBPointVectorReaderContext::Read(), OpenGPS::BinaryMSBPointVectorWriterContext::Write(), and OpenGPS::BinaryLSBPointVectorWriterContext::Write().

13.110.2.6 #define _OPENGPS_DELETE(x) if((x) != NULL) { delete (x); (x) = NULL; }

Referenced by OpenGPS::BinaryPointVectorWriterContext::Close(), OpenGPS::BinaryPointVectorReaderContext::Close(), OpenGPS::VectorBufferBuilder::CreatePointBuffer(), ogps_CloseISO5436_2(), ogps_-CreateListISO5436_2(), ogps_CreateMatrixISO5436_2(), ogps_CreateNextPointIterator(), ogps_CreatePrevPointIterator(), ogps_FreePointIterator(), ogps_FreePointVector(), ogps_-OpenISO5436_2(), OpenGPS::XmlPointVectorReaderContext::Reset(), OpenGPS::ISO5436_-2Container::Reset(), OpenGPS::Environment::Reset(), OpenGPS::PointVectorParser::SetX(), OpenGPS::PointVectorParser::SetY(), OpenGPS::PointVectorParser::SetZ(), OpenGPS::ISO5436_-2::~ISO5436_2(), OpenGPS::ISO5436_2Container::~ISO5436_2Container(), OpenGPS::PointVector::~PointVector(), OpenGPS::PointVectorParser::~PointVectorParser(), OpenGPS::PointVectorParserBuilder::~PointVectorParserBuilder(), OpenGPS::PointVectorProxy::~PointVectorProxy(), OpenGPS::VectorBuffer::~VectorBuffer(), OpenGPS::VectorBufferBuilder::~VectorBufferBuilder(), OpenGPS::XmlPointVectorWriterContext::~XmlPointVectorWriterContext(), and OpenGPS::ZipStreamBuffer::~ZipStreamBuffer().

13.110.2.7 #define _OPENGPS_DELETE_ARRAY(x) if((x) != NULL) { delete[ ] (x); (x) = NULL; }

Referenced by OpenGPS::String::FromChar(), OpenGPS::String::ToChar(), and OpenGPS::String::~String().

13.110.2.8 #define _OPENGPS_DESCRIPTION _T("openGPS class library implementing an xml-version of ISO 5436-2 file format.")

Short description of the library.

Referenced by OpenGPS::Info::GetAbout().

13.110.2.9 #define _OPENGPS_ENV_OPENGPS_LOCATION _T("OPENGPS_LOCATION")

Referenced by OpenGPS::ISO5436_2Container::ConfigureNamespaceMap().

13.110.2.10 #define _OPENGPS_EXCEPTION_MESG(x) x

13.110.2.11 #define _OPENGPS_EXPERIMENTAL _T(" (Experimental Version!)")

13.110.2.12 #define _OPENGPS_FREE(x) if((x) != NULL) { free(x); (x) = NULL; }

Referenced by OpenGPS::ISO5436_2Container::Decompress(), OpenGPS::PointBuffer::Free(), OpenGPS::ValidBuffer::Reset(), and OpenGPS::ISO5436_2Container::WriteVendorSpecific().

13.110.2.13 #define _OPENGPS_ID _OPENGPS_NAME _T(" (V") _OPENGPS_-VERSIONSTRING _T(")")

Define ID string with programm name and Version.

Referenced by OpenGPS::Info::PrintVersion().

13.110.2.14 #define _OPENGPS_ISO5436_LOCATION _T("iso5436_2.xsd")

Referenced by OpenGPS::ISO5436_2Container::ConfigureNamespaceMap().

13.110.2.15 #define _OPENGPS_MSTR(x) #x

Build a version string from version numbers.

13.110.2.16 #define _OPENGPS_NAME _T("openGPS ISO 5436-2 XML") _OPENGPS_EXPERIMENTAL

Name of the program.

Referenced by OpenGPS::Info::GetName().

13.110.2.17 #define _OPENGPS_VERSIONSTRING _OPENGPS_VERSIONSTRING_M(_OPENGPS_VERSION, _OPENGPS_MINVERSION, _OPENGPS_BUILD, _OPENGPS_REVISION)

Build a version string.

Referenced by OpenGPS::Info::GetVersion().

13.110.2.18 #define _OPENGPS_VERSIONSTRING_M(ver, mver, build, rev) _OPENGPS_MSTR(ver) _T(".") _OPENGPS_MSTR(mver) _T(".") _OPENGPS_MSTR(build) _T(".") _T(rev)

13.110.2.19 #define _OPENGPS_XSD_ISO5436_DATALINK_PATH _T("bindata/data.bin")

Referenced by OpenGPS::ISO5436_2Container::CreateDocument().

13.110.2.20 #define _OPENGPS_XSD_ISO5436_LOCATION _T("http://www.opengps.eu/2008/ISO5436_2/ISO5436_2.xsd")

Referenced by OpenGPS::ISO5436_2Container::ConfigureNamespaceMap().

13.110.2.21 #define _OPENGPS_XSD_ISO5436_MAIN_CHECKSUM_PATH _T("md5checksum.hex")

Referenced by OpenGPS::ISO5436_2Container::CreateDocument().

13.110.2.22 #define _OPENGPS_XSD_ISO5436_MAIN_PATH _T("main.xml")

Referenced by OpenGPS::ISO5436_2Container::GetMainArchiveName().

13.110.2.23 #define _OPENGPS_XSD_ISO5436_NAMESPACE _T("http://www.opengps.eu/2008/ISO5436_2")

Referenced by OpenGPS::ISO5436_2Container::ConfigureNamespaceMap().

13.110.2.24 #define _OPENGPS_XSD_ISO5436_VALIDPOINTSLINK_PATH _T("bindata/valid.bin")

Referenced by OpenGPS::ISO5436_2Container::ResetValidPointsLink().

13.110.2.25 #define _VERIFY(func, retval) func

Referenced by OpenGPS::ISO5436_2Container::Compress(), OpenGPS::ISO5436_-2Container::Decompress(), OpenGPS::ISO5436_2Container::DecompressChecksum(), OpenGPS::ISO5436_2Container::DecompressDataBin(), OpenGPS::ISO5436_-2Container::DecompressMain(), OpenGPS::ISO5436_2Container::SavePointBuffer(), OpenGPS::ISO5436_2Container::SaveValidPointsLink(), OpenGPS::ISO5436_-2Container::SaveXmlDocument(), and OpenGPS::ISO5436_2Container::WriteVendorSpecific().

13.111 string.cxx File Reference

```
#include <opengps/cxx/string.hxx>
#include <stdlib.h>
#include <sstream>
#include <iomanip>
#include "stdafx.hxx"
```

Include dependency graph for string.cxx:

13.112 string.hxx File Reference

13.112.1 Detailed Description

An enhanced std::string string type.

```
#include <opengps/cxx/opengps.hxx>
#include <string>
```

Include dependency graph for string.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::String

  Stores an OGPS_Character sequence.

13.113 valid_buffer.cxx File Reference

```
#include "valid_buffer.hxx"
#include "point_buffer.hxx"
#include "stdafx.hxx"
#include <opengps/cxx/exceptions.hxx>
```

Include dependency graph for valid_buffer.cxx:

13.114 valid_buffer.hxx File Reference

13.114.1 Detailed Description

Communicate validity of point vectors through external binary file.

```
#include <opengps/cxx/opengps.hxx>
#include "point_validity_provider.hxx"
#include <iostream>
```

Include dependency graph for valid_buffer.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::Int16ValidBuffer

  Implementation of OpenGPS::ValidBuffer for Z axis of OGPS_Int16 data type.

- class OpenGPS::Int32ValidBuffer

  Implementation of OpenGPS::ValidBuffer for Z axis of OGPS_Int32 data type.

- class OpenGPS::ValidBuffer

  Implements the OpenGPS::PointValidityProvider as an external binary file.

13.115 vector_buffer.cxx File Reference

```
#include "vector_buffer.hxx"
#include "point_vector_proxy.hxx"
#include "point_buffer.hxx"
#include "stdafx.hxx"
```

Include dependency graph for vector_buffer.cxx:

13.116 vector_buffer.hxx File Reference

13.116.1 Detailed Description

The internal memory structure that stores all point measurement data.

```
#include <opengps/data_point_type.h>
#include <opengps/cxx/point_vector_base.hxx>
#include "valid_buffer.hxx"
#include "auto_ptr_types.hxx"
```

Include dependency graph for vector_buffer.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::VectorBuffer

  Implements the memory structure of point measurement data.

13.117 vector_buffer_builder.cxx File Reference

```
#include "vector_buffer_builder.hxx"
#include "vector_buffer.hxx"
#include "inline_validity.hxx"
#include "int16_point_buffer.hxx"
#include "int32_point_buffer.hxx"
#include "float_point_buffer.hxx"
#include "double_point_buffer.hxx"
#include <opengps/cxx/exceptions.hxx>
#include "stdafx.hxx"
```

Include dependency graph for vector_buffer_builder.cxx:

13.118 vector_buffer_builder.hxx File Reference

13.118.1 Detailed Description

Methods to assemble a OpenGPS::VectorBuffer instance.

```
#include <opengps/cxx/opengps.hxx>
#include <opengps/data_point_type.h>
```

Include dependency graph for vector_buffer_builder.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::VectorBufferBuilder

  Creates an object which is able to assemble a OpenGPS::VectorBuffer instance.

13.119 Version.h File Reference

13.119.1 Detailed Description

Include file generated from "Version.in" containing the current version.

Author:

Georg Wiora

Defines

- #define _OPENGPS_BUILD 15

  Build number is automatically incremented after each succesfull release build.

- #define _OPENGPS_BUILD_DATE "2008/10/23 15:38:40"

  Build date.

- #define _OPENGPS_CLEAN_BUILD "Warning! Contains locally modified code!"

  Warning message "Warning! Contains locally modified code!" or empty.

- #define _OPENGPS_CLEAN_BUILD_FLAG 0

  0 if locally modified sources exist, 1 otherwise

- #define _OPENGPS_CLEAN_REVISION

  Warning message "Warning! Contains mixed revisions!" or empty.

- #define _OPENGPS_CLEAN_REVISION_FLAG 1

  0 if it contains mixed revisions, 1 otherwise

- #define _OPENGPS_MINVERSION 1

  Minor program revision. This is increased manually in the release process.

- #define _OPENGPS_REPOSITORY_URL "https://open-gps.svn.sourceforge.net/svnroot/open-gps/ISO5436_XML/trunk/src/ISO5436_2_XML"

  URL of project archive.

- #define _OPENGPS_REVISION "186"

  Version or Version range, automatically extracted from SVN repository.

- #define _OPENGPS_REVISION_DATE "2008/10/23 13:12:49"

  Date of highest revision.

- #define _OPENGPS_VERSION 0

  Major program version. This is increased manually in the release process.

13.120 win32_environment.cxx File Reference

13.121 win32_environment.hxx File Reference

13.121.1 Detailed Description

The environment of Microsoft Windows operating systems.

13.122 xml_point_vector_reader_context.cxx File Reference

```
#include "xml_point_vector_reader_context.hxx"
#include "point_vector_iostream.hxx"
#include "stdafx.hxx"
#include <opengps/cxx/string.hxx>
#include <opengps/cxx/exceptions.hxx>
```

Include dependency graph for xml_point_vector_reader_context.cxx:

Defines

- #define _CHECK_ISGOOD_AND_THROW_EXCEPTION

  Checks whether the underlying stream is valid.

- #define _CHECK_STREAM_AND_THROW_EXCEPTION

  Checks whether the underlying stream is valid.

13.122.1 Define Documentation

13.122.1.1 #define _CHECK_ISGOOD_AND_THROW_EXCEPTION

Value:

```
if(!IsGood()) \
   { \
   throw OpenGPS::Exception( \
      OGPS_ExInvalidOperation, \
      _EX_T("The underlying stream object became invalid."), \
      _EX_T("A read/write error occured."), \
      _EX_T("OpenGPS::XmlPointVectorReaderContext")); \
   }
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

13.122.1.2 #define _CHECK_STREAM_AND_THROW_EXCEPTION

Value:

```
if(!m_Stream) \
   { \
   throw OpenGPS::Exception( \
      OGPS_ExInvalidOperation, \
      _EX_T("No stream object available."), \
      _EX_T("The operation on the stream object failed, because the stream has been released already."), \
      _EX_T("OpenGPS::XmlPointVectorReaderContext")); \
   }
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

13.123 xml_point_vector_reader_context.hxx File Reference

13.123.1 Detailed Description

Implementation of access methods for reading typed point data from a string list of point vectors.

```
#include "point_vector_reader_context.hxx"
```

```
#include <opengps/cxx/iso5436_2_xsd.hxx>
```

Include dependency graph for xml_point_vector_reader_context.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::XmlPointVectorReaderContext

  Specialized OpenGPS::PointVectorReaderContext for point vectors stored as list of strings.

13.124 xml_point_vector_writer_context.cxx File Reference

```
#include "xml_point_vector_writer_context.hxx"
#include "point_vector_iostream.hxx"
#include "stdafx.hxx"
#include <opengps/cxx/string.hxx>
#include <opengps/cxx/exceptions.hxx>
```

Include dependency graph for xml_point_vector_writer_context.cxx:

Defines

- #define _CHECK_ISGOOD_AND_THROW_EXCEPTION

  Checks whether the underlying stream is valid.

- #define _CHECK_STREAM_AND_THROW_EXCEPTION

  Checks whether the underlying stream is valid.

13.124.1 Define Documentation

13.124.1.1 #define _CHECK_ISGOOD_AND_THROW_EXCEPTION

Value:

```
if(!IsGood()) \
   { \
   throw OpenGPS::Exception( \
      OGPS_ExInvalidOperation, \
      _EX_T("The underlying stream object became invalid."), \
      _EX_T("A read/write error occured."), \
      _EX_T("OpenGPS::XmlPointVectorWriterContext")); \
   }
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

13.124.1.2 #define _CHECK_STREAM_AND_THROW_EXCEPTION

Value:

```
if(!m_Stream) \
   { \
   throw OpenGPS::Exception( \
      OGPS_ExInvalidOperation, \
      _EX_T("No stream object available."), \
      _EX_T("The operation on the stream object failed, because the stream has been released already."), \
      _EX_T("OpenGPS::XmlPointVectorWriterContext")); \
   }
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

13.125 xml_point_vector_writer_context.hxx File Reference

13.125.1 Detailed Description

Implementation of access methods for writing typed point data to a string list of point vectors.

`#include "point_vector_writer_context.hxx"`

`#include <opengps/cxx/iso5436_2_xsd.hxx>`

Include dependency graph for xml_point_vector_writer_context.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::XmlPointVectorWriterContext

  Specialized OpenGPS::PointVectorWriterContext for point vectors stored as list of strings.

13.126 xsd_Licence_Header.c File Reference

```
#include <opengps/opengps.h>
```

Include dependency graph for xsd_Licence_Header.c:

13.127 zip_stream_buffer.cxx File Reference

```
#include "zip_stream_buffer.hxx"
#include "stdafx.hxx"
#include <opengps/cxx/string.hxx>
```

Include dependency graph for zip_stream_buffer.cxx:

13.128 zip_stream_buffer.hxx File Reference

13.128.1 Detailed Description

Integrates Info-Zip handles into the common standard io framework.

```
#include <ostream>
#include <zip.h>
#include "../xyssl/md5.h"
```

```
#include <opengps/cxx/opengps.hxx>
```

Include dependency graph for zip_stream_buffer.hxx:

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace OpenGPS

Classes

- class OpenGPS::ZipOutputStream

  Provides an output stream interface to write binary data to Info-Zip archives.

- class OpenGPS::ZipStreamBuffer

  Provides a buffer interface suitable for streaming the zipFile handle defined in the zlib/minizip package.

Index

~BinaryLSBPointVectorReaderContext
OpenGPS::BinaryLSBPointVectorReaderContext, 90
~BinaryLSBPointVectorWriterContext
OpenGPS::BinaryLSBPointVectorWriterContext, 96
~BinaryMSBPointVectorReaderContext
OpenGPS::BinaryMSBPointVectorReaderContext, 101
~BinaryMSBPointVectorWriterContext
OpenGPS::BinaryMSBPointVectorWriterContext, 107
~BinaryPointVectorReaderContext
OpenGPS::BinaryPointVectorReaderContext, 112
~BinaryPointVectorWriterContext
OpenGPS::BinaryPointVectorWriterContext, 118
~DataPointImpl
OpenGPS::DataPointImpl, 139
~DataPointParser
OpenGPS::DataPointParser, 143
~DataPointProxy
OpenGPS::PointVectorProxy::DataPointProxy, 315
~DoubleDataPointParser
OpenGPS::DoubleDataPointParser, 153
~DoubleInlineValidity
OpenGPS::DoubleInlineValidity, 155
~DoublePointBuffer
OpenGPS::DoublePointBuffer, 159
~Environment
OpenGPS::Environment, 167
~ExceptionHistory
OpenGPS::ExceptionHistory, 177
~FloatDataPointParser
OpenGPS::FloatDataPointParser, 183
~FloatInlineValidity
OpenGPS::FloatInlineValidity, 186
~FloatPointBuffer
OpenGPS::FloatPointBuffer, 189
~ISO5436_2Container
OpenGPS::ISO5436_2Container, 239
~InputBinaryFileStream
OpenGPS::InputBinaryFileStream, 195
~Int16DataPointParser
OpenGPS::Int16DataPointParser, 206
~Int16PointBuffer
OpenGPS::Int16PointBuffer, 209
~Int16ValidBuffer
OpenGPS::Int16ValidBuffer, 215
~Int32DataPointParser
OpenGPS::Int32DataPointParser, 219
~Int32PointBuffer
OpenGPS::Int32PointBuffer, 222
~Int32ValidBuffer
OpenGPS::Int32ValidBuffer, 228
~MissingDataPointParser
OpenGPS::MissingDataPointParser, 285
~OutputBinaryFileStream
OpenGPS::OutputBinaryFileStream, 288
~PointBuffer
OpenGPS::PointBuffer, 291
~PointIteratorImpl
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 261
~PointValidityProvider
OpenGPS::PointValidityProvider, 296
~PointVectorInputStringStream
OpenGPS::PointVectorInputStringStream, 299
~PointVectorInvariantLocale
OpenGPS::PointVectorInvariantLocale, 301
~PointVectorOutputStringStream
OpenGPS::PointVectorOutputStringStream, 302
~PointVectorParser
OpenGPS::PointVectorParser, 304
~PointVectorParserBuilder
OpenGPS::PointVectorParserBuilder, 308
~PointVectorProxy
OpenGPS::PointVectorProxy, 312
~PointVectorProxyContext
OpenGPS::PointVectorProxyContext, 319
~PointVectorProxyContextList
OpenGPS::PointVectorProxyContextList, 321
~PointVectorProxyContextMatrix
OpenGPS::PointVectorProxyContextMatrix, 325
~PointVectorReaderContext
OpenGPS::PointVectorReaderContext, 328
~PointVectorWhitespaceFacet
OpenGPS::PointVectorWhitespaceFacet, 332
~PointVectorWriterContext
OpenGPS::PointVectorWriterContext, 334
~ValidBuffer
OpenGPS::ValidBuffer, 408
~VectorBuffer
OpenGPS::VectorBuffer, 413
~VectorBufferBuilder
OpenGPS::VectorBufferBuilder, 419

~XmlPointVectorReaderContext
OpenGPS::XmlPointVectorReaderContext, 424
~XmlPointVectorWriterContext
OpenGPS::XmlPointVectorWriterContext, 431
~ZipOutputStream
OpenGPS::ZipOutputStream, 435
~ZipStreamBuffer
OpenGPS::ZipStreamBuffer, 437
_ASSERT
stdafx.hxx, 547
_CHECK_BUFFER_AND_THROW_READ_-EXCEPTION
data_point_proxy.cxx, 461
_CHECK_BUFFER_AND_THROW_WRITE_-EXCEPTION
data_point_proxy.cxx, 461
_CHECK_ISGOOD_AND_THROW_-EXCEPTION
binary_lsb_point_vector_reader_context.cxx, 443
binary_lsb_point_vector_writer_context.cxx, 445
binary_msb_point_vector_reader_context.cxx, 447
binary_msb_point_vector_writer_context.cxx, 450
binary_point_vector_writer_context.cxx, 454
xml_point_vector_reader_context.cxx, 557
xml_point_vector_writer_context.cxx, 560
_CHECK_STREAM_AND_THROW_-EXCEPTION
binary_lsb_point_vector_reader_context.cxx, 443
binary_lsb_point_vector_writer_context.cxx, 445
binary_msb_point_vector_reader_context.cxx, 448
binary_msb_point_vector_writer_context.cxx, 450
binary_point_vector_reader_context.cxx, 452
binary_point_vector_writer_context.cxx, 454
xml_point_vector_reader_context.cxx, 558
xml_point_vector_writer_context.cxx, 560
_OGPS_DATA_POINT, 60
instance, 61
_OGPS_DOUBLE_SIZE
environment.cxx, 465
_OGPS_FLOAT_SIZE
environment.cxx, 465
_OGPS_INT_SIZE
environment.cxx, 465
_OGPS_ISO5436_2_HANDLE, 61
instance, 61
_OGPS_POINT_ITERATOR, 61
instance, 62
_OGPS_POINT_VECTOR, 62
instance, 63
x, 63
y, 63
z, 63
_OGPS_SHORT_SIZE
environment.cxx, 465
_OPENGPS_BINFORMAT_DOUBLE_SIZE
stdafx.hxx, 547
_OPENGPS_BINFORMAT_FLOAT_SIZE
stdafx.hxx, 547
_OPENGPS_BINFORMAT_INT16_SIZE
stdafx.hxx, 547
_OPENGPS_BINFORMAT_INT32_SIZE
stdafx.hxx, 547
_OPENGPS_BUILD
Version.h, 555
_OPENGPS_BUILD_DATE
Version.h, 555
_OPENGPS_CLEAN_BUILD
Version.h, 555
_OPENGPS_CLEAN_BUILD_FLAG
Version.h, 555
_OPENGPS_CLEAN_REVISION
Version.h, 556
_OPENGPS_CLEAN_REVISION_FLAG
Version.h, 556
_OPENGPS_DELETE
stdafx.hxx, 547
_OPENGPS_DELETE_ARRAY
stdafx.hxx, 547
_OPENGPS_DESCRIPTION
stdafx.hxx, 547
_OPENGPS_ENV_OPENGPS_LOCATION
stdafx.hxx, 547
_OPENGPS_EXCEPTION_MESG
stdafx.hxx, 548
_OPENGPS_EXPERIMENTAL
stdafx.hxx, 548
_OPENGPS_FILE_URI_PREF
iso5436_2_container.cxx, 483
_OPENGPS_FREE
stdafx.hxx, 548
_OPENGPS_GENERIC_EXCEPTION_-HANDLER
messages_c.hxx, 519
_OPENGPS_GENERIC_EXCEPTION_-HANDLER_CLEANUP
messages_c.hxx, 519
_OPENGPS_GENERIC_EXCEPTION_-HANDLER_RETVALBOOL

messages_c.hxx, 520
_OPENGPS_ID
stdafx.hxx, 548
_OPENGPS_ISO5436_LOCATION
stdafx.hxx, 548
_OPENGPS_MINVERSION
Version.h, 556
_OPENGPS_MSTR
stdafx.hxx, 548
_OPENGPS_NAME
stdafx.hxx, 548
_OPENGPS_REPOSITORY_URL
Version.h, 556
_OPENGPS_REVISION
Version.h, 556
_OPENGPS_REVISION_DATE
Version.h, 556
_OPENGPS_VERSION
Version.h, 556
_OPENGPS_VERSIONSTRING
stdafx.hxx, 548
_OPENGPS_VERSIONSTRING_M
stdafx.hxx, 548
_OPENGPS_XSD_ISO5436_DATALINK_PATH
stdafx.hxx, 549
_OPENGPS_XSD_ISO5436_LOCATION
stdafx.hxx, 549
_OPENGPS_XSD_ISO5436_MAIN_-CHECKSUM_PATH
stdafx.hxx, 549
_OPENGPS_XSD_ISO5436_MAIN_PATH
stdafx.hxx, 549
_OPENGPS_XSD_ISO5436_NAMESPACE
stdafx.hxx, 549
_OPENGPS_XSD_ISO5436_-VALIDPOINTSLINK_PATH
stdafx.hxx, 549
_OPENGPS_ZIP_CHUNK_MAX
iso5436_2_container.cxx, 483
_VERIFY
stdafx.hxx, 549
_clone
OpenGPS::Schemas::ISO5436_2::AxesType, 69
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 79
OpenGPS::Schemas::ISO5436_2::AxisType, 87
OpenGPS::Schemas::ISO5436_-2::DataLinkType, 128
OpenGPS::Schemas::ISO5436_-2::DataListType, 136
OpenGPS::Schemas::ISO5436_2::DataType, 148
OpenGPS::Schemas::ISO5436_2::Datum, 151
OpenGPS::Schemas::ISO5436_-2::FeatureType, 182
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 200
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 270
OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType, 280
OpenGPS::Schemas::ISO5436_-2::ProbingSystemType, 340
OpenGPS::Schemas::ISO5436_-2::Record1Type, 346
OpenGPS::Schemas::ISO5436_-2::Record2Type, 357
OpenGPS::Schemas::ISO5436_-2::Record3Type, 371
OpenGPS::Schemas::ISO5436_-2::Record4Type, 379
OpenGPS::Schemas::ISO5436_-2::RotationMatrixElementType, 383
OpenGPS::Schemas::ISO5436_-2::RotationType, 393
OpenGPS::Schemas::ISO5436_2::Type, 405
_tmain
ISO5436_2_XML_Demo.cxx, 486
Kopie von ISO5436_2_XML_Demo_Test.cxx, 506
_xsd_AxisType_convert
OpenGPS::Schemas::ISO5436_2::AxisType, 87
_xsd_AxisType_indexes_
OpenGPS::Schemas::ISO5436_2::AxisType, 88
_xsd_AxisType_literals_
OpenGPS::Schemas::ISO5436_2::AxisType, 88
_xsd_DataType_convert
OpenGPS::Schemas::ISO5436_2::DataType, 148
_xsd_DataType_indexes_
OpenGPS::Schemas::ISO5436_2::DataType, 149
_xsd_DataType_literals_
OpenGPS::Schemas::ISO5436_2::DataType, 149
_xsd_Type_convert
OpenGPS::Schemas::ISO5436_2::Type, 405
_xsd_Type_indexes_
OpenGPS::Schemas::ISO5436_2::Type, 406
_xsd_Type_literals_
OpenGPS::Schemas::ISO5436_2::Type, 406

A

OpenGPS::Schemas::ISO5436_2::AxisType, 85
Allocate
OpenGPS::DoublePointBuffer, 159
OpenGPS::FloatPointBuffer, 189, 190
OpenGPS::Int16PointBuffer, 209
OpenGPS::Int16ValidBuffer, 215
OpenGPS::Int32PointBuffer, 222
OpenGPS::Int32ValidBuffer, 228
OpenGPS::PointBuffer, 291
OpenGPS::ValidBuffer, 408
AllocateRaw
OpenGPS::Int16ValidBuffer, 215
OpenGPS::Int32ValidBuffer, 228
OpenGPS::ValidBuffer, 408
AppendSeparator
OpenGPS::XmlPointVectorWriterContext, 431
AppendVendorSpecific
OpenGPS::ISO5436_2Container, 239
auto_ptr_types.hxx, 439
Axes
OpenGPS::Schemas::ISO5436_-2::Record1Type, 347
Axes_
OpenGPS::Schemas::ISO5436_-2::Record1Type, 350
Axes_traits
OpenGPS::Schemas::ISO5436_-2::Record1Type, 345
Axes_type
OpenGPS::Schemas::ISO5436_-2::Record1Type, 345
AxesType
OpenGPS::Schemas::ISO5436_2::AxesType, 68
AxisDescriptionType
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 78, 79
AxisType
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 79, 80
OpenGPS::Schemas::ISO5436_2::AxisType, 85, 86
AxisType_
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 83
AxisType_traits
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 77
AxisType_type
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 77
base64_binary
xml_schema, 53
BaseType
OpenGPS::InputBinaryFileStream, 194
OpenGPS::OutputBinaryFileStream, 288
OpenGPS::PointVectorInputStringStream, 299
OpenGPS::PointVectorInvariantLocale, 300
OpenGPS::PointVectorOutputStringStream, 302
OpenGPS::PointVectorWhitespaceFacet, 332
OpenGPS::ZipOutputStream, 435
OpenGPS::ZipStreamBuffer, 437
binary_lsb_point_vector_reader_context.cxx, 441
_CHECK_ISGOOD_AND_THROW_-EXCEPTION, 443
_CHECK_STREAM_AND_THROW_-EXCEPTION, 443
binary_lsb_point_vector_reader_context.hxx, 443
binary_lsb_point_vector_writer_context.cxx, 444
_CHECK_ISGOOD_AND_THROW_-EXCEPTION, 445
_CHECK_STREAM_AND_THROW_-EXCEPTION, 445
binary_lsb_point_vector_writer_context.hxx, 446
binary_msb_point_vector_reader_context.cxx, 447
_CHECK_ISGOOD_AND_THROW_-EXCEPTION, 447
_CHECK_STREAM_AND_THROW_-EXCEPTION, 448
binary_msb_point_vector_reader_context.hxx, 448
binary_msb_point_vector_writer_context.cxx, 449
_CHECK_ISGOOD_AND_THROW_-EXCEPTION, 450
_CHECK_STREAM_AND_THROW_-EXCEPTION, 450
binary_msb_point_vector_writer_context.hxx, 451
binary_point_vector_reader_context.cxx, 452
_CHECK_STREAM_AND_THROW_-EXCEPTION, 452
binary_point_vector_reader_context.hxx, 453
binary_point_vector_writer_context.cxx, 453
_CHECK_ISGOOD_AND_THROW_-EXCEPTION, 454
_CHECK_STREAM_AND_THROW_-EXCEPTION, 454
binary_point_vector_writer_context.hxx, 455
BinaryLSBPointVectorReaderContext
OpenGPS::BinaryLSBPointVectorReaderContext, 90
BinaryLSBPointVectorWriterContext
OpenGPS::BinaryLSBPointVectorWriterContext, 96
BinaryMSBPointVectorReaderContext

OpenGPS::BinaryMSBPointVectorReaderContext, 101
BinaryMSBPointVectorWriterContext
OpenGPS::BinaryMSBPointVectorWriterContext, 107
BinaryPointVectorReaderContext
OpenGPS::BinaryPointVectorReaderContext, 112
BinaryPointVectorWriterContext
OpenGPS::BinaryPointVectorWriterContext, 118
boolean
xml_schema, 53
bounds
xml_schema, 54
buffer
md5_context, 284
xml_schema, 54
BuildBuffer
OpenGPS::VectorBufferBuilder, 420
BuildParser
OpenGPS::PointVectorParserBuilder, 308
BuildPointVectorParser
OpenGPS::ISO5436_2Container, 239
BuildValidityProvider
OpenGPS::VectorBufferBuilder, 420
BuildVectorBuffer
OpenGPS::ISO5436_2Container, 240
BuildX
OpenGPS::PointVectorParserBuilder, 308
OpenGPS::VectorBufferBuilder, 420
BuildY
OpenGPS::PointVectorParserBuilder, 309
OpenGPS::VectorBufferBuilder, 420
BuildZ
OpenGPS::PointVectorParserBuilder, 309
OpenGPS::VectorBufferBuilder, 421
byte
xml_schema, 54
ByteSwap
OpenGPS::Environment, 167–169
ByteSwap16
OpenGPS::Environment, 169
ByteSwap32
OpenGPS::Environment, 170, 171
ByteSwap64
OpenGPS::Environment, 171

CalibrationDate
OpenGPS::Schemas::ISO5436_-2::Record2Type, 358
CalibrationDate_
OpenGPS::Schemas::ISO5436_-2::Record2Type, 364
CalibrationDate_traits
OpenGPS::Schemas::ISO5436_-2::Record2Type, 355
CalibrationDate_type
OpenGPS::Schemas::ISO5436_-2::Record2Type, 355
CanIncrementIndex
OpenGPS::PointVectorProxyContext, 319
OpenGPS::PointVectorProxyContextList, 321
OpenGPS::PointVectorProxyContextMatrix, 325
ChecksumFile
OpenGPS::Schemas::ISO5436_-2::Record4Type, 379, 380
ChecksumFile_
OpenGPS::Schemas::ISO5436_-2::Record4Type, 381
ChecksumFile_traits
OpenGPS::Schemas::ISO5436_-2::Record4Type, 378
ChecksumFile_type
OpenGPS::Schemas::ISO5436_-2::Record4Type, 378
Close
OpenGPS::BinaryLSBPointVectorReaderContext, 91
OpenGPS::BinaryLSBPointVectorWriterContext, 96
OpenGPS::BinaryMSBPointVectorReaderContext, 102
OpenGPS::BinaryMSBPointVectorWriterContext, 107
OpenGPS::BinaryPointVectorReaderContext, 113
OpenGPS::BinaryPointVectorWriterContext, 119
OpenGPS::ISO5436_2Container, 240
Comment
OpenGPS::Schemas::ISO5436_-2::Record2Type, 359, 360
Comment_
OpenGPS::Schemas::ISO5436_-2::Record2Type, 364
Comment_optional
OpenGPS::Schemas::ISO5436_-2::Record2Type, 355
Comment_traits
OpenGPS::Schemas::ISO5436_-2::Record2Type, 355
Comment_type
OpenGPS::Schemas::ISO5436_-2::Record2Type, 355
Compress
OpenGPS::ISO5436_2Container, 240

ConcatPathes
OpenGPS::Environment, 172
ConfigureNamespaceMap
OpenGPS::ISO5436_2Container, 240
Contacting
OpenGPS::Schemas::ISO5436_2::Type, 403
convertFormat
ISO5436_2_XML_Demo.cxx, 486
Kopie von ISO5436_2_XML_Demo_Test.cxx, 506
ConvertPointToCoord
OpenGPS::ISO5436_2Container, 241
ConvertULongLongToULong
OpenGPS::ISO5436_2Container, 241
ConvertULongToInt32
OpenGPS::ISO5436_2Container, 241
Create
OpenGPS::ISO5436_2Container, 242
CreateContainerTempFilePath
OpenGPS::ISO5436_2Container, 242
CreateDataPointParser
OpenGPS::PointVectorParser, 304
CreateDir
OpenGPS::Environment, 172
CreateDocument
OpenGPS::ISO5436_2Container, 242
CreateInstance
OpenGPS::Environment, 172
CreateNextPointIterator
OpenGPS::ISO5436_2Container, 243
CreatePointBuffer
OpenGPS::ISO5436_2Container, 243
OpenGPS::VectorBufferBuilder, 421
CreatePointVectorProxyContext
OpenGPS::ISO5436_2Container, 243
CreatePointVectorReaderContext
OpenGPS::ISO5436_2Container, 243
CreatePointVectorWriterContext
OpenGPS::ISO5436_2Container, 244
CreatePrevPointIterator
OpenGPS::ISO5436_2Container, 244
CreateTempDir
OpenGPS::ISO5436_2Container, 244
Creator
OpenGPS::Schemas::ISO5436_-2::Record2Type, 360, 361
Creator_
OpenGPS::Schemas::ISO5436_-2::Record2Type, 364
Creator_optional
OpenGPS::Schemas::ISO5436_-2::Record2Type, 355
Creator_traits
OpenGPS::Schemas::ISO5436_-2::Record2Type, 355
Creator_type
OpenGPS::Schemas::ISO5436_-2::Record2Type, 355
CX
OpenGPS::Schemas::ISO5436_2::AxesType, 69, 70
CX_
OpenGPS::Schemas::ISO5436_2::AxesType, 73
CX_traits
OpenGPS::Schemas::ISO5436_2::AxesType, 67
CX_type
OpenGPS::Schemas::ISO5436_2::AxesType, 67
CY
OpenGPS::Schemas::ISO5436_2::AxesType, 70
CY_
OpenGPS::Schemas::ISO5436_2::AxesType, 73
CY_traits
OpenGPS::Schemas::ISO5436_2::AxesType, 67
CY_type
OpenGPS::Schemas::ISO5436_2::AxesType, 67
CZ
OpenGPS::Schemas::ISO5436_2::AxesType, 71
CZ_
OpenGPS::Schemas::ISO5436_2::AxesType, 73
CZ_traits
OpenGPS::Schemas::ISO5436_2::AxesType, 67
CZ_type
OpenGPS::Schemas::ISO5436_2::AxesType, 67

D
OpenGPS::Schemas::ISO5436_2::DataType, 146
data_point_c.cxx, 456
ogps_GetDouble, 456
ogps_GetFloat, 456
ogps_GetInt16, 457
ogps_GetInt32, 457
ogps_GetPointType, 457
ogps_SetDouble, 457
ogps_SetFloat, 457
ogps_SetInt16, 457

ogps_SetInt32, 457
data_point_c.hxx, 457
OGPS_DataPoint, 458
OGPS_DataPointPtr, 458
data_point_impl.cxx, 458
data_point_impl.hxx, 459
data_point_parser.hxx, 459
data_point_proxy.cxx, 460
_CHECK_BUFFER_AND_THROW_-READ_EXCEPTION, 461
_CHECK_BUFFER_AND_THROW_-WRITE_EXCEPTION, 461
DataLink
OpenGPS::Schemas::ISO5436_-2::Record3Type, 371, 372
DataLink_
OpenGPS::Schemas::ISO5436_-2::Record3Type, 376
DataLink_optional
OpenGPS::Schemas::ISO5436_-2::Record3Type, 368
DataLink_traits
OpenGPS::Schemas::ISO5436_-2::Record3Type, 368
DataLink_type
OpenGPS::Schemas::ISO5436_-2::Record3Type, 369
DataLinkType
OpenGPS::Schemas::ISO5436_-2::DataLinkType, 127, 128
DataList
OpenGPS::Schemas::ISO5436_-2::Record3Type, 372, 373
DataList_
OpenGPS::Schemas::ISO5436_-2::Record3Type, 376
DataList_optional
OpenGPS::Schemas::ISO5436_-2::Record3Type, 369
DataList_traits
OpenGPS::Schemas::ISO5436_-2::Record3Type, 369
DataList_type
OpenGPS::Schemas::ISO5436_-2::Record3Type, 369
DataListType
OpenGPS::Schemas::ISO5436_-2::DataListType, 135, 136
DataPointImpl
OpenGPS::DataPointImpl, 139
DataPointParser
OpenGPS::DataPointParser, 143
DataPointProxy
OpenGPS::PointVectorProxy::DataPointProxy, 315
DataType
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 80, 81
OpenGPS::Schemas::ISO5436_2::DataType, 146, 147
DataType_
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 83
DataType_optional
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 77
DataType_traits
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 77
DataType_type
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 77
Date
OpenGPS::Schemas::ISO5436_-2::Record2Type, 361, 362
date
xml_schema, 54
Date_
OpenGPS::Schemas::ISO5436_-2::Record2Type, 364
date_time
xml_schema, 54
Date_traits
OpenGPS::Schemas::ISO5436_-2::Record2Type, 356
Date_type
OpenGPS::Schemas::ISO5436_-2::Record2Type, 356
Datum
OpenGPS::Schemas::ISO5436_-2::DataListType, 136, 137
OpenGPS::Schemas::ISO5436_2::Datum, 150, 151
Datum_
OpenGPS::Schemas::ISO5436_-2::DataListType, 137
Datum_const_iterator
OpenGPS::Schemas::ISO5436_-2::DataListType, 135
Datum_iterator
OpenGPS::Schemas::ISO5436_-2::DataListType, 135
Datum_sequence
OpenGPS::Schemas::ISO5436_-2::DataListType, 135
Datum_traits

OpenGPS::Schemas::ISO5436_-2::DataListType, 135
Datum_type
OpenGPS::Schemas::ISO5436_-2::DataListType, 135
day
xml_schema, 54
decimal
xml_schema, 54
Decompress
OpenGPS::ISO5436_2Container, 244, 245
DecompressChecksum
OpenGPS::ISO5436_2Container, 245
DecompressDataBin
OpenGPS::ISO5436_2Container, 245
DecompressMain
OpenGPS::ISO5436_2Container, 245
diagnostics
xml_schema, 54
do_is
OpenGPS::PointVectorWhitespaceFacet, 332
double_
xml_schema, 54
double_data_point_parser.cxx, 462
double_data_point_parser.hxx, 462
double_point_buffer.cxx, 463
double_point_buffer.hxx, 463
DoubleDataPointParser
OpenGPS::DoubleDataPointParser, 153
DoubleInlineValidity
OpenGPS::DoubleInlineValidity, 155
DoublePointBuffer
OpenGPS::DoublePointBuffer, 159
doubleValue
OpenGPS::DataPointImpl::_OGPS_DATA_-POINT_VALUE, 142
Doxygen.cpp, 464
DumpIt
OpenGPS::ExceptionHistory, 177
duplicate_id
xml_schema, 54
duration
xml_schema, 54

entities
xml_schema, 55
entity
xml_schema, 55
Environment
OpenGPS::Environment, 167
environment.cxx, 464
_OGPS_DOUBLE_SIZE, 465
_OGPS_FLOAT_SIZE, 465
_OGPS_INT_SIZE, 465
_OGPS_SHORT_SIZE, 465
environment.hxx, 465
error
xml_schema, 55
error_handler
xml_schema, 55
exception
xml_schema, 55
ExceptionHistory
OpenGPS::ExceptionHistory, 177
exceptions.cxx, 466
expected_attribute
xml_schema, 55
expected_element
xml_schema, 55
expected_text_content
xml_schema, 55

F
md5.c, 509
OpenGPS::Schemas::ISO5436_2::DataType, 146
FeatureType
OpenGPS::Schemas::ISO5436_-2::FeatureType, 180, 181
OpenGPS::Schemas::ISO5436_-2::Record1Type, 348
FeatureType_
OpenGPS::Schemas::ISO5436_-2::Record1Type, 350
FeatureType_traits
OpenGPS::Schemas::ISO5436_-2::Record1Type, 345
FeatureType_type
OpenGPS::Schemas::ISO5436_-2::Record1Type, 345
flags
xml_schema, 55
float_
xml_schema, 55
float_data_point_parser.cxx, 466
float_data_point_parser.hxx, 467
float_point_buffer.cxx, 468
float_point_buffer.hxx, 468
FloatDataPointParser
OpenGPS::FloatDataPointParser, 183
FloatInlineValidity
OpenGPS::FloatInlineValidity, 186
FloatPointBuffer
OpenGPS::FloatPointBuffer, 189
floatValue
OpenGPS::DataPointImpl::_OGPS_DATA_-POINT_VALUE, 142
Free

OpenGPS::DoublePointBuffer, 160
OpenGPS::FloatPointBuffer, 190
OpenGPS::Int16PointBuffer, 210
OpenGPS::Int32PointBuffer, 223
OpenGPS::PointBuffer, 291

Get
OpenGPS::DataPointImpl, 139, 140
OpenGPS::DoublePointBuffer, 160, 161
OpenGPS::FloatPointBuffer, 190, 191
OpenGPS::Int16PointBuffer, 210, 211
OpenGPS::Int32PointBuffer, 223, 224
OpenGPS::PointBuffer, 292, 293
OpenGPS::PointVectorProxy, 312
OpenGPS::PointVectorProxy::DataPointProxy, 316
OpenGPS::XmlPointVectorWriterContext, 431
GET_ULONG_LE
md5.c, 509
GetAltDirectorySeparator
OpenGPS::Environment, 172
GetAxisDataType
OpenGPS::ISO5436_2Container, 246
GetBuffer
OpenGPS::VectorBufferBuilder, 421
GetChecksumArchiveName
OpenGPS::ISO5436_2Container, 246
GetChecksumFileName
OpenGPS::ISO5436_2Container, 246
GetCurrent
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 261
GetDirectorySeparator
OpenGPS::Environment, 172
GetDocument
OpenGPS::ISO5436_2Container, 246
GetFileName
OpenGPS::Environment, 173
GetFilePath
OpenGPS::ISO5436_2Container, 246
GetFullFilePath
OpenGPS::ISO5436_2Container, 247
GetIncrementX
OpenGPS::ISO5436_2Container, 247
GetIncrementY
OpenGPS::ISO5436_2Container, 247
GetIndex
OpenGPS::PointVectorProxyContext, 319
OpenGPS::PointVectorProxyContextList, 322
OpenGPS::PointVectorProxyContextMatrix, 325
GetInstance
OpenGPS::Environment, 173
GetLastErrorDescription
OpenGPS::ExceptionHistory, 177
GetLastErrorId
OpenGPS::ExceptionHistory, 178
GetLastErrorMessage
OpenGPS::ExceptionHistory, 178
GetListCoord
OpenGPS::ISO5436_2Container, 247
GetListPoint
OpenGPS::ISO5436_2Container, 247
GetMainArchiveName
OpenGPS::ISO5436_2Container, 247
GetMainFileName
OpenGPS::ISO5436_2Container, 247
GetMatrixCoord
OpenGPS::ISO5436_2Container, 247
GetMatrixPoint
OpenGPS::ISO5436_2Container, 248
GetMaxU
OpenGPS::ISO5436_2Container, 248
GetMaxV
OpenGPS::ISO5436_2Container, 248
GetMaxW
OpenGPS::ISO5436_2Container, 248
GetMd5
OpenGPS::BinaryLSBPointVectorWriterContext, 96
OpenGPS::BinaryMSBPointVectorWriterContext, 107
OpenGPS::BinaryPointVectorWriterContext, 119
OpenGPS::ZipStreamBuffer, 438
GetOffsetX
OpenGPS::ISO5436_2Container, 249
GetOffsetY
OpenGPS::ISO5436_2Container, 249
GetOffsetZ
OpenGPS::ISO5436_2Container, 249
GetParser
OpenGPS::PointVectorParserBuilder, 309
GetPathName
OpenGPS::Environment, 173
GetPointBuffer
OpenGPS::DoubleInlineValidity, 156
OpenGPS::FloatInlineValidity, 186
OpenGPS::Int16ValidBuffer, 216
OpenGPS::Int32ValidBuffer, 229
OpenGPS::PointValidityProvider, 297
OpenGPS::ValidBuffer, 409
GetPointCount
OpenGPS::ISO5436_2Container, 249
GetPointDataArchiveName
OpenGPS::ISO5436_2Container, 249
GetPointDataFileName

OpenGPS::ISO5436_2Container, 249
GetPointType
OpenGPS::DataPointImpl, 140
OpenGPS::DoublePointBuffer, 161
OpenGPS::FloatPointBuffer, 192
OpenGPS::Int16PointBuffer, 211
OpenGPS::Int32PointBuffer, 224
OpenGPS::PointBuffer, 293
OpenGPS::PointVectorProxy::DataPointProxy, 316
GetPointVectorProxy
OpenGPS::VectorBuffer, 413
GetPosition
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 261, 262
GetSize
OpenGPS::DoublePointBuffer, 161
OpenGPS::FloatPointBuffer, 192
OpenGPS::Int16PointBuffer, 211
OpenGPS::Int32PointBuffer, 224
OpenGPS::PointBuffer, 293
GetStream
OpenGPS::BinaryLSBPointVectorReaderContext, 91
OpenGPS::BinaryLSBPointVectorWriterContext, 96
OpenGPS::BinaryMSBPointVectorReaderContext, 102
OpenGPS::BinaryMSBPointVectorWriterContext, 107
OpenGPS::BinaryPointVectorReaderContext, 113
OpenGPS::BinaryPointVectorWriterContext, 119
GetTempDir
OpenGPS::Environment, 173
OpenGPS::ISO5436_2Container, 250
GetUniqueName
OpenGPS::Environment, 174
GetValidityBuffer
OpenGPS::VectorBuffer, 414
GetValidityProvider
OpenGPS::VectorBuffer, 414
GetValidPointsArchiveName
OpenGPS::ISO5436_2Container, 250
GetValidPointsFileName
OpenGPS::ISO5436_2Container, 250
GetVariable
OpenGPS::Environment, 174
GetVectorBuffer
OpenGPS::ISO5436_2Container, 250
GetVendorSpecific
OpenGPS::ISO5436_2Container, 250
GetX
OpenGPS::PointVectorProxy, 312
OpenGPS::VectorBuffer, 415
GetXaxisDataType
OpenGPS::ISO5436_2Container, 250
GetY
OpenGPS::PointVectorProxy, 312, 313
OpenGPS::VectorBuffer, 415
GetYaxisDataType
OpenGPS::ISO5436_2Container, 250
GetZ
OpenGPS::PointVectorProxy, 313
OpenGPS::VectorBuffer, 415
GetZaxisDataType
OpenGPS::ISO5436_2Container, 251

HasDocument
OpenGPS::ISO5436_2Container, 251
HasNext
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 262
HasPrev
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 262
HasStream
OpenGPS::BinaryLSBPointVectorReaderContext, 91
OpenGPS::BinaryLSBPointVectorWriterContext, 96
OpenGPS::BinaryMSBPointVectorReaderContext, 102
OpenGPS::BinaryMSBPointVectorWriterContext, 107
OpenGPS::BinaryPointVectorReaderContext, 113
OpenGPS::BinaryPointVectorWriterContext, 119
HasTempDir
OpenGPS::ISO5436_2Container, 251
HasValidityBuffer
OpenGPS::VectorBuffer, 415
HasValidPointsLink
OpenGPS::ISO5436_2Container, 251
HasVectorBuffer
OpenGPS::ISO5436_2Container, 251
hex_binary
xml_schema, 56

I
OpenGPS::Schemas::ISO5436_2::AxisType, 85
OpenGPS::Schemas::ISO5436_2::DataType, 146
id

xml_schema, 56
Identification
OpenGPS::Schemas::ISO5436_-2::ProbingSystemType, 340
Identification_
OpenGPS::Schemas::ISO5436_-2::ProbingSystemType, 342
Identification_traits
OpenGPS::Schemas::ISO5436_-2::ProbingSystemType, 338
Identification_type
OpenGPS::Schemas::ISO5436_-2::ProbingSystemType, 338
idref
xml_schema, 56
idrefs
xml_schema, 56
Increment
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 81, 82
Increment_
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 83
Increment_optional
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 77
Increment_traits
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 77
Increment_type
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 78
IncrementIndex
OpenGPS::PointVectorProxyContext, 319
OpenGPS::PointVectorProxyContextList, 322
OpenGPS::PointVectorProxyContextMatrix, 325
info.cxx, 469
info_c.cxx, 469
ogps_GetAboutInfo, 470
ogps_GetCopyrightInfo, 470
ogps_GetLicenseInfo, 470
ogps_GetNameInfo, 470
ogps_GetVersionInfo, 470
ogps_PrintCopyrightInfo, 470
ogps_PrintLicenseInfo, 470
ogps_PrintVersionInfo, 471
inline_validity.cxx, 471
inline_validity.hxx, 471
InputBinaryFileStream
OpenGPS::InputBinaryFileStream, 195
instance
_OGPS_DATA_POINT, 61
_OGPS_ISO5436_2_HANDLE, 61
_OGPS_POINT_ITERATOR, 62
_OGPS_POINT_VECTOR, 63
Instrument
OpenGPS::Schemas::ISO5436_-2::Record2Type, 362, 363
Instrument_
OpenGPS::Schemas::ISO5436_-2::Record2Type, 364
Instrument_traits
OpenGPS::Schemas::ISO5436_-2::Record2Type, 356
Instrument_type
OpenGPS::Schemas::ISO5436_-2::Record2Type, 356
InstrumentType
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 199, 200
int16_data_point_parser.cxx, 472
int16_data_point_parser.hxx, 473
int16_point_buffer.cxx, 473
int16_point_buffer.hxx, 474
Int16DataPointParser
OpenGPS::Int16DataPointParser, 205
Int16PointBuffer
OpenGPS::Int16PointBuffer, 209
Int16ValidBuffer
OpenGPS::Int16ValidBuffer, 215
int16Value
OpenGPS::DataPointImpl::_OGPS_DATA_-POINT_VALUE, 142
int32_data_point_parser.cxx, 475
int32_data_point_parser.hxx, 475
int32_point_buffer.cxx, 476
int32_point_buffer.hxx, 476
Int32DataPointParser
OpenGPS::Int32DataPointParser, 218
Int32PointBuffer
OpenGPS::Int32PointBuffer, 222
Int32ValidBuffer
OpenGPS::Int32ValidBuffer, 228
int32Value
OpenGPS::DataPointImpl::_OGPS_DATA_-POINT_VALUE, 142
int_
xml_schema, 56
integer
xml_schema, 56
ipad
md5_context, 284
IsAllocated
OpenGPS::Int16ValidBuffer, 216
OpenGPS::Int32ValidBuffer, 229
OpenGPS::ValidBuffer, 409
IsBinary

OpenGPS::ISO5436_2Container, 251
IsGood
OpenGPS::BinaryLSBPointVectorReaderContext, 91
OpenGPS::BinaryLSBPointVectorWriterContext, 97
OpenGPS::BinaryMSBPointVectorReaderContext, 102
OpenGPS::BinaryMSBPointVectorWriterContext, 108
OpenGPS::BinaryPointVectorReaderContext, 113
OpenGPS::BinaryPointVectorWriterContext, 119
OpenGPS::PointVectorWriterContext, 334
OpenGPS::XmlPointVectorReaderContext, 425
OpenGPS::XmlPointVectorWriterContext, 431
IsIncrementalX
OpenGPS::ISO5436_2Container, 252
IsIncrementalY
OpenGPS::ISO5436_2Container, 252
IsLittleEndian
OpenGPS::Environment, 174
IsMatrix
OpenGPS::ISO5436_2Container, 252
OpenGPS::PointVectorProxyContext, 319
OpenGPS::PointVectorProxyContextList, 322
OpenGPS::PointVectorProxyContextMatrix, 325
IsMatrixCoordValid
OpenGPS::ISO5436_2Container, 252
ISO5436_2
OpenGPS::Schemas::ISO5436_2, 33–42
iso5436_2.cxx, 477
iso5436_2_c.cxx, 477
ogps_AppendVendorSpecific, 479
ogps_CloseISO5436_2, 479
ogps_CreateListISO5436_2, 479
ogps_CreateMatrixISO5436_2, 479
ogps_CreateNextPointIterator, 480
ogps_CreatePrevPointIterator, 480
ogps_GetDocument, 480
ogps_GetListCoord, 480
ogps_GetListPoint, 480
ogps_GetMatrixCoord, 480
ogps_GetMatrixDimensions, 480
ogps_GetMatrixPoint, 480
ogps_GetVendorSpecific, 481
ogps_IsMatrix, 481
ogps_IsMatrixCoordValid, 481
ogps_OpenISO5436_2, 481
ogps_SetListPoint, 481
ogps_SetMatrixPoint, 481
ogps_WriteISO5436_2, 481
iso5436_2_container.cxx, 481
_OPENGPS_FILE_URI_PREF, 483
_OPENGPS_ZIP_CHUNK_MAX, 483
PointVectorReaderContextAutoPtr, 483
PointVectorWriterContextAutoPtr, 483
iso5436_2_container.hxx, 483
iso5436_2_handle_c.hxx, 484
OGPS_ISO5436_2, 485
OGPS_ISO5436_2Handle, 485
ISO5436_2_XML/ Directory Reference, 21
ISO5436_2_XML/c/ Directory Reference, 15
ISO5436_2_XML/cxx/ Directory Reference, 16
ISO5436_2_XML/xyssl/ Directory Reference, 22
ISO5436_2_XML_Demo.cxx, 485
_tmain, 486
convertFormat, 486
mediumComplexExample, 486
performanceDouble, 487
performanceInt16, 487
readonlyExample, 487
readonlyExample2, 487
readonlyExample3, 487
readonlyExample4, 487
readonlyExampleMatrix, 487
simpleExample, 487
ISO5436_2_XML_Demo/ Directory Reference, 21
iso5436_2_xsd.cxx, 488
iso5436_2_xsd.hxx, 494
ISO5436_2Container
OpenGPS::ISO5436_2Container, 239
ISO5436_2Type
OpenGPS::Schemas::ISO5436_2::ISO5436_2Type, 269, 270
IsValid
OpenGPS::BinaryLSBPointVectorReaderContext, 91
OpenGPS::BinaryMSBPointVectorReaderContext, 102
OpenGPS::BinaryPointVectorReaderContext, 113
OpenGPS::DataPointImpl, 140
OpenGPS::DoubleInlineValidity, 156
OpenGPS::FloatInlineValidity, 186
OpenGPS::Int16ValidBuffer, 216
OpenGPS::Int32ValidBuffer, 229
OpenGPS::PointValidityProvider, 297
OpenGPS::PointVectorProxy::DataPointProxy, 316
OpenGPS::PointVectorReaderContext, 328
OpenGPS::ValidBuffer, 409
OpenGPS::XmlPointVectorReaderContext, 425

Kopie von ISO5436_2_XML_Demo_Test.cxx, 505
_tmain, 506
convertFormat, 506
mediumComplexExample, 507
performanceDouble, 507
performanceInt16, 507
readonlyExample, 507
readonlyExample2, 507
readonlyExample3, 507
readonlyExample4, 507
readonlyExampleMatrix, 507
simpleExample, 507

L
OpenGPS::Schemas::ISO5436_2::DataType, 146
language
xml_schema, 56
ListDimension
OpenGPS::Schemas::ISO5436_-2::Record3Type, 373, 374
ListDimension_
OpenGPS::Schemas::ISO5436_-2::Record3Type, 376
ListDimension_optional
OpenGPS::Schemas::ISO5436_-2::Record3Type, 369
ListDimension_traits
OpenGPS::Schemas::ISO5436_-2::Record3Type, 369
ListDimension_type
OpenGPS::Schemas::ISO5436_-2::Record3Type, 369
long_
xml_schema, 56

m_Buffer
OpenGPS::BinaryPointVectorWriterContext, 122
OpenGPS::DoublePointBuffer, 163
OpenGPS::FloatPointBuffer, 193
OpenGPS::Int16PointBuffer, 213
OpenGPS::Int32PointBuffer, 226
OpenGPS::PointVectorProxy, 313
OpenGPS::PointVectorProxy::DataPointProxy, 317
OpenGPS::VectorBufferBuilder, 422
m_CompressionLevel
OpenGPS::ISO5436_2Container, 256
m_Context
OpenGPS::PointVectorProxy, 313
OpenGPS::PointVectorProxy::DataPointProxy, 317
m_DataBinChecksum
OpenGPS::ISO5436_2Container, 257
m_Document
OpenGPS::ISO5436_2Container, 257
m_FilePath
OpenGPS::ISO5436_2Container, 257
m_Handle
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 263
OpenGPS::ZipStreamBuffer, 438
m_Index
OpenGPS::PointVectorProxyContextList, 322
m_Instance
OpenGPS::Environment, 175
m_IsCreating
OpenGPS::ISO5436_2Container, 257
m_IsForward
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 263
m_IsMatrix
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 263
m_IsReadOnly
OpenGPS::ISO5436_2Container, 257
m_IsReset
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 263
m_LastErrorDescription
OpenGPS::ExceptionHistory, 179
m_LastErrorId
OpenGPS::ExceptionHistory, 179
m_LastErrorMessage
OpenGPS::ExceptionHistory, 179
m_LastErrorSource
OpenGPS::ExceptionHistory, 179
m_Locale
OpenGPS::InputBinaryFileStream, 195
OpenGPS::OutputBinaryFileStream, 288
OpenGPS::PointVectorInputStringStream, 299
OpenGPS::PointVectorOutputStringStream, 302
m_MainChecksum
OpenGPS::ISO5436_2Container, 257
m_MaxIndex
OpenGPS::PointVectorProxyContextList, 322
m_MaxU
OpenGPS::PointVectorProxyContextMatrix, 326
m_MaxV
OpenGPS::PointVectorProxyContextMatrix, 326
m_MaxW
OpenGPS::PointVectorProxyContextMatrix, 326
m_Md5Context

OpenGPS::ZipStreamBuffer, 438
m_NeedsSeparator
OpenGPS::XmlPointVectorWriterContext, 434
m_Next
OpenGPS::XmlPointVectorReaderContext, 428
m_Parser
OpenGPS::PointVectorParserBuilder, 310
m_PointBuffer
OpenGPS::PointValidityProvider, 298
m_PointDataFileName
OpenGPS::ISO5436_2Container, 257
m_PointVector
OpenGPS::ISO5436_2Container, 258
m_PointVectorList
OpenGPS::XmlPointVectorReaderContext, 428
OpenGPS::XmlPointVectorWriterContext, 434
m_ProxyContext
OpenGPS::ISO5436_2Container, 258
m_RawSize
OpenGPS::ValidBuffer, 410
m_Size
OpenGPS::PointBuffer, 295
m_Stream
OpenGPS::BinaryPointVectorReaderContext, 116
OpenGPS::BinaryPointVectorWriterContext, 122
OpenGPS::XmlPointVectorReaderContext, 428
OpenGPS::XmlPointVectorWriterContext, 434
m_TempBasePath
OpenGPS::ISO5436_2Container, 258
m_TempPath
OpenGPS::ISO5436_2Container, 258
m_Type
OpenGPS::DataPointImpl, 141
m_U
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 263
OpenGPS::PointVectorProxyContextMatrix, 326
m_V
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 263
OpenGPS::PointVectorProxyContextMatrix, 326
m_ValidBinChecksum
OpenGPS::ISO5436_2Container, 258
m_ValidBuffer
OpenGPS::VectorBuffer, 417
m_ValidityBuffer
OpenGPS::ValidBuffer, 410
m_ValidityProvider
OpenGPS::VectorBuffer, 417
m_ValidPointsFileName
OpenGPS::ISO5436_2Container, 258
m_Value
OpenGPS::DataPointImpl, 141
m_VectorBufferBuilder
OpenGPS::ISO5436_2Container, 258
m_VendorSpecific
OpenGPS::ISO5436_2Container, 258
m_VendorURI
OpenGPS::ISO5436_2Container, 259
m_W
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 263
OpenGPS::PointVectorProxyContextMatrix, 326
m_X
OpenGPS::PointVectorParser, 306
OpenGPS::PointVectorProxy, 313
OpenGPS::VectorBuffer, 417
m_Y
OpenGPS::PointVectorParser, 306
OpenGPS::PointVectorProxy, 313
OpenGPS::VectorBuffer, 417
m_Z
OpenGPS::PointVectorParser, 306
OpenGPS::PointVectorProxy, 314
OpenGPS::VectorBuffer, 417
Manufacturer
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 200, 201
Manufacturer_
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 204
Manufacturer_traits
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 198
Manufacturer_type
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 198
MatrixDimension
OpenGPS::Schemas::ISO5436_-2::Record3Type, 374, 375
MatrixDimension_
OpenGPS::Schemas::ISO5436_-2::Record3Type, 376
MatrixDimension_optional
OpenGPS::Schemas::ISO5436_-2::Record3Type, 370
MatrixDimension_traits

OpenGPS::Schemas::ISO5436_-
2::Record3Type, 370
MatrixDimension_type
OpenGPS::Schemas::ISO5436_-
2::Record3Type, 370
MatrixDimensionType
OpenGPS::Schemas::ISO5436_-
2::MatrixDimensionType, 280
md5
md5.c, 510
md5.h, 514
md5.c, 508
F, 509
GET_ULONG_LE, 509
md5, 510
md5_file, 510
md5_finish, 510
md5_hmac, 511
md5_hmac_finish, 511
md5_hmac_starts, 511
md5_hmac_update, 511
md5_padding, 512
md5_process, 512
md5_starts, 512
md5_update, 512
P, 509
PUT_ULONG_LE, 509
S, 510
XYSSL_MD5_C, 510
md5.h, 513
md5, 514
md5_file, 514
md5_finish, 514
md5_hmac, 515
md5_hmac_finish, 515
md5_hmac_starts, 515
md5_hmac_update, 515
md5_self_test, 516
md5_starts, 516
md5_update, 516
md5_context, 283
buffer, 284
ipad, 284
opad, 284
state, 284
total, 284
md5_file
md5.c, 510
md5.h, 514
md5_finish
md5.c, 510
md5.h, 514
md5_hmac
md5.c, 511
md5.h, 515
md5_hmac_finish
md5.c, 511
md5.h, 515
md5_hmac_starts
md5.c, 511
md5.h, 515
md5_hmac_update
md5.c, 511
md5.h, 515
md5_padding
md5.c, 512
md5_process
md5.c, 512
md5_self_test
md5.h, 516
md5_starts
md5.c, 512
md5.h, 516
md5_update
md5.c, 512
md5.h, 516
MD5ChecksumPointData
OpenGPS::Schemas::ISO5436_-
2::DataLinkType, 128, 129
MD5ChecksumPointData_
OpenGPS::Schemas::ISO5436_-
2::DataLinkType, 132
MD5ChecksumPointData_traits
OpenGPS::Schemas::ISO5436_-
2::DataLinkType, 126
MD5ChecksumPointData_type
OpenGPS::Schemas::ISO5436_-
2::DataLinkType, 126
MD5ChecksumValidPoints
OpenGPS::Schemas::ISO5436_-
2::DataLinkType, 129, 130
MD5ChecksumValidPoints_
OpenGPS::Schemas::ISO5436_-
2::DataLinkType, 132
MD5ChecksumValidPoints_optional
OpenGPS::Schemas::ISO5436_-
2::DataLinkType, 126
MD5ChecksumValidPoints_traits
OpenGPS::Schemas::ISO5436_-
2::DataLinkType, 126
MD5ChecksumValidPoints_type
OpenGPS::Schemas::ISO5436_-
2::DataLinkType, 126
mediumComplexExample
ISO5436_2_XML_Demo.cxx, 486
Kopie von ISO5436_2_XML_Demo_Test.cxx,
507
messages_c.cxx, 517

ogps_GetErrorDescription, 517
ogps_GetErrorId, 517
ogps_GetErrorMessage, 517
ogps_HasError, 517
messages_c.hxx, 517
_OPENGPS_GENERIC_EXCEPTION_-HANDLER, 519
_OPENGPS_GENERIC_EXCEPTION_-HANDLER_CLEANUP, 519
_OPENGPS_GENERIC_EXCEPTION_-HANDLER_RETVALBOOL, 520
missing_data_point_parser.cxx, 521
missing_data_point_parser.hxx, 521
MissingDataPointParser
OpenGPS::MissingDataPointParser, 285
Model
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 201, 202
Model_
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 204
Model_traits
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 198
Model_type
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 199
month
xml_schema, 56
month_day
xml_schema, 56
MoveNext
OpenGPS::BinaryLSBPointVectorReaderContext, 92
OpenGPS::BinaryLSBPointVectorWriterContext, 97
OpenGPS::BinaryMSBPointVectorReaderContext, 103
OpenGPS::BinaryMSBPointVectorWriterContext, 108
OpenGPS::BinaryPointVectorReaderContext, 114
OpenGPS::BinaryPointVectorWriterContext, 120
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 262
OpenGPS::PointVectorReaderContext, 329
OpenGPS::PointVectorWriterContext, 334
OpenGPS::XmlPointVectorReaderContext, 425
OpenGPS::XmlPointVectorWriterContext, 432
MovePrev
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 262

name
xml_schema, 56
namespace_info
xml_schema, 57
namespace_infomap
xml_schema, 57
ncname
xml_schema, 57
negative_integer
xml_schema, 57
nmtoken
xml_schema, 57
nmtokens
xml_schema, 57
no_namespace_mapping
xml_schema, 57
no_prefix_mapping
xml_schema, 57
no_type_info
xml_schema, 57
non_negative_integer
xml_schema, 57
non_positive_integer
xml_schema, 58
NonContacting
OpenGPS::Schemas::ISO5436_2::Type, 403
normalized_string
xml_schema, 58
not_derived
xml_schema, 58

Offset
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 82
Offset_
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 83
Offset_optional
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 78
Offset_traits
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 78
Offset_type
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 78
ogps_AppendVendorSpecific
iso5436_2_c.cxx, 479
ogps_CloseISO5436_2
iso5436_2_c.cxx, 479
ogps_CreateListISO5436_2

iso5436_2_c.cxx, 479
ogps_CreateMatrixISO5436_2
iso5436_2_c.cxx, 479
ogps_CreateNextPointIterator
iso5436_2_c.cxx, 480
ogps_CreatePointVector
point_vector_c.cxx, 530
ogps_CreatePrevPointIterator
iso5436_2_c.cxx, 480
OGPS_DataPoint
data_point_c.hxx, 458
OGPS_DataPointPtr
data_point_c.hxx, 458
OGPS_DataPointValue
OpenGPS::DataPointImpl, 139
ogps_FreePointIterator
point_iterator_c.cxx, 524
ogps_FreePointVector
point_vector_c.cxx, 530
ogps_GetAboutInfo
info_c.cxx, 470
ogps_GetCopyrightInfo
info_c.cxx, 470
ogps_GetCurrentPoint
point_iterator_c.cxx, 524
ogps_GetDocument
iso5436_2_c.cxx, 480
ogps_GetDouble
data_point_c.cxx, 456
ogps_GetDoubleX
point_vector_c.cxx, 530
ogps_GetDoubleY
point_vector_c.cxx, 530
ogps_GetDoubleZ
point_vector_c.cxx, 530
ogps_GetErrorDescription
messages_c.cxx, 517
ogps_GetErrorId
messages_c.cxx, 517
ogps_GetErrorMessage
messages_c.cxx, 517
ogps_GetFloat
data_point_c.cxx, 456
ogps_GetFloatX
point_vector_c.cxx, 530
ogps_GetFloatY
point_vector_c.cxx, 531
ogps_GetFloatZ
point_vector_c.cxx, 531
ogps_GetInt16
data_point_c.cxx, 457
ogps_GetInt16X
point_vector_c.cxx, 531
ogps_GetInt16Y
point_vector_c.cxx, 531
ogps_GetInt16Z
point_vector_c.cxx, 531
ogps_GetInt32
data_point_c.cxx, 457
ogps_GetInt32X
point_vector_c.cxx, 531
ogps_GetInt32Y
point_vector_c.cxx, 531
ogps_GetInt32Z
point_vector_c.cxx, 531
ogps_GetLicenseInfo
info_c.cxx, 470
ogps_GetListCoord
iso5436_2_c.cxx, 480
ogps_GetListPoint
iso5436_2_c.cxx, 480
ogps_GetListPosition
point_iterator_c.cxx, 525
ogps_GetMatrixCoord
iso5436_2_c.cxx, 480
ogps_GetMatrixDimensions
iso5436_2_c.cxx, 480
ogps_GetMatrixPoint
iso5436_2_c.cxx, 480
ogps_GetMatrixPosition
point_iterator_c.cxx, 525
ogps_GetNameInfo
info_c.cxx, 470
ogps_GetPointType
data_point_c.cxx, 457
ogps_GetPointTypeX
point_vector_c.cxx, 531
ogps_GetPointTypeY
point_vector_c.cxx, 531
ogps_GetPointTypeZ
point_vector_c.cxx, 532
ogps_GetVendorSpecific
iso5436_2_c.cxx, 481
ogps_GetVersionInfo
info_c.cxx, 470
ogps_GetX
point_vector_c.cxx, 532
ogps_GetXYZ
point_vector_c.cxx, 532
ogps_GetY
point_vector_c.cxx, 532
ogps_GetZ
point_vector_c.cxx, 532
ogps_HasError
messages_c.cxx, 517
ogps_HasNextPoint
point_iterator_c.cxx, 525
ogps_HasPrevPoint

point_iterator_c.cxx, 525
ogps_IsMatrix
iso5436_2_c.cxx, 481
ogps_IsMatrixCoordValid
iso5436_2_c.cxx, 481
OGPS_ISO5436_2
iso5436_2_handle_c.hxx, 485
OGPS_ISO5436_2Handle
iso5436_2_handle_c.hxx, 485
ogps_IsValidPoint
point_vector_c.cxx, 532
ogps_MoveNextPoint
point_iterator_c.cxx, 525
ogps_MovePrevPoint
point_iterator_c.cxx, 525
ogps_OpenISO5436_2
iso5436_2_c.cxx, 481
OGPS_PointIterator
point_iterator_c.hxx, 527
OGPS_PointIteratorPtr
point_iterator_c.hxx, 527
OGPS_PointVector
point_vector_c.hxx, 534
OGPS_PointVectorPtr
point_vector_c.hxx, 534
ogps_PrintCopyrightInfo
info_c.cxx, 470
ogps_PrintLicenseInfo
info_c.cxx, 470
ogps_PrintVersionInfo
info_c.cxx, 471
ogps_ResetNextPointIterator
point_iterator_c.cxx, 525
ogps_ResetPrevPointIterator
point_iterator_c.cxx, 525
ogps_SetCurrentPoint
point_iterator_c.cxx, 525
ogps_SetDouble
data_point_c.cxx, 457
ogps_SetDoubleX
point_vector_c.cxx, 532
ogps_SetDoubleY
point_vector_c.cxx, 532
ogps_SetDoubleZ
point_vector_c.cxx, 532
ogps_SetFloat
data_point_c.cxx, 457
ogps_SetFloatX
point_vector_c.cxx, 532
ogps_SetFloatY
point_vector_c.cxx, 533
ogps_SetFloatZ
point_vector_c.cxx, 533
ogps_SetInt16
data_point_c.cxx, 457
ogps_SetInt16X
point_vector_c.cxx, 533
ogps_SetInt16Y
point_vector_c.cxx, 533
ogps_SetInt16Z
point_vector_c.cxx, 533
ogps_SetInt32
data_point_c.cxx, 457
ogps_SetInt32X
point_vector_c.cxx, 533
ogps_SetInt32Y
point_vector_c.cxx, 533
ogps_SetInt32Z
point_vector_c.cxx, 533
ogps_SetListPoint
iso5436_2_c.cxx, 481
ogps_SetMatrixPoint
iso5436_2_c.cxx, 481
ogps_WriteISO5436_2
iso5436_2_c.cxx, 481
opad
md5_context, 284
Open
OpenGPS::ISO5436_2Container, 252
OpenGPS, 22
PointIteratorAutoPtr, 26
PointVectorAutoPtr, 26
PointVectorParserBuilderAutoPtr, 26
PointVectorProxyContextAutoPtr, 26
VectorBufferAutoPtr, 26
VectorBufferBuilderAutoPtr, 26
OpenGPS::BinaryLSBPointVectorReaderContext, 88
~BinaryLSBPointVectorReaderContext, 90
BinaryLSBPointVectorReaderContext, 90
Close, 91
GetStream, 91
HasStream, 91
IsGood, 91
IsValid, 91
MoveNext, 92
Read, 92, 93
Skip, 93
OpenGPS::BinaryLSBPointVectorWriterContext, 94
~BinaryLSBPointVectorWriterContext, 96
BinaryLSBPointVectorWriterContext, 96
Close, 96
GetMd5, 96
GetStream, 96
HasStream, 96
IsGood, 97
MoveNext, 97

Skip, 97
Write, 98, 99
OpenGPS::BinaryMSBPointVectorReaderContext, 99
~BinaryMSBPointVectorReaderContext, 101
BinaryMSBPointVectorReaderContext, 101
Close, 102
GetStream, 102
HasStream, 102
IsGood, 102
IsValid, 102
MoveNext, 103
Read, 103, 104
Skip, 104
OpenGPS::BinaryMSBPointVectorWriterContext, 105
~BinaryMSBPointVectorWriterContext, 107
BinaryMSBPointVectorWriterContext, 107
Close, 107
GetMd5, 107
GetStream, 107
HasStream, 107
IsGood, 108
MoveNext, 108
Skip, 108
Write, 109, 110
OpenGPS::BinaryPointVectorReaderContext, 110
~BinaryPointVectorReaderContext, 112
BinaryPointVectorReaderContext, 112
Close, 113
GetStream, 113
HasStream, 113
IsGood, 113
IsValid, 113
m_Stream, 116
MoveNext, 114
Read, 114, 115
Skip, 115
OpenGPS::BinaryPointVectorWriterContext, 116
~BinaryPointVectorWriterContext, 118
BinaryPointVectorWriterContext, 118
Close, 119
GetMd5, 119
GetStream, 119
HasStream, 119
IsGood, 119
m_Buffer, 122
m_Stream, 122
MoveNext, 120
Skip, 120
Write, 120, 121
OpenGPS::DataPointImpl, 137
~DataPointImpl, 139
DataPointImpl, 139
Get, 139, 140
GetPointType, 140
IsValid, 140
m_Type, 141
m_Value, 141
OGPS_DataPointValue, 139
Reset, 140
Set, 140
OpenGPS::DataPointImpl::_OGPS_DATA_-POINT_VALUE, 141
doubleValue, 142
floatValue, 142
int16Value, 142
int32Value, 142
OpenGPS::DataPointParser, 142
~DataPointParser, 143
DataPointParser, 143
Read, 144
Write, 144
OpenGPS::DoubleDataPointParser, 152
~DoubleDataPointParser, 153
DoubleDataPointParser, 153
Read, 153
Write, 153
OpenGPS::DoubleInlineValidity, 154
~DoubleInlineValidity, 155
DoubleInlineValidity, 155
GetPointBuffer, 156
IsValid, 156
SetValid, 156
OpenGPS::DoublePointBuffer, 157
~DoublePointBuffer, 159
Allocate, 159
DoublePointBuffer, 159
Free, 160
Get, 160, 161
GetPointType, 161
GetSize, 161
m_Buffer, 163
Set, 162, 163
OpenGPS::Environment, 163
~Environment, 167
ByteSwap, 167–169
ByteSwap16, 169
ByteSwap32, 170, 171
ByteSwap64, 171
ConcatPathes, 172
CreateDir, 172
CreateInstance, 172
Environment, 167
GetAltDirectorySeparator, 172
GetDirectorySeparator, 172
GetFileName, 173
GetInstance, 173

GetPathName, 173
GetTempDir, 173
GetUniqueName, 174
GetVariable, 174
IsLittleEndian, 174
m_Instance, 175
PathExists, 174
RemoveDir, 174
RemoveFile, 174
RenameFile, 175
Reset, 175
OpenGPS::ExceptionHistory, 176
~ExceptionHistory, 177
DumpIt, 177
ExceptionHistory, 177
GetLastErrorDescription, 177
GetLastErrorId, 178
GetLastErrorMessage, 178
m_LastErrorDescription, 179
m_LastErrorId, 179
m_LastErrorMessage, 179
m_LastErrorSource, 179
Reset, 178
SetLastException, 178
OpenGPS::FloatDataPointParser, 182
~FloatDataPointParser, 183
FloatDataPointParser, 183
Read, 184
Write, 184
OpenGPS::FloatInlineValidity, 185
~FloatInlineValidity, 186
FloatInlineValidity, 186
GetPointBuffer, 186
IsValid, 186
SetValid, 187
OpenGPS::FloatPointBuffer, 187
~FloatPointBuffer, 189
Allocate, 189, 190
FloatPointBuffer, 189
Free, 190
Get, 190, 191
GetPointType, 192
GetSize, 192
m_Buffer, 193
Set, 192, 193
OpenGPS::InputBinaryFileStream, 193
~InputBinaryFileStream, 195
BaseType, 194
InputBinaryFileStream, 195
m_Locale, 195
OpenGPS::Int16DataPointParser, 205
~Int16DataPointParser, 206
Int16DataPointParser, 205
Read, 206
Write, 206
OpenGPS::Int16PointBuffer, 207
~Int16PointBuffer, 209
Allocate, 209
Free, 210
Get, 210, 211
GetPointType, 211
GetSize, 211
Int16PointBuffer, 209
m_Buffer, 213
Set, 211, 212
OpenGPS::Int16ValidBuffer, 213
~Int16ValidBuffer, 215
Allocate, 215
AllocateRaw, 215
GetPointBuffer, 216
Int16ValidBuffer, 215
IsAllocated, 216
IsValid, 216
Read, 216
Reset, 217
SetValid, 217
Write, 217
OpenGPS::Int32DataPointParser, 218
~Int32DataPointParser, 219
Int32DataPointParser, 218
Read, 219
Write, 219
OpenGPS::Int32PointBuffer, 220
~Int32PointBuffer, 222
Allocate, 222
Free, 223
Get, 223, 224
GetPointType, 224
GetSize, 224
Int32PointBuffer, 222
m_Buffer, 226
Set, 224, 225
OpenGPS::Int32ValidBuffer, 226
~Int32ValidBuffer, 228
Allocate, 228
AllocateRaw, 228
GetPointBuffer, 229
Int32ValidBuffer, 228
IsAllocated, 229
IsValid, 229
Read, 229
Reset, 230
SetValid, 230
Write, 230
OpenGPS::ISO5436_2Container, 231
~ISO5436_2Container, 239
AppendVendorSpecific, 239
BuildPointVectorParser, 239

BuildVectorBuffer, 240
Close, 240
Compress, 240
ConfigureNamespaceMap, 240
ConvertPointToCoord, 241
ConvertULongLongToULong, 241
ConvertULongToInt32, 241
Create, 242
CreateContainerTempFilePath, 242
CreateDocument, 242
CreateNextPointIterator, 243
CreatePointBuffer, 243
CreatePointVectorProxyContext, 243
CreatePointVectorReaderContext, 243
CreatePointVectorWriterContext, 244
CreatePrevPointIterator, 244
CreateTempDir, 244
Decompress, 244, 245
DecompressChecksum, 245
DecompressDataBin, 245
DecompressMain, 245
GetAxisDataType, 246
GetChecksumArchiveName, 246
GetChecksumFileName, 246
GetDocument, 246
GetFilePath, 246
GetFullFilePath, 247
GetIncrementX, 247
GetIncrementY, 247
GetListCoord, 247
GetListPoint, 247
GetMainArchiveName, 247
GetMainFileName, 247
GetMatrixCoord, 247
GetMatrixPoint, 248
GetMaxU, 248
GetMaxV, 248
GetMaxW, 248
GetOffsetX, 249
GetOffsetY, 249
GetOffsetZ, 249
GetPointCount, 249
GetPointDataArchiveName, 249
GetPointDataFileName, 249
GetTempDir, 250
GetValidPointsArchiveName, 250
GetValidPointsFileName, 250
GetVectorBuffer, 250
GetVendorSpecific, 250
GetXaxisDataType, 250
GetYaxisDataType, 250
GetZaxisDataType, 251
HasDocument, 251
HasTempDir, 251
HasValidPointsLink, 251
HasVectorBuffer, 251
IsBinary, 251
IsIncrementalX, 252
IsIncrementalY, 252
IsMatrix, 252
IsMatrixCoordValid, 252
ISO5436_2Container, 239
m_CompressionLevel, 256
m_DataBinChecksum, 257
m_Document, 257
m_FilePath, 257
m_IsCreating, 257
m_IsReadOnly, 257
m_MainChecksum, 257
m_PointDataFileName, 257
m_PointVector, 258
m_ProxyContext, 258
m_TempBasePath, 258
m_TempPath, 258
m_ValidBinChecksum, 258
m_ValidPointsFileName, 258
m_VectorBufferBuilder, 258
m_VendorSpecific, 258
m_VendorURI, 259
Open, 252
ReadDocument, 252
ReadMd5FromFile, 252
ReadXmlDocument, 253
RemoveTempDir, 253
Reset, 253
ResetValidPointsLink, 253
ResetXmlPointList, 253
SafeMultipilcation, 253
SaveChecksumFile, 254
SavePointBuffer, 254
SaveValidPointsLink, 254
SaveXmlDocument, 255
SetListPoint, 255
SetMatrixPoint, 255
StringList, 239
TestChecksums, 255
VerifyChecksum, 255
VerifyDataBinChecksum, 256
VerifyMainChecksum, 256
VerifyValidBinChecksum, 256
Write, 256
WriteVendorSpecific, 256
OpenGPS::ISO5436_-
2Container::PointIteratorImpl, 259
~PointIteratorImpl, 261
GetCurrent, 261
GetPosition, 261, 262
HasNext, 262

HasPrev, 262
m_Handle, 263
m_IsForward, 263
m_IsMatrix, 263
m_IsReset, 263
m_U, 263
m_V, 263
m_W, 263
MoveNext, 262
MovePrev, 262
operator=, 262
PointIteratorImpl, 261
ResetNext, 262
ResetPrev, 262
SetCurrent, 262
OpenGPS::MissingDataPointParser, 284
~MissingDataPointParser, 285
MissingDataPointParser, 285
Read, 286
Write, 286
OpenGPS::OutputBinaryFileStream, 287
~OutputBinaryFileStream, 288
BaseType, 288
m_Locale, 288
OutputBinaryFileStream, 288
OpenGPS::PointBuffer, 288
~PointBuffer, 291
Allocate, 291
Free, 291
Get, 292, 293
GetPointType, 293
GetSize, 293
m_Size, 295
PointBuffer, 291
Set, 293, 294
OpenGPS::PointValidityProvider, 295
~PointValidityProvider, 296
GetPointBuffer, 297
IsValid, 297
m_PointBuffer, 298
PointValidityProvider, 296
SetValid, 297
OpenGPS::PointVectorInputStringStream, 298
~PointVectorInputStringStream, 299
BaseType, 299
m_Locale, 299
PointVectorInputStringStream, 299
OpenGPS::PointVectorInvariantLocale, 300
~PointVectorInvariantLocale, 301
BaseType, 300
PointVectorInvariantLocale, 300
OpenGPS::PointVectorOutputStringStream, 301
~PointVectorOutputStringStream, 302
BaseType, 302
m_Locale, 302
PointVectorOutputStringStream, 302
OpenGPS::PointVectorParser, 302
~PointVectorParser, 304
CreateDataPointParser, 304
m_X, 306
m_Y, 306
m_Z, 306
PointVectorParser, 304
Read, 304
SetX, 305
SetY, 305
SetZ, 305
Write, 306
OpenGPS::PointVectorParserBuilder, 307
~PointVectorParserBuilder, 308
BuildParser, 308
BuildX, 308
BuildY, 309
BuildZ, 309
GetParser, 309
m_Parser, 310
PointVectorParserBuilder, 308
OpenGPS::PointVectorProxy, 310
~PointVectorProxy, 312
Get, 312
GetX, 312
GetY, 312, 313
GetZ, 313
m_Buffer, 313
m_Context, 313
m_X, 313
m_Y, 313
m_Z, 314
operator=, 313
PointVectorProxy, 312
Set, 313
OpenGPS::PointVectorProxy::DataPointProxy, 314
~DataPointProxy, 315
DataPointProxy, 315
Get, 316
GetPointType, 316
IsValid, 316
m_Buffer, 317
m_Context, 317
operator=, 316
Reset, 316
Set, 317
OpenGPS::PointVectorProxyContext, 318
~PointVectorProxyContext, 319
CanIncrementIndex, 319
GetIndex, 319
IncrementIndex, 319
IsMatrix, 319

PointVectorProxyContext, 319
OpenGPS::PointVectorProxyContextList, 320
~PointVectorProxyContextList, 321
CanIncrementIndex, 321
GetIndex, 322
IncrementIndex, 322
IsMatrix, 322
m_Index, 322
m_MaxIndex, 322
PointVectorProxyContextList, 321
SetIndex, 322
OpenGPS::PointVectorProxyContextMatrix, 323
~PointVectorProxyContextMatrix, 325
CanIncrementIndex, 325
GetIndex, 325
IncrementIndex, 325
IsMatrix, 325
m_MaxU, 326
m_MaxV, 326
m_MaxW, 326
m_U, 326
m_V, 326
m_W, 326
PointVectorProxyContextMatrix, 324
SetIndex, 325
OpenGPS::PointVectorReaderContext, 327
~PointVectorReaderContext, 328
IsValid, 328
MoveNext, 329
PointVectorReaderContext, 328
Read, 329, 330
Skip, 330
OpenGPS::PointVectorWhitespaceFacet, 331
~PointVectorWhitespaceFacet, 332
BaseType, 332
do_is, 332
PointVectorWhitespaceFacet, 332
OpenGPS::PointVectorWriterContext, 333
~PointVectorWriterContext, 334
IsGood, 334
MoveNext, 334
PointVectorWriterContext, 334
Skip, 335
Write, 335, 336
OpenGPS::Schemas, 26
OpenGPS::Schemas::ISO5436_2::AxisType
A, 85
I, 85
OpenGPS::Schemas::ISO5436_2::DataType
D, 146
F, 146
I, 146
L, 146
OpenGPS::Schemas::ISO5436_2::Type
Contacting, 403
NonContacting, 403
OpenGPS::Schemas::ISO5436_2, 27
ISO5436_2, 33–42
operator!=, 42–44
operator<<, 44–50
operator==, 50, 51
OpenGPS::Schemas::ISO5436_2::AxesType, 64
_clone, 69
AxesType, 68
CX, 69, 70
CX_, 73
CX_traits, 67
CX_type, 67
CY, 70
CY_, 73
CY_traits, 67
CY_type, 67
CZ, 71
CZ_, 73
CZ_traits, 67
CZ_type, 67
parse, 71
Rotation, 72
Rotation_, 73
Rotation_optional, 67
Rotation_traits, 68
Rotation_type, 68
OpenGPS::Schemas::ISO5436_-
2::AxisDescriptionType, 73
_clone, 79
AxisDescriptionType, 78, 79
AxisType, 79, 80
AxisType_, 83
AxisType_traits, 77
AxisType_type, 77
DataType, 80, 81
DataType_, 83
DataType_optional, 77
DataType_traits, 77
DataType_type, 77
Increment, 81, 82
Increment_, 83
Increment_optional, 77
Increment_traits, 77
Increment_type, 78
Offset, 82
Offset_, 83
Offset_optional, 78
Offset_traits, 78
Offset_type, 78
parse, 83
OpenGPS::Schemas::ISO5436_2::AxisType, 83
_clone, 87

_xsd_AxisType_convert, 87
_xsd_AxisType_indexes_, 88
_xsd_AxisType_literals_, 88
AxisType, 85, 86
operator value, 87
operator=, 87
value, 85
OpenGPS::Schemas::ISO5436_2::DataLinkType, 122
_clone, 128
DataLinkType, 127, 128
MD5ChecksumPointData, 128, 129
MD5ChecksumPointData_, 132
MD5ChecksumPointData_traits, 126
MD5ChecksumPointData_type, 126
MD5ChecksumValidPoints, 129, 130
MD5ChecksumValidPoints_, 132
MD5ChecksumValidPoints_optional, 126
MD5ChecksumValidPoints_traits, 126
MD5ChecksumValidPoints_type, 126
parse, 130
PointDataLink, 130, 131
PointDataLink_, 132
PointDataLink_traits, 126
PointDataLink_type, 127
ValidPointsLink, 131, 132
ValidPointsLink_, 133
ValidPointsLink_optional, 127
ValidPointsLink_traits, 127
ValidPointsLink_type, 127
OpenGPS::Schemas::ISO5436_2::DataListType, 133
_clone, 136
DataListType, 135, 136
Datum, 136, 137
Datum_, 137
Datum_const_iterator, 135
Datum_iterator, 135
Datum_sequence, 135
Datum_traits, 135
Datum_type, 135
parse, 137
OpenGPS::Schemas::ISO5436_2::DataType, 145
_clone, 148
_xsd_DataType_convert, 148
_xsd_DataType_indexes_, 149
_xsd_DataType_literals_, 149
DataType, 146, 147
operator value, 148
operator=, 148
value, 146
OpenGPS::Schemas::ISO5436_2::Datum, 149
_clone, 151
Datum, 150, 151
OpenGPS::Schemas::ISO5436_2::FeatureType, 179
_clone, 182
FeatureType, 180, 181
OpenGPS::Schemas::ISO5436_2::InstrumentType, 195
_clone, 200
InstrumentType, 199, 200
Manufacturer, 200, 201
Manufacturer_, 204
Manufacturer_traits, 198
Manufacturer_type, 198
Model, 201, 202
Model_, 204
Model_traits, 198
Model_type, 199
parse, 202
Serial, 202, 203
Serial_, 204
Serial_traits, 199
Serial_type, 199
Version, 203, 204
Version_, 204
Version_traits, 199
Version_type, 199
OpenGPS::Schemas::ISO5436_2::ISO5436_2Type, 264
_clone, 270
ISO5436_2Type, 269, 270
parse, 270
Record1, 271
Record1_, 276
Record1_traits, 268
Record1_type, 268
Record2, 271, 272
Record2_, 276
Record2_optional, 268
Record2_traits, 268
Record2_type, 268
Record3, 273
Record3_, 276
Record3_traits, 268
Record3_type, 268
Record4, 274
Record4_, 276
Record4_traits, 269
Record4_type, 269
VendorSpecificID, 274, 275
VendorSpecificID_, 276
VendorSpecificID_optional, 269
VendorSpecificID_traits, 269
VendorSpecificID_type, 269
OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType, 276

_clone, 280
MatrixDimensionType, 280
parse, 281
SizeX, 281
SizeX_, 283
SizeX_traits, 279
SizeX_type, 279
SizeY, 281, 282
SizeY_, 283
SizeY_traits, 279
SizeY_type, 279
SizeZ, 282
SizeZ_, 283
SizeZ_traits, 279
SizeZ_type, 279
OpenGPS::Schemas::ISO5436_-2::ProbingSystemType, 336
_clone, 340
Identification, 340
Identification_, 342
Identification_traits, 338
Identification_type, 338
parse, 341
ProbingSystemType, 339
Type, 341
Type_, 342
Type_traits, 338
Type_type, 339
OpenGPS::Schemas::ISO5436_2::Record1Type, 342
_clone, 346
Axes, 347
Axes_, 350
Axes_traits, 345
Axes_type, 345
FeatureType, 348
FeatureType_, 350
FeatureType_traits, 345
FeatureType_type, 345
parse, 348
Record1Type, 346
Revision, 349
Revision_, 350
Revision_traits, 345
Revision_type, 345
OpenGPS::Schemas::ISO5436_2::Record2Type, 350
_clone, 357
CalibrationDate, 358
CalibrationDate_, 364
CalibrationDate_traits, 355
CalibrationDate_type, 355
Comment, 359, 360
Comment_, 364
Comment_optional, 355
Comment_traits, 355
Comment_type, 355
Creator, 360, 361
Creator_, 364
Creator_optional, 355
Creator_traits, 355
Creator_type, 355
Date, 361, 362
Date_, 364
Date_traits, 356
Date_type, 356
Instrument, 362, 363
Instrument_, 364
Instrument_traits, 356
Instrument_type, 356
parse, 363
ProbingSystem, 363, 364
ProbingSystem_, 364
ProbingSystem_traits, 356
ProbingSystem_type, 356
Record2Type, 357
OpenGPS::Schemas::ISO5436_2::Record3Type, 365
_clone, 371
DataLink, 371, 372
DataLink_, 376
DataLink_optional, 368
DataLink_traits, 368
DataLink_type, 369
DataList, 372, 373
DataList_, 376
DataList_optional, 369
DataList_traits, 369
DataList_type, 369
ListDimension, 373, 374
ListDimension_, 376
ListDimension_optional, 369
ListDimension_traits, 369
ListDimension_type, 369
MatrixDimension, 374, 375
MatrixDimension_, 376
MatrixDimension_optional, 370
MatrixDimension_traits, 370
MatrixDimension_type, 370
parse, 376
Record3Type, 370
OpenGPS::Schemas::ISO5436_2::Record4Type, 376
_clone, 379
ChecksumFile, 379, 380
ChecksumFile_, 381
ChecksumFile_traits, 378
ChecksumFile_type, 378

parse, 380
Record4Type, 378, 379
OpenGPS::Schemas::ISO5436_-2::RotationMatrixElementType, 381
_clone, 383
RotationMatrixElementType, 382, 383
OpenGPS::Schemas::ISO5436_2::RotationType, 384
_clone, 393
parse, 393
r11, 393, 394
r11_, 401
r11_traits, 389
r11_type, 389
r12, 394, 395
r12_, 401
r12_traits, 390
r12_type, 390
r13, 395
r13_, 401
r13_traits, 390
r13_type, 390
r21, 396
r21_, 401
r21_traits, 390
r21_type, 390
r22, 396, 397
r22_, 401
r22_traits, 390
r22_type, 391
r23, 397, 398
r23_, 401
r23_traits, 391
r23_type, 391
r31, 398, 399
r31_, 401
r31_traits, 391
r31_type, 391
r32, 399
r32_, 401
r32_traits, 391
r32_type, 391
r33, 400
r33_, 401
r33_traits, 392
r33_type, 392
RotationType, 392
OpenGPS::Schemas::ISO5436_2::Type, 402
_clone, 405
_xsd_Type_convert, 405
_xsd_Type_indexes_, 406
_xsd_Type_literals_, 406
operator value, 405
operator=, 405
Type, 403, 404
value, 403
OpenGPS::ValidBuffer, 406
~ValidBuffer, 408
Allocate, 408
AllocateRaw, 408
GetPointBuffer, 409
IsAllocated, 409
IsValid, 409
m_RawSize, 410
m_ValidityBuffer, 410
Read, 409
Reset, 410
SetValid, 410
ValidBuffer, 408
Write, 410
OpenGPS::VectorBuffer, 411
~VectorBuffer, 413
GetPointVectorProxy, 413
GetValidityBuffer, 414
GetValidityProvider, 414
GetX, 415
GetY, 415
GetZ, 415
HasValidityBuffer, 415
m_ValidBuffer, 417
m_ValidityProvider, 417
m_X, 417
m_Y, 417
m_Z, 417
SetValidityProvider, 416
SetX, 416
SetY, 416
SetZ, 417
VectorBuffer, 413
OpenGPS::VectorBufferBuilder, 418
~VectorBufferBuilder, 419
BuildBuffer, 420
BuildValidityProvider, 420
BuildX, 420
BuildY, 420
BuildZ, 421
CreatePointBuffer, 421
GetBuffer, 421
m_Buffer, 422
VectorBufferBuilder, 419
OpenGPS::XmlPointVectorReaderContext, 422
~XmlPointVectorReaderContext, 424
IsGood, 425
IsValid, 425
m_Next, 428
m_PointVectorList, 428
m_Stream, 428
MoveNext, 425

Read, 426, 427
Reset, 427
Set, 427
Skip, 427
StringList, 424
XmlPointVectorReaderContext, 424
OpenGPS::XmlPointVectorWriterContext, 428
~XmlPointVectorWriterContext, 431
AppendSeparator, 431
Get, 431
IsGood, 431
m_NeedsSeparator, 434
m_PointVectorList, 434
m_Stream, 434
MoveNext, 432
Reset, 432
Skip, 432
StringList, 431
Write, 433
XmlPointVectorWriterContext, 431
OpenGPS::ZipOutputStream, 434
~ZipOutputStream, 435
BaseType, 435
write, 436
ZipOutputStream, 435
OpenGPS::ZipStreamBuffer, 436
~ZipStreamBuffer, 437
BaseType, 437
GetMd5, 438
m_Handle, 438
m_Md5Context, 438
xsputn, 438
ZipStreamBuffer, 437
operator value
OpenGPS::Schemas::ISO5436_2::AxisType, 87
OpenGPS::Schemas::ISO5436_2::DataType, 148
OpenGPS::Schemas::ISO5436_2::Type, 405
operator!=
OpenGPS::Schemas::ISO5436_2, 42–44
operator<<
OpenGPS::Schemas::ISO5436_2, 44–50
operator=
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 262
OpenGPS::PointVectorProxy, 313
OpenGPS::PointVectorProxy::DataPointProxy, 316
OpenGPS::Schemas::ISO5436_2::AxisType, 87
OpenGPS::Schemas::ISO5436_2::DataType, 148
OpenGPS::Schemas::ISO5436_2::Type, 405
operator==
OpenGPS::Schemas::ISO5436_2, 50, 51
OutputBinaryFileStream
OpenGPS::OutputBinaryFileStream, 288

P
md5.c, 509
parse
OpenGPS::Schemas::ISO5436_2::AxesType, 71
OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType, 83
OpenGPS::Schemas::ISO5436_-2::DataLinkType, 130
OpenGPS::Schemas::ISO5436_-2::DataListType, 137
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 202
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 270
OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType, 281
OpenGPS::Schemas::ISO5436_-2::ProbingSystemType, 341
OpenGPS::Schemas::ISO5436_-2::Record1Type, 348
OpenGPS::Schemas::ISO5436_-2::Record2Type, 363
OpenGPS::Schemas::ISO5436_-2::Record3Type, 376
OpenGPS::Schemas::ISO5436_-2::Record4Type, 380
OpenGPS::Schemas::ISO5436_-2::RotationType, 393
parsing
xml_schema, 58
PathExists
OpenGPS::Environment, 174
performanceDouble
ISO5436_2_XML_Demo.cxx, 487
Kopie von ISO5436_2_XML_Demo_Test.cxx, 507
performanceInt16
ISO5436_2_XML_Demo.cxx, 487
Kopie von ISO5436_2_XML_Demo_Test.cxx, 507
point_buffer.cxx, 522
point_buffer.hxx, 522
point_iterator.cxx, 523
point_iterator_c.cxx, 523
ogps_FreePointIterator, 524
ogps_GetCurrentPoint, 524
ogps_GetListPosition, 525
ogps_GetMatrixPosition, 525

ogps_HasNextPoint, 525
ogps_HasPrevPoint, 525
ogps_MoveNextPoint, 525
ogps_MovePrevPoint, 525
ogps_ResetNextPointIterator, 525
ogps_ResetPrevPointIterator, 525
ogps_SetCurrentPoint, 525
point_iterator_c.hxx, 526
OGPS_PointIterator, 527
OGPS_PointIteratorPtr, 527
point_validity_provider.cxx, 527
point_validity_provider.hxx, 527
point_vector.cxx, 528
point_vector_c.cxx, 528
ogps_CreatePointVector, 530
ogps_FreePointVector, 530
ogps_GetDoubleX, 530
ogps_GetDoubleY, 530
ogps_GetDoubleZ, 530
ogps_GetFloatX, 530
ogps_GetFloatY, 531
ogps_GetFloatZ, 531
ogps_GetInt16X, 531
ogps_GetInt16Y, 531
ogps_GetInt16Z, 531
ogps_GetInt32X, 531
ogps_GetInt32Y, 531
ogps_GetInt32Z, 531
ogps_GetPointTypeX, 531
ogps_GetPointTypeY, 531
ogps_GetPointTypeZ, 532
ogps_GetX, 532
ogps_GetXYZ, 532
ogps_GetY, 532
ogps_GetZ, 532
ogps_IsValidPoint, 532
ogps_SetDoubleX, 532
ogps_SetDoubleY, 532
ogps_SetDoubleZ, 532
ogps_SetFloatX, 532
ogps_SetFloatY, 533
ogps_SetFloatZ, 533
ogps_SetInt16X, 533
ogps_SetInt16Y, 533
ogps_SetInt16Z, 533
ogps_SetInt32X, 533
ogps_SetInt32Y, 533
ogps_SetInt32Z, 533
point_vector_c.hxx, 534
OGPS_PointVector, 534
OGPS_PointVectorPtr, 534
point_vector_iostream.cxx, 535
point_vector_iostream.hxx, 535
point_vector_parser.cxx, 536
point_vector_parser.hxx, 537
point_vector_parser_builder.cxx, 537
point_vector_parser_builder.hxx, 538
point_vector_proxy.cxx, 539
point_vector_proxy.hxx, 539
point_vector_proxy_context.cxx, 540
point_vector_proxy_context.hxx, 540
point_vector_proxy_context_list.cxx, 541
point_vector_proxy_context_list.hxx, 541
point_vector_proxy_context_matrix.cxx, 542
point_vector_proxy_context_matrix.hxx, 543
point_vector_reader_context.hxx, 544
point_vector_writer_context.hxx, 544
PointBuffer
OpenGPS::PointBuffer, 291
PointDataLink
OpenGPS::Schemas::ISO5436_-2::DataLinkType, 130, 131
PointDataLink_
OpenGPS::Schemas::ISO5436_-2::DataLinkType, 132
PointDataLink_traits
OpenGPS::Schemas::ISO5436_-2::DataLinkType, 126
PointDataLink_type
OpenGPS::Schemas::ISO5436_-2::DataLinkType, 127
PointIteratorAutoPtr
OpenGPS, 26
PointIteratorImpl
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 261
PointValidityProvider
OpenGPS::PointValidityProvider, 296
PointVectorAutoPtr
OpenGPS, 26
PointVectorInputStringStream
OpenGPS::PointVectorInputStringStream, 299
PointVectorInvariantLocale
OpenGPS::PointVectorInvariantLocale, 300
PointVectorOutputStringStream
OpenGPS::PointVectorOutputStringStream, 302
PointVectorParser
OpenGPS::PointVectorParser, 304
PointVectorParserBuilder
OpenGPS::PointVectorParserBuilder, 308
PointVectorParserBuilderAutoPtr
OpenGPS, 26
PointVectorProxy
OpenGPS::PointVectorProxy, 312
PointVectorProxyContext
OpenGPS::PointVectorProxyContext, 319
PointVectorProxyContextAutoPtr

OpenGPS, 26
PointVectorProxyContextList
OpenGPS::PointVectorProxyContextList, 321
PointVectorProxyContextMatrix
OpenGPS::PointVectorProxyContextMatrix, 324
PointVectorReaderContext
OpenGPS::PointVectorReaderContext, 328
PointVectorReaderContextAutoPtr
iso5436_2_container.cxx, 483
PointVectorWhitespaceFacet
OpenGPS::PointVectorWhitespaceFacet, 332
PointVectorWriterContext
OpenGPS::PointVectorWriterContext, 334
PointVectorWriterContextAutoPtr
iso5436_2_container.cxx, 483
positive_integer
xml_schema, 58
ProbingSystem
OpenGPS::Schemas::ISO5436_-2::Record2Type, 363, 364
ProbingSystem_
OpenGPS::Schemas::ISO5436_-2::Record2Type, 364
ProbingSystem_traits
OpenGPS::Schemas::ISO5436_-2::Record2Type, 356
ProbingSystem_type
OpenGPS::Schemas::ISO5436_-2::Record2Type, 356
ProbingSystemType
OpenGPS::Schemas::ISO5436_-2::ProbingSystemType, 339
properties
xml_schema, 58
PUT_ULONG_LE
md5.c, 509

qname
xml_schema, 58

r11
OpenGPS::Schemas::ISO5436_-2::RotationType, 393, 394
r11_
OpenGPS::Schemas::ISO5436_-2::RotationType, 401
r11_traits
OpenGPS::Schemas::ISO5436_-2::RotationType, 389
r11_type
OpenGPS::Schemas::ISO5436_-2::RotationType, 389
r12
OpenGPS::Schemas::ISO5436_-2::RotationType, 394, 395
r12_
OpenGPS::Schemas::ISO5436_-2::RotationType, 401
r12_traits
OpenGPS::Schemas::ISO5436_-2::RotationType, 390
r12_type
OpenGPS::Schemas::ISO5436_-2::RotationType, 390
r13
OpenGPS::Schemas::ISO5436_-2::RotationType, 395
r13_
OpenGPS::Schemas::ISO5436_-2::RotationType, 401
r13_traits
OpenGPS::Schemas::ISO5436_-2::RotationType, 390
r13_type
OpenGPS::Schemas::ISO5436_-2::RotationType, 390
r21
OpenGPS::Schemas::ISO5436_-2::RotationType, 396
r21_
OpenGPS::Schemas::ISO5436_-2::RotationType, 401
r21_traits
OpenGPS::Schemas::ISO5436_-2::RotationType, 390
r21_type
OpenGPS::Schemas::ISO5436_-2::RotationType, 390
r22
OpenGPS::Schemas::ISO5436_-2::RotationType, 396, 397
r22_
OpenGPS::Schemas::ISO5436_-2::RotationType, 401
r22_traits
OpenGPS::Schemas::ISO5436_-2::RotationType, 390
r22_type
OpenGPS::Schemas::ISO5436_-2::RotationType, 391
r23
OpenGPS::Schemas::ISO5436_-2::RotationType, 397, 398
r23_
OpenGPS::Schemas::ISO5436_-2::RotationType, 401
r23_traits

OpenGPS::Schemas::ISO5436_-2::RotationType, 391
r23_type
OpenGPS::Schemas::ISO5436_-2::RotationType, 391
r31
OpenGPS::Schemas::ISO5436_-2::RotationType, 398, 399
r31_
OpenGPS::Schemas::ISO5436_-2::RotationType, 401
r31_traits
OpenGPS::Schemas::ISO5436_-2::RotationType, 391
r31_type
OpenGPS::Schemas::ISO5436_-2::RotationType, 391
r32
OpenGPS::Schemas::ISO5436_-2::RotationType, 399
r32_
OpenGPS::Schemas::ISO5436_-2::RotationType, 401
r32_traits
OpenGPS::Schemas::ISO5436_-2::RotationType, 391
r32_type
OpenGPS::Schemas::ISO5436_-2::RotationType, 391
r33
OpenGPS::Schemas::ISO5436_-2::RotationType, 400
r33_
OpenGPS::Schemas::ISO5436_-2::RotationType, 401
r33_traits
OpenGPS::Schemas::ISO5436_-2::RotationType, 392
r33_type
OpenGPS::Schemas::ISO5436_-2::RotationType, 392
Read
OpenGPS::BinaryLSBPointVectorReaderContext, 92, 93
OpenGPS::BinaryMSBPointVectorReaderContext, 103, 104
OpenGPS::BinaryPointVectorReaderContext, 114, 115
OpenGPS::DataPointParser, 144
OpenGPS::DoubleDataPointParser, 153
OpenGPS::FloatDataPointParser, 184
OpenGPS::Int16DataPointParser, 206
OpenGPS::Int16ValidBuffer, 216
OpenGPS::Int32DataPointParser, 219
OpenGPS::Int32ValidBuffer, 229
OpenGPS::MissingDataPointParser, 286
OpenGPS::PointVectorParser, 304
OpenGPS::PointVectorReaderContext, 329, 330
OpenGPS::ValidBuffer, 409
OpenGPS::XmlPointVectorReaderContext, 426, 427
ReadDocument
OpenGPS::ISO5436_2Container, 252
ReadMd5FromFile
OpenGPS::ISO5436_2Container, 252
readonlyExample
ISO5436_2_XML_Demo.cxx, 487
Kopie von ISO5436_2_XML_Demo_Test.cxx, 507
readonlyExample2
ISO5436_2_XML_Demo.cxx, 487
Kopie von ISO5436_2_XML_Demo_Test.cxx, 507
readonlyExample3
ISO5436_2_XML_Demo.cxx, 487
Kopie von ISO5436_2_XML_Demo_Test.cxx, 507
readonlyExample4
ISO5436_2_XML_Demo.cxx, 487
Kopie von ISO5436_2_XML_Demo_Test.cxx, 507
readonlyExampleMatrix
ISO5436_2_XML_Demo.cxx, 487
Kopie von ISO5436_2_XML_Demo_Test.cxx, 507
ReadXmlDocument
OpenGPS::ISO5436_2Container, 253
Record1
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 271
Record1_
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 276
Record1_traits
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 268
Record1_type
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 268
Record1Type
OpenGPS::Schemas::ISO5436_-2::Record1Type, 346
Record2
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 271, 272
Record2_

OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 276
Record2_optional
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 268
Record2_traits
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 268
Record2_type
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 268
Record2Type
OpenGPS::Schemas::ISO5436_-2::Record2Type, 357
Record3
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 273
Record3_
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 276
Record3_traits
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 268
Record3_type
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 268
Record3Type
OpenGPS::Schemas::ISO5436_-2::Record3Type, 370
Record4
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 274
Record4_
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 276
Record4_traits
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 269
Record4_type
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 269
Record4Type
OpenGPS::Schemas::ISO5436_-2::Record4Type, 378, 379
RemoveDir
OpenGPS::Environment, 174
RemoveFile
OpenGPS::Environment, 174
RemoveTempDir
OpenGPS::ISO5436_2Container, 253
RenameFile
OpenGPS::Environment, 175
Reset
OpenGPS::DataPointImpl, 140
OpenGPS::Environment, 175
OpenGPS::ExceptionHistory, 178
OpenGPS::Int16ValidBuffer, 217
OpenGPS::Int32ValidBuffer, 230
OpenGPS::ISO5436_2Container, 253
OpenGPS::PointVectorProxy::DataPointProxy, 316
OpenGPS::ValidBuffer, 410
OpenGPS::XmlPointVectorReaderContext, 427
OpenGPS::XmlPointVectorWriterContext, 432
ResetNext
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 262
ResetPrev
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 262
ResetValidPointsLink
OpenGPS::ISO5436_2Container, 253
ResetXmlPointList
OpenGPS::ISO5436_2Container, 253
Revision
OpenGPS::Schemas::ISO5436_-2::Record1Type, 349
Revision_
OpenGPS::Schemas::ISO5436_-2::Record1Type, 350
Revision_traits
OpenGPS::Schemas::ISO5436_-2::Record1Type, 345
Revision_type
OpenGPS::Schemas::ISO5436_-2::Record1Type, 345
Rotation
OpenGPS::Schemas::ISO5436_2::AxesType, 72
Rotation_
OpenGPS::Schemas::ISO5436_2::AxesType, 73
Rotation_optional
OpenGPS::Schemas::ISO5436_2::AxesType, 67
Rotation_traits
OpenGPS::Schemas::ISO5436_2::AxesType, 68
Rotation_type
OpenGPS::Schemas::ISO5436_2::AxesType, 68
RotationMatrixElementType
OpenGPS::Schemas::ISO5436_-2::RotationMatrixElementType, 382, 383
RotationType

OpenGPS::Schemas::ISO5436_-2::RotationType, 392

S
md5.c, 510
SafeMultipilcation
OpenGPS::ISO5436_2Container, 253
SaveChecksumFile
OpenGPS::ISO5436_2Container, 254
SavePointBuffer
OpenGPS::ISO5436_2Container, 254
SaveValidPointsLink
OpenGPS::ISO5436_2Container, 254
SaveXmlDocument
OpenGPS::ISO5436_2Container, 255
Serial
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 202, 203
Serial_
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 204
Serial_traits
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 199
Serial_type
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 199
serialization
xml_schema, 58
Set
OpenGPS::DataPointImpl, 140
OpenGPS::DoublePointBuffer, 162, 163
OpenGPS::FloatPointBuffer, 192, 193
OpenGPS::Int16PointBuffer, 211, 212
OpenGPS::Int32PointBuffer, 224, 225
OpenGPS::PointBuffer, 293, 294
OpenGPS::PointVectorProxy, 313
OpenGPS::PointVectorProxy::DataPointProxy, 317
OpenGPS::XmlPointVectorReaderContext, 427
SetCurrent
OpenGPS::ISO5436_-2Container::PointIteratorImpl, 262
SetIndex
OpenGPS::PointVectorProxyContextList, 322
OpenGPS::PointVectorProxyContextMatrix, 325
SetLastException
OpenGPS::ExceptionHistory, 178
SetListPoint
OpenGPS::ISO5436_2Container, 255
SetMatrixPoint
OpenGPS::ISO5436_2Container, 255
SetValid
OpenGPS::DoubleInlineValidity, 156
OpenGPS::FloatInlineValidity, 187
OpenGPS::Int16ValidBuffer, 217
OpenGPS::Int32ValidBuffer, 230
OpenGPS::PointValidityProvider, 297
OpenGPS::ValidBuffer, 410
SetValidityProvider
OpenGPS::VectorBuffer, 416
SetX
OpenGPS::PointVectorParser, 305
OpenGPS::VectorBuffer, 416
SetY
OpenGPS::PointVectorParser, 305
OpenGPS::VectorBuffer, 416
SetZ
OpenGPS::PointVectorParser, 305
OpenGPS::VectorBuffer, 417
severity
xml_schema, 58
short_
xml_schema, 58
simple_type
xml_schema, 58
simpleExample
ISO5436_2_XML_Demo.cxx, 487
Kopie von ISO5436_2_XML_Demo_Test.cxx, 507
SizeX
OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType, 281
SizeX_
OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType, 283
SizeX_traits
OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType, 279
SizeX_type
OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType, 279
SizeY
OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType, 281, 282
SizeY_
OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType, 283
SizeY_traits
OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType, 279
SizeY_type
OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType, 279
SizeZ

OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType, 282
SizeZ_
OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType, 283
SizeZ_traits
OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType, 279
SizeZ_type
OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType, 279
Skip
OpenGPS::BinaryLSBPointVectorReaderContext, 93
OpenGPS::BinaryLSBPointVectorWriterContext, 97
OpenGPS::BinaryMSBPointVectorReaderContext, 104
OpenGPS::BinaryMSBPointVectorWriterContext, 108
OpenGPS::BinaryPointVectorReaderContext, 115
OpenGPS::BinaryPointVectorWriterContext, 120
OpenGPS::PointVectorReaderContext, 330
OpenGPS::PointVectorWriterContext, 335
OpenGPS::XmlPointVectorReaderContext, 427
OpenGPS::XmlPointVectorWriterContext, 432
state
md5_context, 284
std, 51
stdafx.hxx, 545
_ASSERT, 547
_OPENGPS_BINFORMAT_DOUBLE_SIZE, 547
_OPENGPS_BINFORMAT_FLOAT_SIZE, 547
_OPENGPS_BINFORMAT_INT16_SIZE, 547
_OPENGPS_BINFORMAT_INT32_SIZE, 547
_OPENGPS_DELETE, 547
_OPENGPS_DELETE_ARRAY, 547
_OPENGPS_DESCRIPTION, 547
_OPENGPS_ENV_OPENGPS_LOCATION, 547
_OPENGPS_EXCEPTION_MESG, 548
_OPENGPS_EXPERIMENTAL, 548
_OPENGPS_FREE, 548
_OPENGPS_ID, 548
_OPENGPS_ISO5436_LOCATION, 548
_OPENGPS_MSTR, 548
_OPENGPS_NAME, 548
_OPENGPS_VERSIONSTRING, 548
_OPENGPS_VERSIONSTRING_M, 548
_OPENGPS_XSD_ISO5436_DATALINK_-PATH, 549
_OPENGPS_XSD_ISO5436_LOCATION, 549
_OPENGPS_XSD_ISO5436_MAIN_-CHECKSUM_PATH, 549
_OPENGPS_XSD_ISO5436_MAIN_PATH, 549
_OPENGPS_XSD_ISO5436_NAMESPACE, 549
_OPENGPS_XSD_ISO5436_-VALIDPOINTSLINK_PATH, 549
_VERIFY, 549
string
xml_schema, 59
string.cxx, 549
StringList
OpenGPS::ISO5436_2Container, 239
OpenGPS::XmlPointVectorReaderContext, 424
OpenGPS::XmlPointVectorWriterContext, 431

TestChecksums
OpenGPS::ISO5436_2Container, 255
time
xml_schema, 59
token
xml_schema, 59
total
md5_context, 284
tree_node_key
xml_schema, 60
Type
OpenGPS::Schemas::ISO5436_-2::ProbingSystemType, 341
OpenGPS::Schemas::ISO5436_2::Type, 403, 404
type
xml_schema, 59
Type_
OpenGPS::Schemas::ISO5436_-2::ProbingSystemType, 342
Type_traits
OpenGPS::Schemas::ISO5436_-2::ProbingSystemType, 338
Type_type
OpenGPS::Schemas::ISO5436_-2::ProbingSystemType, 339

unexpected_element

xml_schema, 59
unexpected_enumerator
xml_schema, 59
unsigned_byte
xml_schema, 59
unsigned_int
xml_schema, 59
unsigned_long
xml_schema, 59
unsigned_short
xml_schema, 59
uri
xml_schema, 59

valid_buffer.cxx, 550
valid_buffer.hxx, 550
ValidBuffer
OpenGPS::ValidBuffer, 408
ValidPointsLink
OpenGPS::Schemas::ISO5436_-2::DataLinkType, 131, 132
ValidPointsLink_
OpenGPS::Schemas::ISO5436_-2::DataLinkType, 133
ValidPointsLink_optional
OpenGPS::Schemas::ISO5436_-2::DataLinkType, 127
ValidPointsLink_traits
OpenGPS::Schemas::ISO5436_-2::DataLinkType, 127
ValidPointsLink_type
OpenGPS::Schemas::ISO5436_-2::DataLinkType, 127
value
OpenGPS::Schemas::ISO5436_2::AxisType, 85
OpenGPS::Schemas::ISO5436_2::DataType, 146
OpenGPS::Schemas::ISO5436_2::Type, 403
vector_buffer.cxx, 551
vector_buffer.hxx, 552
vector_buffer_builder.cxx, 553
vector_buffer_builder.hxx, 553
VectorBuffer
OpenGPS::VectorBuffer, 413
VectorBufferAutoPtr
OpenGPS, 26
VectorBufferBuilder
OpenGPS::VectorBufferBuilder, 419
VectorBufferBuilderAutoPtr
OpenGPS, 26
VendorSpecificID
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 274, 275
VendorSpecificID_
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 276
VendorSpecificID_optional
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 269
VendorSpecificID_traits
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 269
VendorSpecificID_type
OpenGPS::Schemas::ISO5436_2::ISO5436_-2Type, 269
VerifyChecksum
OpenGPS::ISO5436_2Container, 255
VerifyDataBinChecksum
OpenGPS::ISO5436_2Container, 256
VerifyMainChecksum
OpenGPS::ISO5436_2Container, 256
VerifyValidBinChecksum
OpenGPS::ISO5436_2Container, 256
Version
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 203, 204
Version.h, 554
_OPENGPS_BUILD, 555
_OPENGPS_BUILD_DATE, 555
_OPENGPS_CLEAN_BUILD, 555
_OPENGPS_CLEAN_BUILD_FLAG, 555
_OPENGPS_CLEAN_REVISION, 556
_OPENGPS_CLEAN_REVISION_FLAG, 556
_OPENGPS_MINVERSION, 556
_OPENGPS_REPOSITORY_URL, 556
_OPENGPS_REVISION, 556
_OPENGPS_REVISION_DATE, 556
_OPENGPS_VERSION, 556
Version_
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 204
Version_traits
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 199
Version_type
OpenGPS::Schemas::ISO5436_-2::InstrumentType, 199

win32_environment.cxx, 557
win32_environment.hxx, 557
Write
OpenGPS::BinaryLSBPointVectorWriterContext, 98, 99
OpenGPS::BinaryMSBPointVectorWriterContext, 109, 110

OpenGPS::BinaryPointVectorWriterContext, 120, 121
OpenGPS::DataPointParser, 144
OpenGPS::DoubleDataPointParser, 153
OpenGPS::FloatDataPointParser, 184
OpenGPS::Int16DataPointParser, 206
OpenGPS::Int16ValidBuffer, 217
OpenGPS::Int32DataPointParser, 219
OpenGPS::Int32ValidBuffer, 230
OpenGPS::ISO5436_2Container, 256
OpenGPS::MissingDataPointParser, 286
OpenGPS::PointVectorParser, 306
OpenGPS::PointVectorWriterContext, 335, 336
OpenGPS::ValidBuffer, 410
OpenGPS::XmlPointVectorWriterContext, 433
write
OpenGPS::ZipOutputStream, 436
WriteVendorSpecific
OpenGPS::ISO5436_2Container, 256

x
_OGPS_POINT_VECTOR, 63
xml_point_vector_reader_context.cxx, 557
_CHECK_ISGOOD_AND_THROW_-EXCEPTION, 557
_CHECK_STREAM_AND_THROW_-EXCEPTION, 558
xml_point_vector_reader_context.hxx, 558
xml_point_vector_writer_context.cxx, 559
_CHECK_ISGOOD_AND_THROW_-EXCEPTION, 560
_CHECK_STREAM_AND_THROW_-EXCEPTION, 560
xml_point_vector_writer_context.hxx, 560
xml_schema, 52
base64_binary, 53
boolean, 53
bounds, 54
buffer, 54
byte, 54
date, 54
date_time, 54
day, 54
decimal, 54
diagnostics, 54
double_, 54
duplicate_id, 54
duration, 54
entities, 55
entity, 55
error, 55
error_handler, 55
exception, 55
expected_attribute, 55
expected_element, 55
expected_text_content, 55
flags, 55
float_, 55
hex_binary, 56
id, 56
idref, 56
idrefs, 56
int_, 56
integer, 56
language, 56
long_, 56
month, 56
month_day, 56
name, 56
namespace_info, 57
namespace_infomap, 57
ncname, 57
negative_integer, 57
nmtoken, 57
nmtokens, 57
no_namespace_mapping, 57
no_prefix_mapping, 57
no_type_info, 57
non_negative_integer, 57
non_positive_integer, 58
normalized_string, 58
not_derived, 58
parsing, 58
positive_integer, 58
properties, 58
qname, 58
serialization, 58
severity, 58
short_, 58
simple_type, 58
string, 59
time, 59
token, 59
tree_node_key, 60
type, 59
unexpected_element, 59
unexpected_enumerator, 59
unsigned_byte, 59
unsigned_int, 59
unsigned_long, 59
unsigned_short, 59
uri, 59
xsi_already_in_use, 60
year, 60
year_month, 60
XmlPointVectorReaderContext

OpenGPS::XmlPointVectorReaderContext, 424
XmlPointVectorWriterContext
OpenGPS::XmlPointVectorWriterContext, 431
xsd_Licence_Header.c, 561
xsi_already_in_use
xml_schema, 60
xsputn
OpenGPS::ZipStreamBuffer, 438
XYSSL_MD5_C
md5.c, 510

y
_OGPS_POINT_VECTOR, 63
year
xml_schema, 60
year_month
xml_schema, 60

z
_OGPS_POINT_VECTOR, 63
zip_stream_buffer.cxx, 561
zip_stream_buffer.hxx, 562
ZipOutputStream
OpenGPS::ZipOutputStream, 435
ZipStreamBuffer
OpenGPS::ZipStreamBuffer, 437