

openGPS ISO 5436-2 XML Reference Manual  
0.01

Generated by Doxygen 1.5.4

Tue Apr 15 14:50:44 2008

## Contents

<b>1</b>	<b>openGPS ISO 5436-2 XML Library Documentation</b>	<b>1</b>
<b>2</b>	<b>openGPS ISO 5436-2 XML Directory Hierarchy</b>	<b>2</b>
<b>3</b>	<b>openGPS ISO 5436-2 XML Namespace Index</b>	<b>2</b>
<b>4</b>	<b>openGPS ISO 5436-2 XML Hierarchical Index</b>	<b>2</b>
<b>5</b>	<b>openGPS ISO 5436-2 XML Class Index</b>	<b>6</b>
<b>6</b>	<b>openGPS ISO 5436-2 XML File Index</b>	<b>11</b>
<b>7</b>	<b>openGPS ISO 5436-2 XML Directory Documentation</b>	<b>18</b>
<b>8</b>	<b>openGPS ISO 5436-2 XML Namespace Documentation</b>	<b>26</b>
<b>9</b>	<b>openGPS ISO 5436-2 XML Class Documentation</b>	<b>66</b>
<b>10</b>	<b>openGPS ISO 5436-2 XML File Documentation</b>	<b>592</b>

## 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](http://www.opengps.eu).

## 2 openGPS ISO 5436-2 XML Directory Hierarchy

### 2.1 openGPS ISO 5436-2 XML Directories

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

<b>ISO5436_2_XML</b>	<b>24</b>
<b>c</b>	<b>18</b>
<b>cxx</b>	<b>20</b>
<b>xyssl</b>	<b>26</b>
<b>ISO5436_2_XML_Demo</b>	<b>24</b>
<b>opengps</b>	<b>25</b>
<b>cxx</b>	<b>19</b>

## 3 openGPS ISO 5436-2 XML Namespace Index

### 3.1 openGPS ISO 5436-2 XML Namespace List

Here is a list of all namespaces with brief descriptions:

<b>OpenGPS</b> (The standard namespace for the openGPS software library )	<b>26</b>
<b>OpenGPS::Schemas</b> (Holds the C++ representations of the structure of all supported XML documents )	<b>32</b>
<b>OpenGPS::Schemas::ISO5436_2</b> (C++ namespace for the http://www.opengps.eu/2008/ISO5436_2 schema namespace )	<b>33</b>
<b>std</b> (The Standard namespace for the C++ library )	<b>55</b>
<b>xml_schema</b>	<b>60</b>

## 4 openGPS ISO 5436-2 XML Hierarchical Index

### 4.1 openGPS ISO 5436-2 XML Class Hierarchy

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

_OGPS_DATA_POINT	66
_OGPS_ISO5436_2_HANDLE	67
_OGPS_POINT_ITERATOR	68
_OGPS_POINT_VECTOR	69
OpenGPS::Schemas::ISO5436_2::AxesType	70
OpenGPS::Schemas::ISO5436_2::AxisDescriptionType	88
OpenGPS::Schemas::ISO5436_2::AxisType	107
std::basic_string< char >	
std::string	
OpenGPS::String	549
OpenGPS::Schemas::ISO5436_2::DataLinkType	135
OpenGPS::Schemas::ISO5436_2::DataListType	156
OpenGPS::DataPoint	164
OpenGPS::DataPointImpl	171
OpenGPS::PointVectorProxy::DataPointProxy	407
OpenGPS::DataPointImpl::_OGPS_DATA_POINT_- VALUE	178
OpenGPS::DataPointParser	180
OpenGPS::DoubleDataPointParser	195
OpenGPS::FloatDataPointParser	224
OpenGPS::Int16DataPointParser	253
OpenGPS::Int32DataPointParser	259
OpenGPS::MissingDataPointParser	357
OpenGPS::Schemas::ISO5436_2::DataType	182
OpenGPS::Schemas::ISO5436_2::Datum	190
OpenGPS::Environment	202
std::exception	
OpenGPS::Exception	213
OpenGPS::ExceptionHistory	216

<b>OpenGPS::Schemas::ISO5436_2::FeatureType</b>	<b>220</b>
<b>OpenGPS::Info</b>	<b>231</b>
<b>OpenGPS::Schemas::ISO5436_2::InstrumentType</b>	<b>236</b>
<b>std::ios_base</b>	
<b>std::basic_ios</b>	
<b>std::basic_istream</b>	
<b>std::basic_ifstream</b>	
<b>OpenGPS::InputBinaryFileStream</b>	<b>234</b>
<b>std::basic_istringstream</b>	
<b>OpenGPS::PointVectorInputStream</b>	<b>390</b>
<b>std::basic_ostream</b>	
<b>std::basic_ofstream</b>	
<b>OpenGPS::OutputBinaryFileStream</b>	<b>359</b>
<b>std::basic_ostringstream</b>	
<b>OpenGPS::PointVectorOutputStream</b>	<b>393</b>
<b>std::basic_ios&lt; char &gt;</b>	
<b>std::basic_ostream&lt; char &gt;</b>	
<b>std::ostream</b>	
<b>OpenGPS::ZipOutputStream</b>	<b>588</b>
<b>OpenGPS::ISO5436_2</b>	<b>266</b>
<b>OpenGPS::ISO5436_2Container</b>	<b>279</b>
<b>OpenGPS::Schemas::ISO5436_2::ISO5436_2Type</b>	<b>322</b>
<b>OpenGPS::Schemas::ISO5436_2::MatrixDimensionType</b>	<b>344</b>
<b>md5_context</b>	<b>356</b>
<b>OpenGPS::PointBuffer</b>	<b>362</b>
<b>OpenGPS::DoublePointBuffer</b>	<b>199</b>
<b>OpenGPS::FloatPointBuffer</b>	<b>229</b>
<b>OpenGPS::Int16PointBuffer</b>	<b>255</b>
<b>OpenGPS::Int32PointBuffer</b>	<b>261</b>
<b>OpenGPS::PointIterator</b>	<b>368</b>
<b>OpenGPS::ISO5436_2Container::PointIteratorImpl</b>	<b>314</b>
<b>OpenGPS::PointValidityProvider</b>	<b>373</b>

OpenGPS::DoubleInlineValidity	197
OpenGPS::FloatInlineValidity	226
OpenGPS::ValidBuffer	560
OpenGPS::Int16ValidBuffer	258
OpenGPS::Int32ValidBuffer	264
OpenGPS::PointVectorBase	387
OpenGPS::PointVector	376
OpenGPS::PointVectorProxy	402
OpenGPS::PointVectorInvariantLocale	392
OpenGPS::PointVectorParser	396
OpenGPS::PointVectorParserBuilder	400
OpenGPS::PointVectorProxyContext	414
OpenGPS::PointVectorProxyContextList	416
OpenGPS::PointVectorProxyContextMatrix	419
OpenGPS::PointVectorReaderContext	423
OpenGPS::BinaryPointVectorReaderContext	127
OpenGPS::BinaryLSBPointVectorReaderContext	115
OpenGPS::BinaryMSBPointVectorReaderContext	121
OpenGPS::XmlPointVectorReaderContext	575
OpenGPS::PointVectorWhitespaceFacet	427
OpenGPS::PointVectorWriterContext	428
OpenGPS::BinaryPointVectorWriterContext	132
OpenGPS::BinaryLSBPointVectorWriterContext	119
OpenGPS::BinaryMSBPointVectorWriterContext	125
OpenGPS::XmlPointVectorWriterContext	582
OpenGPS::Schemas::ISO5436_2::ProbingSystemType	432
OpenGPS::Schemas::ISO5436_2::Record1Type	443

OpenGPS::Schemas::ISO5436_2::Record2Type	457
OpenGPS::Schemas::ISO5436_2::Record3Type	482
OpenGPS::Schemas::ISO5436_2::Record4Type	504
OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType	511
OpenGPS::Schemas::ISO5436_2::RotationType	516
OpenGPS::Schemas::ISO5436_2::Type	553
OpenGPS::VectorBuffer	564
OpenGPS::VectorBufferBuilder	571
OpenGPS::ZipStreamBuffer	590

## 5 openGPS ISO 5436-2 XML Class Index

### 5.1 openGPS ISO 5436-2 XML Class List

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

<b>_OGPS_DATA_POINT</b> (Encapsulates the C++ structure of a data point handle used internally within the C interface )	66
<b>_OGPS_ISO5436_2_HANDLE</b> (Encapsulates the internal C++ structure of an ISO5436-2 handle used within the C interface )	67
<b>_OGPS_POINT_ITERATOR</b> (Encapsulates the internal C++ structure of a point iterator handle used within the C interface )	68
<b>_OGPS_POINT_VECTOR</b> (Encapsulates the internal C++ structure of a point vector handle used within the C interface )	69
<b>OpenGPS::Schemas::ISO5436_2::AxesType</b> (Class corresponding to the AxesType schema type )	70
<b>OpenGPS::Schemas::ISO5436_2::AxisDescriptionType</b> (Class corresponding to the AxisDescriptionType schema type )	88
<b>OpenGPS::Schemas::ISO5436_2::AxisType</b> (Enumeration class corresponding to the AxisType schema type )	107
<b>OpenGPS::BinaryLSBPointVectorReaderContext</b> (Implements OpenGPS::BinaryPointVectorReaderContext for binary	

files to be parsed on machines reading in least significant byte order )	115
<b>OpenGPS::BinaryLSBPointVectorWriterContext</b> (Implements <b>OpenGPS::BinaryPointVectorWriterContext</b> for binary files to be written on machines operating in least significant byte order )	119
<b>OpenGPS::BinaryMSBPointVectorReaderContext</b> (Implements <b>OpenGPS::BinaryPointVectorReaderContext</b> for binary files to be parsed on machines reading in most significant byte order )	121
<b>OpenGPS::BinaryMSBPointVectorWriterContext</b> (Implements <b>OpenGPS::BinaryPointVectorWriterContext</b> for binary files to be written on machines operating in most significant byte order )	125
<b>OpenGPS::BinaryPointVectorReaderContext</b> (Specialized <b>OpenGPS::PointVectorReaderContext</b> for binary streams )	127
<b>OpenGPS::BinaryPointVectorWriterContext</b> (Implements <b>OpenGPS::PointVectorWriterContext</b> for writing to compressed binary streams of point vectors )	132
<b>OpenGPS::Schemas::ISO5436_2::DataLinkType</b> (Class corresponding to the DataLinkType schema type )	135
<b>OpenGPS::Schemas::ISO5436_2::DataListType</b> (Class corresponding to the DataListType schema type )	156
<b>OpenGPS::DataPoint</b> (Typesafe representation of a single data point value )	164
<b>OpenGPS::DataPointImpl</b> (A straightforward implementation of <b>OpenGPS::DataPoint</b> )	171
<b>OpenGPS::DataPointImpl::_OGPS_DATA_POINT_VALUE</b> (Typesafe storage for every possible type of point data )	178
<b>OpenGPS::DataPointParser</b> (Interface for reading/writing point data )	180
<b>OpenGPS::Schemas::ISO5436_2::DataType</b> (Enumeration class corresponding to the DataType schema type )	182
<b>OpenGPS::Schemas::ISO5436_2::Datum</b> (Class corresponding to the Datum schema type )	190
<b>OpenGPS::DoubleDataPointParser</b> (Reads/Writes instances of OpenGPS::DataPoint of <b>OGPS_Double</b> point data )	195

<b>OpenGPS::DoubleInlineValidity</b> (Implements OpenGPS::PointValidityProvider as a lookup of a spe- cial IEEE754 value )	<b>197</b>
<b>OpenGPS::DoublePointBuffer</b> (Manages static memory and typesafe access )	<b>199</b>
<b>OpenGPS::Environment</b> (Interface for communicating with the operating system and related subjects )	<b>202</b>
<b>OpenGPS::Exception</b> (Describes a general exception )	<b>213</b>
<b>OpenGPS::ExceptionHistory</b> (Maintains a history of exceptions )	<b>216</b>
<b>OpenGPS::Schemas::ISO5436_2::FeatureType</b> (Class corre- sponding to the FeatureType schema type )	<b>220</b>
<b>OpenGPS::FloatDataPointParser</b> (Reads/Writes instances of OpenGPS::DataPoint of OGPS_Float point data )	<b>224</b>
<b>OpenGPS::FloatInlineValidity</b> (Implements OpenGPS::PointValidityProvider as a lookup of a spe- cial IEEE754 value )	<b>226</b>
<b>OpenGPS::FloatPointBuffer</b> (Manages static memory and type- safe access )	<b>229</b>
<b>OpenGPS::Info</b> (Publishes license text, ownership and similar information )	<b>231</b>
<b>OpenGPS::InputBinaryFileStream</b> (A binary stream class used for reading from binary files )	<b>234</b>
<b>OpenGPS::Schemas::ISO5436_2::InstrumentType</b> (Class cor- responding to the InstrumentType schema type )	<b>236</b>
<b>OpenGPS::Int16DataPointParser</b> (Reads/Writes instances of OpenGPS::DataPoint of OGPS_Int16 point data )	<b>253</b>
<b>OpenGPS::Int16PointBuffer</b> (Manages static memory and typesafe access )	<b>255</b>
<b>OpenGPS::Int16ValidBuffer</b> (Implementation of OpenGPS::ValidBuffer for Z axis of OGPS_Int16 data type )	<b>258</b>
<b>OpenGPS::Int32DataPointParser</b> (Reads/Writes instances of OpenGPS::DataPoint of OGPS_Int32 point data )	<b>259</b>
<b>OpenGPS::Int32PointBuffer</b> (Manages static memory and typesafe access )	<b>261</b>

<b>OpenGPS::Int32ValidBuffer</b> (Implementation of OpenGPS::ValidBuffer for Z axis of <code>OGPS_Int32</code> data type )	<b>264</b>
<b>OpenGPS::ISO5436_2</b> (Represents the ISO5436-2 XML X3P file format container )	<b>266</b>
<b>OpenGPS::ISO5436_2Container</b> (This is the main gate to this software library )	<b>279</b>
<b>OpenGPS::ISO5436_2Container::PointIteratorImpl</b> (Implementation of the point iterator interface )	<b>314</b>
<b>OpenGPS::Schemas::ISO5436_2::ISO5436_2Type</b> (Class corresponding to the ISO5436_2Type schema type )	<b>322</b>
<b>OpenGPS::Schemas::ISO5436_2::MatrixDimensionType</b> (Class corresponding to the MatrixDimensionType schema type )	<b>344</b>
<b>md5_context</b> (MD5 context structure )	<b>356</b>
<b>OpenGPS::MissingDataPointParser</b> (Reads/Writes instances of OpenGPS::DataPoint of missing point data )	<b>357</b>
<b>OpenGPS::OutputBinaryFileStream</b> (A binary stream class used for writing to binary files )	<b>359</b>
<b>OpenGPS::PointBuffer</b> (A static memory buffer that stores all point data belonging to a single axis )	<b>362</b>
<b>OpenGPS::PointIterator</b> (Interface to a point iterator )	<b>368</b>
<b>OpenGPS::PointValidityProvider</b> (Interface to communicate the validity of a point vector at a given location )	<b>373</b>
<b>OpenGPS::PointVector</b> (Typesafe representation of three-dimensional point measurement data )	<b>376</b>
<b>OpenGPS::PointVectorBase</b> (Typesafe and very fundamental representation of three-dimensional point measurement data )	<b>387</b>
<b>OpenGPS::PointVectorInputStream</b> (A string stream used for parsing a point vector as defined in ISO5436-2 XML )	<b>390</b>
<b>OpenGPS::PointVectorInvariantLocale</b> (The culture invariant locale )	<b>392</b>
<b>OpenGPS::PointVectorOutputStream</b> (A locale invariant string stream used to convert a point vector object to its rep-	

resentation as text according to the ISO5436-2 XML specification )	393
<b>OpenGPS::PointVectorParser</b> (Generic parser of a point vector )	396
<b>OpenGPS::PointVectorParserBuilder</b> (Component which handles the building process of a specialized parser object for reading and writing typed point data as a three-vector )	400
<b>OpenGPS::PointVectorProxy</b> (Serves one row of the three distinct buffers of data points managed by <b>OpenGPS::VectorBuffer</b> and raises the expression of accessing a single point vector )	402
<b>OpenGPS::PointVectorProxy::DataPointProxy</b> (Proxies one single data point pointed to by the current context information )	407
<b>OpenGPS::PointVectorProxyContext</b> (Indexing of point data managed by <b>OpenGPS::VectorBuffer</b> )	414
<b>OpenGPS::PointVectorProxyContextList</b> (Indexing of point data managed by <b>OpenGPS::VectorBuffer</b> )	416
<b>OpenGPS::PointVectorProxyContextMatrix</b> (Indexing of point data managed by <b>OpenGPS::VectorBuffer</b> )	419
<b>OpenGPS::PointVectorReaderContext</b> (Encapsulates an underlying stream of point vector data )	423
<b>OpenGPS::PointVectorWhitespaceFacet</b> (Redefines the white space character set to make parsing of a point vector easier )	427
<b>OpenGPS::PointVectorWriterContext</b> (Encapsulates an underlying stream of point vector data )	428
<b>OpenGPS::Schemas::ISO5436_2::ProbingSystemType</b> (Class corresponding to the ProbingSystemType schema type )	432
<b>OpenGPS::Schemas::ISO5436_2::Record1Type</b> (Class corresponding to the Record1Type schema type )	443
<b>OpenGPS::Schemas::ISO5436_2::Record2Type</b> (Class corresponding to the Record2Type schema type )	457
<b>OpenGPS::Schemas::ISO5436_2::Record3Type</b> (Class corresponding to the Record3Type schema type )	482
<b>OpenGPS::Schemas::ISO5436_2::Record4Type</b> (Class corresponding to the Record4Type schema type )	504

<b>OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType</b> (Class corresponding to the RotationMatrixElementType schema type )	<b>511</b>
<b>OpenGPS::Schemas::ISO5436_2::RotationType</b> (Class corresponding to the RotationType schema type )	<b>516</b>
<b>OpenGPS::String</b> (Stores an OGPS_Character sequence )	<b>549</b>
<b>OpenGPS::Schemas::ISO5436_2::Type</b> (Enumeration class corresponding to the Type schema type )	<b>553</b>
<b>OpenGPS::ValidBuffer</b> (Implements the OpenGPS::PointValidityProvider as an external binary file )	<b>560</b>
<b>OpenGPS::VectorBuffer</b> (Implements the memory structure of point measurement data )	<b>564</b>
<b>OpenGPS::VectorBufferBuilder</b> (Creates an object which is able to assemble a OpenGPS::VectorBuffer instance )	<b>571</b>
<b>OpenGPS::XmlPointVectorReaderContext</b> (Specialized OpenGPS::PointVectorReaderContext for point vectors stored as list of strings )	<b>575</b>
<b>OpenGPS::XmlPointVectorWriterContext</b> (Specialized OpenGPS::PointVectorWriterContext for point vectors stored as list of strings )	<b>582</b>
<b>OpenGPS::ZipOutputStream</b> (Provides an output stream interface to write binary data to Info-Zip archives )	<b>588</b>
<b>OpenGPS::ZipStreamBuffer</b> (Provides a buffer interface suitable for streaming the zipFile handle defined in the zlib/minizip package )	<b>590</b>

## 6 openGPS ISO 5436-2 XML File Index

### 6.1 openGPS ISO 5436-2 XML File List

Here is a list of all files with brief descriptions:

<b>auto_ptr_types.hxx</b> (Agglomerated definition of std::auto_ptr commonly used )	<b>592</b>
<b>binary_lsb_point_vector_reader_context.cxx</b>	<b>593</b>
<b>binary_lsb_point_vector_reader_context.hxx</b> (Implementation of access methods for reading typed point data from	

a binary file of point vectors )	595
<b>binary_lsb_point_vector_writer_context.cxx</b>	596
<b>binary_lsb_point_vector_writer_context.hxx</b> (Implementation of access methods for writing typed point data to a binary file of point vectors )	597
<b>binary_msb_point_vector_reader_context.cxx</b>	598
<b>binary_msb_point_vector_reader_context.hxx</b> (Implementation of access methods for reading typed point data from a binary file of point vectors )	600
<b>binary_msb_point_vector_writer_context.cxx</b>	601
<b>binary_msb_point_vector_writer_context.hxx</b> (Implementation of access methods for writing typed point data to a binary file of point vectors )	602
<b>binary_point_vector_reader_context.cxx</b>	603
<b>binary_point_vector_reader_context.hxx</b> (Interface for reading typed point data from an underlying binary file stream of point vectors )	604
<b>binary_point_vector_writer_context.cxx</b>	605
<b>binary_point_vector_writer_context.hxx</b> (Interface for writing typed point data to an underlying binary of point vectors )	607
<b>data_point.h</b> (The abstract data type of the typesafe representation of a single data point value )	608
<b>data_point.hxx</b> (Typesafe representation of a single data point value )	613
<b>data_point_c.cxx</b>	613
<b>data_point_c.hxx</b> (Handle mechanism that makes a C++ <code>OpenGPS::DataPoint</code> object accessible through its corresponding C interface )	618
<b>data_point_impl.cxx</b>	619
<b>data_point_impl.hxx</b> (An implementation of a data point to be used for typesafe storage and access of point data )	619
<b>data_point_parser.hxx</b> (Generic interface of a data point parser for reading/writing point data of any supported data type from/to any supported media )	620

<a href="#">data_point_proxy.cxx</a>	621
<a href="#">data_point_type.h</a> (An enumeration type which may describe the data type of a current data point value )	622
<a href="#">double_data_point_parser.cxx</a>	623
<a href="#">double_data_point_parser.hxx</a> (A data point parser for point data of type <code>OGPS_DoublePointType</code> )	624
<a href="#">double_point_buffer.cxx</a>	625
<a href="#">double_point_buffer.hxx</a> (Allocate static memory to store point data )	625
<a href="#">Doxygen.cpp</a> (Title page of documentation, no source code )	626
<a href="#">environment.cxx</a>	626
<a href="#">environment.hxx</a> (An interface to provide access to the envi- ronment of different architectures and operating systems )	627
<a href="#">exceptions.cxx</a>	628
<a href="#">exceptions.hxx</a> (Exception types thrown within this software library )	629
<a href="#">float_data_point_parser.cxx</a>	630
<a href="#">float_data_point_parser.hxx</a> (A data point parser for point data of type <code>OGPS_FloatPointType</code> )	630
<a href="#">float_point_buffer.cxx</a>	631
<a href="#">float_point_buffer.hxx</a> (Allocate static memory to store point data )	632
<a href="#">info.cxx</a>	633
<a href="#">info.h</a> (Copyright and license information )	633
<a href="#">info.hxx</a> (Copyright and license information )	636
<a href="#">info_c.cxx</a>	637
<a href="#">inline_validity.cxx</a>	640
<a href="#">inline_validity.hxx</a> (Communicate validity of point vectors through special IEEE754 values )	641
<a href="#">int16_data_point_parser.cxx</a>	642

<code>int16_data_point_parser.hxx</code> (A data point parser for point data of type <code>OGPS_Int16PointType</code> )	642
<code>int16_point_buffer.cxx</code>	643
<code>int16_point_buffer.hxx</code> (Allocate static memory to store point data )	644
<code>int32_data_point_parser.cxx</code>	645
<code>int32_data_point_parser.hxx</code> (A data point parser for point data of type <code>OGPS_Int32PointType</code> )	645
<code>int32_point_buffer.cxx</code>	646
<code>int32_point_buffer.hxx</code> (Allocate static memory to store point data )	647
<code>iso5436_2.cxx</code>	648
<code>iso5436_2.h</code> (The abstract data type of the ISO 5436-2 X3P file specification )	648
<code>iso5436_2.hxx</code> (Represents the ISO5436-2 XML X3P file for- mat container )	660
<code>iso5436_2_c.cxx</code>	661
<code>iso5436_2_container.cxx</code>	674
<code>iso5436_2_container.hxx</code> (Concrete implementation of the in- terface of an X3P container )	675
<code>iso5436_2_handle.hxx</code> (Enhancing C++ part of the C interface of the abstract data type of the ISO 5436-2 X3P file format )	676
<code>iso5436_2_handle_c.hxx</code> (Handle mechanism which makes a C++ <code>OpenGPS::ISO5436_2</code> object accessible through the corresponding C interface of an ISO5436-2 X3P file container )	680
<code>ISO5436_2_XML_Demo.cxx</code>	680
<code>iso5436_2_xsd.cxx</code>	682
<code>src/ISO5436_2_XML/iso5436_2_xsd.hxx</code> (Generated from <code>iso5436_2.xsd</code> )	689
<code>include/opengps/cxx/iso5436_2_xsd.hxx</code> (Generated from <code>iso5436_2.xsd</code> )	700
<code>md5.c</code>	709

<b>md5.h</b>	<b>714</b>
<b>messages.h</b> (Global handling of error and warning messages )	<b>717</b>
<b>messages_c.cxx</b>	<b>720</b>
<b>messages_c.hxx</b> (Error and warning messages during application )	<b>722</b>
<b>missing_data_point_parser.cxx</b>	<b>726</b>
<b>missing_data_point_parser.hxx</b> (A data point parser for point data of type <code>OGPS_MissingPointType</code> )	<b>726</b>
<b>opengps.h</b> (Common define's and typedef's )	<b>727</b>
<b>opengps.hxx</b> (Common define's and typedef's )	<b>730</b>
<b>point_buffer.cxx</b>	<b>731</b>
<b>point_buffer.hxx</b> (Allocate static memory to store point data )	<b>731</b>
<b>point_iterator.cxx</b>	<b>732</b>
<b>point_iterator.h</b> (The abstract data type of a point iterator used for direct manipulation of point data stored in an ISO 5436-2 X3P file )	<b>732</b>
<b>point_iterator.hxx</b> (Interface to a point iterator used for direct manipulation of point data stored in an ISO 5436-2 X3P file )	<b>739</b>
<b>point_iterator_c.cxx</b>	<b>740</b>
<b>point_iterator_c.hxx</b> (Handle mechanism which makes a C++ <code>OpenGPS::PointIterator</code> object accessible through the corresponding C interface of a point iterator )	<b>745</b>
<b>point_validity_provider.cxx</b>	<b>746</b>
<b>point_validity_provider.hxx</b> (Interface to communicate whether the point vector at a given location contains valid data )	<b>747</b>
<b>point_vector.cxx</b>	<b>748</b>
<b>point_vector.h</b> (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 )	<b>748</b>
<b>point_vector.hxx</b> (Typesafe representation of three-dimensional point measurement data )	<b>761</b>

<code>point_vector_base.hxx</code> (Very fundamental representation of three-dimensional point measurement data )	762
<code>point_vector_c.hxx</code>	763
<code>point_vector_c.hxx</code> (Handle mechanism which makes a C++ <code>OpenGPS::PointVector</code> object accessible through the corresponding C interface of a point vector )	776
<code>point_vector_iostream.hxx</code>	777
<code>point_vector_iostream.hxx</code> (Locale related stuff to support locale invariant parsing of point vector data )	777
<code>point_vector_parser.hxx</code>	778
<code>point_vector_parser.hxx</code> (Generic parser of a point vector )	779
<code>point_vector_parser_builder.hxx</code>	780
<code>point_vector_parser_builder.hxx</code> (Builder which assembles a concrete instance of the generic point vector parser )	780
<code>point_vector_proxy.hxx</code>	781
<code>point_vector_proxy.hxx</code> (Make the internal memory structure of distinct data point buffers accessible as point vector data )	781
<code>point_vector_proxy_context.hxx</code>	782
<code>point_vector_proxy_context.hxx</code> (Indexing of point data managed by <code>OpenGPS::VectorBuffer</code> )	783
<code>point_vector_proxy_context_list.hxx</code>	784
<code>point_vector_proxy_context_list.hxx</code> (Indexing of point data managed by <code>OpenGPS::VectorBuffer</code> )	784
<code>point_vector_proxy_context_matrix.hxx</code>	785
<code>point_vector_proxy_context_matrix.hxx</code> (Indexing of point data managed by <code>OpenGPS::VectorBuffer</code> )	785
<code>point_vector_reader_context.hxx</code> (Interface for reading typed point data from an underlying stream of point vectors )	786
<code>point_vector_writer_context.hxx</code> (Interface for writing typed point data to an underlying stream of point vectors )	787
<code>stdafx.hxx</code> (Standard includes and defines )	788
<code>string.hxx</code>	792

<a href="#">string.hxx</a> (An enhanced std::string string type )	793
<a href="#">valid_buffer.cxx</a>	793
<a href="#">valid_buffer.hxx</a> (Communicate validity of point vectors through external binary file )	794
<a href="#">vector_buffer.cxx</a>	795
<a href="#">vector_buffer.hxx</a> (The internal memory structure that stores all point measurement data )	796
<a href="#">vector_buffer_builder.cxx</a>	796
<a href="#">vector_buffer_builder.hxx</a> (Methods to assemble a OpenGPS::VectorBuffer instance )	797
<a href="#">win32_environment.cxx</a>	798
<a href="#">win32_environment.hxx</a> (The environment of Microsoft Windows operating systems )	798
<a href="#">xml_point_vector_reader_context.cxx</a>	798
<a href="#">xml_point_vector_reader_context.hxx</a> (Implementation of access methods for reading typed point data from a string list of point vectors )	799
<a href="#">xml_point_vector_writer_context.cxx</a>	800
<a href="#">xml_point_vector_writer_context.hxx</a> (Implementation of access methods for writing typed point data to a string list of point vectors )	802
<a href="#">xsd_Licence_Header.c</a>	802
<a href="#">zip_stream_buffer.cxx</a>	803
<a href="#">zip_stream_buffer.hxx</a> (Integrates Info-Zip handles into the common standard io framework )	803

## 7 openGPS ISO 5436-2 XML Directory Documentation

### 7.1 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.*

## 7.2 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.*

## 7.3 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 `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.*

## 7.4 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](#)

## 7.5 ISO5436\_2\_XML\_Demo/ Directory Reference

### Files

- file [ISO5436\\_2\\_XML\\_Demo.cxx](#)

## 7.6 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.*

## 7.7 ISO5436\_2\_XML/xyssl/ Directory Reference

### Files

- file [md5.c](#)
- file [md5.h](#)

## 8 openGPS ISO 5436-2 XML Namespace Documentation

### 8.1 OpenGPS Namespace Reference

#### 8.1.1 Detailed Description

The standard namespace for the openGPS software 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 [PointVectorInputStream](#)  
*A string stream used for parsing a point vector as defined in ISO5436-2 XML.*
- class [PointVectorInvariantLocale](#)  
*The culture invariant locale.*
- class [PointVectorOutputStream](#)  
*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.*

### 8.1.2 Typedef Documentation

#### 8.1.2.1 **typedef char OpenGPS::Byte**

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

#### 8.1.2.2 **typedef Byte\* OpenGPS::BytePtr**

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

#### 8.1.2.3 **typedef char OpenGPS::OGPS\_ExceptionChar**

Character type used to provide more information about an exception.

#### 8.1.2.4 **typedef std::auto\_ptr<PointBuffer> OpenGPS::PointBufferAutoPtr**

std::auto\_ptr for usage with OpenGPS::PointBuffer type.

**8.1.2.5** `typedef std::auto_ptr< PointIterator >`  
**OpenGPS::PointIteratorAutoPtr**

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

**8.1.2.6** `typedef std::auto_ptr< PointVectorBase >`  
**OpenGPS::PointVectorAutoPtr**

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

**8.1.2.7** `typedef std::auto_ptr< PointVectorParserBuilder >`  
**OpenGPS::PointVectorParserBuilderAutoPtr**

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

**8.1.2.8** `typedef std::auto_ptr<PointVectorProxyContext>`  
**OpenGPS::PointVectorProxyContextAutoPtr**

**8.1.2.9** `typedef unsigned char OpenGPS::UnsignedByte`

Data type of the size of one unsigned byte.

**8.1.2.10** `typedef UnsignedByte* OpenGPS::UnsignedBytePtr`

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

**8.1.2.11** `typedef std::auto_ptr< VectorBuffer >`  
**OpenGPS::VectorBufferAutoPtr**

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

**8.1.2.12** `typedef std::auto_ptr< VectorBufferBuilder >`  
**OpenGPS::VectorBufferBuilderAutoPtr**

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

## 8.2 OpenGPS::Schemas Namespace Reference

### 8.2.1 Detailed Description

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

#### Namespaces

- namespace `ISO5436_2`

*C++ namespace for the [http://www.opengps.eu/2008/ISO5436\\_2](http://www.opengps.eu/2008/ISO5436_2) schema namespace.*

## 8.3 OpenGPS::Schemas::ISO5436\_2 Namespace Reference

### 8.3.1 Detailed Description

C++ namespace for the [http://www.opengps.eu/2008/ISO5436\\_2](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

- bool [operator!=](#) (const [RotationType](#) &x, const [RotationType](#) &y)
- bool [operator!=](#) (const [MatrixDimensionType](#) &x, const [MatrixDimensionType](#) &y)
- bool [operator!=](#) (const [DataLinkType](#) &x, const [DataLinkType](#) &y)
- bool [operator!=](#) (const [DataListType](#) &x, const [DataListType](#) &y)
- bool [operator!=](#) (const [ProbingSystemType](#) &x, const [ProbingSystemType](#) &y)
- bool [operator!=](#) (const [InstrumentType](#) &x, const [InstrumentType](#) &y)
- bool [operator!=](#) (const [AxisDescriptionType](#) &x, const [AxisDescriptionType](#) &y)
- bool [operator!=](#) (const [AxesType](#) &x, const [AxesType](#) &y)
- bool [operator!=](#) (const [Record4Type](#) &x, const [Record4Type](#) &y)
- bool [operator!=](#) (const [Record3Type](#) &x, const [Record3Type](#) &y)
- bool [operator!=](#) (const [Record2Type](#) &x, const [Record2Type](#) &y)
- bool [operator!=](#) (const [ISO5436\\_2Type](#) &x, const [ISO5436\\_2Type](#) &y)
- bool [operator!=](#) (const [Record1Type](#) &x, const [Record1Type](#) &y)
- void [operator<<](#) (::xsd::cxx::tree::list\_stream<wchar\_t> &l, const [Datum](#) &i)
- void [operator<<](#) (::xercesc::DOMAttr &a, const [Datum](#) &i)
- void [operator<<](#) (::xercesc::DOMElement &e, const [Datum](#) &i)

- void `operator<<` (::xsd::cxx::tree::list\_stream< wchar\_t > &l, const Type &i)
- void `operator<<` (::xercesc::DOMAttr &a, const Type &i)
- void `operator<<` (::xercesc::DOMElement &e, const Type &i)
- void `operator<<` (::xsd::cxx::tree::list\_stream< wchar\_t > &l, const DataType &i)
- void `operator<<` (::xercesc::DOMAttr &a, const DataType &i)
- void `operator<<` (::xercesc::DOMElement &e, const DataType &i)
- void `operator<<` (::xsd::cxx::tree::list\_stream< wchar\_t > &l, const AxisType &i)
- void `operator<<` (::xercesc::DOMAttr &a, const AxisType &i)
- void `operator<<` (::xercesc::DOMElement &e, const AxisType &i)
- void `operator<<` (::xsd::cxx::tree::list\_stream< wchar\_t > &l, const FeatureType &i)
- void `operator<<` (::xercesc::DOMAttr &a, const FeatureType &i)
- void `operator<<` (::xercesc::DOMElement &e, const FeatureType &i)
- void `operator<<` (::xsd::cxx::tree::list\_stream< wchar\_t > &l, const RotationMatrixElementType &i)
- void `operator<<` (::xercesc::DOMAttr &a, const RotationMatrixElementType &i)
- void `operator<<` (::xercesc::DOMElement &e, const RotationMatrixElementType &i)
- void `operator<<` (::xercesc::DOMElement &e, const RotationType &i)
- void `operator<<` (::xercesc::DOMElement &e, const MatrixDimensionType &i)
- void `operator<<` (::xercesc::DOMElement &e, const DataLinkType &i)
- void `operator<<` (::xercesc::DOMElement &e, const DataListType &i)
- void `operator<<` (::xercesc::DOMElement &e, const ProbingSystemType &i)
- void `operator<<` (::xercesc::DOMElement &e, const InstrumentType &i)
- void `operator<<` (::xercesc::DOMElement &e, const AxisDescriptionType &i)
- void `operator<<` (::xercesc::DOMElement &e, const AxesType &i)
- void `operator<<` (::xercesc::DOMElement &e, const Record4Type &i)
- void `operator<<` (::xercesc::DOMElement &e, const Record3Type &i)
- void `operator<<` (::xercesc::DOMElement &e, const Record2Type &i)
- void `operator<<` (::xercesc::DOMElement &e, const ISO5436\_2Type &i)
- void `operator<<` (::xercesc::DOMElement &e, const Record1Type &i)
- ::std::wostream & `operator<<` (::std::wostream &o, const Datum &i)
- ::std::wostream & `operator<<` (::std::wostream &o, const Type &i)
- ::std::wostream & `operator<<` (::std::wostream &o, Type::value i)
- ::std::wostream & `operator<<` (::std::wostream &o, const DataType &i)
- ::std::wostream & `operator<<` (::std::wostream &o, DataType::value i)
- ::std::wostream & `operator<<` (::std::wostream &o, const AxisType &i)
- ::std::wostream & `operator<<` (::std::wostream &o, AxisType::value i)
- ::std::wostream & `operator<<` (::std::wostream &o, const FeatureType &i)

- `::std::wostream & operator<< (::std::wostream &o, const RotationMatrixElementType &i)`
- `::std::wostream & operator<< (::std::wostream &o, const RotationType &i)`
- `::std::wostream & operator<< (::std::wostream &o, const MatrixDimensionType &i)`
- `::std::wostream & operator<< (::std::wostream &o, const DataLinkType &i)`
- `::std::wostream & operator<< (::std::wostream &o, const DataListType &i)`
- `::std::wostream & operator<< (::std::wostream &o, const ProbingSystemType &i)`
- `::std::wostream & operator<< (::std::wostream &o, const InstrumentType &i)`
- `::std::wostream & operator<< (::std::wostream &o, const AxisDescriptionType &i)`
- `::std::wostream & operator<< (::std::wostream &o, const AxesType &i)`
- `::std::wostream & operator<< (::std::wostream &o, const Record4Type &i)`
- `::std::wostream & operator<< (::std::wostream &o, const Record3Type &i)`
- `::std::wostream & operator<< (::std::wostream &o, const Record2Type &i)`
- `::std::wostream & operator<< (::std::wostream &o, const ISO5436_2Type &i)`
- `::std::wostream & operator<< (::std::wostream &o, const Record1Type &i)`
- `bool operator==(const RotationType &x, const RotationType &y)`
- `bool operator==(const MatrixDimensionType &x, const MatrixDimensionType &y)`
- `bool operator==(const DataLinkType &x, const DataLinkType &y)`
- `bool operator==(const DataListType &x, const DataListType &y)`
- `bool operator==(const ProbingSystemType &x, const ProbingSystemType &y)`
- `bool operator==(const InstrumentType &x, const InstrumentType &y)`
- `bool operator==(const AxisDescriptionType &x, const AxisDescriptionType &y)`
- `bool operator==(const AxesType &x, const AxesType &y)`
- `bool operator==(const Record4Type &x, const Record4Type &y)`
- `bool operator==(const Record3Type &x, const Record3Type &y)`
- `bool operator==(const Record2Type &x, const Record2Type &y)`
- `bool operator==(const ISO5436_2Type &x, const ISO5436_2Type &y)`
- `bool operator==(const Record1Type &x, const Record1Type &y)`

**Serialization functions for the ISO5436\_2 document root.**

*The only global element: The root node*

- `::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.*
- `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.*
- `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.*
- `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.*
- `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.*
- `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.*
- `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.*
- `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*

- `::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.*
- `::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.*
- `::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.*
- `::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.*
- `::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.*
- `::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.*
- `::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.*
- `::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.*

- `::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.*
- `::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.*
- `::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.*
- `::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.*
- `::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.*
- `::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.*

### 8.3.2 Function Documentation

**8.3.2.1 \_OPENGPS\_EXPORT::xsd::cxx::xml::dom::auto\_ptr<::xercesc::DOMDocument > OpenGPS::Schemas::ISO5436\_2::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.

```
8.3.2.2 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.

```
8.3.2.3 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.

```
8.3.2.4 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.

```
8.3.2.5 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.

```
8.3.2.6 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.

```
8.3.2.7 OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2::operator<<(::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.

```
8.3.2.8 OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_2::ISO5436_2::operator<<(::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.

```
8.3.2.9 OPENGPS_EXPORT::std::auto_ptr<::OpenGPS::Schemas::ISO5436_2::ISO5436_2Type> OpenGPS::Schemas::ISO5436_2::ISO5436_2::parse(::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.

```
8.3.2.10 _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.

```
8.3.2.11 _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.

```
8.3.2.12 _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.

```
8.3.2.13 _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.

```
8.3.2.14 _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.

```
8.3.2.15 _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.

```
8.3.2.16 _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.

```
8.3.2.17 _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.

```
8.3.2.18 _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.

```
8.3.2.19 _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.

```
8.3.2.20 _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.

---

```
8.3.2.21 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.

---

```
8.3.2.22 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.

---

```
8.3.2.23 OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const RotationType & x, const RotationType & y)
```

---

```
8.3.2.24 OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator!= (const MatrixDimensionType & x, const MatrixDimensionType & y)
```

8.3.2.25 **OPENGPS\_EXPORT** **bool** OpenGPS::Schemas::ISO5436\_-  
2::operator!= (const DataLinkType & *x*, const DataLinkType & *y*)

8.3.2.26 **OPENGPS\_EXPORT** **bool** OpenGPS::Schemas::ISO5436\_-  
2::operator!= (const DataListType & *x*, const DataListType & *y*)

8.3.2.27 **OPENGPS\_EXPORT** **bool** OpenGPS::Schemas::ISO5436\_-  
2::operator!= (const ProbingSystemType & *x*, const ProbingSystem-  
Type & *y*)

8.3.2.28 **OPENGPS\_EXPORT** **bool** OpenGPS::Schemas::ISO5436\_-  
2::operator!= (const InstrumentType & *x*, const InstrumentType &  
*y*)

8.3.2.29 **OPENGPS\_EXPORT** **bool** OpenGPS::Schemas::ISO5436\_-  
2::operator!= (const AxisDescriptionType & *x*, const AxisDescrip-  
tionType & *y*)

8.3.2.30 **OPENGPS\_EXPORT** **bool** OpenGPS::Schemas::ISO5436\_-  
2::operator!= (const AxesType & *x*, const AxesType & *y*)

8.3.2.31 **OPENGPS\_EXPORT** **bool** OpenGPS::Schemas::ISO5436\_-  
2::operator!= (const Record4Type & *x*, const Record4Type & *y*)

8.3.2.32 **OPENGPS\_EXPORT** **bool** OpenGPS::Schemas::ISO5436\_-  
2::operator!= (const Record3Type & *x*, const Record3Type & *y*)

8.3.2.33 **OPENGPS\_EXPORT** **bool** OpenGPS::Schemas::ISO5436\_-  
2::operator!= (const Record2Type & *x*, const Record2Type & *y*)

8.3.2.34 **OPENGPS\_EXPORT** **bool** OpenGPS::Schemas::ISO5436\_-  
2::operator!= (const ISO5436\_2Type & *x*, const ISO5436\_2Type &  
*y*)

8.3.2.35 **OPENGPS\_EXPORT** **bool** OpenGPS::Schemas::ISO5436\_-  
2::operator!= (const Record1Type & *x*, const Record1Type & *y*)

8.3.2.36 `_OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > & l, const Datum & i)`

8.3.2.37 `_OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-2::operator<< (::xercesc::DOMAttr & a, const Datum & i)`

8.3.2.38 `_OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-2::operator<< (::xercesc::DOMElement & e, const Datum & i)`

8.3.2.39 `_OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > & l, const Type & i)`

8.3.2.40 `_OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-2::operator<< (::xercesc::DOMAttr & a, const Type & i)`

8.3.2.41 `_OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-2::operator<< (::xercesc::DOMElement & e, const Type & i)`

8.3.2.42 `_OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > & l, const DataType & i)`

8.3.2.43 `_OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-2::operator<< (::xercesc::DOMAttr & a, const DataType & i)`

8.3.2.44 `_OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-2::operator<< (::xercesc::DOMElement & e, const DataType & i)`

8.3.2.45 `_OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > & l, const AxisType & i)`

8.3.2.46 `_OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-2::operator<< (::xercesc::DOMAttr & a, const AxisType & i)`

8.3.2.47 `_OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-2::operator<< (::xercesc::DOMElement & e, const AxisType & i)`

---

```
8.3.2.48 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > & l, const  
FeatureType & i)
```

```
8.3.2.49 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMAttr & a, const FeatureType & i)
```

```
8.3.2.50 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMElement & e, const FeatureType & i)
```

```
8.3.2.51 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > & l, const  
RotationMatrixElementType & i)
```

```
8.3.2.52 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMAttr & a, const RotationMatrixEle-  
mentType & i)
```

```
8.3.2.53 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMElement & e, const RotationMatrix-  
ElementType & i)
```

```
8.3.2.54 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMElement & e, const RotationType &  
i)
```

```
8.3.2.55 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMElement & e, const MatrixDimen-  
sionType & i)
```

```
8.3.2.56 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMElement & e, const DataLinkType &  
i)
```

```
8.3.2.57 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMElement & e, const DataListType &  
i)
```

```
8.3.2.58 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMElement & e, const ProbingSystem-  
Type & i)
```

---

```
8.3.2.59 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMElement & e, const InstrumentType  
& i)
```

```
8.3.2.60 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMElement & e, const AxisDescription-  
Type & i)
```

```
8.3.2.61 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMElement & e, const AxesType & i)
```

```
8.3.2.62 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMElement & e, const Record4Type &  
i)
```

```
8.3.2.63 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMElement & e, const Record3Type &  
i)
```

```
8.3.2.64 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMElement & e, const Record2Type &  
i)
```

```
8.3.2.65 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMElement & e, const ISO5436_2Type  
& i)
```

```
8.3.2.66 _OPENGPS_EXPORT void OpenGPS::Schemas::ISO5436_-  
2::operator<< (::xercesc::DOMElement & e, const Record1Type &  
i)
```

```
8.3.2.67 _OPENGPS_EXPORT::std::wostream &  
OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &  
o, const Datum & i)
```

```
8.3.2.68 _OPENGPS_EXPORT::std::wostream &  
OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &  
o, const Type & i)
```

```
8.3.2.69 _OPENGPS_EXPORT::std::wostream &  
OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &  
o, Type::value i)
```

8.3.2.70 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const DataType & i)`

8.3.2.71 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, DataType::value i)`

8.3.2.72 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const AxisType & i)`

8.3.2.73 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, AxisType::value i)`

8.3.2.74 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const FeatureType & i)`

8.3.2.75 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const RotationMatrixElementType & i)`

8.3.2.76 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const RotationType & i)`

8.3.2.77 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const MatrixDimensionType & i)`

8.3.2.78 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const DataLinkType & i)`

8.3.2.79 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const DataListType & i)`

8.3.2.80 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const ProbingSystemType & i)`

8.3.2.81 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const InstrumentType & i)`

8.3.2.82 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const AxisDescriptionType & i)`

8.3.2.83 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const AxesType & i)`

8.3.2.84 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const Record4Type & i)`

8.3.2.85 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const Record3Type & i)`

8.3.2.86 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const Record2Type & i)`

8.3.2.87 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const ISO5436_2Type & i)`

8.3.2.88 `_OPENGPS_EXPORT::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream & o, const Record1Type & i)`

8.3.2.89 `_OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_-2::operator== (const RotationType & x, const RotationType & y)`

8.3.2.90 `_OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_-2::operator== (const MatrixDimensionType & x, const MatrixDimensionType & y)`

8.3.2.91 `_OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_-2::operator== (const DataLinkType & x, const DataLinkType & y)`

8.3.2.92 `_OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator==(const DataListType & x, const DataListType & y)`

8.3.2.93 `_OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator==(const ProbingSystemType & x, const ProbingSystemType & y)`

8.3.2.94 `_OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator==(const InstrumentType & x, const InstrumentType & y)`

8.3.2.95 `_OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator==(const AxisDescriptionType & x, const AxisDescriptionType & y)`

8.3.2.96 `_OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator==(const AxesType & x, const AxesType & y)`

8.3.2.97 `_OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator==(const Record4Type & x, const Record4Type & y)`

8.3.2.98 `_OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator==(const Record3Type & x, const Record3Type & y)`

8.3.2.99 `_OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator==(const Record2Type & x, const Record2Type & y)`

8.3.2.100 `_OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator==(const ISO5436_2Type & x, const ISO5436_2Type & y)`

8.3.2.101 `_OPENGPS_EXPORT bool OpenGPS::Schemas::ISO5436_2::operator==(const Record1Type & x, const Record1Type & y)`

## 8.4 std Namespace Reference

### 8.4.1 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\_ostringstream**  
*STL class.*

- class **basic\_ostream**  
*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 **wostreamstream**  
*STL class.*
- class **wstring**  
*STL class.*

- class **wstringstream**

*STL class.*

## 8.5 `xml_schema` Namespace Reference

### 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

**8.5.1 Typedef Documentation**

**8.5.1.1 `typedef::xsd::cxx::tree::base64_binary< wchar_t, simple_type > xml_schema::base64_binary`**

**8.5.1.2 `typedef bool xml_schema::boolean`**

**8.5.1.3 `typedef::xsd::cxx::tree::bounds< wchar_t > xml_schema::bounds`**

**8.5.1.4 `typedef::xsd::cxx::tree::buffer< wchar_t > xml_schema::buffer`**

**8.5.1.5 `typedef signed char xml_schema::byte`**

**8.5.1.6 `typedef::xsd::cxx::tree::date< wchar_t, simple_type > xml_schema::date`**

**8.5.1.7 `typedef::xsd::cxx::tree::date_time< wchar_t, simple_type > xml_schema::date_time`**

**8.5.1.8 `typedef::xsd::cxx::tree::day< wchar_t, simple_type > xml_schema::day`**

**8.5.1.9 `typedef double xml_schema::decimal`**

**8.5.1.10 `typedef::xsd::cxx::tree::diagnostics< wchar_t > xml_schema::diagnostics`**

**8.5.1.11 `typedef double xml_schema::double_`**

**8.5.1.12 `typedef::xsd::cxx::tree::duplicate_id< wchar_t > xml_schema::duplicate_id`**

**8.5.1.13 `typedef::xsd::cxx::tree::duration< wchar_t, simple_type > xml_schema::duration`**

8.5.1.14 `typedef::xsd::cxx::tree::entities< wchar_t, simple_type, entity > xml_schema::entities`

8.5.1.15 `typedef::xsd::cxx::tree::entity< wchar_t, ncname > xml_schema::entity`

8.5.1.16 `typedef::xsd::cxx::tree::error< wchar_t > xml_schema::error`

8.5.1.17 `typedef::xsd::cxx::xml::error_handler< wchar_t > xml_schema::error_handler`

8.5.1.18 `typedef::xsd::cxx::tree::exception< wchar_t > xml_schema::exception`

8.5.1.19 `typedef::xsd::cxx::tree::expected_attribute< wchar_t > xml_schema::expected_attribute`

8.5.1.20 `typedef::xsd::cxx::tree::expected_element< wchar_t > xml_schema::expected_element`

8.5.1.21 `typedef::xsd::cxx::tree::expected_text_content< wchar_t > xml_schema::expected_text_content`

8.5.1.22 `typedef::xsd::cxx::tree::flags xml_schema::flags`

8.5.1.23 `typedef float xml_schema::float_`

8.5.1.24 `typedef::xsd::cxx::tree::hex_binary< wchar_t, simple_type > xml_schema::hex_binary`

8.5.1.25 `typedef::xsd::cxx::tree::id< wchar_t, ncname > xml_schema::id`

8.5.1.26 `typedef::xsd::cxx::tree::idref< type, wchar_t, ncname > xml_schema::idref`

8.5.1.27 `typedef::xsd::cxx::tree::idrefs< wchar_t, simple_type, idref > xml_schema::idrefs`

8.5.1.28 `typedef int xml_schema::int_`

8.5.1.29 `typedef long long xml_schema::integer`

8.5.1.30 `typedef::xsd::cxx::tree::language< wchar_t, token > xml_schema::language`

8.5.1.31 `typedef long long xml_schema::long_`

8.5.1.32 `typedef::xsd::cxx::tree::month< wchar_t, simple_type > xml_schema::month`

8.5.1.33 `typedef::xsd::cxx::tree::month_day< wchar_t, simple_type > xml_schema::month_day`

8.5.1.34 `typedef::xsd::cxx::tree::name< wchar_t, token > xml_schema::name`

8.5.1.35 `typedef::xsd::cxx::xml::dom::namespace_info< wchar_t > xml_schema::namespace_info`

8.5.1.36 `typedef::xsd::cxx::xml::dom::namespace_infomap< wchar_t > xml_schema::namespace_infomap`

8.5.1.37 `typedef::xsd::cxx::tree::ncname< wchar_t, name > xml_schema::ncname`

8.5.1.38 `typedef integer xml_schema::negative_integer`

8.5.1.39 `typedef::xsd::cxx::tree::nmtoken< wchar_t, token > xml_schema::nmtoken`

8.5.1.40 `typedef::xsd::cxx::tree::nmtokens< wchar_t, simple_type, nmtoken > xml_schema::nmtokens`

8.5.1.41 `typedef::xsd::cxx::tree::no_namespace_mapping< wchar_t > xml_schema::no_namespace_mapping`

8.5.1.42 `typedef::xsd::cxx::tree::no_prefix_mapping< wchar_t > xml_schema::no_prefix_mapping`

8.5.1.43 `typedef::xsd::cxx::tree::no_type_info< wchar_t > xml_schema::no_type_info`

8.5.1.44 `typedef integer xml_schema::non_negative_integer`

8.5.1.45 `typedef integer xml_schema::non_positive_integer`

8.5.1.46 `typedef::xsd::cxx::tree::normalized_string< wchar_t, string > xml_schema::normalized_string`

8.5.1.47 `typedef::xsd::cxx::tree::not_derived< wchar_t > xml_schema::not_derived`

8.5.1.48 `typedef::xsd::cxx::tree::parsing< wchar_t > xml_schema::parsing`

8.5.1.49 `typedef integer xml_schema::positive_integer`

8.5.1.50 `typedef::xsd::cxx::tree::properties< wchar_t > xml_schema::properties`

8.5.1.51 `typedef::xsd::cxx::tree::qname< wchar_t, simple_type, uri, ncname > xml_schema::qname`

8.5.1.52 `typedef::xsd::cxx::tree::serialization< wchar_t > xml_schema::serialization`

8.5.1.53 `typedef::xsd::cxx::tree::severity xml_schema::severity`

8.5.1.54 `typedef short xml_schema::short_`

8.5.1.55 `typedef::xsd::cxx::tree::simple_type< type > xml_schema::simple_type`

8.5.1.56 `typedef::xsd::cxx::tree::string< wchar_t, simple_type > xml_schema::string`

8.5.1.57 `typedef::xsd::cxx::tree::time< wchar_t, simple_type > xml_schema::time`

8.5.1.58 `typedef::xsd::cxx::tree::token< wchar_t, normalized_string > xml_schema::token`

8.5.1.59 `typedef::xsd::cxx::tree::type xml_schema::type`

8.5.1.60 `typedef::xsd::cxx::tree::unexpected_element< wchar_t > xml_schema::unexpected_element`

8.5.1.61 `typedef::xsd::cxx::tree::unexpected_enumerator< wchar_t > xml_schema::unexpected_enumerator`

8.5.1.62 `typedef unsigned char xml_schema::unsigned_byte`

8.5.1.63 `typedef unsigned int xml_schema::unsigned_int`

8.5.1.64 `typedef unsigned long long xml_schema::unsigned_long`

8.5.1.65 `typedef unsigned short xml_schema::unsigned_short`

8.5.1.66 `typedef::xsd::cxx::tree::uri< wchar_t, simple_type > xml_schema::uri`

8.5.1.67 `typedef::xsd::cxx::tree::xsi_already_in_use< wchar_t > xml_schema::xsi_already_in_use`

8.5.1.68 `typedef::xsd::cxx::tree::year< wchar_t, simple_type > xml_schema::year`

8.5.1.69 `typedef::xsd::cxx::tree::year_month< wchar_t, simple_type > xml_schema::year_month`

## 8.5.2 Variable Documentation

8.5.2.1 `const XMLCh *const xml_schema::tree_node_key = ::xsd::cxx::tree::user_data_keys::node`

# 9 openGPS ISO 5436-2 XML Class Documentation

## 9.1 \_OGPS\_DATA\_POINT Struct Reference

```
#include <data_point_c.hxx>
```

Collaboration diagram for \_OGPS\_DATA\_POINT:

### 9.1.1 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.*

### 9.1.2 Member Data Documentation

#### 9.1.2.1 OpenGPS::DataPoint\* \_OGPS\_DATA\_POINT::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](#)

## 9.2 \_OGPS\_ISO5436\_2\_HANDLE Struct Reference

```
#include <iso5436_2_handle_c.hxx>
```

Collaboration diagram for \_OGPS\_ISO5436\_2\_HANDLE:

### 9.2.1 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.*

### 9.2.2 Member Data Documentation

#### 9.2.2.1 [OpenGPS::ISO5436\\_2\\* \\_OGPS\\_ISO5436\\_2\\_-HANDLE::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](#)

## 9.3 \_OGPS\_POINT\_ITERATOR Struct Reference

```
#include <point_iterator_c.hxx>
```

Collaboration diagram for \_OGPS\_POINT\_ITERATOR:

### 9.3.1 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.*

### 9.3.2 Member Data Documentation

#### 9.3.2.1 OpenGPS::PointIterator\* \_OGPS\_POINT\_-ITERATOR::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](#)

## 9.4 \_OGPS\_POINT\_VECTOR Struct Reference

```
#include <point_vector_c.hxx>
```

Collaboration diagram for \_OGPS\_POINT\_VECTOR:

### 9.4.1 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.*

#### 9.4.2 Member Data Documentation

##### 9.4.2.1 [OpenGPS::PointVector](#) \_OGPS\_POINT\_- VECTOR::instance

Gets/Sets the pointer to the internal C++ object behind the scenes.

##### 9.4.2.2 [OGPS\\_DataPointPtr \\_OGPS\\_POINT\\_VECTOR::x](#)

Gets/Sets the buffered C interface wrapper for typesafe access to the X member of the internal C++ [OpenGPS::PointVector](#) instance.

##### 9.4.2.3 [OGPS\\_DataPointPtr \\_OGPS\\_POINT\\_VECTOR::y](#)

Gets/Sets the buffered C interface wrapper for typesafe access to the Y member of the internal C++ [OpenGPS::PointVector](#) instance.

##### 9.4.2.4 [OGPS\\_DataPointPtr \\_OGPS\\_POINT\\_VECTOR::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](#)

## 9.5 OpenGPS::Schemas::ISO5436\_2::AxesType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

### 9.5.1 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\_

#### 9.5.2 Member Typedef Documentation

**9.5.2.1 typedef ::xsd::cxx::tree::traits< CX\_type, wchar\_t > OpenGPS::Schemas::ISO5436\_2::AxesType::CX\_traits**  
Element traits type.

**9.5.2.2 `typedef ::xsd::cxx::tree::traits< CX_type, wchar_t >`**  
OpenGPS::Schemas::ISO5436\_2::AxesType::CX\_traits

Element traits type.

**9.5.2.3 `typedef ::OpenGPS::Schemas::ISO5436_-`**  
**2::AxisDescriptionType** `::OpenGPS::Schemas::ISO5436_-`  
**2::AxesType::CX\_type**

Element type.

**9.5.2.4 `typedef ::OpenGPS::Schemas::ISO5436_-`**  
**2::AxisDescriptionType** `::OpenGPS::Schemas::ISO5436_-`  
**2::AxesType::CX\_type**

Element type.

**9.5.2.5 `typedef ::xsd::cxx::tree::traits< CY_type, wchar_t >`**  
OpenGPS::Schemas::ISO5436\_2::AxesType::CY\_traits

Element traits type.

**9.5.2.6 `typedef ::xsd::cxx::tree::traits< CY_type, wchar_t >`**  
OpenGPS::Schemas::ISO5436\_2::AxesType::CY\_traits

Element traits type.

**9.5.2.7 `typedef ::OpenGPS::Schemas::ISO5436_-`**  
**2::AxisDescriptionType** `::OpenGPS::Schemas::ISO5436_-`  
**2::AxesType::CY\_type**

Element type.

**9.5.2.8 `typedef ::OpenGPS::Schemas::ISO5436_-`**  
**2::AxisDescriptionType** `::OpenGPS::Schemas::ISO5436_-`  
**2::AxesType::CY\_type**

Element type.

**9.5.2.9 `typedef ::xsd::cxx::tree::traits< CZ_type, wchar_t >`**  
OpenGPS::Schemas::ISO5436\_2::AxesType::CZ\_traits

Element traits type.

**9.5.2.10 `typedef ::xsd::cxx::tree::traits< CZ_type, wchar_t >`**  
OpenGPS::Schemas::ISO5436\_2::AxesType::CZ\_traits

Element traits type.

**9.5.2.11 `typedef ::OpenGPS::Schemas::ISO5436_-OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType 2::AxesType::CZ_type`**

Element type.

**9.5.2.12 `typedef ::OpenGPS::Schemas::ISO5436_-OpenGPS::Schemas::ISO5436_-2::AxisDescriptionType 2::AxesType::CZ_type`**

Element type.

**9.5.2.13 `typedef ::xsd::cxx::tree::optional< Rotation_type > OpenGPS::Schemas::ISO5436_2::AxesType::Rotation_optional`**

Element optional container type.

**9.5.2.14 `typedef ::xsd::cxx::tree::optional< Rotation_type > OpenGPS::Schemas::ISO5436_2::AxesType::Rotation_optional`**

Element optional container type.

**9.5.2.15 `typedef ::xsd::cxx::tree::traits< Rotation_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxesType::Rotation_traits`**

Element traits type.

**9.5.2.16 `typedef ::xsd::cxx::tree::traits< Rotation_type, wchar_t > OpenGPS::Schemas::ISO5436_2::AxesType::Rotation_traits`**

Element traits type.

**9.5.2.17 `typedef ::OpenGPS::Schemas::ISO5436_2::RotationType OpenGPS::Schemas::ISO5436_2::AxesType::Rotation_type`**

Element type.

**9.5.2.18 `typedef ::OpenGPS::Schemas::ISO5436_2::RotationType OpenGPS::Schemas::ISO5436_2::AxesType::Rotation_type`**

Element type.

### **9.5.3 Constructor & Destructor Documentation**

**9.5.3.1 `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.

**9.5.3.2 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.

**9.5.3.3 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.

**9.5.3.4 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.

**9.5.3.5 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.

**9.5.3.6 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.

**9.5.4 Member Function Documentation**

**9.5.4.1 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.

**9.5.4.2 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.

**9.5.4.3 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.

**9.5.4.4 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.

**9.5.4.5 CX\_type& OpenGPS::Schemas::ISO5436\_-  
2::AxesType::CX ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.5.4.6 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.

**9.5.4.7 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.

## **9.5 OpenGPS::Schemas::ISO5436\_2::AxesType Class Reference 82**

**9.5.4.8 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.

**9.5.4.9 CX\_type& OpenGPS::Schemas::ISO5436\_2::AxesType::CX ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.5.4.10 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.

**9.5.4.11 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.

**9.5.4.12 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.

**9.5.4.13 CY\_type& OpenGPS::Schemas::ISO5436\_-  
2::AxesType::CY ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.5.4.14 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.

**9.5.4.15 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.

**9.5.4.16 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.

**9.5.4.17 CY\_type& OpenGPS::Schemas::ISO5436\_-  
2::AxesType::CY ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.5.4.18 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.

**9.5.4.19 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.

**9.5.4.20 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.

**9.5.4.21 CZ\_type& OpenGPS::Schemas::ISO5436\_2::AxesType::CZ ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.5.4.22 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.

## 9.5 OpenGPS::Schemas::ISO5436\_2::AxesType Class Reference 85

**9.5.4.23 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.

**9.5.4.24 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.

**9.5.4.25 CZ\_type& OpenGPS::Schemas::ISO5436\_-  
2::AxesType::CZ ()**

Return a read-write reference to the element.

### **Returns:**

A reference to the element.

**9.5.4.26 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.

**9.5.4.27 void OpenGPS::Schemas::ISO5436\_2::AxesType::parse  
(:xsd::cxx::xml::dom::parser< wchar\_t > &, ::xml\_schema::flags)  
[protected]**

**9.5.4.28 void OpenGPS::Schemas::ISO5436\_2::AxesType::parse  
(:xsd::cxx::xml::dom::parser< wchar\_t > & p, ::xml\_schema::flags  
f) [protected]**

**9.5.4.29 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.

**9.5.4.30 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.

**9.5.4.31 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.

**9.5.4.32 Rotation\_optional& OpenGPS::Schemas::ISO5436\_-  
2::AxesType::Rotation ()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.5.4.33 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.

**9.5.4.34 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.

**9.5.4.35 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.

**9.5.4.36 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.

**9.5.4.37 Rotation\_optional& OpenGPS::Schemas::ISO5436\_-  
2::AxesType::Rotation ()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.5.4.38 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.

### 9.5.5 Member Data Documentation

9.5.5.1 xsd::cxx::tree::one< CX\_type >  
OpenGPS::Schemas::ISO5436\_2::AxesType::CX\_ [private]

9.5.5.2 xsd::cxx::tree::one< CY\_type >  
OpenGPS::Schemas::ISO5436\_2::AxesType::CY\_ [private]

9.5.5.3 xsd::cxx::tree::one< CZ\_type >  
OpenGPS::Schemas::ISO5436\_2::AxesType::CZ\_ [private]

9.5.5.4 Rotation\_optional OpenGPS::Schemas::ISO5436\_-  
2::AxesType::Rotation\_ [private]

The documentation for this class was generated from the following files:

- src/ISO5436\_2\_XML/iso5436\_2\_xsd.hxx
- include/opengps/cxx/iso5436\_2\_xsd.hxx
- iso5436\_2\_xsd.cxx

## 9.6 OpenGPS::Schemas::ISO5436\_- 2::AxisDescriptionType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

### 9.6.1 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 multiplier 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 multiplier 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 &z)`  
*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 &z)`  
*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_`

### 9.6.2 Member Typedef Documentation

**9.6.2.1 `typedef ::xsd::cxx::tree::traits< AxisType_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::AxisType\_-  
traits**

Element traits type.

**9.6.2.2 `typedef ::xsd::cxx::tree::traits< AxisType_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::AxisType\_-  
traits**

Element traits type.

**9.6.2.3 `typedef ::OpenGPS::Schemas::ISO5436_2::AxisType`  
OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::AxisType\_-  
type**

Element type.

**9.6.2.4 `typedef ::OpenGPS::Schemas::ISO5436_2::AxisType`  
OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::AxisType\_-  
type**

Element type.

**9.6.2.5 `typedef ::xsd::cxx::tree::optional< DataType_type >`  
OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::DataType\_-  
optional**

Element optional container type.

**9.6.2.6 `typedef ::xsd::cxx::tree::optional< DataType_type >`  
OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::DataType\_-  
optional**

Element optional container type.

**9.6.2.7 `typedef ::xsd::cxx::tree::traits< DataType_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::DataType\_-  
traits**

Element traits type.

**9.6.2.8 `typedef ::xsd::cxx::tree::traits< DataType_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::DataType\_-  
traits**

Element traits type.

**9.6.2.9 `typedef ::OpenGPS::Schemas::ISO5436_2::DataType  
OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType_-  
type`**

Element type.

**9.6.2.10 `typedef ::OpenGPS::Schemas::ISO5436_2::DataType  
OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::DataType_-  
type`**

Element type.

**9.6.2.11 `typedef ::xsd::cxx::tree::optional<  
Increment_type > OpenGPS::Schemas::ISO5436_-  
2::AxisDescriptionType::Increment_optional`**

Element optional container type.

**9.6.2.12 `typedef ::xsd::cxx::tree::optional<  
Increment_type > OpenGPS::Schemas::ISO5436_-  
2::AxisDescriptionType::Increment_optional`**

Element optional container type.

**9.6.2.13 `typedef ::xsd::cxx::tree::traits< Increment_-  
type, wchar_t > OpenGPS::Schemas::ISO5436_-  
2::AxisDescriptionType::Increment_traits`**

Element traits type.

**9.6.2.14 `typedef ::xsd::cxx::tree::traits< Increment_-  
type, wchar_t > OpenGPS::Schemas::ISO5436_-  
2::AxisDescriptionType::Increment_traits`**

Element traits type.

**9.6.2.15 `typedef ::xml_schema::double  
OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment_-  
type`**

Element type.

**9.6.2.16 `typedef ::xml_schema::double  
OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::Increment_-  
type`**

Element type.

**9.6.2.17 `typedef ::xsd::cxx::tree::optional< Offset_type >`  
OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::Offset\_-  
optional**

Element optional container type.

**9.6.2.18 `typedef ::xsd::cxx::tree::optional< Offset_type >`  
OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::Offset\_-  
optional**

Element optional container type.

**9.6.2.19 `typedef ::xsd::cxx::tree::traits< Offset_type, wchar_-  
t >` OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::Offset\_-  
traits**

Element traits type.

**9.6.2.20 `typedef ::xsd::cxx::tree::traits< Offset_type, wchar_-  
t >` OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::Offset\_-  
traits**

Element traits type.

**9.6.2.21 `typedef ::xml_schema::double_-`  
OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::Offset\_type**

Element type.

**9.6.2.22 `typedef ::xml_schema::double_-`  
OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::Offset\_type**

Element type.

### 9.6.3 Constructor & Destructor Documentation

**9.6.3.1 `OpenGPS::Schemas::ISO5436_2::AxisDescriptionType::AxisDescriptionType  
(const AxisType_type & AxisType)`**

Construct an instance from the ultimate base and initializers for required elements and attributes.

**9.6.3.2 `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.

**9.6.3.3 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.

**9.6.3.4 OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::AxisDescriptionType**  
**(const AxisType\_type &)**

Construct an instance from the ultimate base and initializers for required elements and attributes.

**9.6.3.5 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.

**9.6.3.6 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.

#### 9.6.4 Member Function Documentation

**9.6.4.1 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.

**9.6.4.2 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.

**9.6.4.3 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.

**9.6.4.4 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.

**9.6.4.5 AxisType\_type& OpenGPS::Schemas::ISO5436\_-  
2::AxisDescriptionType::AxisType ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.6.4.6 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.

**9.6.4.7 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.

**9.6.4.8 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.

**9.6.4.9 AxisType\_type& OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::AxisType ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.6.4.10 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.

**9.6.4.11 void OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::Data\_Type (const 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.

**9.6.4.12 void OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::Data\_Type (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.

**9.6.4.13 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.

**9.6.4.14 DataType\_optional& OpenGPS::Schemas::ISO5436\_-  
2::AxisDescriptionType::DataTypE ()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.6.4.15 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.

**9.6.4.16 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.

**9.6.4.17 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.

**9.6.4.18 void OpenGPS::Schemas::ISO5436\_-  
2::AxisDescriptionType::Data\_Type (const Data\_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.

**9.6.4.19 Data\_Type\_optional& OpenGPS::Schemas::ISO5436\_-  
2::AxisDescriptionType::Data\_Type ()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.6.4.20 const Data\_Type\_optional& OpenGPS::Schemas::ISO5436\_-  
2::AxisDescriptionType::Data\_Type () const**

Return a read-only (constant) reference to the element container.

**Returns:**

A constant reference to the optional container.

**9.6.4.21 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.

**9.6.4.22 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.

**9.6.4.23 Increment\_optional& OpenGPS::Schemas::ISO5436\_-  
2::AxisDescriptionType::Increment ()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.6.4.24 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.

**9.6.4.25 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.

**9.6.4.26 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.

**9.6.4.27 Increment\_optional& OpenGPS::Schemas::ISO5436\_-2::AxisDescriptionType::Increment ()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.6.4.28 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.

**9.6.4.29 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.

**9.6.4.30 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.

**9.6.4.31 Offset\_optional& OpenGPS::Schemas::ISO5436\_-  
2::AxisDescriptionType::Offset ()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.6.4.32 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.

**9.6.4.33 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.

**9.6.4.34 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.

**9.6.4.35 Offset\_optional& OpenGPS::Schemas::ISO5436\_-  
2::AxisDescriptionType::Offset ()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

## **9.7 OpenGPS::Schemas::ISO5436\_2::AxisType Class Reference**

**9.6.4.36 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.

**9.6.4.37 void OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::parse (::xsd::cxx::xml::dom::parser< wchar\_t > &, ::xml\_schema::flags) [protected]**

**9.6.4.38 void OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::parse (::xsd::cxx::xml::dom::parser< wchar\_t > & p, ::xml\_schema::flags f) [protected]**

### **9.6.5 Member Data Documentation**

**9.6.5.1 xsd::cxx::tree::one< AxisType\_type > OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::AxisType\_ [private]**

**9.6.5.2 DataType\_optional OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::DataType\_ [private]**

**9.6.5.3 Increment\_optional OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::Increment\_ [private]**

**9.6.5.4 Offset\_optional OpenGPS::Schemas::ISO5436\_2::AxisDescriptionType::Offset\_ [private]**

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

## **9.7 OpenGPS::Schemas::ISO5436\_2::AxisType Class Reference**

```
#include <iso5436_2_xsd.hxx>
```

### **9.7.1 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)`

## **9.7 OpenGPS::Schemas::ISO5436\_2::AxisType Class Reference**

---

*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]
- static const wchar\_t \*const **\_xsd\_AxisType\_literals\_** [2]

### **Protected Member Functions**

- **value \_xsd\_AxisType\_convert () const**
- **value \_xsd\_AxisType\_convert () const**

#### **9.7.2 Member Enumeration Documentation**

##### **9.7.2.1 enum OpenGPS::Schemas::ISO5436\_2::AxisType::value**

Underlying enum type.

## 9.7 OpenGPS::Schemas::ISO5436\_2::AxisType Class Reference 110

**Enumerator:**

*A*

*I*

*A*

*I*

### 9.7.2.2 enum OpenGPS::Schemas::ISO5436\_2::AxisType::value

Underlying enum type.

**Enumerator:**

*A*

*I*

*A*

*I*

### 9.7.3 Constructor & Destructor Documentation

#### 9.7.3.1 OpenGPS::Schemas::ISO5436\_2::AxisType::AxisType (value *v*)

Construct an instance from the underlying enum value.

**Parameters:**

*v* A enum value.

#### 9.7.3.2 OpenGPS::Schemas::ISO5436\_2::AxisType::AxisType (const ::xml\_schema::token & *v*)

Construct an instance from the base value.

**Parameters:**

*v* A base value.

#### 9.7.3.3 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.

**9.7.3.4 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.

**9.7.3.5 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.

**9.7.3.6 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.

**9.7.3.7 OpenGPS::Schemas::ISO5436\_2::AxisType::AxisType**  
`(value v)`

Construct an instance from the underlying enum value.

**Parameters:**

- v* A enum value.

**9.7.3.8 OpenGPS::Schemas::ISO5436\_2::AxisType::AxisType  
(const ::xml\_schema::token & v)**

Construct an instance from the base value.

**Parameters:**

*v* A base value.

**9.7.3.9 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.

**9.7.3.10 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.

**9.7.3.11 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.

**9.7.3.12 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.

**9.7.4 Member Function Documentation**

**9.7.4.1 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.

**9.7.4.2 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.

**9.7.4.3 value OpenGPS::Schemas::ISO5436\_2::AxisType::\_xsd\_AxisType\_convert () const [protected]**

**9.7.4.4 AxisType::value OpenGPS::Schemas::ISO5436\_2::AxisType::\_xsd\_AxisType\_convert () const [protected]**

**9.7.4.5 virtual OpenGPS::Schemas::ISO5436\_2::AxisType::operator value () const [inline, virtual]**

Implicit conversion operator to the underlying enum value.

**Returns:**

A enum value.

**9.7.4.6 virtual OpenGPS::Schemas::ISO5436\_2::AxisType::operator value () const [inline, virtual]**

Implicit conversion operator to the underlying enum value.

**Returns:**

A enum value.

**9.7.4.7 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.

**9.7.4.8 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.

### 9.7.5 Member Data Documentation

9.7.5.1 static const value OpenGPS::Schemas::ISO5436\_-  
2::AxisType::\_xsd\_AxisType\_indexes\_ [static]

9.7.5.2 const wchar\_t\* const OpenGPS::Schemas::ISO5436\_-  
2::AxisType::\_xsd\_AxisType\_literals\_[2] [static]

9.7.5.3 const wchar\_t\* const OpenGPS::Schemas::ISO5436\_-  
2::AxisType::\_xsd\_AxisType\_literals\_[2] [static]

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

## 9.8 OpenGPS::BinaryLSBPointVectorReaderContext Class Reference

#include <binary\_lsb\_point\_vector\_reader\_context.hxx>

Inheritance diagram for OpenGPS::BinaryLSBPointVectorReaderContext:

Collaboration diagram for OpenGPS::BinaryLSBPointVectorReaderContext:

#### **9.8.1 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 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 `~BinaryLSBPointVectorReaderContext ()`  
*Destroys this instance.*

### 9.8.2 Constructor & Destructor Documentation

#### 9.8.2.1 BinaryLSBPointVectorReaderContext::BinaryLSBPointVectorReaderContext (const OpenGPS::String & filePath)

Creates a new instance.

**Parameters:**

*filePath* Absolute path to the binary file streamed herein.

#### 9.8.2.2 BinaryLSBPointVectorReaderContext::~BinaryLSBPointVectorReaderContext () [virtual]

Destroys this instance.

### 9.8.3 Member Function Documentation

#### 9.8.3.1 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`.

**9.8.3.2 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](#).

**9.8.3.3 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](#).

**9.8.3.4 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](#).

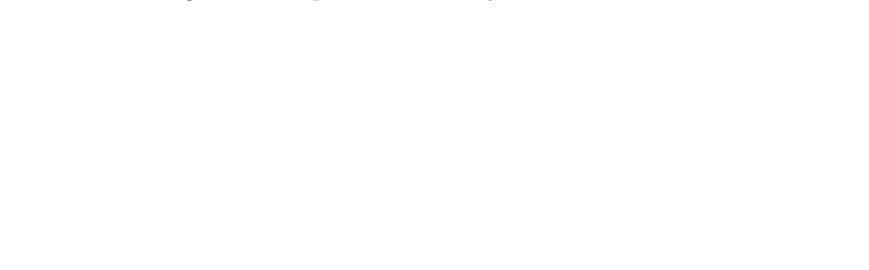
The documentation for this class was generated from the following files:

- [binary\\_lsb\\_point\\_vector\\_reader\\_context.hxx](#)
- [binary\\_lsb\\_point\\_vector\\_reader\\_context.cxx](#)

## **9.9 OpenGPS::BinaryLSBPointVectorWriterContext Class Reference**

```
#include <binary_lsb_point_vector_writer_context.hxx>
```

Inheritance diagram for OpenGPS::BinaryLSBPointVectorWriterContext:



Collaboration diagram for OpenGPS::BinaryLSBPointVectorWriterContext:



### **9.9.1 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 [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.*

### 9.9.2 Constructor & Destructor Documentation

#### 9.9.2.1 BinaryLSBPointVectorWriterContext::BinaryLSBPointVectorWriterContext (*zipFile handle*)

Creates a new instance.

##### Parameters:

*handle* The zip-stream where binary data is written to.

#### 9.9.2.2 BinaryLSBPointVectorWriterContext::~BinaryLSBPointVectorWriterContext () [virtual]

Destroys this instance.

### 9.9.3 Member Function Documentation

#### 9.9.3.1 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](#).

#### 9.9.3.2 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](#).

**9.9.3.3 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`.

**9.9.3.4 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`.

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`

## 9.10 OpenGPS::BinaryMSBPointVectorReaderContext Class Reference

```
#include <binary_msb_point_vector_reader_context.hxx>
```

Inheritance diagram for OpenGPS::BinaryMSBPointVectorReaderContext:

Collaboration diagram for OpenGPS::BinaryMSBPointVectorReaderContext:

### **9.10.1 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 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 `~BinaryMSBPointVectorReaderContext ()`  
*Destroys this instance.*

### 9.10.2 Constructor & Destructor Documentation

**9.10.2.1 BinaryMSBPointVectorReaderContext::BinaryMSBPointVectorReaderContext (const OpenGPS::String & filePath)**

Creates a new instance.

**Parameters:**

*filePath* Absolute path to the binary file streamed herein.

**9.10.2.2 BinaryMSBPointVectorReaderContext::~BinaryMSBPointVectorReaderContext () [virtual]**

Destroys this instance.

### 9.10.3 Member Function Documentation

**9.10.3.1 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`.

**9.10.3.2 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](#).

**9.10.3.3 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](#).

**9.10.3.4 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](#).

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](#)

## 9.11 OpenGPS::BinaryMSBPointVectorWriterContext Class Reference

```
#include <binary_msb_point_vector_writer_context.hxx>
```

Inheritance diagram for OpenGPS::BinaryMSBPointVectorWriterContext:



Collaboration diagram for OpenGPS::BinaryMSBPointVectorWriterContext:



### 9.11.1 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 [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.*

### 9.11.2 Constructor & Destructor Documentation

#### 9.11.2.1 BinaryMSBPointVectorWriterContext::BinaryMSBPointVectorWriterContext (*zipFile handle*)

Creates a new instance.

**Parameters:**

*handle* The zip-stream where binary data is written to.

#### 9.11.2.2 BinaryMSBPointVectorWriterContext::~BinaryMSBPointVectorWriterContext () [virtual]

Destroys this instance.

### 9.11.3 Member Function Documentation

#### 9.11.3.1 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](#).

#### 9.11.3.2 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](#).

## **9.12 OpenGPS::BinaryPointVectorReaderContext Class Reference**

**9.11.3.3 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`.

**9.11.3.4 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`.

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`

## **9.12 OpenGPS::BinaryPointVectorReaderContext Class Reference**

```
#include <binary_point_vector_reader_context.hxx>
```

Inheritance diagram for `OpenGPS::BinaryPointVectorReaderContext`:

## **9.12 OpenGPS::BinaryPointVectorReaderContext Class Reference**

Collaboration diagram for OpenGPS::BinaryPointVectorReaderContext:

### **9.12.1 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 [Skip \(\) throw \(...\)](#)  
*Skips reading of the currently underlying data.*

## **9.12 OpenGPS::BinaryPointVectorReaderContext Class Reference**

### **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.*

#### **9.12.2 Constructor & Destructor Documentation**

##### **9.12.2.1 BinaryPointVectorReaderContext::BinaryPointVectorReaderContext (const OpenGPS::String & filePath) [protected]**

Creates a new instance.

###### **Parameters:**

*filePath* Absolute path to the binary file streamed herein.

##### **9.12.2.2 BinaryPointVectorReaderContext::~BinaryPointVectorReaderContext () [protected, virtual]**

Destroys this instance.

## **9.12 OpenGPS::BinaryPointVectorReaderContext Class Reference**

### **9.12.3 Member Function Documentation**

**9.12.3.1 void BinaryPointVectorReaderContext::Close () [protected, virtual]**

Closes the internal handle to the binary file and releases its resources.

**9.12.3.2 InputBinaryFileStream \* BinaryPointVectorReaderContext::GetStream () [protected]**

Returns the underlying data stream for read access.

**9.12.3.3 OGPS\_Boolean BinaryPointVectorReaderContext::HasStream () const [protected]**

Returns TRUE if the underlying stream object had successfully been allocated, FALSE otherwise.

**9.12.3.4 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 occurred.

**9.12.3.5 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](#).

## **9.12 OpenGPS::BinaryPointVectorReaderContext Class Reference**

### **9.12.3.6 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](#).

### **9.12.3.7 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](#).

### **9.12.4 Member Data Documentation**

#### **9.12.4.1 InputBinaryFileStream\* OpenGPS::BinaryPointVectorReaderContext::m\_Stream [private]**

Pointer to the underlying data stream of binary point vectors.

The documentation for this class was generated from the following files:

- [binary\\_point\\_vector\\_reader\\_context.hxx](#)
- [binary\\_point\\_vector\\_reader\\_context.cxx](#)

## **9.13 OpenGPS::BinaryPointVectorWriterContext Class Reference**

### **9.13 OpenGPS::BinaryPointVectorWriterContext Class Reference**

```
#include <binary_point_vector_writer_context.hxx>
```

Inheritance diagram for OpenGPS::BinaryPointVectorWriterContext:

Collaboration diagram for OpenGPS::BinaryPointVectorWriterContext:

#### **9.13.1 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.*

## **9.13 OpenGPS::BinaryPointVectorWriterContext Class Reference**

### **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.*

### **9.13.2 Constructor & Destructor Documentation**

#### **9.13.2.1 BinaryPointVectorWriterContext::BinaryPointVectorWriterContext (zipFile handle)**

Creates a new instance.

##### **Parameters:**

*handle* The zip-stream where binary data is written to.

#### **9.13.2.2 BinaryPointVectorWriterContext::~BinaryPointVectorWriterContext () [protected, virtual]**

Destroys this instance.

### **9.13.3 Member Function Documentation**

#### **9.13.3.1 void BinaryPointVectorWriterContext::Close () [virtual]**

Closes the internal handle to the binary stream and frees its resources.

## **9.13 OpenGPS::BinaryPointVectorWriterContext Class Reference**

**9.13.3.2 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.

**9.13.3.3 std::ostream \* BinaryPointVectorWriterContext::GetStream () [protected]**

Gets the internal binary stream.

### **Returns:**

Returns the target binary stream or NULL.

**9.13.3.4 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](#)

**9.13.3.5 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 occurred.

Implements [OpenGPS::PointVectorWriterContext](#).

**9.13.3.6 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](#).

**9.13.3.7 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](#).

**9.13.4 Member Data Documentation**

**9.13.4.1 ZipStreamBuffer\* OpenGPS::BinaryPointVectorWriterContext::m\_Buffer [private]**

The target buffer where compressed binary data gets stored.

**9.13.4.2 ZipOutputStream\* OpenGPS::BinaryPointVectorWriterContext::m\_Stream [private]**

The stream buffer which makes [BinaryPointVectorWriterContext::m\\_Buffer](#) accessible through the more abstract std::ostream interface.

The documentation for this class was generated from the following files:

- [binary\\_point\\_vector\\_writer\\_context.hxx](#)
- [binary\\_point\\_vector\\_writer\\_context.cxx](#)

**9.14 OpenGPS::Schemas::ISO5436\_2::DataLinkType Class Reference**

#include <iso5436\_2\_xsd.hxx>

### 9.14.1 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 (MD5ChecksumPointData_type &x)` `(::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 (MD5ChecksumValidPoints_type &x)` `(::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\_

#### 9.14.2 Member Typedef Documentation

**9.14.2.1** `typedef ::xsd::cxx::tree::traits< MD5ChecksumPointData_type, wchar_t > OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData_traits`

Element traits type.

**9.14.2.2** `typedef ::xsd::cxx::tree::traits< MD5ChecksumPointData_type, wchar_t > OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData_traits`

Element traits type.

**9.14.2.3** `typedef ::xml_schema::hex_binary OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData_type`

Element type.

**9.14.2.4** `typedef ::xml_schema::hex_binary OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData_type`

Element type.

**9.14.2.5** `typedef ::xsd::cxx::tree::optional< MD5ChecksumValidPoints_type > OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints_optional`

Element optional container type.

**9.14.2.6** `typedef ::xsd::cxx::tree::optional< MD5ChecksumValidPoints_type > OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints_optional`

Element optional container type.

**9.14.2.7** `typedef ::xsd::cxx::tree::traits< MD5ChecksumValidPoints_-  
type, wchar_t > OpenGPS::Schemas::ISO5436_-  
2::DataLinkType::MD5ChecksumValidPoints_traits`

Element traits type.

**9.14.2.8** `typedef ::xsd::cxx::tree::traits< MD5ChecksumValidPoints_-  
type, wchar_t > OpenGPS::Schemas::ISO5436_-  
2::DataLinkType::MD5ChecksumValidPoints_traits`

Element traits type.

**9.14.2.9** `typedef ::xml_schema::hex_binary  
OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints_-  
type`

Element type.

**9.14.2.10** `typedef ::xml_schema::hex_binary  
OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumValidPoints_-  
type`

Element type.

**9.14.2.11** `typedef ::xsd::cxx::tree::traits< PointDataLink_-  
type, wchar_t > OpenGPS::Schemas::ISO5436_-  
2::DataLinkType::PointDataLink_traits`

Element traits type.

**9.14.2.12** `typedef ::xsd::cxx::tree::traits< PointDataLink_-  
type, wchar_t > OpenGPS::Schemas::ISO5436_-  
2::DataLinkType::PointDataLink_traits`

Element traits type.

**9.14.2.13** `typedef ::xml_schema::string OpenGPS::Schemas::ISO5436_-  
2::DataLinkType::PointDataLink_type`

Element type.

**9.14.2.14** `typedef ::xml_schema::string OpenGPS::Schemas::ISO5436_-  
2::DataLinkType::PointDataLink_type`

Element type.

**9.14.2.15** `typedef ::xsd::cxx::tree::optional<  
ValidPointsLink_type > OpenGPS::Schemas::ISO5436_-  
2::DataLinkType::ValidPointsLink_optional`

Element optional container type.

**9.14.2.16** `typedef ::xsd::cxx::tree::optional<ValidPointsLink_type> OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink_optional`

Element optional container type.

**9.14.2.17** `typedef ::xsd::cxx::tree::traits<ValidPointsLink_type, wchar_t> OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink_traits`

Element traits type.

**9.14.2.18** `typedef ::xsd::cxx::tree::traits<ValidPointsLink_type, wchar_t> OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink_traits`

Element traits type.

**9.14.2.19** `typedef ::xml_schema::string OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink_type`

Element type.

**9.14.2.20** `typedef ::xml_schema::string OpenGPS::Schemas::ISO5436_2::DataLinkType::ValidPointsLink_type`

Element type.

### 9.14.3 Constructor & Destructor Documentation

**9.14.3.1** `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.

**9.14.3.2** `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.

**9.14.3.3 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.

**9.14.3.4 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.

**9.14.3.5 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.

**9.14.3.6 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.

#### 9.14.4 Member Function Documentation

**9.14.4.1 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.

**9.14.4.2 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.

**9.14.4.3 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.

**9.14.4.4 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.

**9.14.4.5 MD5ChecksumPointData\_type&**  
**OpenGPS::Schemas::ISO5436\_-2::DataLinkType::MD5ChecksumPointData**  
**()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.14.4.6 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.

**9.14.4.7 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.

**9.14.4.8 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.

**9.14.4.9** `MD5ChecksumPointData_type&`  
`OpenGPS::Schemas::ISO5436_2::DataLinkType::MD5ChecksumPointData`  
(*)*

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.14.4.10** `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.

**9.14.4.11** `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.

**9.14.4.12** `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.

---

**9.14.4.13 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.

**9.14.4.14 MD5ChecksumValidPoints\_optional&**  
**OpenGPS::Schemas::ISO5436\_-2::DataLinkType::MD5ChecksumValidPoints**  
**()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.14.4.15 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.

**9.14.4.16 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.

**9.14.4.17 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.

**9.14.4.18 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.

**9.14.4.19 MD5ChecksumValidPoints\_optional&**  
**OpenGPS::Schemas::ISO5436\_-2::DataLinkType::MD5ChecksumValidPoints**  
**()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.14.4.20 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.

**9.14.4.21 void** **OpenGPS::Schemas::ISO5436\_-  
2::DataLinkType::parse** **(::xsd::cxx::xml::dom::parser< wchar\_t  
> &, ::xml\_schema::flags)** **[protected]**

**9.14.4.22 void** **OpenGPS::Schemas::ISO5436\_-  
2::DataLinkType::parse** **(::xsd::cxx::xml::dom::parser< wchar\_t  
> & *p*, ::xml\_schema::flags *f*)** **[protected]**

**9.14.4.23 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.

**9.14.4.24 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.

**9.14.4.25 PointDataLink\_type&** *OpenGPS::Schemas::ISO5436\_2::DataLinkType::PointDataLink* ()

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.14.4.26 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.

**9.14.4.27 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.

**9.14.4.28 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.

**9.14.4.29 PointDataLink\_type& OpenGPS::Schemas::ISO5436\_-  
2::DataLinkType::PointDataLink ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.14.4.30 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.

**9.14.4.31 void OpenGPS::Schemas::ISO5436\_-  
2::DataLinkType::ValidPointsLink (const 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.

**9.14.4.32 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.

**9.14.4.33 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.

**9.14.4.34 ValidPointsLink\_optional& OpenGPS::Schemas::ISO5436\_-  
2::DataLinkType::ValidPointsLink ()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.14.4.35 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.

**9.14.4.36 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.

**9.14.4.37 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.

**9.14.4.38 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.

**9.14.4.39 ValidPointsLink\_optional& OpenGPS::Schemas::ISO5436\_2::DataLinkType::ValidPointsLink ()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.14.4.40 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.

#### 9.14.5 Member Data Documentation

9.14.5.1 xsd::cxx::tree::one< type > MD5ChecksumPointData\_-  
 OpenGPS::Schemas::ISO5436\_-  
 2::DataLinkType::MD5ChecksumPointData\_- [private]

9.14.5.2 MD5ChecksumValidPoints\_ optional  
 OpenGPS::Schemas::ISO5436\_-2::DataLinkType::MD5ChecksumValidPoints\_-  
 [private]

9.14.5.3 xsd::cxx::tree::one< OpenGPS::Schemas::ISO5436\_-2::DataLinkType::PointDataLink\_->  
 [private]

9.14.5.4 ValidPointsLink\_ optional OpenGPS::Schemas::ISO5436\_-2::DataLinkType::ValidPointsLink\_- [private]

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

## 9.15 OpenGPS::Schemas::ISO5436\_2::DataListType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

### 9.15.1 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 neighbourhood. A list is always an 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_`

#### 9.15.2 Member Typedef Documentation

**9.15.2.1 typedef `Datum_sequence::const_iterator`**  
**OpenGPS::Schemas::ISO5436\_2::DataListType::Datum\_const\_-  
iterator**

Element constant iterator type.

**9.15.2.2 typedef `Datum_sequence::const_iterator`**  
**OpenGPS::Schemas::ISO5436\_2::DataListType::Datum\_const\_-  
iterator**

Element constant iterator type.

**9.15.2.3 typedef `Datum_sequence::iterator`**  
**OpenGPS::Schemas::ISO5436\_2::DataListType::Datum\_iterator**

Element iterator type.

**9.15.2.4 typedef `Datum_sequence::iterator`**  
**OpenGPS::Schemas::ISO5436\_2::DataListType::Datum\_iterator**

Element iterator type.

**9.15.2.5 typedef `::xsd::cxx::tree::sequence< Datum_type >`**  
**OpenGPS::Schemas::ISO5436\_2::DataListType::Datum\_sequence**

Element sequence container type.

**9.15.2.6 typedef `::xsd::cxx::tree::sequence< Datum_type >`**  
**OpenGPS::Schemas::ISO5436\_2::DataListType::Datum\_sequence**

Element sequence container type.

**9.15.2.7** `typedef ::xsd::cxx::tree::traits< Datum_type, wchar_t >`  
`OpenGPS::Schemas::ISO5436_2::DataListType::Datum_traits`

Element traits type.

**9.15.2.8** `typedef ::xsd::cxx::tree::traits< Datum_type, wchar_t >`  
`OpenGPS::Schemas::ISO5436_2::DataListType::Datum_traits`

Element traits type.

**9.15.2.9** `typedef ::OpenGPS::Schemas::ISO5436_2::Datum`  
`OpenGPS::Schemas::ISO5436_2::DataListType::Datum_type`

Element type.

**9.15.2.10** `typedef ::OpenGPS::Schemas::ISO5436_2::Datum`  
`OpenGPS::Schemas::ISO5436_2::DataListType::Datum_type`

Element type.

### 9.15.3 Constructor & Destructor Documentation

**9.15.3.1** `OpenGPS::Schemas::ISO5436_2::DataListType::DataListType()`

Construct an instance from the ultimate base and initializers for required elements and attributes.

**9.15.3.2** `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.

**9.15.3.3** `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.

**9.15.3.4 OpenGPS::Schemas::ISO5436\_2::DataListType::DataListType()**

Construct an instance from the ultimate base and initializers for required elements and attributes.

**9.15.3.5 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.

**9.15.3.6 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.

**9.15.4 Member Function Documentation**

**9.15.4.1 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.

**9.15.4.2 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.

**9.15.4.3 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.

**9.15.4.4 Datum\_sequence& OpenGPS::Schemas::ISO5436\_2::DataListType::Datum ()**

Return a read-write reference to the element sequence.

**Returns:**

A reference to the sequence container.

**9.15.4.5 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.

**9.15.4.6 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.

**9.15.4.7 Datum\_sequence& OpenGPS::Schemas::ISO5436\_2::DataListType::Datum ()**

Return a read-write reference to the element sequence.

**Returns:**

A reference to the sequence container.

**9.15.4.8 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.

**9.15.4.9 void OpenGPS::Schemas::ISO5436\_2::DataListType::parse (::xsd::cxx::xml::dom::parser< wchar\_t > &, ::xml\_schema::flags) [protected]**

**9.15.4.10 void OpenGPS::Schemas::ISO5436\_2::DataListType::parse (::xsd::cxx::xml::dom::parser< wchar\_t > & p, ::xml\_schema::flags f) [protected]**

### 9.15.5 Member Data Documentation

#### 9.15.5.1 Datum\_sequence OpenGPS::Schemas::ISO5436\_-2::DataListType::Datum\_ [private]

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

## 9.16 OpenGPS::DataPoint Class Reference

```
#include <data_point.hxx>
```

Inheritance diagram for OpenGPS::DataPoint:

### 9.16.1 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 &zsrc)=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.*

### 9.16.2 Constructor & Destructor Documentation

#### 9.16.2.1 DataPoint::DataPoint () [protected, pure virtual]

Creates and initializes a new object.

#### 9.16.2.2 DataPoint::~DataPoint () [pure virtual]

Destructs this object.

#### 9.16.2.3 OpenGPS::DataPoint::DataPoint (const DataPoint & src) [private]

The copy-ctor is not implemented.

This prevents its usage.

### 9.16.3 Member Function Documentation

#### 9.16.3.1 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](#).

#### 9.16.3.2 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](#).

**9.16.3.3 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](#).

**9.16.3.4 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](#).

**9.16.3.5 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](#).

**9.16.3.6 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](#).

**9.16.3.7 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](#).

**9.16.3.8 DataPoint& OpenGPS::DataPoint::operator= (const DataPoint & *src*) [private]**

The assignment-operator is not implemented.

This prevents its usage.

**9.16.3.9 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](#).

**9.16.3.10 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](#).

**9.16.3.11 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](#).

**9.16.3.12 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](#).

**9.16.3.13 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](#).

**9.16.3.14 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](#),  
[OpenGPS::PointVectorProxy::DataPointProxy](#).

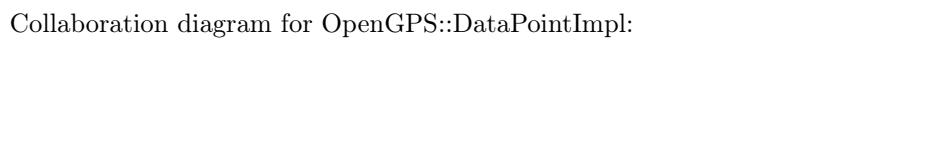
The documentation for this class was generated from the following files:

- [data\\_point.hxx](#)
- [data\\_point\\_impl.cxx](#)

## 9.17 OpenGPS::DataPointImpl Class Reference

```
#include <data_point_impl.hxx>
```

Inheritance diagram for OpenGPS::DataPointImpl:



```
graph TD; DataPointImpl --> DataPoint; DataPointImpl --> PointVectorProxy
```

Collaboration diagram for OpenGPS::DataPointImpl:



```
graph TD; DataPointImpl --> DataPoint; DataPointImpl --> PointVectorProxy
```

### 9.17.1 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 Types

- [typedef union OpenGPS::DataPointImpl::\\_OGPS\\_DATA\\_POINT\\_-  
VALUE OGPS\\_DataPointValue](#)

### 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.*

### Classes

- [union \\_OGPS\\_DATA\\_POINT\\_VALUE](#)  
*Typesafe storage for every possible type of point data.*

### 9.17.2 Member Typedef Documentation

**9.17.2.1 [typedef union OpenGPS::DataPointImpl::\\_OGPS\\_-  
DATA\\_POINT\\_-VALUE OpenGPS::DataPointImpl::OGPS\\_-  
DataPointValue \[private\]](#)**

### 9.17.3 Constructor & Destructor Documentation

#### 9.17.3.1 [DataPointImpl::DataPointImpl \(\)](#)

Creates a new instance.

#### 9.17.3.2 [DataPointImpl::~DataPointImpl \(\) \[virtual\]](#)

Destroys this instance.

#### 9.17.4 Member Function Documentation

##### 9.17.4.1 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](#).

##### 9.17.4.2 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](#).

##### 9.17.4.3 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](#).

**9.17.4.4 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](#).

**9.17.4.5 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](#).

**9.17.4.6 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](#).

**9.17.4.7 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](#).

**9.17.4.8 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](#).

**9.17.4.9 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](#).

**9.17.4.10 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](#).

**9.17.4.11 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](#).

**9.17.4.12 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](#).

**9.17.4.13 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](#).

**9.17.5 Member Data Documentation****9.17.5.1 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](#).

**9.17.5.2 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](#)

The documentation for this class was generated from the following files:

- [data\\_point\\_impl.hxx](#)
- [data\\_point\\_impl.cxx](#)

**9.18 OpenGPS::DataPointImpl::\_OGPS\_DATA\_POINT\_VALUE Union Reference****9.18.1 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.*

**9.18.2 Member Data Documentation****9.18.2.1 OGPS\_Double OpenGPS::DataPointImpl::\_OGPS\_DATA\_POINT\_VALUE::doubleValue**

Stores a value of type OGPS\_Double.

It is undefined if DataPointImpl::m\_Type does not equal OGPS\_DoublePointType.

**9.18.2.2 OGPS\_Float OpenGPS::DataPointImpl::\_OGPS\_DATA\_POINT\_VALUE::floatValue**

Stores a value of type OGPS\_Float.

It is undefined if DataPointImpl::m\_Type does not equal OGPS\_FloatPointType.

**9.18.2.3 OGPS\_Int16 OpenGPS::DataPointImpl::\_OGPS\_DATA\_POINT\_VALUE::int16Value**

Stores a value of type OGPS\_Int16.

It is undefined if DataPointImpl::m\_Type does not equal OGPS\_Int16PointType.

**9.18.2.4 OGPS\_Int32 OpenGPS::DataPointImpl::\_OGPS\_DATA\_POINT\_VALUE::int32Value**

Stores a value of type OGPS\_Int32.

It is undefined if DataPointImpl::m\_Type does not equal OGPS\_Int32PointType.

The documentation for this union was generated from the following file:

- [data\\_point\\_impl.hxx](#)

## 9.19 OpenGPS::DataPointParser Class Reference

```
#include <data_point_parser.hxx>
```

Inheritance diagram for OpenGPS::DataPointParser:

### 9.19.1 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.*

### 9.19.2 Constructor & Destructor Documentation

#### 9.19.2.1 DataPointParser::~DataPointParser () [virtual]

Destroys this instance.

**9.19.2.2 DataPointParser::DataPointParser () [protected]**

Creates a new instance.

**9.19.3 Member Function Documentation****9.19.3.1 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](#).

**9.19.3.2 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 exactly 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](#).

The documentation for this class was generated from the following files:

## **9.20 OpenGPS::Schemas::ISO5436\_2::DataType Class Reference**

---

- [data\\_point\\_parser.hxx](#)
- [int16\\_data\\_point\\_parser.cxx](#)

### **9.20 OpenGPS::Schemas::ISO5436\_2::DataType Class Reference**

```
#include <iso5436_2_xsd.hxx>
```

#### **9.20.1 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)

## **9.20 OpenGPS::Schemas::ISO5436\_2::DataType Class Reference**

*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.*

## **9.20 OpenGPS::Schemas::ISO5436\_2::DataType Class Reference**

---

### **Static Public Attributes**

- static const `value _xsd(DataType_indexes_ [4]`
- static const `wchar_t *const _xsd(DataType_literals_ [4]`
- static const `wchar_t *const _xsd(DataType_literals_ [4]`

### **Protected Member Functions**

- `value _xsd(DataType_convert () const`
- `value _xsd(DataType_convert () const`

#### **9.20.2 Member Enumeration Documentation**

##### **9.20.2.1 enum OpenGPS::Schemas::ISO5436\_2::DataType::value**

Underlying enum type.

**Enumerator:**

*I*

*L*

*F*

*D*

*I*

*L*

*F*

*D*

##### **9.20.2.2 enum OpenGPS::Schemas::ISO5436\_2::DataType::value**

Underlying enum type.

**Enumerator:**

*I*

*L*

*F*

*D*

*I*

*L*

*F*

*D*

## **9.20 OpenGPS::Schemas::ISO5436\_2::DataType Class Reference**

---

### **9.20.3 Constructor & Destructor Documentation**

#### **9.20.3.1 OpenGPS::Schemas::ISO5436\_2::DataType::DataType (value *v*)**

Construct an instance from the underlying enum value.

##### **Parameters:**

*v* A enum value.

#### **9.20.3.2 OpenGPS::Schemas::ISO5436\_2::DataType::DataType (const ::xml\_schema::token & *v*)**

Construct an instance from the base value.

##### **Parameters:**

*v* A base value.

#### **9.20.3.3 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.

#### **9.20.3.4 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.

## 9.20 OpenGPS::Schemas::ISO5436\_2::DataType Class Reference

**9.20.3.5 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.

**9.20.3.6 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.

**9.20.3.7 OpenGPS::Schemas::ISO5436\_2::DataType::DataType**  
(value *v*)

Construct an instance from the underlying enum value.

### **Parameters:**

- v* A enum value.

**9.20.3.8 OpenGPS::Schemas::ISO5436\_2::DataType::DataType**  
(const ::xml\_schema::token & *v*)

Construct an instance from the base value.

### **Parameters:**

- v* A base value.

## **9.20 OpenGPS::Schemas::ISO5436\_2::DataType Class Reference**

---

**9.20.3.9 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.

**9.20.3.10 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.

**9.20.3.11 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.

**9.20.3.12 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.

## **9.20 OpenGPS::Schemas::ISO5436\_2::DataType Class Reference**

---

### **9.20.4 Member Function Documentation**

**9.20.4.1 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.

**9.20.4.2      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.

**9.20.4.3      value      OpenGPS::Schemas::ISO5436\_2::DataType::\_xsd(DataType\_convert) const [protected]**

**9.20.4.4      DataType::value                OpenGPS::Schemas::ISO5436\_2::DataType::\_xsd(DataType\_convert) const [protected]**

**9.20.4.5      virtual                        OpenGPS::Schemas::ISO5436\_2::DataType::operator value () const [inline, virtual]**

Implicit conversion operator to the underlying enum value.

## **9.20 OpenGPS::Schemas::ISO5436\_2::DataType Class Reference**

---

### **Returns:**

A enum value.

**9.20.4.6 virtual OpenGPS::Schemas::ISO5436\_-  
2::DataType::operator value () const [inline, virtual]**

Implicit conversion operator to the underlying enum value.

### **Returns:**

A enum value.

**9.20.4.7 DataType& OpenGPS::Schemas::ISO5436\_-  
2::DataType::operator= (value v)**

Assign the underlying enum value.

### **Parameters:**

*v* A enum value.

### **Returns:**

A reference to the instance.

**9.20.4.8 DataType& OpenGPS::Schemas::ISO5436\_-  
2::DataType::operator= (value v)**

Assign the underlying enum value.

### **Parameters:**

*v* A enum value.

### **Returns:**

A reference to the instance.

## **9.20.5 Member Data Documentation**

**9.20.5.1 static const value OpenGPS::Schemas::ISO5436\_-  
2::DataType::\_xsd\_DataType\_indexes\_ [static]**

**9.20.5.2 const wchar\_t\* const OpenGPS::Schemas::ISO5436\_-  
2::DataType::\_xsd\_DataType\_literals\_[4] [static]**

**9.20.5.3 const wchar\_t\* const OpenGPS::Schemas::ISO5436\_2::DataType::\_xsd\_DataType\_literals\_[4] [static]**

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

**9.21 OpenGPS::Schemas::ISO5436\_2::Datum Class Reference**

```
#include <iso5436_2_xsd.hxx>
```

**9.21.1 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

- **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.*

### 9.21.2 Constructor & Destructor Documentation

#### 9.21.2.1 OpenGPS::Schemas::ISO5436\_2::Datum::Datum()

Construct an instance from initializers for required elements and attributes.

#### 9.21.2.2 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.

#### 9.21.2.3 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.

**9.21.2.4 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.

**9.21.2.5 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.

**9.21.2.6 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.

**9.21.2.7 OpenGPS::Schemas::ISO5436\_2::Datum::Datum ()**

Construct an instance from initializers for required elements and attributes.

**9.21.2.8 OpenGPS::Schemas::ISO5436\_2::Datum::Datum (const ::xml\_schema::token &)**

Construct an instance from the ultimate base and initializers for required elements and attributes.

**9.21.2.9 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.

**9.21.2.10 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.

**9.21.2.11 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.

**9.21.2.12 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.

**9.21.3 Member Function Documentation**

**9.21.3.1 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.

**9.21.3.2 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.

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

## 9.22 OpenGPS::DoubleDataPointParser Class Reference

```
#include <double_data_point_parser.hxx>
```

Inheritance diagram for OpenGPS::DoubleDataPointParser:

Collaboration diagram for OpenGPS::DoubleDataPointParser:

### 9.22.1 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.*

### 9.22.2 Constructor & Destructor Documentation

#### 9.22.2.1 DoubleDataPointParser::DoubleDataPointParser ()

Creates a new instance.

#### 9.22.2.2 DoubleDataPointParser::~DoubleDataPointParser () [virtual]

Destroys this instance.

### 9.22.3 Member Function Documentation

#### 9.22.3.1 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`.

#### 9.22.3.2 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 exactly 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](#).

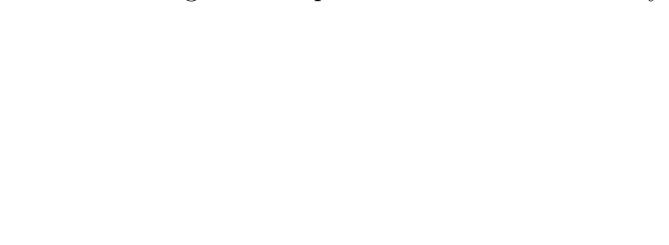
The documentation for this class was generated from the following files:

- [double\\_data\\_point\\_parser.hxx](#)
- [double\\_data\\_point\\_parser.cxx](#)

## 9.23 OpenGPS::DoubleInlineValidity Class Reference

```
#include <inline_validity.hxx>
```

Inheritance diagram for OpenGPS::DoubleInlineValidity:



```
graph TD; DoubleInlineValidity --> PointValidityProvider
```

Collaboration diagram for OpenGPS::DoubleInlineValidity:

### 9.23.1 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.*

### 9.23.2 Constructor & Destructor Documentation

#### 9.23.2.1 DoubleInlineValidity::DoubleInlineValidity (`PointBuffer *const value`)

Creates a new instance.

##### Parameters:

*value* The point buffer of the Z axis.

#### 9.23.2.2 DoubleInlineValidity::~DoubleInlineValidity ()

Destroys this instance.

### 9.23.3 Member Function Documentation

#### 9.23.3.1 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`.

**9.23.3.2 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](#).

The documentation for this class was generated from the following files:

- [inline\\_validity.hxx](#)
- [inline\\_validity.cxx](#)

## 9.24 OpenGPS::DoublePointBuffer Class Reference

```
#include <double_point_buffer.hxx>
```

Inheritance diagram for OpenGPS::DoublePointBuffer:

Collaboration diagram for OpenGPS::DoublePointBuffer:

### 9.24.1 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 unsigned long size) throw (...)

*Allocates internal memory.*

- **DoublePointBuffer ()**  
*Create a new instance.*
- virtual void **Get** (const unsigned long index, OGPS\_Double &value) const  
throw (...)  
*Gets the value of the internal memory at the given position.*
- virtual OGPS\_DataPointType **GetPointType ()** const  
*Gets the type of point data that can be stored within this instance.*
- virtual void **Set** (const unsigned long index, const OGPS\_Double value)  
throw (...)  
*Sets the value of the internal memory at the given position.*
- virtual ~**DoublePointBuffer ()**  
*Destroys this instance.*

### Private Attributes

- OGPS\_Double \* **m\_Buffer**  
*Pointer to internal memory.*

### 9.24.2 Constructor & Destructor Documentation

#### 9.24.2.1 DoublePointBuffer::DoublePointBuffer ()

Create a new instance.

#### 9.24.2.2 DoublePointBuffer::~DoublePointBuffer () [virtual]

Destroys this instance.

### 9.24.3 Member Function Documentation

#### 9.24.3.1 void DoublePointBuffer::Allocate (const unsigned long *size*) throw (...) [virtual]

Allocates internal memory.

Throws an exception on failure.

#### Parameters:

*size* Amount of point data to be stored.

Reimplemented from [OpenGPS::PointBuffer](#).

**9.24.3.2 void DoublePointBuffer::Get (const unsigned long *index*, OGPS\_Double & *value*) const throw (...) [virtual]**

Gets the value of the internal 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](#).

**9.24.3.3 OGPS\_DataPointType DoublePointBuffer::GetPointType () const [virtual]**

Gets the type of point data that can be stored within this instance.

Reimplemented from [OpenGPS::PointBuffer](#).

**9.24.3.4 void DoublePointBuffer::Set (const unsigned long *index*, const OGPS\_Double *value*) throw (...) [virtual]**

Sets the value of the internal 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](#).

#### 9.24.4 Member Data Documentation

**9.24.4.1 OGPS\_Double\* OpenGPS::DoublePointBuffer::m\_Buffer [private]**

Pointer to internal memory.

The documentation for this class was generated from the following files:

- [double\\_point\\_buffer.hxx](#)
- [double\\_point\\_buffer.cxx](#)

## 9.25 OpenGPS::Environment Class Reference

```
#include <environment.hxx>
```

Collaboration diagram for OpenGPS::Environment:

### 9.25.1 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.*

## 9.25.2 Constructor & Destructor Documentation

### 9.25.2.1 Environment::Environment () [protected]

Creates a new instance.

### 9.25.2.2 Environment::~Environment () [protected, virtual]

Destroys this instance.

## 9.25.3 Member Function Documentation

### 9.25.3.1 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.

**9.25.3.2 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.

**9.25.3.3 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.

**9.25.3.4 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.

**9.25.3.5 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.

**9.25.3.6 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.

**9.25.3.7 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.

**9.25.3.8 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.

**9.25.3.9 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.

**9.25.3.10 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.

**9.25.3.11 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.

**9.25.3.12 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.

**9.25.3.13 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.

**9.25.3.14 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.

**9.25.3.15 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.

**9.25.3.16 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.

**9.25.3.17 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.

**9.25.3.18 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.

**9.25.3.19 static Environment\* OpenGPS::Environment::CreateInstance  
() [static, protected]**

Creates the appropriate instance for the current system.

**9.25.3.20 virtual OGPS\_Character OpenGPS::Environment::GetAltDirectorySeparator  
() const [pure virtual]**

E.g.

gets the forward slash on Microsoft Windows systems.

**9.25.3.21 virtual OGPS\_Character OpenGPS::Environment::GetDirectorySeparator  
() const [pure virtual]**

E.g.

gets the backslash on Microsoft Windows systems.

**9.25.3.22 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.

**9.25.3.23 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.

**9.25.3.24 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.

**9.25.3.25 virtual OpenGPS::String OpenGPS::Environment::GetTempDir () const [pure virtual]**

Returns the path to the temporary directory or an empty string on error.

**9.25.3.26 virtual OpenGPS::String OpenGPS::Environment::GetUniqueName () const [pure virtual]**

Gets a random sequence of numeric characters.

**9.25.3.27 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.

**9.25.3.28 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.

**9.25.3.29 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.

**9.25.3.30 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.

**9.25.3.31 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.

**9.25.3.32 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.

**9.25.3.33 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](#)

**9.25.4 Member Data Documentation**

**9.25.4.1 Environment \* Environment::m\_Instance = NULL  
[static, private]**

Pointer to the environment.

The documentation for this class was generated from the following files:

- [environment.hxx](#)
- [environment.cxx](#)

**9.26 OpenGPS::Exception Class Reference**

#include <exceptions.hxx>

Inheritance diagram for OpenGPS::Exception:

Collaboration diagram for OpenGPS::Exception:

### 9.26.1 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 exception.*
- const `OpenGPS::String & method () const throw ()`  
*Gets the name of the method where the exception occurred 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 occurred.*

### 9.26.2 Constructor & Destructor Documentation

**9.26.2.1 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 occurred.  
May be set to NULL.

**9.26.2.2 Exception::Exception (const Exception & *rhs*) throw ()**

Creates a new instance.

#### Parameters:

- rhs* The instance to copy from.

**9.26.2.3 Exception::~Exception () throw () [virtual]**

Destroys this instance.

### 9.26.3 Member Function Documentation

**9.26.3.1 const OpenGPS::String & Exception::details () const throw ()**

Gets a detailed description possible with hints to its avoidance.

**9.26.3.2 OGPS\_ExceptionId Exception::id () const throw ()**

Gets an identifier for the current type of exception.

**9.26.3.3 const OpenGPS::String & Exception::method () const throw ()**

Gets the name of the method where the exception occured or NULL.

#### 9.26.4 Member Data Documentation

**9.26.4.1 OpenGPS::String OpenGPS::Exception::m\_Details [private]**

Detailed description of the failure.

**9.26.4.2 OGPS\_ExceptionId OpenGPS::Exception::m\_Id [private]**

Describes the type of failure.

**9.26.4.3 OpenGPS::String OpenGPS::Exception::m\_Method [private]**

Name of the method wherein the failure occurred.

The documentation for this class was generated from the following files:

- [exceptions.hxx](#)
- [exceptions.cxx](#)

## 9.27 OpenGPS::ExceptionHistory Class Reference

```
#include <messages_c.hxx>
```

Collaboration diagram for OpenGPS::ExceptionHistory:

### 9.27.1 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 occurred.*
- static void `SetLastException (const std::exception &ex)`  
*Sets an exception object as the last one that occurred during execution.*
- static void `SetLastException (const OpenGPS::Exception &ex)`  
*Sets an exception object as the last one that occurred 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.*

### 9.27.2 Constructor & Destructor Documentation

**9.27.2.1 OpenGPS::ExceptionHistory::ExceptionHistory ()**  
[private]

Creates a new instance.

**9.27.2.2 OpenGPS::ExceptionHistory::~ExceptionHistory ()**  
[private]

Destroys this instance.

### 9.27.3 Member Function Documentation

**9.27.3.1 void OpenGPS::ExceptionHistory::DumpIt () [static,**  
[private]

Dumps a message to the error console.

This works in \_DEBUG only.

**9.27.3.2 const OGPS\_Character \* OpenGPS::ExceptionHistory::GetLastErrorDescription () [static]**

Gets the detailed failure description of the last known exception or NULL.

**9.27.3.3 OGPS\_ExceptionId OpenGPS::ExceptionHistory::GetLastErrorCode () [static]**

Gets the classifier of the last known exception or OGPS\_ExNone if none is maintained so far.

**9.27.3.4 const OGPS\_Character \* OpenGPS::ExceptionHistory::GetLastErrorMessage () [static]**

Gets the brief failure description of the last known exception or NULL.

**9.27.3.5 void OpenGPS::ExceptionHistory::Reset () [static]**

Reset the history of exceptions.

**9.27.3.6 void OpenGPS::ExceptionHistory::SetLastException () [static]**

Notify that an unknown type of exception occured.

**9.27.3.7 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.

**9.27.3.8 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.

#### 9.27.4 Member Data Documentation

**9.27.4.1 OpenGPS::String OpenGPS::ExceptionHistory::m\_-LastErrorDescription [static, private]**

The detailed description of the last known failure condition or empty.

**9.27.4.2 OGPS\_ExceptionId OpenGPS::ExceptionHistory::m\_-LastErrorId = OGPS\_ExNone [static, private]**

The classifier of the last known failure condition or [OGPS\\_ExNone](#).

**9.27.4.3 OpenGPS::String OpenGPS::ExceptionHistory::m\_-LastErrorMessage [static, private]**

The brief description of the last known failure condition or empty.

**9.27.4.4 OpenGPS::String OpenGPS::ExceptionHistory::m\_-LastErrorSource [static, private]**

Source of the last error condition.

The documentation for this class was generated from the following files:

- [messages\\_c.hxx](#)
- [messages\\_c.hxx](#)

## **9.28 OpenGPS::Schemas::ISO5436\_2::FeatureType Class Reference**

```
#include <iso5436_2_xsd.hxx>
```

### **9.28.1 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.*

### 9.28.2 Constructor & Destructor Documentation

#### 9.28.2.1 OpenGPS::Schemas::ISO5436\_2::FeatureType::FeatureType()

Construct an instance from initializers for required elements and attributes.

#### 9.28.2.2 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.

#### 9.28.2.3 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.

**9.28.2.4 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.

**9.28.2.5 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.

**9.28.2.6 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.

**9.28.2.7 OpenGPS::Schemas::ISO5436\_2::FeatureType::FeatureType  
()**

Construct an instance from initializers for required elements and attributes.

**9.28.2.8 OpenGPS::Schemas::ISO5436\_2::FeatureType::FeatureType  
(const ::xml\_schema::token &)**

Construct an instance from the ultimate base and initializers for required elements and attributes.

**9.28.2.9 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.

**9.28.2.10 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.

**9.28.2.11 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.

**9.28.2.12 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.

### 9.28.3 Member Function Documentation

**9.28.3.1 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.

**9.28.3.2 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.

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

## 9.29 OpenGPS::FloatDataPointParser Class Reference

```
#include <float_data_point_parser.hxx>
```

Inheritance diagram for OpenGPS::FloatDataPointParser:



```
graph TD; A[OpenGPS::FloatDataPointParser] --> B[OpenGPS::DataPointParser]; A --> C[OpenGPS::VectorDataPointParser]; A --> D[OpenGPS::TextDataPointParser]; A --> E[OpenGPS::BinaryDataPointParser]; A --> F[OpenGPS::ImageDataPointParser]; A --> G[OpenGPS::TextDataPointParser]; A --> H[OpenGPS::BinaryDataPointParser]; A --> I[OpenGPS::ImageDataPointParser];
```

Collaboration diagram for OpenGPS::FloatDataPointParser:



```
graph TD; A[OpenGPS::DataPointParser] --- B[PointVectorReaderContext]; A --- C[DataPoint]; A --- D[PointVectorWriterContext]; A --- E[DataPoint]; A --- F[DataPoint]; A --- G[DataPoint]; A --- H[DataPoint]; A --- I[DataPoint];
```

### 9.29.1 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.*

### 9.29.2 Constructor & Destructor Documentation

#### 9.29.2.1 [FloatDataPointParser::FloatDataPointParser \(\)](#)

Creates a new instance.

#### 9.29.2.2 [FloatDataPointParser::~FloatDataPointParser \(\)](#) [virtual]

Destroys this instance.

### 9.29.3 Member Function Documentation

#### 9.29.3.1 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](#).

#### 9.29.3.2 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 exactly 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](#)

## 9.30 OpenGPS::FloatInlineValidity Class Reference

```
#include <inline_validity.hxx>
```

Inheritance diagram for OpenGPS::FloatInlineValidity:

Collaboration diagram for OpenGPS::FloatInlineValidity:

### 9.30.1 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.*

### 9.30.2 Constructor & Destructor Documentation

#### 9.30.2.1 `FloatInlineValidity::FloatInlineValidity (PointBuffer *const value)`

Creates a new instance.

##### Parameters:

*value* The point buffer of the Z axis.

#### 9.30.2.2 `FloatInlineValidity::~FloatInlineValidity ()`

Destroys this instance.

### 9.30.3 Member Function Documentation

#### 9.30.3.1 `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](#).

#### 9.30.3.2 `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](#).

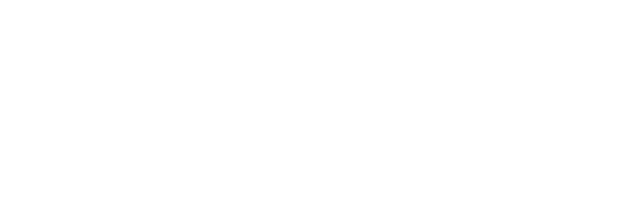
The documentation for this class was generated from the following files:

- [inline\\_validity.hxx](#)
- [inline\\_validity.cxx](#)

## 9.31 OpenGPS::FloatPointBuffer Class Reference

```
#include <float_point_buffer.hxx>
```

Inheritance diagram for OpenGPS::FloatPointBuffer:



Collaboration diagram for OpenGPS::FloatPointBuffer:



### 9.31.1 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 unsigned long size) throw (...)  
*Allocates internal memory.*
- [FloatPointBuffer](#) ()  
*Create a new instance.*
- virtual void [Get](#) (const unsigned long index, [OGPS\\_Float](#) &value) const  
throw (...)  
*Gets the value of the internal memory at the given position.*
- virtual [OGPS\\_DataPointType](#) [GetPointType](#) () const  
*Gets the type of point data that can be stored within this instance.*
- virtual void [Set](#) (const unsigned long index, const [OGPS\\_Float](#) value)  
throw (...)

*Sets the value of the internal memory at the given position.*

- virtual [~FloatPointBuffer \(\)](#)

*Destroys this instance.*

### Private Attributes

- [OGPS\\_Float \\* m\\_Buffer](#)

*Pointer to internal memory.*

### 9.31.2 Constructor & Destructor Documentation

#### 9.31.2.1 [FloatPointBuffer::FloatPointBuffer \(\)](#)

Create a new instance.

#### 9.31.2.2 [FloatPointBuffer::~FloatPointBuffer \(\) \[virtual\]](#)

Destroys this instance.

### 9.31.3 Member Function Documentation

#### 9.31.3.1 [void FloatPointBuffer::Allocate \(const unsigned long size\) throw \(...\) \[virtual\]](#)

Allocates internal memory.

Throws an exception on failure.

##### Parameters:

*size* Amount of point data to be stored.

Reimplemented from [OpenGPS::PointBuffer](#).

#### 9.31.3.2 [void FloatPointBuffer::Get \(const unsigned long index, OGPS\\_Float & value\) const throw \(...\) \[virtual\]](#)

Gets the value of the internal 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](#).

**9.31.3.3 OGPS\_DataPointType FloatPointBuffer::GetPointType ()  
const [virtual]**

Gets the type of point data that can be stored within this instance.

Reimplemented from [OpenGPS::PointBuffer](#).

**9.31.3.4 void FloatPointBuffer::Set (const unsigned long *index*,  
const OGPS\_Float *value*) throw (...) [virtual]**

Sets the value of the internal 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](#).

**9.31.4 Member Data Documentation****9.31.4.1 OGPS\_Float\* OpenGPS::FloatPointBuffer::m\_Buffer  
[private]**

Pointer to internal memory.

The documentation for this class was generated from the following files:

- [float\\_point\\_buffer.hxx](#)
- [float\\_point\\_buffer.cxx](#)

**9.32 OpenGPS::Info Class Reference**

```
#include <info.hxx>
```

**9.32.1 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](#) ()

### 9.32.2 Constructor & Destructor Documentation

#### 9.32.2.1 [Info::Info](#) () [private]

#### 9.32.2.2 [Info::~Info](#) () [private]

### 9.32.3 Member Function Documentation

#### 9.32.3.1 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.

**9.32.3.2 void Info::GetCopyright (OpenGPS::String \*const *text*) [static]**

Gets the copyright information of this software library.

**Parameters:**

*text* Contains the resulting message.

**9.32.3.3 void Info::GetLicense (OpenGPS::String \*const *text*) [static]**

Gets the license information of this software library.

**Parameters:**

*text* Contains the resulting message.

**9.32.3.4 void Info::GetName (OpenGPS::String \*const *text*) [static]**

Gets a short name identifying this software library.

**Parameters:**

*text* Contains the resulting message.

**9.32.3.5 void Info::GetVersion (OpenGPS::String \*const *text*) [static]**

Gets the version identifier of this software library.

**Parameters:**

*text* Contains the resulting message.

**9.32.3.6 void Info::PrintCopyright () [static]**

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

**9.32.3.7 void Info::PrintLicense () [static]**

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

**9.32.3.8 void Info::PrintVersion () [static]**

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

The documentation for this class was generated from the following files:

- [info.hxx](#)
- [info.cxx](#)

### **9.33 OpenGPS::InputBinaryFileStream Class Reference**

```
#include <point_vector_iostream.hxx>
```

Inheritance diagram for OpenGPS::InputBinaryFileStream:

Collaboration diagram for OpenGPS::InputBinaryFileStream:

### 9.33.1 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.*

### 9.33.2 Member Typedef Documentation

#### 9.33.2.1 [typedef std::basic\\_ifstream<OpenGPS::UnsignedByte> OpenGPS::InputBinaryFileStream::BaseType \[private\]](#)

The type of the super class.

### 9.33.3 Constructor & Destructor Documentation

#### 9.33.3.1 [InputBinaryFileStream::InputBinaryFileStream \(const OpenGPS::String & filePath\)](#)

Creates a new instance.

##### Parameters:

**filePath** Full path to the binary file to be read.

#### 9.33.3.2 [InputBinaryFileStream::~InputBinaryFileStream \(\)](#)

Destroys this instance.

### 9.33.4 Member Data Documentation

#### 9.33.4.1 PointVectorInvariantLocale OpenGPS::InputBinaryFileStream::m\_- Locale [private]

Invariant locale.

The documentation for this class was generated from the following files:

- [point\\_vector\\_iostream.hxx](#)
- [point\\_vector\\_iostream.cxx](#)

## 9.34 OpenGPS::Schemas::ISO5436\_2::InstrumentType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

### 9.34.1 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\_

### 9.34.2 Member Typedef Documentation

**9.34.2.1 `typedef ::xsd::cxx::tree::traits< Manufacturer_type, wchar_t > OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer_traits`**

Element traits type.

**9.34.2.2 `typedef ::xsd::cxx::tree::traits< Manufacturer_type, wchar_t > OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer_traits`**

Element traits type.

**9.34.2.3 `typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer_type`**

Element type.

**9.34.2.4 `typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_2::InstrumentType::Manufacturer_type`**

Element type.

**9.34.2.5 `typedef ::xsd::cxx::tree::traits< Model_type, wchar_t > OpenGPS::Schemas::ISO5436_2::InstrumentType::Model_traits`**

Element traits type.

**9.34.2.6 `typedef ::xsd::cxx::tree::traits< Model_type, wchar_t > OpenGPS::Schemas::ISO5436_2::InstrumentType::Model_traits`**

Element traits type.

**9.34.2.7 `typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_2::InstrumentType::Model_type`**

Element type.

**9.34.2.8 `typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_2::InstrumentType::Model_type`**

Element type.

**9.34.2.9 `typedef ::xsd::cxx::tree::traits< Serial_type, wchar_t > OpenGPS::Schemas::ISO5436_2::InstrumentType::Serial_traits`**

Element traits type.

---

**9.34.2.10 `typedef ::xsd::cxx::tree::traits< Serial_type, wchar_t >`**  
**OpenGPS::Schemas::ISO5436\_2::InstrumentType::Serial\_traits**

Element traits type.

**9.34.2.11 `typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_-2::InstrumentType::Serial_type`**

Element type.

**9.34.2.12 `typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_-2::InstrumentType::Serial_type`**

Element type.

**9.34.2.13 `typedef ::xsd::cxx::tree::traits< Version_type, wchar_t >`**  
**OpenGPS::Schemas::ISO5436\_2::InstrumentType::Version\_traits**

Element traits type.

**9.34.2.14 `typedef ::xsd::cxx::tree::traits< Version_type, wchar_t >`**  
**OpenGPS::Schemas::ISO5436\_2::InstrumentType::Version\_traits**

Element traits type.

**9.34.2.15 `typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_-2::InstrumentType::Version_type`**

Element type.

**9.34.2.16 `typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_-2::InstrumentType::Version_type`**

Element type.

### 9.34.3 Constructor & Destructor Documentation

**9.34.3.1 `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.

**9.34.3.2 `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.

**9.34.3.3 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.

**9.34.3.4 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.

**9.34.3.5 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.

**9.34.3.6 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.

#### 9.34.4 Member Function Documentation

**9.34.4.1 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.

**9.34.4.2 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.

**9.34.4.3 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.

**9.34.4.4 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.

**9.34.4.5 Manufacturer\_type& OpenGPS::Schemas::ISO5436\_2::InstrumentType::Manufacturer ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.34.4.6 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.

**9.34.4.7 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.

---

**9.34.4.8 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.

**9.34.4.9 Manufacturer\_type& OpenGPS::Schemas::ISO5436\_2::InstrumentType::Manufacturer ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.34.4.10 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.

**9.34.4.11 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.

**9.34.4.12 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.

**9.34.4.13 Model\_type& 2::InstrumentType::Model ()****OpenGPS::Schemas::ISO5436\_-**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.34.4.14 const Model\_type& 2::InstrumentType::Model () const****OpenGPS::Schemas::ISO5436\_-**

Return a read-only (constant) reference to the element.

**Returns:**

A constant reference to the element.

**9.34.4.15 void 2::InstrumentType::Model (::std::auto\_ptr< Model\_type > p)****OpenGPS::Schemas::ISO5436\_-**

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.

**9.34.4.16 void 2::InstrumentType::Model (const Model\_type & x)****OpenGPS::Schemas::ISO5436\_-**

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.

---

**9.34.4.17 Model\_type& OpenGPS::Schemas::ISO5436\_2::InstrumentType::Model ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.34.4.18 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.

**9.34.4.19 void OpenGPS::Schemas::ISO5436\_2::InstrumentType::parse (::xsd::cxx::xml::dom::parser< wchar\_t > &, ::xml\_schema::flags) [protected]**

**9.34.4.20 void OpenGPS::Schemas::ISO5436\_2::InstrumentType::parse (::xsd::cxx::xml::dom::parser< wchar\_t > & p, ::xml\_schema::flags f) [protected]**

**9.34.4.21 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.

**9.34.4.22 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.

**9.34.4.23 Serial\_type& OpenGPS::Schemas::ISO5436\_-  
2::InstrumentType::Serial ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.34.4.24 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.

**9.34.4.25 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.

**9.34.4.26 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.

**9.34.4.27 Serial\_type& OpenGPS::Schemas::ISO5436\_-  
2::InstrumentType::Serial ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.34.4.28 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.

**9.34.4.29 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.

**9.34.4.30 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.

**9.34.4.31 Version\_type& OpenGPS::Schemas::ISO5436\_2::InstrumentType::Version ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.34.4.32 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.

---

**9.34.4.33 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.

**9.34.4.34 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.

**9.34.4.35 Version\_type& OpenGPS::Schemas::ISO5436\_2::InstrumentType::Version ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.34.4.36 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.

## 9.34.5 Member Data Documentation

**9.34.5.1 xsd::cxx::tree::one< Manufacturer\_type > OpenGPS::Schemas::ISO5436\_2::InstrumentType::Manufacturer\_[private]**

```
9.34.5.2 xsd::cxx::tree::one<           Model_type           >
OpenGPS::Schemas::ISO5436_2::InstrumentType::Model_-
[private]
```

```
9.34.5.3 xsd::cxx::tree::one<           Serial_type           >
OpenGPS::Schemas::ISO5436_2::InstrumentType::Serial_-
[private]
```

```
9.34.5.4 xsd::cxx::tree::one<           Version_type           >
OpenGPS::Schemas::ISO5436_2::InstrumentType::Version_-
[private]
```

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

## 9.35 OpenGPS::Int16DataPointParser Class Reference

```
#include <int16_data_point_parser.hxx>
```

Inheritance diagram for OpenGPS::Int16DataPointParser:

Collaboration diagram for OpenGPS::Int16DataPointParser:

### 9.35.1 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.*

### 9.35.2 Constructor & Destructor Documentation

#### 9.35.2.1 Int16DataPointParser::Int16DataPointParser ()

Creates a new instance.

#### 9.35.2.2 Int16DataPointParser::~Int16DataPointParser () [virtual]

Destroys this instance.

### 9.35.3 Member Function Documentation

#### 9.35.3.1 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`.

#### 9.35.3.2 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 exactly 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\\_point\\_parser.hxx](#)
- [int16\\_point\\_parser.cxx](#)

## 9.36 OpenGPS::Int16PointBuffer Class Reference

```
#include <int16_point_buffer.hxx>
```

Inheritance diagram for OpenGPS::Int16PointBuffer:

Collaboration diagram for OpenGPS::Int16PointBuffer:

### 9.36.1 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 unsigned long size) throw (...)  
*Allocates internal memory.*
- virtual void [Get](#) (const unsigned long index, [OGPS\\_Int16](#) &value) const  
throw (...)  
*Gets the value of the internal memory at the given position.*
- virtual [OGPS\\_DataPointType](#) [GetPointType](#) () const  
*Gets the type of point data that can be stored within this instance.*
- [Int16PointBuffer](#) ()  
*Create a new instance.*
- virtual void [Set](#) (const unsigned long index, const [OGPS\\_Int16](#) value)  
throw (...)  
*Sets the value of the internal memory at the given position.*
- virtual [~Int16PointBuffer](#) ()  
*Destroys this instance.*

### Private Attributes

- [OGPS\\_Int16](#) \* [m\\_Buffer](#)  
*Pointer to internal memory.*

### 9.36.2 Constructor & Destructor Documentation

#### 9.36.2.1 Int16PointBuffer::Int16PointBuffer ()

Create a new instance.

#### 9.36.2.2 Int16PointBuffer::~Int16PointBuffer () [virtual]

Destroys this instance.

### 9.36.3 Member Function Documentation

#### 9.36.3.1 void Int16PointBuffer::Allocate (const unsigned long *size*) throw (...) [virtual]

Allocates internal memory.

Throws an exception on failure.

**Parameters:**

*size* Amount of point data to be stored.

Reimplemented from [OpenGPS::PointBuffer](#).

**9.36.3.2 void Int16PointBuffer::Get (const unsigned long *index*, OGPS\_Int16 & *value*) const throw (...) [virtual]**

Gets the value of the internal 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](#).

**9.36.3.3 OGPS\_DataPointType Int16PointBuffer::GetPointType () const [virtual]**

Gets the type of point data that can be stored within this instance.

Reimplemented from [OpenGPS::PointBuffer](#).

**9.36.3.4 void Int16PointBuffer::Set (const unsigned long *index*, const OGPS\_Int16 *value*) throw (...) [virtual]**

Sets the value of the internal 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](#).

**9.36.4 Member Data Documentation****9.36.4.1 OGPS\_Int16\* OpenGPS::Int16PointBuffer::m\_Buffer [private]**

Pointer to internal memory.

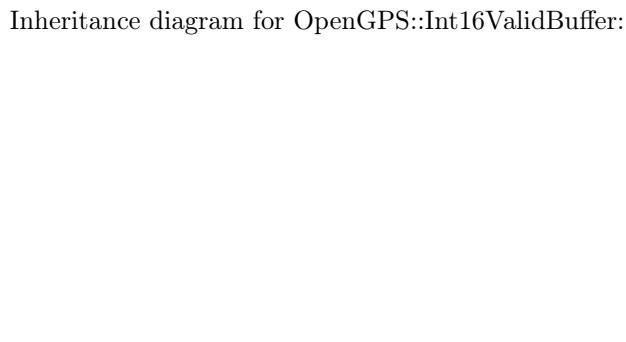
The documentation for this class was generated from the following files:

- [int16\\_point\\_buffer.hxx](#)
- [int16\\_point\\_buffer.cxx](#)

## 9.37 OpenGPS::Int16ValidBuffer Class Reference

```
#include <valid_buffer.hxx>
```

Inheritance diagram for OpenGPS::Int16ValidBuffer:



Collaboration diagram for OpenGPS::Int16ValidBuffer:



### 9.37.1 Detailed Description

Implementation of [OpenGPS::ValidBuffer](#) for Z axis of [OGPS\\_Int16](#) data type.

#### Public Member Functions

- [`Int16ValidBuffer \(PointBuffer \*const value\)`](#)  
*Creates a new instance.*
- [`virtual void SetValid \(const unsigned int index, const OGPS\_Boolean value\) throw \(...\)`](#)  
*Sets the validity of a point vector at a given location.*
- [`~Int16ValidBuffer \(\)`](#)  
*Destroys this instance.*

### 9.37.2 Constructor & Destructor Documentation

#### 9.37.2.1 Int16ValidBuffer::Int16ValidBuffer (PointBuffer \*const *value*)

Creates a new instance.

##### Parameters:

*value* The point buffer of the Z axis.

#### 9.37.2.2 Int16ValidBuffer::~Int16ValidBuffer ()

Destroys this instance.

### 9.37.3 Member Function Documentation

#### 9.37.3.1 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](#).

The documentation for this class was generated from the following files:

- [valid\\_buffer.hxx](#)
- [valid\\_buffer.cxx](#)

## 9.38 OpenGPS::Int32DataPointParser Class Reference

```
#include <int32_data_point_parser.hxx>
```

Inheritance diagram for OpenGPS::Int32DataPointParser:

Collaboration diagram for OpenGPS::Int32DataPointParser:

### 9.38.1 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.*

### 9.38.2 Constructor & Destructor Documentation

#### 9.38.2.1 Int32DataPointParser::Int32DataPointParser ()

Creates a new instance.

#### 9.38.2.2 Int32DataPointParser::~Int32DataPointParser () [virtual]

Destroys this instance.

### 9.38.3 Member Function Documentation

#### 9.38.3.1 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](#).

**9.38.3.2 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 exactly 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](#)

## 9.39 OpenGPS::Int32PointBuffer Class Reference

```
#include <int32_point_buffer.hxx>
```

Inheritance diagram for OpenGPS::Int32PointBuffer:

Collaboration diagram for OpenGPS::Int32PointBuffer:

### 9.39.1 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 unsigned long size) throw (...)  
*Allocates internal memory.*
- virtual void [Get](#) (const unsigned long index, [OGPS\\_Int32](#) &value) const  
throw (...)  
*Gets the value of the internal memory at the given position.*
- virtual [OGPS\\_DataPointType](#) [GetPointType](#) () const  
*Gets the type of point data that can be stored within this instance.*
- [Int32PointBuffer](#) ()  
*Create a new instance.*
- virtual [~Int32PointBuffer](#) ()  
*Destroys this instance.*

#### Private Attributes

- [OGPS\\_Int32](#) \* [m\\_Buffer](#)  
*Pointer to internal memory.*

### 9.39.2 Constructor & Destructor Documentation

#### 9.39.2.1 Int32PointBuffer::Int32PointBuffer ()

Create a new instance.

**9.39.2.2 Int32PointBuffer::~Int32PointBuffer () [virtual]**

Destroys this instance.

**9.39.3 Member Function Documentation****9.39.3.1 void Int32PointBuffer::Allocate (const unsigned long *size*)  
throw (...) [virtual]**

Allocates internal memory.

Throws an exception on failure.

**Parameters:**

*size* Amount of point data to be stored.

Reimplemented from [OpenGPS::PointBuffer](#).

**9.39.3.2 void Int32PointBuffer::Get (const unsigned long *index*,  
OGPS\_Int32 & *value*) const throw (...) [virtual]**

Gets the value of the internal 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](#).

**9.39.3.3 OGPS\_DataPointType Int32PointBuffer::GetPointType ()  
const [virtual]**

Gets the type of point data that can be stored within this instance.

Reimplemented from [OpenGPS::PointBuffer](#).

**9.39.3.4 void Int32PointBuffer::Set (const unsigned long *index*,  
const OGPS\_Int32 *value*) throw (...) [virtual]**

Sets the value of the internal 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](#).

#### 9.39.4 Member Data Documentation

**9.39.4.1 OGPS\_Int32\* OpenGPS::Int32PointBuffer::m\_Buffer  
[private]**

Pointer to internal memory.

The documentation for this class was generated from the following files:

- [int32\\_point\\_buffer.hxx](#)
- [int32\\_point\\_buffer.cxx](#)

## 9.40 OpenGPS::Int32ValidBuffer Class Reference

```
#include <valid_buffer.hxx>
```

Inheritance diagram for OpenGPS::Int32ValidBuffer:

Collaboration diagram for OpenGPS::Int32ValidBuffer:

#### 9.40.1 Detailed Description

Implementation of [OpenGPS::ValidBuffer](#) for Z axis of [OGPS\\_Int32](#) data type.

## Public Member Functions

- [Int32ValidBuffer \(PointBuffer \\*const value\)](#)  
*Creates a new instance.*
- virtual void [SetValid \(const unsigned int index, const OGPS\\_Boolean value\) throw \(...\)](#)  
*Sets the validity of a point vector at a given location.*
- [~Int32ValidBuffer \(\)](#)  
*Destroys this instance.*

### 9.40.2 Constructor & Destructor Documentation

#### 9.40.2.1 Int32ValidBuffer::Int32ValidBuffer (PointBuffer \*const value)

Creates a new instance.

##### Parameters:

*value* The point buffer of the Z axis.

#### 9.40.2.2 Int32ValidBuffer::~Int32ValidBuffer ()

Destroys this instance.

### 9.40.3 Member Function Documentation

#### 9.40.3.1 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](#).

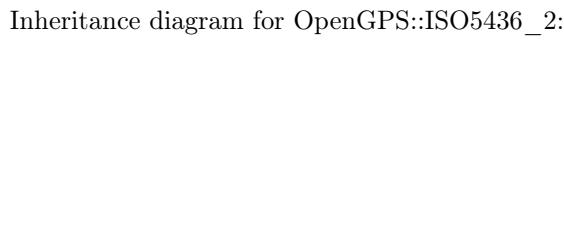
The documentation for this class was generated from the following files:

- [valid\\_buffer.hxx](#)
- [valid\\_buffer.cxx](#)

## 9.41 OpenGPS::ISO5436\_2 Class Reference

```
#include <iso5436_2.hxx>
```

Inheritance diagram for OpenGPS::ISO5436\_2:



Collaboration diagram for OpenGPS::ISO5436\_2:



### 9.41.1 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 unsigned long  
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](#) ()  
*Creates an iterator to access point data contained in an ISO5436-2 X3P file.*

- virtual `PointIteratorAutoPtr CreatePrevPointIterator ()`  
*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 unsigned long 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 void `GetListPoint (const unsigned long index, PointVector &vector) throw (...)`  
*Gets the raw value of a data point vector at a given index position.*
- virtual void `GetMatrixCoord (const unsigned long u, const unsigned long v, const unsigned long 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 `GetMatrixPoint (const unsigned long u, const unsigned long v, const unsigned long 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 vendor specific data from the current archive to a given file location.*
- virtual `OGPS_Boolean IsMatrixCoordValid (unsigned long u, unsigned long v, unsigned long 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=TRUE) throw (...)`  
*Opens an existing ISO5436-2 XML X3P file.*

- virtual void `SetListPoint` (const unsigned long index, const `PointVector` &vector) throw (...)  
*Sets the value of a three-dimensional data point vector at a given index position.*
- virtual void `SetMatrixPoint` (const unsigned long u, const unsigned long v, const unsigned long 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.*

### 9.41.2 Constructor & Destructor Documentation

#### 9.41.2.1 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.

**9.41.2.2 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.

**9.41.2.3 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.

**9.41.2.4 ISO5436\_2::~ISO5436\_2 () [virtual]**

Destructs this object.

**9.41.2.5 OpenGPS::ISO5436\_2::ISO5436\_2 (const ISO5436\_2 & src) [private]**

The copy-ctor is not implemented.

This prevents its usage.

### 9.41.3 Member Function Documentation

**9.41.3.1 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](#).

**9.41.3.2 void ISO5436\_2::Close () [virtual]**

Closes an open file handle and frees its resources.

**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](#).

**9.41.3.3 void ISO5436\_2::Create (const Schemas::ISO5436\_2::Record1Type & *record1*, const Schemas::ISO5436\_2::Record2Type & *record2*, const unsigned long *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.  
The given object instance must be valid.
- 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](#).

**9.41.3.4 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.  
The given object instance must be valid.
- 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](#).

**9.41.3.5 PointIteratorAutoPtr ISO5436\_2::CreateNextPointIterator () [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](#).

**9.41.3.6 PointIteratorAutoPtr ISO5436\_-2::CreatePrevPointIterator () [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](#).

**9.41.3.7 OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type \*const ISO5436\_2::GetDocument () [virtual]**

Gets access to the [ISO5436\\_2](#) XML document.

Reimplemented in [OpenGPS::ISO5436\\_2Container](#).

**9.41.3.8 void ISO5436\_2::GetListCoord (const unsigned long *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](#).

**9.41.3.9 void ISO5436\_2::GetListPoint (const unsigned long *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](#).

**9.41.3.10 void ISO5436\_2::GetMatrixCoord (const unsigned long *u*, const unsigned long *v*, const unsigned long *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](#).

---

**9.41.3.11 void ISO5436\_2::GetMatrixPoint (const unsigned long *u*, const unsigned long *v*, const unsigned long *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](#).

**9.41.3.12 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](#).

**9.41.3.13 OGPS\_Boolean ISO5436\_2::IsMatrixCoordValid (unsigned long *u*, unsigned long *v*, unsigned long *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](#).

**9.41.3.14 void ISO5436\_2::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 in [OpenGPS::ISO5436\\_2Container](#).

#### **9.41.3.15 ISO5436\_2& OpenGPS::ISO5436\_2::operator= (const ISO5436\_2 & src) [private]**

The assignment-operator is not implemented.

This prevents its usage.

#### **9.41.3.16 void ISO5436\_2::SetListPoint (const unsigned long 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](#).

#### **9.41.3.17 void ISO5436\_2::SetMatrixPoint (const unsigned long u, const unsigned long v, const unsigned long 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](#).

#### 9.41.3.18 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](#).

#### 9.41.4 Member Data Documentation

**9.41.4.1 ISO5436\_2\***                    **OpenGPS::ISO5436\_2::m\_Instance**  
[private]

Internal object instance.

Either "this" or ISO5436\_2::ISO5436\_2Container instance.

**9.41.4.2 const OGPS\_Boolean OpenGPS::ISO5436\_2::m\_IsProtected [private]**

When TRUE: do not allow any modifications to the data points.

The documentation for this class was generated from the following files:

- [iso5436\\_2.hxx](#)
- [iso5436\\_2.cxx](#)

## 9.42 OpenGPS::ISO5436\_2Container Class Reference

```
#include <iso5436_2_container.hxx>
```

Inheritance diagram for OpenGPS::ISO5436\_2Container:

Collaboration diagram for OpenGPS::ISO5436\_2Container:

#### 9.42.1 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 unsigned long 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](#) ()  
*Creates an iterator to access point data contained in an ISO5436-2 X3P file.*
- virtual [PointIteratorAutoPtr](#) [CreatePrevPointIterator](#) ()  
*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 unsigned long 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 void `GetListPoint (const unsigned long index, PointVector &vector) throw (...)`  
*Gets the raw value of a data point vector at a given index position.*
- virtual void `GetMatrixCoord (const unsigned long u, const unsigned long v, const unsigned long 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 `GetMatrixPoint (const unsigned long u, const unsigned long v, const unsigned long 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 vendor-specific data from the current archive to a given file location.*
- virtual `OGPS_Boolean IsMatrixCoordValid (unsigned long u, unsigned long v, unsigned long 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 unsigned long index, const PointVector &vector) throw (...)`  
*Sets the value of a three-dimensional data point vector at a given index position.*
- virtual void `SetMatrixPoint (const unsigned long u, const unsigned long v, const unsigned long 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 [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 unsigned long 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  
*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 GetFullPath () 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.*
- **unsigned long GetMaxU () const**  
*Gets the maximum index value possible in X direction.*
- **unsigned long GetMaxV () const**  
*Gets the maximum index value possible in Y direction.*
- **unsigned long GetMaxW () const**  
*Gets the maximum index value possible in Z direction.*
- **unsigned long 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.*

- **OGPS\_Boolean IsMatrix () const**

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

- **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.*
- `unsigned long 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.*

- unsigned long [SafeMultiplication \(const unsigned long long value1, const unsigned long long value2\) 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.*

### 9.42.2 Member Typedef Documentation

**9.42.2.1 `typedef std::vector<OpenGPS::String>`**  
**OpenGPS::ISO5436\_2Container::StringList [private]**

### 9.42.3 Constructor & Destructor Documentation

**9.42.3.1 `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 served 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 alternative 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.

### 9.42.3.2 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](#).

### 9.42.4 Member Function Documentation

#### 9.42.4.1 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](#).

#### 9.42.4.2 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.

#### 9.42.4.3 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.

**9.42.4.4 void ISO5436\_2Container::Close () [virtual]**

Closes an open file handle and frees its resources.

**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](#).

**9.42.4.5 void ISO5436\_2Container::Compress () throw (...) [protected]**

(Over)writes the current X3P archive file with the actual content.

**9.42.4.6 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.

**9.42.4.7 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.

**9.42.4.8 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.

**9.42.4.9 unsigned long 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.

**9.42.4.10 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.

```
9.42.4.11 void ISO5436_2Container::Create (const
Schemas::ISO5436_2::Record1Type & record1, const
Schemas::ISO5436_2::Record2Type & record2, const unsigned
long 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.  
The given object instance must be valid.

*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](#).

```
9.42.4.12 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.  
The given object instance must be valid.

***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](#).

#### 9.42.4.13 String ISO5436\_2Container::CreateContainerTempFilePath () const [protected]

Creates a unique temporary file name.

**9.42.4.14 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 unsigned long 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 stored within the ISO5436-2 XML document directly.

#### 9.42.4.15 PointIteratorAutoPtr ISO5436\_- 2Container::CreateNextPointIterator () [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](#).

**9.42.4.16 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.

**9.42.4.17 PointVectorProxyContext \* ISO5436\_-2Container::CreatePointVectorProxyContext () const [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 is responsible of releasing this resource.

**9.42.4.18 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.

**9.42.4.19 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.

**9.42.4.20 PointIteratorAutoPtr ISO5436\_-  
2Container::CreatePrevPointIterator () [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](#).

**9.42.4.21 void ISO5436\_2Container::CreateTempDir () throw (...) [private]**

Creates a temporary directory in the file system and sets [ISO5436\\_-2Container::m\\_TempPath](#).

**9.42.4.22 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.

**9.42.4.23 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.

**9.42.4.24 void ISO5436\_2Container::DecompressChecksum ()  
throw (...) [protected]**

Decompresses the md5 checksum file contained within the X3P archive.

**See also:**

<a href="#">ISO5436_2Container::Decompress</a> , <a href="#">ISO5436_2Container::GetChecksumArchiveName</a> , <a href="#">ISO5436_2Container::GetChecksumFileName</a>	<a href="#">ISO5436_-</a> <a href="#">ISO5436_-</a>
---	--

**9.42.4.25 void ISO5436\_2Container::DecompressDataBin () throw  
(...) [protected]**

Decompresses the binary point data file contained within the X3P archive.

**See also:**

<a href="#">ISO5436_2Container::Decompress</a> , <a href="#">ISO5436_2Container::GetPointDataArchiveName</a> , <a href="#">ISO5436_2Container::GetPointDataFileName</a>	<a href="#">ISO5436_-</a> <a href="#">ISO5436_-</a>
---	--

**9.42.4.26 void ISO5436\_2Container::DecompressMain () const  
throw (...) [protected]**

Decompresses the main xml document contained within the X3P archive.

**See also:**

<a href="#">ISO5436_2Container::Decompress</a> , <a href="#">ISO5436_2Container::GetMainArchiveName</a> , <a href="#">ISO5436_2Container::GetMainFileName</a>	<a href="#">ISO5436_-</a> <a href="#">ISO5436_-</a>
---	--

**9.42.4.27 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 implicit 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 incremental axis definition.

**9.42.4.28 String ISO5436\_2Container::GetChecksumArchiveName () const [protected]**

Gets the relative path of the md5 checksum file within the zip archive.

**9.42.4.29 String ISO5436\_2Container::GetChecksumFileName () const [protected]**

Gets the full path to the temporaryily decompressed md5 checksum file.

**9.42.4.30 OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type \*const ISO5436\_2Container::GetDocument () [virtual]**

Gets access to the [ISO5436\\_2](#) XML document.

Reimplemented from [OpenGPS::ISO5436\\_2](#).

**9.42.4.31 const OpenGPS::String & ISO5436\_2Container::GetFilePath () const [protected]**

Gets the file path to the X3P archive the current instance is an interface for.

**9.42.4.32 String ISO5436\_2Container::GetFullPath () const [protected]**

Gets the absolute file path to the X3P archive the current instance is an interface for.

**9.42.4.33 double ISO5436\_2Container::GetIncrementX () const [private]**

Gets the increment of the X axis definition or 1.0 if there is no incremental step.

**9.42.4.34 double ISO5436\_2Container::GetIncrementY () const [private]**

Gets the increment of the Y axis definition or 1.0 if there is no incremental step.

**9.42.4.35 void ISO5436\_2Container::GetListCoord (const unsigned long 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](#).

#### 9.42.4.36 void ISO5436\_2Container::GetListPoint (const unsigned long *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](#).

**9.42.4.37 String ISO5436\_2Container::GetMainArchiveName () const [protected]**

Gets the relative path of the main xml document file within the zip archive.

**9.42.4.38 String ISO5436\_2Container::GetMainFileName () const [protected]**

Gets the full path to the temporaryily decompressed main xml document file.

**9.42.4.39 void ISO5436\_2Container::GetMatrixCoord (const unsigned long *u*, const unsigned long *v*, const unsigned long *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](#).

**9.42.4.40 void ISO5436\_2Container::GetMatrixPoint (const unsigned long *u*, const unsigned long *v*, const unsigned long *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](#).

**9.42.4.41 unsigned long ISO5436\_2Container::GetMaxU () const [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.

**9.42.4.42 unsigned long ISO5436\_2Container::GetMaxV () const [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.

**9.42.4.43 unsigned long ISO5436\_2Container::GetMaxW () const [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.

**9.42.4.44 double ISO5436\_2Container::GetOffsetX () const [private]**

Gets the offset of the X axis definition or 0.0 if there is no offset at all.

**9.42.4.45 double ISO5436\_2Container::GetOffsetY () const [private]**

Gets the offset of the Y axis definition or 0.0 if there is no offset at all.

**9.42.4.46 double ISO5436\_2Container::GetOffsetZ () const [private]**

Gets the offset of the Z axis definition or 0.0 if there is no offset at all.

**9.42.4.47 unsigned long ISO5436\_2Container::GetPointCount () const throw (...) [protected]**

Gets the amount of point data that is stored in the archive.

**9.42.4.48 String ISO5436\_2Container::GetPointDataArchiveName () const [protected]**

Gets the relative path of the binary point data file within the zip archive.

**9.42.4.49 String ISO5436\_2Container::GetPointDataFileName () [protected]**

Gets the full path to the temporaryily decompressed binary point data file.

**9.42.4.50 const OpenGPS::String & ISO5436\_2Container::GetTempDir () const [protected]**

Gets the full path to the directory where temporary files get stored.

**9.42.4.51 String ISO5436\_2Container::GetValidPointsArchiveName () const [protected]**

Gets the relative path of the binary point validity data file within the zip archive.

**9.42.4.52 String ISO5436\_2Container::GetValidPointsFileName () [protected]**

Gets the full path to the temporaryily decompressed binary point validity data file.

**9.42.4.53 VectorBuffer \* ISO5436\_2Container::GetVectorBuffer () [private]**

Gets a pointer to the vector buffer or NULL.

**9.42.4.54 OGPS\_Boolean ISO5436\_2Container::GetVendorSpecific  
(const OpenGPS::String & *vendorURI*, const OpenGPS::String & *fileName*, const OpenGPS::String & *targetPath*) [virtual]**

Extracts vendor-specific data from the current archive to a given file location.

If the current X3P archive contains vendor-specific 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](#).

**9.42.4.55 OGPS\_DataPointType ISO5436\_-  
2Container::GetXaxisDataType () const [protected]**

Gets the data type of point data of the X component of a vector.

**9.42.4.56 OGPS\_DataPointType ISO5436\_-  
2Container::GetYaxisDataType () const [protected]**

Gets the data type of point data of the Y component of a vector.

**9.42.4.57 OGPS\_DataPointType ISO5436\_-  
2Container::GetZaxisDataType () const [protected]**

Gets the data type of point data of the Z component of a vector.

**9.42.4.58 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.

**9.42.4.59 OGPS\_Boolean ISO5436\_2Container::HasTempDir () const [private]**

Returns TRUE if a unique temporary directory has been created or FALSE if not.

**9.42.4.60 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.

**9.42.4.61 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.

**9.42.4.62 OGPS\_Boolean ISO5436\_2Container::IsBinary () const [protected]**

Checks whether points are sotred 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.

**9.42.4.63 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.

**9.42.4.64 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.

**9.42.4.65 OGPS\_Boolean ISO5436\_2Container::IsMatrix () const [protected]**

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.

**9.42.4.66 OGPS\_Boolean ISO5436\_2Container::IsMatrixCoordValid (unsigned long *u*, unsigned long *v*, unsigned long *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](#).

**9.42.4.67 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](#).

**9.42.4.68 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.

**9.42.4.69 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.

**9.42.4.70 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.

**9.42.4.71 void ISO5436\_2Container::RemoveTempDir () [private]**

Removes the current temporary directory from the file system and erases all of its content.

**9.42.4.72 void ISO5436\_2Container::Reset () [protected]**

Releases any resources allocated to handle the xml document tree or internal storage of vectors in memory.

**9.42.4.73 void ISO5436\_2Container::ResetValidPointsLink () [protected]**

Rests the valid points link xml tag of the xml document handle to its default value.

**9.42.4.74 void ISO5436\_2Container::ResetXmlPointList () [protected]**

Removes the point list xml tag and its content from the xml document handle.

**9.42.4.75 unsigned long ISO5436\_2Container::SafeMultiplication (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.

**9.42.4.76 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.

**9.42.4.77 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.

**9.42.4.78 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.

**9.42.4.79 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.

**9.42.4.80 void ISO5436\_2Container::SetListPoint (const unsigned long 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](#).

**9.42.4.81 void ISO5436\_2Container::SetMatrixPoint (const unsigned long *u*, const unsigned long *v*, const unsigned long *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](#).

**9.42.4.82 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.

**9.42.4.83 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.

**9.42.4.84 void ISO5436\_2Container::VerifyDataBinChecksum ()**  
[private]

Verifies the checksum of the binary point data file.

**9.42.4.85 void ISO5436\_2Container::VerifyMainChecksum ()**  
[private]

Verifies the checksum of the main document ISO5436-2 XML file.

**9.42.4.86 void ISO5436\_2Container::VerifyValidBinChecksum ()**  
[private]

Verifies the checksum of the binary point validity data file.

**9.42.4.87 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](#).

**9.42.4.88 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.

**9.42.5 Member Data Documentation****9.42.5.1 int OpenGPS::ISO5436\_2Container::m\_-  
CompressionLevel [private]**

The level of compression of the zip archive.

**9.42.5.2 OGPS\_Boolean OpenGPS::ISO5436\_2Container::m\_-  
DataBinChecksum [private]**

FALSE, if the md5 checksum could not be verified after reading.

**9.42.5.3 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.

**9.42.5.4 OpenGPS::String OpenGPS::ISO5436\_2Container::m\_-  
FilePath [private]**

The path of the X3P archive handles.

**9.42.5.5 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.

**9.42.5.6 OGPS\_Boolean OpenGPS::ISO5436\_2Container::m\_-  
IsReadOnly [private]**

TRUE if the user wants readonly access to the X3P file only.

FALSE otherwise.

**9.42.5.7 OGPS\_Boolean OpenGPS::ISO5436\_2Container::m\_-MainChecksum [private]**

FALSE, if the md5 checksum could not be verified after reading.

**9.42.5.8 OpenGPS::String OpenGPS::ISO5436\_2Container::m\_-PointDataFileName [private]**

The temporary target path of the uncompressed binary point data file.

**9.42.5.9 PointVectorAutoPtr OpenGPS::ISO5436\_-2Container::m\_-PointVector [private]**

A point vector which serves as a temporary buffer.

**9.42.5.10 PointVectorProxyContextAutoPtr OpenGPS::ISO5436\_-2Container::m\_-ProxyContext [private]**

A global point vector proxy used for index based read/writes of point vectors.

**9.42.5.11 OpenGPS::String OpenGPS::ISO5436\_2Container::m\_-TempBasePath [private]**

The path to the global directory for temporary files.

**9.42.5.12 OpenGPS::String OpenGPS::ISO5436\_2Container::m\_-TempPath [private]**

The path to a temporary directory unique to the current instance.

**9.42.5.13 OGPS\_Boolean OpenGPS::ISO5436\_2Container::m\_-ValidBinChecksum [private]**

FALSE, if the md5 checksum could not be verified after reading.

**9.42.5.14 OpenGPS::String OpenGPS::ISO5436\_2Container::m\_-ValidPointsFileName [private]**

The temporary target path of the uncompressed binary point validity data file.

**9.42.5.15 VectorBufferBuilderAutoPtr OpenGPS::ISO5436\_-2Container::m\_-VectorBufferBuilder [private]**

The builder used to assemble the current vector buffer.

**9.42.5.16 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](#)

**9.42.5.17 OpenGPS::String OpenGPS::ISO5436\_2Container::m\_-VendorURI [private]**

ID of vendor-specific data or empty.

**See also:**

[ISO5436\\_2Container::m\\_VendorSpecific.](#)

The documentation for this class was generated from the following files:

- [iso5436\\_2\\_container.hxx](#)
- [iso5436\\_2\\_container.cxx](#)

## **9.43 OpenGPS::ISO5436\_2Container::PointIteratorImpl Class Reference**

Inheritance diagram for OpenGPS::ISO5436\_2Container::PointIteratorImpl:

Collaboration diagram for OpenGPS::ISO5436\_2Container::PointIteratorImpl:

### **9.43.1 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` (`unsigned long *const u`, `unsigned long *const v`, `unsigned long *const w`) const  
*Gets the current position of the iterator in topology coordinates.*
- virtual `OGPS_Boolean GetPosition` (`unsigned long *const index`) const  
*Gets the current position of the iterator.*
- virtual `OGPS_Boolean HasNext` () const  
*Asks if there is another point available to iterate.*
- virtual `OGPS_Boolean HasPrev` () const  
*Asks if there is another point available to iterate.*
- virtual `OGPS_Boolean MoveNext` ()  
*Moves the iterator forward.*
- virtual `OGPS_Boolean MovePrev` ()  
*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 popoint 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.*
- `unsigned long m_U`  
*The current index in X direction.*
- `unsigned long m_V`  
*The current index in Y direction.*
- `unsigned long m_W`  
*The current index in Z direction.*

#### 9.43.2 Constructor & Destructor Documentation

##### 9.43.2.1 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.

#### 9.43.2.2 ISO5436\_2Container::PointIteratorImpl::~PointIteratorImpl() () [virtual]

Destroys this instance.

#### 9.43.2.3 OpenGPS::ISO5436\_2Container::PointIteratorImpl::PointIteratorImpl(const PointIteratorImpl & *src*) [private]

The copy-ctor is not implemented.

This prevents its usage.

### 9.43.3 Member Function Documentation

#### 9.43.3.1 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](#).

#### 9.43.3.2 OGPS\_Boolean ISO5436\_2Container::PointIteratorImpl::GetPosition(unsigned long \*const *u*, unsigned long \*const *v*, unsigned long \*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](#).

**9.43.3.3 OGPS\_Boolean ISO5436\_2Container::PointIteratorImpl::GetPosition  
(*unsigned long \*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](#).

**9.43.3.4 OGPS\_Boolean ISO5436\_2Container::PointIteratorImpl::HasNext  
() const [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](#).

**9.43.3.5 OGPS\_Boolean ISO5436\_2Container::PointIteratorImpl::HasPrev  
() const [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](#).

**9.43.3.6 OGPS\_Boolean ISO5436\_2Container::PointIteratorImpl::MoveNext  
() [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](#).

**9.43.3.7 OGPS\_Boolean ISO5436\_2Container::PointIteratorImpl::MovePrev  
() [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](#).

**9.43.3.8 PointIteratorImpl& OpenGPS::ISO5436\_-2Container::PointIteratorImpl::operator= (const PointIteratorImpl & src) [private]**

The assignment-operator is not implemented.

This prevents its usage.

**9.43.3.9 void ISO5436\_2Container::PointIteratorImpl::ResetNext () [virtual]**

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

Implements [OpenGPS::PointIterator](#).

**9.43.3.10 void ISO5436\_2Container::PointIteratorImpl::ResetPrev () [virtual]**

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

Implements [OpenGPS::PointIterator](#).

**9.43.3.11 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](#).

#### 9.43.4 Member Data Documentation

**9.43.4.1 OGPS::ISO5436\_2Container\* const OpenGPS::ISO5436\_2Container::PointIteratorImpl::m\_Handle [private]**

Instance of the ISO5436-2 X3P to be accessed through this point iterator.

**9.43.4.2 OGPS::ISO5436\_2Container::PointIteratorImpl::m\_IsForward [private]**

This is the forward- or backward iterator.

**9.43.4.3 OGPS::ISO5436\_2Container::PointIteratorImpl::m\_IsMatrix [private]**

TRUE to use matrix indexes and access methods, but FALSE for the simple list interface.

**9.43.4.4 OGPS::ISO5436\_2Container::PointIteratorImpl::m\_IsReset [private]**

TRUE if in an untouched state.

**9.43.4.5 unsigned long OpenGPS::ISO5436\_2Container::PointIteratorImpl::m\_U [private]**

The current index in X direction.

Used when iterating both matrices and lists.

**9.43.4.6 unsigned long OpenGPS::ISO5436\_2Container::PointIteratorImpl::m\_V [private]**

The current index in Y direction.

Used with matrices only.

**9.43.4.7 unsigned long OpenGPS::ISO5436\_2Container::PointIteratorImpl::m\_W [private]**

The current index in Z direction.

Used with matrices only.

The documentation for this class was generated from the following files:

- [iso5436\\_2\\_container.hxx](#)
- [point\\_iterator.cxx](#)

## 9.44 OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type Class Reference

```
#include <iso5436_2_xsd.hxx>
```

### 9.44.1 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_`

#### 9.44.2 Member Typedef Documentation

##### 9.44.2.1 `typedef ::xsd::cxx::tree::traits< Record1_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1_traits`

Element traits type.

##### 9.44.2.2 `typedef ::xsd::cxx::tree::traits< Record1_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1_traits`

Element traits type.

##### 9.44.2.3 `typedef ::OpenGPS::Schemas::ISO5436_2::Record1Type OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1_type`

Element type.

##### 9.44.2.4 `typedef ::OpenGPS::Schemas::ISO5436_2::Record1Type OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record1_type`

Element type.

**9.44.2.5** `typedef ::xsd::cxx::tree::optional< Record2_type >`  
`OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2_-`  
`optional`

Element optional container type.

**9.44.2.6** `typedef ::xsd::cxx::tree::optional< Record2_type >`  
`OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2_-`  
`optional`

Element optional container type.

**9.44.2.7** `typedef ::xsd::cxx::tree::traits< Record2_type, wchar_-`  
`t >` `OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2_-`  
`traits`

Element traits type.

**9.44.2.8** `typedef ::xsd::cxx::tree::traits< Record2_type, wchar_-`  
`t >` `OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2_-`  
`traits`

Element traits type.

**9.44.2.9** `typedef ::OpenGPS::Schemas::ISO5436_2::Record2Type`  
`OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2_type`

Element type.

**9.44.2.10** `typedef ::OpenGPS::Schemas::ISO5436_2::Record2Type`  
`OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record2_type`

Element type.

**9.44.2.11** `typedef ::xsd::cxx::tree::traits< Record3_type, wchar_-`  
`t >` `OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3_-`  
`traits`

Element traits type.

**9.44.2.12** `typedef ::xsd::cxx::tree::traits< Record3_type, wchar_-`  
`t >` `OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3_-`  
`traits`

Element traits type.

**9.44.2.13** `typedef ::OpenGPS::Schemas::ISO5436_2::Record3Type`  
`OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::Record3_type`

Element type.

**9.44.2.14 `typedef ::OpenGPS::Schemas::ISO5436_2::Record3Type`**  
**OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type::Record3\_type**

Element type.

**9.44.2.15 `typedef ::xsd::cxx::tree::traits< Record4_type, wchar_t >`**  
**OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type::Record4\_traits**

Element traits type.

**9.44.2.16 `typedef ::xsd::cxx::tree::traits< Record4_type, wchar_t >`**  
**OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type::Record4\_traits**

Element traits type.

**9.44.2.17 `typedef ::OpenGPS::Schemas::ISO5436_2::Record4Type`**  
**OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type::Record4\_type**

Element type.

**9.44.2.18 `typedef ::OpenGPS::Schemas::ISO5436_2::Record4Type`**  
**OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type::Record4\_type**

Element type.

**9.44.2.19 `typedef ::xsd::cxx::tree::optional< VendorSpecificID_type >`**  
**OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type::VendorSpecificID\_optional**

Element optional container type.

**9.44.2.20 `typedef ::xsd::cxx::tree::optional< VendorSpecificID_type >`**  
**OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type::VendorSpecificID\_optional**

Element optional container type.

**9.44.2.21 `typedef ::xsd::cxx::tree::traits< VendorSpecificID_type, wchar_t >`**  
**OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type::VendorSpecificID\_traits**

Element traits type.

**9.44.2.22 `typedef ::xsd::cxx::tree::traits< VendorSpecificID_type, wchar_t >`**  
**OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type::VendorSpecificID\_traits**

Element traits type.

**9.44.2.23** `typedef ::xml_schema::uri OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID_type`

Element type.

**9.44.2.24** `typedef ::xml_schema::uri OpenGPS::Schemas::ISO5436_2::ISO5436_2Type::VendorSpecificID_type`

Element type.

### 9.44.3 Constructor & Destructor Documentation

**9.44.3.1** `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.

**9.44.3.2** `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.

**9.44.3.3** `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.

**9.44.3.4** `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.

**9.44.3.5 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.

**9.44.3.6 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.

#### 9.44.4 Member Function Documentation

**9.44.4.1 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.

**9.44.4.2 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.

**9.44.4.3 void OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type::parse (::xsd::cxx::xml::dom::parser< wchar\_t > &, ::xml\_schema::flags) [protected]**

**9.44.4.4 void OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type::parse (::xsd::cxx::xml::dom::parser< wchar\_t > & p, ::xml\_schema::flags f) [protected]**

**9.44.4.5 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.

**9.44.4.6 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.

**9.44.4.7 Record1\_type& OpenGPS::Schemas::ISO5436\_-  
2::ISO5436\_2Type::Record1 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.44.4.8 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.

**9.44.4.9 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.

**9.44.4.10 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.

**9.44.4.11 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.

**9.44.4.12 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.

**9.44.4.13 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.

**9.44.4.14 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.

**9.44.4.15 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.

**9.44.4.16 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.

**9.44.4.17 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.

**9.44.4.18 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.

**9.44.4.19 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.

**9.44.4.20 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.

**9.44.4.21 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.

**9.44.4.22 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.

**9.44.4.23 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.

**9.44.4.24 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.

**9.44.4.25 Record3\_type& OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type::Record3 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.44.4.26 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.

**9.44.4.27 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.

**9.44.4.28 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.

**9.44.4.29 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.

**9.44.4.30 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.

**9.44.4.31 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.

**9.44.4.32 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.

**9.44.4.33 Record4\_type& OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type::Record4 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.44.4.34 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.

**9.44.4.35 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.

---

**9.44.4.36 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.

**9.44.4.37 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.

**9.44.4.38 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.

**9.44.4.39 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.

**9.44.4.40 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.

**9.44.4.41 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.

**9.44.4.42 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.

**9.44.4.43 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.

**9.44.4.44 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.

---

**9.44.4.45 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.

**9.44.4.46 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.

**9.44.4.47 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.

**9.44.4.48 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.

## 9.44.5 Member Data Documentation

**9.44.5.1 xsd:cxx:tree::one< Record1\_type >  
OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type::Record1\_-  
[private]**

9.44.5.2 Record2\_optional OpenGPS::Schemas::ISO5436\_-  
2::ISO5436\_2Type::Record2\_ [private]

9.44.5.3 xsd::cxx::tree::one< Record3\_type >  
OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type::Record3\_-  
[private]

9.44.5.4 xsd::cxx::tree::one< Record4\_type >  
OpenGPS::Schemas::ISO5436\_2::ISO5436\_2Type::Record4\_-  
[private]

9.44.5.5 VendorSpecificID\_optional OpenGPS::Schemas::ISO5436\_-  
2::ISO5436\_2Type::VendorSpecificID\_- [private]

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

## 9.45 OpenGPS::Schemas::ISO5436\_- 2::MatrixDimensionType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

### 9.45.1 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_`

#### 9.45.2 Member Typedef Documentation

##### 9.45.2.1 `typedef ::xsd::cxx::tree::traits< SizeX_type, wchar_t > OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX_traits`

Element traits type.

##### 9.45.2.2 `typedef ::xsd::cxx::tree::traits< SizeX_type, wchar_t > OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX_traits`

Element traits type.

##### 9.45.2.3 `typedef ::xml_schema::unsigned_long OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX_type`

Element type.

**9.45.2.4 `typedef ::xml_schema::unsigned_long  
OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeX_-  
type`**

Element type.

**9.45.2.5 `typedef ::xsd::cxx::tree::traits< SizeY_type, wchar_t  
> OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY_-  
traits`**

Element traits type.

**9.45.2.6 `typedef ::xsd::cxx::tree::traits< SizeY_type, wchar_t  
> OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY_-  
traits`**

Element traits type.

**9.45.2.7 `typedef ::xml_schema::unsigned_long  
OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY_-  
type`**

Element type.

**9.45.2.8 `typedef ::xml_schema::unsigned_long  
OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeY_-  
type`**

Element type.

**9.45.2.9 `typedef ::xsd::cxx::tree::traits< SizeZ_type, wchar_t >  
OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ_-  
traits`**

Element traits type.

**9.45.2.10 `typedef ::xsd::cxx::tree::traits< SizeZ_type, wchar_t >  
OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ_-  
traits`**

Element traits type.

**9.45.2.11 `typedef ::xml_schema::unsigned_long  
OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ_-  
type`**

Element type.

**9.45.2.12** `typedef ::xml_schema::unsigned_long OpenGPS::Schemas::ISO5436_2::MatrixDimensionType::SizeZ_-type`

Element type.

### 9.45.3 Constructor & Destructor Documentation

**9.45.3.1** `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.

**9.45.3.2** `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.

**9.45.3.3** `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.

**9.45.3.4** `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.

**9.45.3.5 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.

**9.45.3.6 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.

**9.45.4 Member Function Documentation**

**9.45.4.1 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.

**9.45.4.2 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.

**9.45.4.3 void OpenGPS::Schemas::ISO5436\_2::MatrixDimensionType::parse (::xsd::cxx::xml::dom::parser< wchar\_t > &, ::xml\_schema::flags) [protected]**

**9.45.4.4 void OpenGPS::Schemas::ISO5436\_2::MatrixDimensionType::parse (::xsd::cxx::xml::dom::parser< wchar\_t > & *p*, ::xml\_schema::flags *f*) [protected]**

**9.45.4.5 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.

**9.45.4.6 SizeX\_type& OpenGPS::Schemas::ISO5436\_2::MatrixDimensionType::SizeX ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.45.4.7 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.

**9.45.4.8 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.

**9.45.4.9 SizeX\_type& OpenGPS::Schemas::ISO5436\_2::MatrixDimensionType::SizeX ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.45.4.10 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.

**9.45.4.11 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.

**9.45.4.12 SizeY\_type& OpenGPS::Schemas::ISO5436\_-  
2::MatrixDimensionType::SizeY ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.45.4.13 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.

**9.45.4.14 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.

**9.45.4.15 SizeY\_type& OpenGPS::Schemas::ISO5436\_-  
2::MatrixDimensionType::SizeY ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.45.4.16 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.

**9.45.4.17 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.

**9.45.4.18 SizeZ\_type& OpenGPS::Schemas::ISO5436\_-  
2::MatrixDimensionType::SizeZ ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.45.4.19 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.

**9.45.4.20 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.

**9.45.4.21 SizeZ\_type& OpenGPS::Schemas::ISO5436\_-  
2::MatrixDimensionType::SizeZ ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.45.4.22 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.

### 9.45.5 Member Data Documentation

**9.45.5.1 xsd::cxx::tree::one< SizeX\_type >**  
**OpenGPS::Schemas::ISO5436\_2::MatrixDimensionType::SizeX\_**  
**[private]**

**9.45.5.2 xsd::cxx::tree::one< SizeY\_type >**  
**OpenGPS::Schemas::ISO5436\_2::MatrixDimensionType::SizeY\_**  
**[private]**

**9.45.5.3 xsd::cxx::tree::one< SizeZ\_type >**  
**OpenGPS::Schemas::ISO5436\_2::MatrixDimensionType::SizeZ\_**  
**[private]**

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

## 9.46 md5\_context Struct Reference

```
#include <md5.h>
```

### 9.46.1 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*

#### 9.46.2 Member Data Documentation

**9.46.2.1 unsigned char md5\_context::buffer[64]**  
data block being processed

**9.46.2.2 unsigned char md5\_context::ipad[64]**  
HMAC: inner padding.

**9.46.2.3 unsigned char md5\_context::opad[64]**  
HMAC: outer padding.

**9.46.2.4 unsigned long md5\_context::state[4]**  
intermediate digest state

**9.46.2.5 unsigned long md5\_context::total[2]**  
number of bytes processed

The documentation for this struct was generated from the following file:

- [md5.h](#)

#### 9.47 OpenGPS::MissingDataPointParser Class Reference

```
#include <missing_data_point_parser.hxx>
```

Inheritance diagram for OpenGPS::MissingDataPointParser:

Collaboration diagram for OpenGPS::MissingDataPointParser:



### 9.47.1 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.*

### 9.47.2 Constructor & Destructor Documentation

#### 9.47.2.1 MissingDataPointParser::MissingDataPointParser ()

Creates a new instance.

#### 9.47.2.2 MissingDataPointParser::~MissingDataPointParser () [virtual]

Destroys this instance.

### 9.47.3 Member Function Documentation

#### 9.47.3.1 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](#).

#### 9.47.3.2 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 exactly 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](#)

## 9.48 OpenGPS::OutputBinaryFileStream Class Reference

```
#include <point_vector_iostream.hxx>
```

Inheritance diagram for OpenGPS::OutputBinaryFileStream:



```
graph TD; OutputBinaryFileStream --> basic_ostream[std::basic_ostream<char>];
```

Collaboration diagram for OpenGPS::OutputBinaryFileStream:



```
graph TD; OutputBinaryFileStream <--> ios_base[std::ios_base]; OutputBinaryFileStream <--> ios[std::ios];
```

#### **9.48.1 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\\_ostream<OpenGPS::UnsignedByte> BaseType](#)  
*The type of the super class.*

### Private Attributes

- [PointVectorInvariantLocale m\\_Locale](#)  
*Invariant locale.*

#### 9.48.2 Member Typedef Documentation

##### 9.48.2.1 [typedef std::basic\\_ostream<OpenGPS::UnsignedByte> OpenGPS::OutputBinaryFileStream::BaseType \[private\]](#)

The type of the super class.

#### 9.48.3 Constructor & Destructor Documentation

##### 9.48.3.1 [OutputBinaryFileStream::OutputBinaryFileStream \(const OpenGPS::String & filePath\)](#)

Creates a new instance.

#### Parameters:

*filePath* Full path to the binary file to be written.

##### 9.48.3.2 [OutputBinaryFileStream::~OutputBinaryFileStream \(\)](#)

Destroys this instance.

#### 9.48.4 Member Data Documentation

##### 9.48.4.1 PointVectorInvariantLocale OpenGPS::OutputBinaryFileStream::m\_-Locale [private]

Invariant locale.

The documentation for this class was generated from the following files:

- [point\\_vector\\_iostream.hxx](#)
- [point\\_vector\\_iostream.cxx](#)

## 9.49 OpenGPS::PointBuffer Class Reference

```
#include <point_buffer.hxx>
```

Inheritance diagram for OpenGPS::PointBuffer:

### 9.49.1 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 typesafe 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 unsigned long size) throw (...)  
*Allocates internal memory.*

- virtual void `Get` (const unsigned long index, `OGPS_Double` &value) const  
throw (...)  
*Gets the value of the internal memory at the given position.*
- virtual void `Get` (const unsigned long index, `OGPS_Float` &value) const  
throw (...)  
*Gets the value of the internal memory at the given position.*
- virtual void `Get` (const unsigned long index, `OGPS_Int32` &value) const  
throw (...)  
*Gets the value of the internal memory at the given position.*
- virtual void `Get` (const unsigned long index, `OGPS_Int16` &value) const  
throw (...)  
*Gets the value of the internal memory at the given position.*
- virtual `OGPS_DataPointType GetPointType () const`  
*Gets the type of point data that can be stored within this instance.*
- virtual unsigned long `GetSize () const`  
*Gets the maximum amount of point data that can be stored.*
- virtual void `Set` (const unsigned long index, const `OGPS_Double` value)  
throw (...)  
*Sets the value of the internal memory at the given position.*
- virtual void `Set` (const unsigned long index, const `OGPS_Float` value)  
throw (...)  
*Sets the value of the internal memory at the given position.*
- virtual void `Set` (const unsigned long index, const `OGPS_Int32` value)  
throw (...)  
*Sets the value of the internal memory at the given position.*
- virtual void `Set` (const unsigned long index, const `OGPS_Int16` value)  
throw (...)  
*Sets the value of the internal memory at the given position.*
- virtual `~PointBuffer ()`  
*Destroys this instance.*

### Protected Member Functions

- `OpenGPS::UnsignedBytePtr Allocate (const unsigned long size, const size_t typeSize) throw (...)`  
*Allocates internal memory.*

- void [Free \(OpenGPS::UnsignedBytePtr \\*value\)](#)

*Frees allocated memory.*

- [PointBuffer \(\)](#)

*Creates a new instance.*

### Private Attributes

- unsigned long [m\\_Size](#)

*Logical size or amount of point data that can be stored.*

### 9.49.2 Constructor & Destructor Documentation

#### 9.49.2.1 PointBuffer::PointBuffer () [protected]

Creates a new instance.

#### 9.49.2.2 PointBuffer::~PointBuffer () [virtual]

Destroys this instance.

### 9.49.3 Member Function Documentation

#### 9.49.3.1 OpenGPS::UnsignedBytePtr PointBuffer::Allocate (const unsigned long *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.

**9.49.3.2 void PointBuffer::Allocate (const unsigned long *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](#).

**9.49.3.3 void PointBuffer::Free (OpenGPS::UnsignedBytePtr \* *value*) [protected]**

Frees allocated memory.

**See also:**

[PointBuffer::Allocate](#)

**Parameters:**

*value* The pointer to the memory to be freed.

**9.49.3.4 void PointBuffer::Get (const unsigned long *index*, OGPS\_-Double & *value*) const throw (...) [virtual]**

Gets the value of the internal 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](#).

**9.49.3.5 void PointBuffer::Get (const unsigned long *index*, OGPS\_-Float & *value*) const throw (...) [virtual]**

Gets the value of the internal 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](#).

**9.49.3.6 void PointBuffer::Get (const unsigned long *index*, OGPS\_-  
Int32 & *value*) const throw (...) [virtual]**

Gets the value of the internal 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](#).

**9.49.3.7 void PointBuffer::Get (const unsigned long *index*, OGPS\_-  
Int16 & *value*) const throw (...) [virtual]**

Gets the value of the internal 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](#).

**9.49.3.8 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](#).

**9.49.3.9 unsigned long PointBuffer::GetSize () const [virtual]**

Gets the maximum amount of point data that can be stored.

**9.49.3.10 void PointBuffer::Set (const unsigned long *index*, const  
OGPS\_Double *value*) throw (...) [virtual]**

Sets the value of the internal 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](#).

**9.49.3.11 void PointBuffer::Set (const unsigned long *index*, const OGPS\_Float *value*) throw (...) [virtual]**

Sets the value of the internal 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](#).

**9.49.3.12 void PointBuffer::Set (const unsigned long *index*, const OGPS\_Int32 *value*) throw (...) [virtual]**

Sets the value of the internal 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](#).

**9.49.3.13 void PointBuffer::Set (const unsigned long *index*, const OGPS\_Int16 *value*) throw (...) [virtual]**

Sets the value of the internal 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](#).

#### 9.49.4 Member Data Documentation

##### 9.49.4.1 unsigned long OpenGPS::PointBuffer::m\_Size [private]

Logical size or amount of point data that can be stored.

The documentation for this class was generated from the following files:

- [point\\_buffer.hxx](#)
- [point\\_buffer.cxx](#)

## 9.50 OpenGPS::PointIterator Class Reference

```
#include <point_iterator.hxx>
```

Inheritance diagram for OpenGPS::PointIterator:

### 9.50.1 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 \(unsigned long \\*const index\) const =0](#)  
*Gets the current position of the iterator.*
- virtual [OGPS\\_Boolean GetPosition \(unsigned long \\*const u, unsigned long \\*const v, unsigned long \\*const w\) const =0](#)  
*Gets the current position of the iterator in topology coordinates.*

- virtual `OGPS_Boolean HasNext () const =0`  
*Asks if there is another point available to iterate.*
- virtual `OGPS_Boolean HasPrev () const =0`  
*Asks if there is another point available to iterate.*
- virtual `OGPS_Boolean MoveNext ()=0`  
*Moves the iterator forward.*
- virtual `OGPS_Boolean MovePrev ()=0`  
*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.*

### 9.50.2 Constructor & Destructor Documentation

#### 9.50.2.1 `PointIterator::PointIterator () [protected]`

Creates a new instance.

#### 9.50.2.2 `PointIterator::~PointIterator () [virtual]`

Destructs this instance.

### 9.50.3 Member Function Documentation

**9.50.3.1 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](#).

**9.50.3.2 virtual OGPS\_Boolean OpenGPS::PointIterator::GetPosition (unsigned long \*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](#).

**9.50.3.3 virtual OGPS\_Boolean OpenGPS::PointIterator::GetPosition (unsigned long \*const *u*, unsigned long \*const *v*, unsigned long \*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](#).

**9.50.3.4 virtual OGPS\_Boolean OpenGPS::PointIterator::HasNext  
() const [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](#).

**9.50.3.5 virtual OGPS\_Boolean OpenGPS::PointIterator::HasPrev  
() const [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](#).

**9.50.3.6 virtual OGPS\_Boolean OpenGPS::PointIterator::MoveNext()  
[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](#).

**9.50.3.7 virtual OGPS\_Boolean OpenGPS::PointIterator::MovePrev()  
[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](#).

**9.50.3.8 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](#).

**9.50.3.9 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](#).

**9.50.3.10 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](#)

## 9.51 OpenGPS::PointValidityProvider Class Reference

```
#include <point_validity_provider.hxx>
```

Inheritance diagram for OpenGPS::PointValidityProvider:

Collaboration diagram for OpenGPS::PointValidityProvider:

### 9.51.1 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 Attributes

- `PointBuffer * m_PointBuffer`  
*Point buffer of the Z axis.*

### 9.51.2 Constructor & Destructor Documentation

#### 9.51.2.1 `PointValidityProvider::~PointValidityProvider` () [virtual]

Destroys this instance.

**9.51.2.2 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.

**9.51.3 Member Function Documentation****9.51.3.1 PointBuffer \* PointValidityProvider::GetPointBuffer () [protected]**

Returns the point buffer of the Z axis.

**9.51.3.2 const PointBuffer \* PointValidityProvider::GetPointBuffer () const [protected]**

Returns the point buffer of the Z axis.

**9.51.3.3 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](#).

**9.51.3.4 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](#).

#### 9.51.4 Member Data Documentation

##### 9.51.4.1 PointBuffer\* [OpenGPS::PointValidityProvider::m\\_pointBuffer](#) [private]

Point buffer of the Z axis.

The documentation for this class was generated from the following files:

- [point\\_validity\\_provider.hxx](#)
- [point\\_validity\\_provider.cxx](#)

## 9.52 OpenGPS::PointVector Class Reference

```
#include <point_vector.hxx>
```

Inheritance diagram for OpenGPS::PointVector:



```
graph TD; PointVectorBase --> PointVector
```

Collaboration diagram for OpenGPS::PointVector:

### 9.52.1 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](#)

### 9.52.2 Constructor & Destructor Documentation

#### 9.52.2.1 PointVector::PointVector ()

Creates a new instance.

#### 9.52.2.2 PointVector::~PointVector ()

Destructs this object.

### 9.52.3 Member Function Documentation

#### 9.52.3.1 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](#).

**9.52.3.2 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.

**9.52.3.3 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.

**9.52.3.4 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.

**9.52.3.5 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.

#### 9.52.3.6 `DataPoint * PointVector::GetX () [virtual]`

Gets typesafe direct access to the x component.

Implements [OpenGPS::PointVectorBase](#).

#### 9.52.3.7 `const DataPoint * PointVector::GetX () const [virtual]`

Gets typesafe direct read-only access to the x component.

Implements [OpenGPS::PointVectorBase](#).

#### 9.52.3.8 `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.

#### 9.52.3.9 `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.

**9.52.3.10 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.

**9.52.3.11 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.

**9.52.3.12 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.

**9.52.3.13 DataPoint \* PointVector::GetY () [virtual]**

Gets typesafe direct access to the y component.

Implements [OpenGPS::PointVectorBase](#).

**9.52.3.14 const DataPoint \* PointVector::GetY () const [virtual]**

Gets typesafe direct read-only access to the y component.

Implements [OpenGPS::PointVectorBase](#).

**9.52.3.15 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.

**9.52.3.16 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.

**9.52.3.17 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.

**9.52.3.18 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.

**9.52.3.19 DataPoint \* PointVector::GetZ () [virtual]**

Gets typesafe direct access to the z component.

Implements [OpenGPS::PointVectorBase](#).

**9.52.3.20 const DataPoint \* PointVector::GetZ () const [virtual]**

Gets typesafe direct read-only access to the z component.

Implements [OpenGPS::PointVectorBase](#).

**9.52.3.21 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.

**9.52.3.22 PointVector & PointVector::operator= (const PointVector & *src*) throw (...)****9.52.3.23 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](#).

**9.52.3.24 void PointVector::SetX (const OGPS\_Double *value*) throw (...)**

Sets the new value for the x component.

**Parameters:**

*value* The new value.

**9.52.3.25 void PointVector::SetX (const OGPS\_Float *value*) throw (...)**

Sets the new value for the x component.

**Parameters:**

*value* The new value.

**9.52.3.26 void PointVector::SetX (const OGPS\_Int32 *value*) throw (...)**

Sets the new value for the x component.

**Parameters:**

*value* The new value.

**9.52.3.27 void PointVector::SetX (const OGPS\_Int16 *value*) throw (...)**

Sets the new value for the x component.

**Parameters:**

*value* The new value.

**9.52.3.28 void PointVector::SetY (const OGPS\_Double *value*) throw (...)**

Sets the new value for the y component.

**Parameters:**

*value* The new value.

**9.52.3.29 void PointVector::SetY (const OGPS\_Float *value*) throw (...)**

Sets the new value for the y component.

**Parameters:**

*value* The new value.

**9.52.3.30 void PointVector::SetY (const OGPS\_Int32 *value*) throw (...)**

Sets the new value for the y component.

**Parameters:**

*value* The new value.

**9.52.3.31 void PointVector::SetY (const OGPS\_Int16 *value*) throw (...)**

Sets the new value for the y component.

**Parameters:**

*value* The new value.

**9.52.3.32 void PointVector::SetZ (const OGPS\_Double *value*) throw (...)**

Sets the new value for the z component.

**Parameters:**

*value* The new value.

**9.52.3.33 void PointVector::SetZ (const OGPS\_Float *value*) throw (...)**

Sets the new value for the z component.

**Parameters:**

*value* The new value.

**9.52.3.34 void PointVector::SetZ (const OGPS\_Int32 *value*) throw (...)**

Sets the new value for the z component.

**Parameters:**

*value* The new value.

**9.52.3.35 void PointVector::SetZ (const OGPS\_Int16 *value*) throw (...)**

Sets the new value for the z component.

**Parameters:**

*value* The new value.

## 9.52.4 Member Data Documentation

**9.52.4.1 DataPoint\* OpenGPS::PointVector::m\_X [private]**

**9.52.4.2 DataPoint\* OpenGPS::PointVector::m\_Y [private]**

**9.52.4.3 DataPoint\* OpenGPS::PointVector::m\_Z [private]**

The documentation for this class was generated from the following files:

- [point\\_vector.hxx](#)
- [point\\_vector.cxx](#)

## 9.53 OpenGPS::PointVectorBase Class Reference

```
#include <point_vector_base.hxx>
```

Inheritance diagram for OpenGPS::PointVectorBase:

### 9.53.1 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.*

**9.53.2 Constructor & Destructor Documentation****9.53.2.1 PointVectorBase::~PointVectorBase () [virtual]**  
Destructs this object.

**9.53.2.2 PointVectorBase::PointVectorBase () [protected]**

Creates a new instance.

**9.53.3 Member Function Documentation****9.53.3.1 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::PointVectorProxy](#), and [OpenGPS::PointVector](#).

**9.53.3.2 virtual DataPoint\* OpenGPS::PointVectorBase::GetX () [pure virtual]**

Gets typesafe direct access to the x component.

Implemented in [OpenGPS::PointVectorProxy](#), and [OpenGPS::PointVector](#).

**9.53.3.3 virtual const DataPoint\* OpenGPS::PointVectorBase::GetX () const [pure virtual]**

Gets typesafe direct read-only access to the x component.

Implemented in [OpenGPS::PointVectorProxy](#), and [OpenGPS::PointVector](#).

**9.53.3.4 virtual DataPoint\* OpenGPS::PointVectorBase::GetY () [pure virtual]**

Gets typesafe direct access to the y component.

Implemented in [OpenGPS::PointVectorProxy](#), and [OpenGPS::PointVector](#).

**9.53.3.5 virtual const DataPoint\* OpenGPS::PointVectorBase::GetY () const [pure virtual]**

Gets typesafe direct read-only access to the y component.

Implemented in [OpenGPS::PointVectorProxy](#), and [OpenGPS::PointVector](#).

**9.53.3.6 virtual DataPoint\* OpenGPS::PointVectorBase::GetZ () [pure virtual]**

Gets typesafe direct access to the z component.

Implemented in [OpenGPS::PointVectorProxy](#), and [OpenGPS::PointVector](#).

## **9.54 OpenGPS::PointVectorInputStream Class Reference**

**9.53.3.7 virtual const DataPoint\* OpenGPS::PointVectorBase::GetZ()  
() const [pure virtual]**

Gets typesafe direct read-only access to the z component.

Implemented in [OpenGPS::PointVectorProxy](#), and [OpenGPS::PointVector](#).

**9.53.3.8 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::PointVectorProxy](#), and [OpenGPS::PointVector](#).

The documentation for this class was generated from the following files:

- [point\\_vector\\_base.hxx](#)
- [point\\_vector.cxx](#)

## **9.54 OpenGPS::PointVectorInputStream Class Reference**

```
#include <point_vector_iostream.hxx>
```

Inheritance diagram for OpenGPS::PointVectorInputStream:

## **9.54 OpenGPS::PointVectorInputStream Class Reference**

Collaboration diagram for OpenGPS::PointVectorInputStream:

### **9.54.1 Detailed Description**

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

#### **Public Member Functions**

- [PointVectorInputStream \(const OpenGPS::String &s\)](#)  
*Creates a new instance.*
- [PointVectorInputStream \(\)](#)  
*Creates a new instance.*
- [~PointVectorInputStream \(\)](#)  
*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.*

#### **9.54.2 Member Typedef Documentation**

**9.54.2.1 typedef std::basic\_istringstream<OGPS\_Character> OpenGPS::PointVectorInputStream::BaseType [private]**

The type of the super class.

#### **9.54.3 Constructor & Destructor Documentation**

**9.54.3.1 PointVectorInputStream::PointVectorInputStream()**

Creates a new instance.

**9.54.3.2 PointVectorInputStream::PointVectorInputStream(const OpenGPS::String & s)**

Creates a new instance.

##### **Parameters:**

*s* The source which is streamed.

**9.54.3.3 PointVectorInputStream::~PointVectorInputStream()**

Destroys this instance.

#### **9.54.4 Member Data Documentation**

**9.54.4.1 PointVectorInvariantLocale OpenGPS::PointVectorInputStream::m\_Locale [private]**

Invariant locale.

The documentation for this class was generated from the following files:

- [point\\_vector\\_iostream.hxx](#)
- [point\\_vector\\_iostream.cxx](#)

## **9.55 OpenGPS::PointVectorInvariantLocale Class Reference**

```
#include <point_vector_iostream.hxx>
```

#### **9.55.1 Detailed Description**

The culture invariant locale.

## **9.56 OpenGPS::PointVectorOutputStream Class Reference**

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.*

#### **9.55.2 Member Typedef Documentation**

##### **9.55.2.1 [typedef std::locale OpenGPS::PointVectorInvariantLocale::BaseType](#) [private]**

The type of the super class.

#### **9.55.3 Constructor & Destructor Documentation**

##### **9.55.3.1 [PointVectorInvariantLocale::PointVectorInvariantLocale \(\)](#)**

Creates a new instance.

##### **9.55.3.2 [PointVectorInvariantLocale::~PointVectorInvariantLocale \(\)](#)**

Destroys this instance.

The documentation for this class was generated from the following files:

- [point\\_vector\\_iostream.hxx](#)
- [point\\_vector\\_iostream.cxx](#)

## **9.56 OpenGPS::PointVectorOutputStream Class Reference**

```
#include <point_vector_iostream.hxx>
```

## **9.56 OpenGPS::PointVectorOutputStream Class Reference**

Inheritance diagram for OpenGPS::PointVectorOutputStream:

Collaboration diagram for OpenGPS::PointVectorOutputStream:

### **9.56.1 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.

## **9.56 OpenGPS::PointVectorOutputStream Class Reference**

---

### **Public Member Functions**

- [PointVectorOutputStream \(\)](#)  
*Creates a new instance.*
- [~PointVectorOutputStream \(\)](#)  
*Destroys this instance.*

### **Private Types**

- [typedef std::basic\\_ostream< OGPS\\_Character > BaseType](#)  
*The type of the super class.*

### **Private Attributes**

- [PointVectorInvariantLocale m\\_Locale](#)  
*Invariant locale.*

#### **9.56.2 Member Typedef Documentation**

##### **9.56.2.1 [typedef std::basic\\_ostream<OGPS\\_Character> OpenGPS::PointVectorOutputStream::BaseType \[private\]](#)**

The type of the super class.

#### **9.56.3 Constructor & Destructor Documentation**

##### **9.56.3.1 [PointVectorOutputStream::PointVectorOutputStream\(\)](#)**

Creates a new instance.

##### **9.56.3.2 [PointVectorOutputStream::~PointVectorOutputStream\(\)](#)**

Destroys this instance.

#### **9.56.4 Member Data Documentation**

##### **9.56.4.1 [PointVectorInvariantLocale OpenGPS::PointVectorOutputStream::m\\_Locale \[private\]](#)**

Invariant locale.

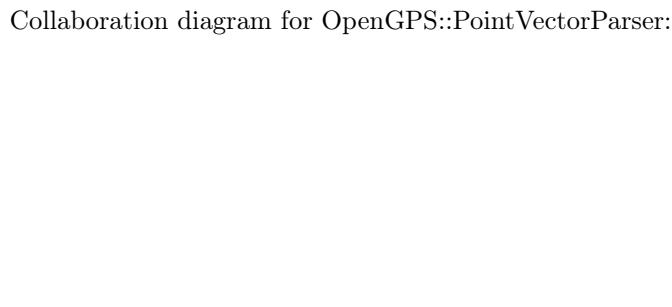
The documentation for this class was generated from the following files:

- [point\\_vector\\_iostream.hxx](#)
- [point\\_vector\\_iostream.cxx](#)

## 9.57 OpenGPS::PointVectorParser Class Reference

```
#include <point_vector_parser.hxx>
```

Collaboration diagram for OpenGPS::PointVectorParser:



### 9.57.1 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.*

### 9.57.2 Constructor & Destructor Documentation

#### 9.57.2.1 PointVectorParser::PointVectorParser ()

Creates a new instance.

#### 9.57.2.2 PointVectorParser::~PointVectorParser () [virtual]

Destroys this instance.

### 9.57.3 Member Function Documentation

#### 9.57.3.1 DataPointParser \* Parser::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.

**9.57.3.2 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.

**9.57.3.3 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](#)

**9.57.3.4 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](#)

**9.57.3.5 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](#)

**9.57.3.6 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.

**9.57.4 Member Data Documentation****9.57.4.1 DataPointParser\* OpenGPS::PointVectorParser::m\_X [private]**

Instance of the point parser of the X component.

**9.57.4.2 DataPointParser\* OpenGPS::PointVectorParser::m\_Y [private]**

Instance of the point parser of the Y component.

**9.57.4.3 DataPointParser\* OpenGPS::PointVectorParser::m\_Z [private]**

Instance of the point parser of the Z component.

The documentation for this class was generated from the following files:

- [point\\_vector\\_parser.hxx](#)
- [point\\_vector\\_parser.cxx](#)

## 9.58 OpenGPS::PointVectorParserBuilder Class Reference

```
#include <point_vector_parser_builder.hxx>
```

Collaboration diagram for OpenGPS::PointVectorParserBuilder:

### 9.58.1 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.*

#### 9.58.2 Constructor & Destructor Documentation

##### 9.58.2.1 PointVectorParserBuilder::PointVectorParserBuilder ()

Creates a new instance.

##### 9.58.2.2 PointVectorParserBuilder::~PointVectorParserBuilder () [virtual]

Destroys this instance.

#### 9.58.3 Member Function Documentation

##### 9.58.3.1 void PointVectorParserBuilder::BuildParser ()

Builds the main parser object to be assembled.

##### Remarks:

This must be called firstly when starting the build process.

##### 9.58.3.2 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](#)

##### 9.58.3.3 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](#)

**9.58.3.4 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](#)

**9.58.3.5 PointVectorParser \* PointVectorParserBuilder::GetParser () [virtual]**

Gets the assembled point vector parser.

**9.58.4 Member Data Documentation****9.58.4.1 PointVectorParser\* OpenGPS::PointVectorParserBuilder::m\_-Parser [private]**

The object to be built.

The documentation for this class was generated from the following files:

- [point\\_vector\\_parser\\_builder.hxx](#)
- [point\\_vector\\_parser\\_builder.cxx](#)

**9.59 OpenGPS::PointVectorProxy Class Reference**

```
#include <point_vector_proxy.hxx>
```

Inheritance diagram for OpenGPS::PointVectorProxy:

Collaboration diagram for OpenGPS::PointVectorProxy:

### 9.59.1 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.*

### 9.59.2 Constructor & Destructor Documentation

#### 9.59.2.1 `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.

**9.59.2.2 PointVectorProxy::~PointVectorProxy () [virtual]**

Destroys this instance.

**9.59.2.3 OpenGPS::PointVectorProxy::PointVectorProxy (const PointVectorProxy & src) [private]**

The copy-ctor is not implemented.

This prevents its usage.

**9.59.3 Member Function Documentation****9.59.3.1 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](#).

**9.59.3.2 DataPoint \* PointVectorProxy::GetX () [virtual]**

Gets typesafe direct access to the x component.

Implements [OpenGPS::PointVectorBase](#).

**9.59.3.3 const DataPoint \* PointVectorProxy::GetX () const [virtual]**

Gets typesafe direct read-only access to the x component.

Implements [OpenGPS::PointVectorBase](#).

**9.59.3.4 DataPoint \* PointVectorProxy::GetY () [virtual]**

Gets typesafe direct access to the y component.

Implements [OpenGPS::PointVectorBase](#).

**9.59.3.5 const DataPoint \* PointVectorProxy::GetY () const [virtual]**

Gets typesafe direct read-only access to the y component.

Implements [OpenGPS::PointVectorBase](#).

**9.59.3.6 DataPoint \* PointVectorProxy::GetZ () [virtual]**

Gets typesafe direct access to the z component.

Implements [OpenGPS::PointVectorBase](#).

**9.59.3.7 const DataPoint \* PointVectorProxy::GetZ () const [virtual]**

Gets typesafe direct read-only access to the z component.

Implements [OpenGPS::PointVectorBase](#).

**9.59.3.8 PointVectorProxy& OpenGPS::PointVectorProxy::operator=(const PointVectorProxy & src) [private]**

The assignment-operator is not implemented.

This prevents its usage.

**9.59.3.9 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](#).

### 9.59.4 Member Data Documentation

**9.59.4.1 VectorBuffer\* OpenGPS::PointVectorProxy::m\_Buffer [private]**

Provides access to the point data of all three axes.

**9.59.4.2 const PointVectorProxyContext\* OpenGPS::PointVectorProxy::m\_Context [private] const**

Provides information which data row is currently proxied.

**9.59.4.3 DataPoint\*                   OpenGPS::PointVectorProxy::m\_X  
[private]**

Provides typesafe access to the point data of the current X component.

**9.59.4.4 DataPoint\*                   OpenGPS::PointVectorProxy::m\_Y  
[private]**

Provides typesafe access to the point data of the current Y component.

**9.59.4.5 DataPoint\* OpenGPS::PointVectorProxy::m\_Z [private]**

Provides typesafe access to the point data of the current Z component.

The documentation for this class was generated from the following files:

- [point\\_vector\\_proxy.hxx](#)
- [point\\_vector\\_proxy.cxx](#)

**9.60 OpenGPS::PointVectorProxy::DataPointProxy  
Class Reference**

Inheritance diagram for OpenGPS::PointVectorProxy::DataPointProxy:

Collaboration diagram for OpenGPS::PointVectorProxy::DataPointProxy:

**9.60.1 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.*

### 9.60.2 Constructor & Destructor Documentation

#### 9.60.2.1 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.

#### 9.60.2.2 PointVectorProxy::DataPointProxy::~DataPointProxy () [virtual]

Destroys this instance.

#### 9.60.2.3 OpenGPS::PointVectorProxy::DataPointProxy::DataPointProxy (`const DataPointProxy & src`) [private]

The copy-ctor is not implemented.

This prevents its usage.

### 9.60.3 Member Function Documentation

#### 9.60.3.1 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](#).

**9.60.3.2 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](#).

**9.60.3.3 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](#).

**9.60.3.4 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](#).

**9.60.3.5 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](#).

**9.60.3.6 OGPS\_DataPointType  
Proxy::DataPointProxy::GetPointType () const throw (...) [virtual]**

PointVector-  
Proxy::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](#).

**9.60.3.7 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](#).

**9.60.3.8 DataPointProxy& OpenGPS::PointVectorProxy::DataPointProxy::operator= (const DataPointProxy & src) [private]**

The assignment-operator is not implemented.

This prevents its usage.

**9.60.3.9 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](#).

**9.60.3.10 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](#).

**9.60.3.11 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](#).

**9.60.3.12 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](#).

**9.60.3.13 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](#).

**9.60.3.14 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](#).

#### 9.60.4 Member Data Documentation

**9.60.4.1 PointBuffer\* OpenGPS::PointVectorProxy::DataPointProxy::m\_Buffer [private]**

Provides access to the point data.

**9.60.4.2 const PointVectorProxyContext\* OpenGPS::PointVectorProxy::DataPointProxy::m\_Context [private]**

Provides information which data row is currently proxied.

The documentation for this class was generated from the following files:

- [point\\_vector\\_proxy.hxx](#)
- [data\\_point\\_proxy.cxx](#)

#### 9.61 OpenGPS::PointVectorProxyContext Class Reference

```
#include <point_vector_proxy_context.hxx>
```

Inheritance diagram for OpenGPS::PointVectorProxyContext:

### 9.61.1 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 unsigned long [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.*

### 9.61.2 Constructor & Destructor Documentation

#### 9.61.2.1 PointVectorProxyContext::~PointVectorProxyContext () [virtual]

Destroys this instance.

#### 9.61.2.2 PointVectorProxyContext::PointVectorProxyContext () [protected]

Creates a new instance.

### 9.61.3 Member Function Documentation

#### 9.61.3.1 OGPS\_Boolean PointVectorProxyContext::CanIncrementIndex () const [virtual]

Can the current index be incremented?

Reimplemented in [OpenGPS::PointVectorProxyContextList](#), and [OpenGPS::PointVectorProxyContextMatrix](#).

**9.61.3.2 unsigned long PointVectorProxyContext::GetIndex () const [virtual]**

Gets the current index.

Reimplemented in [OpenGPS::PointVectorProxyContextList](#), and [OpenGPS::PointVectorProxyContextMatrix](#).

**9.61.3.3 OGPS\_Boolean PointVectorProxyContext::IncrementIndex () [virtual]**

Increments the current index by one.

Reimplemented in [OpenGPS::PointVectorProxyContextList](#), and [OpenGPS::PointVectorProxyContextMatrix](#).

**9.61.3.4 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](#)

**9.62 OpenGPS::PointVectorProxyContextList Class Reference**

```
#include <point_vector_proxy_context_list.hxx>
```

Inheritance diagram for OpenGPS::PointVectorProxyContextList:

Collaboration diagram for OpenGPS::PointVectorProxyContextList:

### **9.62.1 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 unsigned long [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 \(const unsigned long maxIndex\)](#)  
*Creates a new instance.*
- virtual [void SetIndex \(const unsigned long index\) throw \(...\)](#)  
*Sets the current index.*
- virtual [~PointVectorProxyContextList \(\)](#)  
*Destroys this instance.*

#### **Private Attributes**

- unsigned long [m\\_Index](#)  
*The current index.*
- unsigned long [m\\_MaxIndex](#)  
*The maximum index possible.*

### **9.62.2 Constructor & Destructor Documentation**

#### **9.62.2.1 PointVectorProxyContextList::PointVectorProxyContextList (const unsigned long *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.

**9.62.2.2 PointVectorProxyContextList::~PointVectorProxyContextList () [virtual]**

Destroys this instance.

**9.62.3 Member Function Documentation**

**9.62.3.1 OGPS\_Boolean PointVectorProxyContextList::CanIncrementIndex () const [virtual]**

Can the current index be incremented?

Reimplemented from [OpenGPS::PointVectorProxyContext](#).

**9.62.3.2 unsigned long PointVectorProxyContextList::GetIndex () const [virtual]**

Gets the current index.

Reimplemented from [OpenGPS::PointVectorProxyContext](#).

**9.62.3.3 OGPS\_Boolean PointVectorProxyContextList::IncrementIndex () [virtual]**

Increments the current index by one.

Reimplemented from [OpenGPS::PointVectorProxyContext](#).

**9.62.3.4 OGPS\_Boolean PointVectorProxyContextList::IsMatrix () const [virtual]**

Asks whether this context is for points managed in matrices or lists.

Implements [OpenGPS::PointVectorProxyContext](#).

**9.62.3.5 void PointVectorProxyContextList::SetIndex (const unsigned long *index*) throw (...) [virtual]**

Sets the current index.

**Parameters:**

***index*** The new arbitrary index.

**Returns:**

Returns TRUE on success, FALSE otherwise.

## **9.63 OpenGPS::PointVectorProxyContextMatrix Class Reference**

### **9.62.4 Member Data Documentation**

#### **9.62.4.1 unsigned long OpenGPS::PointVectorProxyContextList::m\_Index [private]**

The current index.

#### **9.62.4.2 unsigned long OpenGPS::PointVectorProxyContextList::m\_MaxIndex [private]**

The maximum index possible.

The documentation for this class was generated from the following files:

- [point\\_vector\\_proxy\\_context\\_list.hxx](#)
- [point\\_vector\\_proxy\\_context\\_list.cxx](#)

## **9.63 OpenGPS::PointVectorProxyContextMatrix Class Reference**

```
#include <point_vector_proxy_context_matrix.hxx>
```

Inheritance diagram for OpenGPS::PointVectorProxyContextMatrix:

Collaboration diagram for OpenGPS::PointVectorProxyContextMatrix:

### **9.63.1 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](#)

## **9.63 OpenGPS::PointVectorProxyContextMatrix Class Reference**

*Can the current index be incremented?*

- virtual unsigned long [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 \(const unsigned long maxU, const unsigned long maxV, const unsigned long maxW\)](#)  
*Creates a new instance.*
- void [SetIndex \(const unsigned long u, const unsigned long v, const unsigned long w\) throw \(...\)](#)  
*Sets the current index.*
- virtual [~PointVectorProxyContextMatrix \(\)](#)  
*Destroys this instance.*

### **Private Attributes**

- unsigned long [m\\_MaxU](#)  
*The maximum index possible in X direction.*
- unsigned long [m\\_MaxV](#)  
*The maximum index possible in Y direction.*
- unsigned long [m\\_MaxW](#)  
*The maximum index possible in Z direction.*
- unsigned long [m\\_U](#)  
*The current index in X direction of the topology mapping.*
- unsigned long [m\\_V](#)  
*The current index in Y direction of the topology mapping.*
- unsigned long [m\\_W](#)  
*The current index in Z direction of the topology mapping.*

## **9.63 OpenGPS::PointVectorProxyContextMatrix Class Reference**

### **9.63.2 Constructor & Destructor Documentation**

**9.63.2.1 PointVectorProxyContextMatrix::PointVectorProxyContextMatrix  
(const unsigned long *maxU*, const unsigned long *maxV*, const unsigned long *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.

**9.63.2.2 PointVectorProxyContextMatrix::~PointVectorProxyContextMatrix  
() [virtual]**

Destroys this instance.

### **9.63.3 Member Function Documentation**

**9.63.3.1 OGPS\_Boolean PointVectorProxyContextMatrix::CanIncrementIndex () const [virtual]**

Can the current index be incremented?

Reimplemented from [OpenGPS::PointVectorProxyContext](#).

**9.63.3.2 unsigned long PointVectorProxyContextMatrix::GetIndex () const [virtual]**

Gets the current index.

Reimplemented from [OpenGPS::PointVectorProxyContext](#).

**9.63.3.3 OGPS\_Boolean PointVectorProxyContextMatrix::IncrementIndex () [virtual]**

Increments the current index by one.

Reimplemented from [OpenGPS::PointVectorProxyContext](#).

**9.63.3.4 OGPS\_Boolean PointVectorProxyContextMatrix::IsMatrix () const [virtual]**

Asks whether this context is for points managed in matrices or lists.

Implements [OpenGPS::PointVectorProxyContext](#).

## **9.63 OpenGPS::PointVectorProxyContextMatrix Class Reference**

**9.63.3.5 void PointVectorProxyContextMatrix::SetIndex (const unsigned long *u*, const unsigned long *v*, const unsigned long *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.

## **9.63.4 Member Data Documentation**

**9.63.4.1 unsigned long OpenGPS::PointVectorProxyContextMatrix::m\_-MaxU [private]**

The maximum index possible in X direction.

**9.63.4.2 unsigned long OpenGPS::PointVectorProxyContextMatrix::m\_-MaxV [private]**

The maximum index possible in Y direction.

**9.63.4.3 unsigned long OpenGPS::PointVectorProxyContextMatrix::m\_-MaxW [private]**

The maximum index possible in Z direction.

**9.63.4.4 unsigned long OpenGPS::PointVectorProxyContextMatrix::m\_-U [private]**

The current index in X direction of the topology mapping.

**9.63.4.5 unsigned long OpenGPS::PointVectorProxyContextMatrix::m\_-V [private]**

The current index in Y direction of the topology mapping.

**9.63.4.6 unsigned long OpenGPS::PointVectorProxyContextMatrix::m\_-W [private]**

The current index in Z direction of the topology mapping.

The documentation for this class was generated from the following files:

- [point\\_vector\\_proxy\\_context\\_matrix.hxx](#)
- [point\\_vector\\_proxy\\_context\\_matrix.cxx](#)

## 9.64 OpenGPS::PointVectorReaderContext Class Reference

```
#include <point_vector_reader_context.hxx>
```

Inheritance diagram for OpenGPS::PointVectorReaderContext:

### 9.64.1 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.*

### 9.64.2 Constructor & Destructor Documentation

#### 9.64.2.1 PointVectorReaderContext::~PointVectorReaderContext () [virtual]

Destroys this instance.

#### 9.64.2.2 PointVectorReaderContext::PointVectorReaderContext () [protected]

Creates a new instance.

### 9.64.3 Member Function Documentation

#### 9.64.3.1 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](#).

**9.64.3.2 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](#).

**9.64.3.3 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](#).

**9.64.3.4 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](#).

**9.64.3.5 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](#).

**9.64.3.6 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](#).

**9.64.3.7 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](#)

## **9.65 OpenGPS::PointVectorWhitespaceFacet Class Reference**

```
#include <point_vector_iostream.hxx>
```

### **9.65.1 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.*

### 9.65.2 Member Typedef Documentation

**9.65.2.1 typedef std::ctype<OGPS\_Character>**  
OpenGPS::PointVectorWhitespaceFacet::BaseType [private]

The type of the super class.

### 9.65.3 Constructor & Destructor Documentation

**9.65.3.1 PointVectorWhitespaceFacet::PointVectorWhitespaceFacet (const size\_t refs = 0)**

Creates a new instance.

**9.65.3.2 PointVectorWhitespaceFacet::~PointVectorWhitespaceFacet ()**

Destroys this instance.

### 9.65.4 Member Function Documentation

**9.65.4.1 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](#)

## 9.66 OpenGPS::PointVectorWriterContext Class Reference

```
#include <point_vector_writer_context.hxx>
```

Inheritance diagram for OpenGPS::PointVectorWriterContext:

### 9.66.1 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.*

### 9.66.2 Constructor & Destructor Documentation

**9.66.2.1 PointVectorWriterContext::~PointVectorWriterContext ()**  
`[virtual]`

Destroys this instance.

**9.66.2.2 PointVectorWriterContext::PointVectorWriterContext ()**  
`[protected]`

Creates a new instance.

### 9.66.3 Member Function Documentation

**9.66.3.1 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 occurred.

Implemented in [OpenGPS::BinaryPointVectorWriterContext](#), and [OpenGPS::XmlPointVectorWriterContext](#).

**9.66.3.2 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](#).

**9.66.3.3 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](#).

**9.66.3.4 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](#).

**9.66.3.5 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](#).

**9.66.3.6 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](#).

**9.66.3.7 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](#)

**9.67 OpenGPS::Schemas::ISO5436\_-  
2::ProbingSystemType Class Reference**

```
#include <iso5436_2_xsd.hxx>
```

**9.67.1 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\\_](#)

#### 9.67.2 Member Typedef Documentation

**9.67.2.1 `typedef ::xsd::cxx::tree::traits< Identification_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification_traits`**

Element traits type.

**9.67.2.2 `typedef ::xsd::cxx::tree::traits< Identification_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification_traits`**

Element traits type.

**9.67.2.3 `typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification_type`**

Element type.

**9.67.2.4 `typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Identification_type`**

Element type.

**9.67.2.5 `typedef ::xsd::cxx::tree::traits< Type_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Type_traits`**

Element traits type.

**9.67.2.6 `typedef ::xsd::cxx::tree::traits< Type_type, wchar_t > OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Type_traits`**

Element traits type.

**9.67.2.7 `typedef ::OpenGPS::Schemas::ISO5436_2::Type OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Type_type`**

Element type.

**9.67.2.8 `typedef ::OpenGPS::Schemas::ISO5436_2::Type OpenGPS::Schemas::ISO5436_2::ProbingSystemType::Type_type`**

Element type.

**9.67.3 Constructor & Destructor Documentation**

**9.67.3.1 `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.

**9.67.3.2 `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.

**9.67.3.3 `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.

**9.67.3.4 `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.

**9.67.3.5 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.

**9.67.3.6 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.

**9.67.4 Member Function Documentation**

**9.67.4.1 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.

**9.67.4.2 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.

**9.67.4.3 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.

**9.67.4.4 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.

**9.67.4.5 Identification\_type& OpenGPS::Schemas::ISO5436\_2::ProbingSystemType::Identification ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.67.4.6 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.

**9.67.4.7 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.

**9.67.4.8 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.

**9.67.4.9 Identification\_type& OpenGPS::Schemas::ISO5436\_2::ProbingSystemType::Identification ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.67.4.10 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.

**9.67.4.11 void** OpenGPS::Schemas::ISO5436\_-  
2::ProbingSystemType::parse (::xsd::cxx::xml::dom::parser< wchar\_-  
t > &, ::xml\_schema::flags) [protected]

**9.67.4.12 void** OpenGPS::Schemas::ISO5436\_-  
2::ProbingSystemType::parse (::xsd::cxx::xml::dom::parser< wchar\_-  
t > & p, ::xml\_schema::flags f) [protected]

**9.67.4.13 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.

**9.67.4.14 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.

**9.67.4.15 Type\_type&** OpenGPS::Schemas::ISO5436\_-  
2::ProbingSystemType::Type ()

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.67.4.16 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.

**9.67.4.17 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.

**9.67.4.18 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.

**9.67.4.19 Type\_type& OpenGPS::Schemas::ISO5436\_2::ProbingSystemType::Type ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.67.4.20 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.

## 9.67.5 Member Data Documentation

**9.67.5.1 xsd::cxx::tree::one< type > Identification\_OpenGPS::Schemas::ISO5436\_2::ProbingSystemType::Identification\_ [private]**

**9.67.5.2 xsd::cxx::tree::one< Type\_type >**  
**OpenGPS::Schemas::ISO5436\_2::ProbingSystemType::Type\_-**  
[private]

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

## **9.68 OpenGPS::Schemas::ISO5436\_2::Record1Type Class Reference**

#include <[iso5436\\_2\\_xsd.hxx](#)>

### **9.68.1 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_`

### 9.68.2 Member Typedef Documentation

**9.68.2.1** `typedef ::xsd::cxx::tree::traits< Axes_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::Record1Type::Axes\_traits

Element traits type.

**9.68.2.2** `typedef ::xsd::cxx::tree::traits< Axes_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::Record1Type::Axes\_traits

Element traits type.

**9.68.2.3** `typedef ::OpenGPS::Schemas::ISO5436_2::AxesType`  
OpenGPS::Schemas::ISO5436\_2::Record1Type::Axes\_type

Element type.

**9.68.2.4** `typedef ::OpenGPS::Schemas::ISO5436_2::AxesType`  
OpenGPS::Schemas::ISO5436\_2::Record1Type::Axes\_type

Element type.

**9.68.2.5** `typedef ::xsd::cxx::tree::traits< FeatureType_-  
type, wchar_t >` OpenGPS::Schemas::ISO5436\_-  
2::Record1Type::FeatureType\_traits

Element traits type.

**9.68.2.6** `typedef ::xsd::cxx::tree::traits< FeatureType_-  
type, wchar_t >` OpenGPS::Schemas::ISO5436\_-  
2::Record1Type::FeatureType\_traits

Element traits type.

**9.68.2.7** `typedef ::OpenGPS::Schemas::ISO5436_2::FeatureType`  
OpenGPS::Schemas::ISO5436\_2::Record1Type::FeatureType\_type

Element type.

**9.68.2.8** `typedef ::OpenGPS::Schemas::ISO5436_2::FeatureType`  
OpenGPS::Schemas::ISO5436\_2::Record1Type::FeatureType\_type

Element type.

**9.68.2.9** `typedef ::xsd::cxx::tree::traits< Revision_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::Record1Type::Revision\_traits

Element traits type.

**9.68.2.10** `typedef ::xsd::cxx::tree::traits< Revision_type, wchar_t >` OpenGPS::Schemas::ISO5436\_2::Record1Type::Revision\_traits

Element traits type.

**9.68.2.11** `typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_2::Record1Type::Revision_type`

Element type.

**9.68.2.12** `typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_2::Record1Type::Revision_type`

Element type.

### 9.68.3 Constructor & Destructor Documentation

**9.68.3.1** `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.

**9.68.3.2** `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.

**9.68.3.3** `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.

**9.68.3.4 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.

**9.68.3.5 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.

**9.68.3.6 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.

#### 9.68.4 Member Function Documentation

**9.68.4.1 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.

**9.68.4.2 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.

**9.68.4.3 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.

**9.68.4.4 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.

**9.68.4.5 Axes\_type& OpenGPS::Schemas::ISO5436\_2::Record1Type::Axes ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.68.4.6 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.

**9.68.4.7 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.

**9.68.4.8 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.

**9.68.4.9 Record1Type::Axes\_type & OpenGPS::Schemas::ISO5436\_2::Record1Type::Axes ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.68.4.10 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.

**9.68.4.11 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.

**9.68.4.12 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.

**9.68.4.13 FeatureType\_type& OpenGPS::Schemas::ISO5436\_2::Record1Type::FeatureType ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.68.4.14 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.

**9.68.4.15 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.

**9.68.4.16 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.

**9.68.4.17 Record1Type::FeatureType\_type &**  
**OpenGPS::Schemas::ISO5436\_2::Record1Type::FeatureType ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.68.4.18 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.

**9.68.4.19 void** **OpenGPS::Schemas::ISO5436\_2::Record1Type::parse** (**::xsd::cxx::xml::dom::parser< wchar\_t > &, ::xml\_schema::flags)** [protected]

9.68.4.20 void OpenGPS::Schemas::ISO5436\_-  
2::Record1Type::parse (::xsd::cxx::xml::dom::parser< wchar\_t >  
& p, ::xml\_schema::flags f) [protected]

9.68.4.21 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.

9.68.4.22 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.

9.68.4.23 Revision\_type& OpenGPS::Schemas::ISO5436\_-  
2::Record1Type::Revision ()

Return a read-write reference to the element.

**Returns:**

A reference to the element.

9.68.4.24 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.

**9.68.4.25 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.

**9.68.4.26 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.

**9.68.4.27 Record1Type::Revision\_type &  
OpenGPS::Schemas::ISO5436\_2::Record1Type::Revision ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.68.4.28 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.

## 9.68.5 Member Data Documentation

**9.68.5.1 xsd::cxx::tree::one< Axes\_type >  
OpenGPS::Schemas::ISO5436\_2::Record1Type::Axes\_ [private]**

**9.68.5.2 xsd::cxx::tree::one< FeatureType\_type >  
OpenGPS::Schemas::ISO5436\_2::Record1Type::FeatureType\_ [private]**

```
9.68.5.3 xsd::cxx::tree::one< Revision_type >
OpenGPS::Schemas::ISO5436_2::Record1Type::Revision_-
[private]
```

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

## **9.69 OpenGPS::Schemas::ISO5436\_2::Record2Type Class Reference**

```
#include <iso5436_2_xsd.hxx>
```

### **9.69.1 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`  
`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`  
`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\_

### 9.69.2 Member Typedef Documentation

**9.69.2.1 typedef ::xsd::cxx::tree::traits< CalibrationDate\_type, wchar\_t > OpenGPS::Schemas::ISO5436\_2::Record2Type::CalibrationDate\_traits**

Element traits type.

**9.69.2.2 typedef ::xsd::cxx::tree::traits< CalibrationDate\_type, wchar\_t > OpenGPS::Schemas::ISO5436\_2::Record2Type::CalibrationDate\_traits**

Element traits type.

**9.69.2.3 `typedef ::xml_schema::date_time  
OpenGPS::Schemas::ISO5436_2::Record2Type::CalibrationDate_-  
type`**

Element type.

**9.69.2.4 `typedef ::xml_schema::date_time  
OpenGPS::Schemas::ISO5436_2::Record2Type::CalibrationDate_-  
type`**

Element type.

**9.69.2.5 `typedef ::xsd::cxx::tree::optional< Comment_type >  
OpenGPS::Schemas::ISO5436_2::Record2Type::Comment_optional`**

Element optional container type.

**9.69.2.6 `typedef ::xsd::cxx::tree::optional< Comment_type >  
OpenGPS::Schemas::ISO5436_2::Record2Type::Comment_optional`**

Element optional container type.

**9.69.2.7 `typedef ::xsd::cxx::tree::traits< Comment_type, wchar_t  
> OpenGPS::Schemas::ISO5436_2::Record2Type::Comment_traits`**

Element traits type.

**9.69.2.8 `typedef ::xsd::cxx::tree::traits< Comment_type, wchar_t  
> OpenGPS::Schemas::ISO5436_2::Record2Type::Comment_traits`**

Element traits type.

**9.69.2.9 `typedef ::xml_schema::string OpenGPS::Schemas::ISO5436_-  
2::Record2Type::Comment_type`**

Element type.

**9.69.2.10 `typedef ::xml_schema::string OpenGPS::Schemas::ISO5436_-  
2::Record2Type::Comment_type`**

Element type.

**9.69.2.11 `typedef ::xsd::cxx::tree::optional< Creator_type >  
OpenGPS::Schemas::ISO5436_2::Record2Type::Creator_optional`**

Element optional container type.

**9.69.2.12 `typedef ::xsd::cxx::tree::optional< Creator_type >`  
OpenGPS::Schemas::ISO5436\_2::Record2Type::Creator\_optional**

Element optional container type.

**9.69.2.13 `typedef ::xsd::cxx::tree::traits< Creator_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::Record2Type::Creator\_traits**

Element traits type.

**9.69.2.14 `typedef ::xsd::cxx::tree::traits< Creator_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::Record2Type::Creator\_traits**

Element traits type.

**9.69.2.15 `typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_-  
2::Record2Type::Creator_type`**

Element type.

**9.69.2.16 `typedef ::xml_schema::token OpenGPS::Schemas::ISO5436_-  
2::Record2Type::Creator_type`**

Element type.

**9.69.2.17 `typedef ::xsd::cxx::tree::traits< Date_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::Record2Type::Date\_traits**

Element traits type.

**9.69.2.18 `typedef ::xsd::cxx::tree::traits< Date_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::Record2Type::Date\_traits**

Element traits type.

**9.69.2.19 `typedef ::xml_schema::date_time  
OpenGPS::Schemas::ISO5436_2::Record2Type::Date_type`**

Element type.

**9.69.2.20 `typedef ::xml_schema::date_time  
OpenGPS::Schemas::ISO5436_2::Record2Type::Date_type`**

Element type.

**9.69.2.21 `typedef ::xsd::cxx::tree::traits< Instrument_-  
type, wchar_t >` OpenGPS::Schemas::ISO5436\_-  
2::Record2Type::Instrument\_traits**

Element traits type.

9.69.2.22 `typedef ::xsd::cxx::tree::traits< Instrument_-  
type, wchar_t > OpenGPS::Schemas::ISO5436_-  
2::Record2Type::Instrument_traits`

Element traits type.

9.69.2.23 `typedef ::OpenGPS::Schemas::ISO5436_-  
2::InstrumentType OpenGPS::Schemas::ISO5436_-  
2::Record2Type::Instrument_type`

Element type.

9.69.2.24 `typedef ::OpenGPS::Schemas::ISO5436_-  
2::InstrumentType OpenGPS::Schemas::ISO5436_-  
2::Record2Type::Instrument_type`

Element type.

9.69.2.25 `typedef ::xsd::cxx::tree::traits< ProbingSystem_-  
type, wchar_t > OpenGPS::Schemas::ISO5436_-  
2::Record2Type::ProbingSystem_traits`

Element traits type.

9.69.2.26 `typedef ::xsd::cxx::tree::traits< ProbingSystem_-  
type, wchar_t > OpenGPS::Schemas::ISO5436_-  
2::Record2Type::ProbingSystem_traits`

Element traits type.

9.69.2.27 `typedef ::OpenGPS::Schemas::ISO5436_-  
2::ProbingSystemType OpenGPS::Schemas::ISO5436_-  
2::Record2Type::ProbingSystem_type`

Element type.

9.69.2.28 `typedef ::OpenGPS::Schemas::ISO5436_-  
2::ProbingSystemType OpenGPS::Schemas::ISO5436_-  
2::Record2Type::ProbingSystem_type`

Element type.

### 9.69.3 Constructor & Destructor Documentation

9.69.3.1 `OpenGPS::Schemas::ISO5436_2::Record2Type::Record2Type  
(const Date_type & Date, const Instrument_type & Instru-  
ment, const CalibrationDate_type & CalibrationDate, const  
ProbingSystem_type & ProbingSystem)`

Construct an instance from the ultimate base and initializers for required elements and attributes.

**9.69.3.2 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.

**9.69.3.3 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.

**9.69.3.4 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.

**9.69.3.5 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.

---

**9.69.3.6 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.

#### 9.69.4 Member Function Documentation

**9.69.4.1 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.

**9.69.4.2 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.

**9.69.4.3 void OpenGPS::Schemas::ISO5436\_-  
2::Record2Type::CalibrationDate (const 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.

**9.69.4.4 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.

**9.69.4.5 CalibrationDate\_type& OpenGPS::Schemas::ISO5436\_-  
2::Record2Type::CalibrationDate ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.69.4.6 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.

**9.69.4.7 void OpenGPS::Schemas::ISO5436\_-  
2::Record2Type::CalibrationDate (const 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.

**9.69.4.8 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.

**9.69.4.9 Record2Type::CalibrationDate\_type & OpenGPS::Schemas::ISO5436\_2::Record2Type::CalibrationDate ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.69.4.10 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.

**9.69.4.11 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.

**9.69.4.12 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.

**9.69.4.13 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.

**9.69.4.14 Comment\_optional& OpenGPS::Schemas::ISO5436\_-  
2::Record2Type::Comment ()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.69.4.15 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.

**9.69.4.16 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.

**9.69.4.17 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.

**9.69.4.18 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.

**9.69.4.19 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.

**9.69.4.20 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.

**9.69.4.21 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.

**9.69.4.22 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.

**9.69.4.23 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.

**9.69.4.24 Creator\_optional& OpenGPS::Schemas::ISO5436\_-  
2::Record2Type::Creator ()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.69.4.25 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.

**9.69.4.26 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.

**9.69.4.27 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.

**9.69.4.28 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.

**9.69.4.29 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.

**9.69.4.30 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.

**9.69.4.31 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.

**9.69.4.32 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.

**9.69.4.33 Date\_type& OpenGPS::Schemas::ISO5436\_-  
2::Record2Type::Date ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.69.4.34 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.

**9.69.4.35 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.

**9.69.4.36 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.

**9.69.4.37 Record2Type::Date\_type & OpenGPS::Schemas::ISO5436\_2::Record2Type::Date ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.69.4.38 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.

**9.69.4.39 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.

**9.69.4.40 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.

**9.69.4.41 Instrument\_type& OpenGPS::Schemas::ISO5436\_-  
2::Record2Type::Instrument ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.69.4.42 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.

**9.69.4.43 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.

**9.69.4.44 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.

**9.69.4.45 Record2Type::Instrument\_type &**  
**OpenGPS::Schemas::ISO5436\_2::Record2Type::Instrument ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.69.4.46 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.

**9.69.4.47 void OpenGPS::Schemas::ISO5436\_-  
2::Record2Type::parse (::xsd::cxx::xml::dom::parser< wchar\_t >  
&, ::xml\_schema::flags) [protected]**

**9.69.4.48 void OpenGPS::Schemas::ISO5436\_-  
2::Record2Type::parse (::xsd::cxx::xml::dom::parser< wchar\_t >  
& p, ::xml\_schema::flags f) [protected]**

**9.69.4.49 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.

**9.69.4.50 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.

**9.69.4.51 ProbingSystem\_type& OpenGPS::Schemas::ISO5436\_2::Record2Type::ProbingSystem ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.69.4.52 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.

**9.69.4.53 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.

**9.69.4.54 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.

**9.69.4.55 Record2Type::ProbingSystem\_type & OpenGPS::Schemas::ISO5436\_2::Record2Type::ProbingSystem ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.69.4.56 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.

### 9.69.5 Member Data Documentation

**9.69.5.1 xsd::cxx::tree::one< CalibrationDate\_type > OpenGPS::Schemas::ISO5436\_2::Record2Type::CalibrationDate\_ [private]**

**9.69.5.2 Comment\_optional OpenGPS::Schemas::ISO5436\_-2::Record2Type::Comment\_ [private]**

**9.69.5.3 Creator\_optional OpenGPS::Schemas::ISO5436\_-2::Record2Type::Creator\_ [private]**

**9.69.5.4 xsd::cxx::tree::one< Date\_type > OpenGPS::Schemas::ISO5436\_2::Record2Type::Date\_ [private]**

**9.69.5.5 xsd::cxx::tree::one< Instrument\_type > OpenGPS::Schemas::ISO5436\_2::Record2Type::Instrument\_- [private]**

**9.69.5.6 xsd::cxx::tree::one< ProbingSystem\_type > OpenGPS::Schemas::ISO5436\_2::Record2Type::ProbingSystem\_ [private]**

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

### 9.70 OpenGPS::Schemas::ISO5436\_2::Record3Type Class Reference

```
#include <iso5436_2_xsd.hxx>
```

### **9.70.1 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_`

### 9.70.2 Member Typedef Documentation

#### 9.70.2.1 `typedef ::xsd::cxx::tree::optional< DataLink_type >` OpenGPS::Schemas::ISO5436\_2::Record3Type::DataLink\_optional

Element optional container type.

**9.70.2.2 `typedef ::xsd::cxx::tree::optional< DataLink_type >`  
OpenGPS::Schemas::ISO5436\_2::Record3Type::DataLink\_optional**

Element optional container type.

**9.70.2.3 `typedef ::xsd::cxx::tree::traits< DataLink_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::Record3Type::DataLink\_traits**

Element traits type.

**9.70.2.4 `typedef ::xsd::cxx::tree::traits< DataLink_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::Record3Type::DataLink\_traits**

Element traits type.

**9.70.2.5 `typedef ::OpenGPS::Schemas::ISO5436_2::DataLinkType`  
OpenGPS::Schemas::ISO5436\_2::Record3Type::DataLink\_type**

Element type.

**9.70.2.6 `typedef ::OpenGPS::Schemas::ISO5436_2::DataLinkType`  
OpenGPS::Schemas::ISO5436\_2::Record3Type::DataLink\_type**

Element type.

**9.70.2.7 `typedef ::xsd::cxx::tree::optional< DataList_type >`  
OpenGPS::Schemas::ISO5436\_2::Record3Type::DataList\_optional**

Element optional container type.

**9.70.2.8 `typedef ::xsd::cxx::tree::optional< DataList_type >`  
OpenGPS::Schemas::ISO5436\_2::Record3Type::DataList\_optional**

Element optional container type.

**9.70.2.9 `typedef ::xsd::cxx::tree::traits< DataList_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::Record3Type::DataList\_traits**

Element traits type.

**9.70.2.10 `typedef ::xsd::cxx::tree::traits< DataList_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::Record3Type::DataList\_traits**

Element traits type.

**9.70.2.11 `typedef ::OpenGPS::Schemas::ISO5436_2::DataListType  
OpenGPS::Schemas::ISO5436_2::Record3Type::DataList_type`**

Element type.

**9.70.2.12 `typedef ::OpenGPS::Schemas::ISO5436_2::DataListType  
OpenGPS::Schemas::ISO5436_2::Record3Type::DataList_type`**

Element type.

**9.70.2.13 `typedef ::xsd::cxx::tree::optional< ListDimension_type  
> OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension_-  
optional`**

Element optional container type.

**9.70.2.14 `typedef ::xsd::cxx::tree::optional< ListDimension_type  
> OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension_-  
optional`**

Element optional container type.

**9.70.2.15 `typedef ::xsd::cxx::tree::traits< ListDimension_-  
type, wchar_t > OpenGPS::Schemas::ISO5436_-  
2::Record3Type::ListDimension_traits`**

Element traits type.

**9.70.2.16 `typedef ::xsd::cxx::tree::traits< ListDimension_-  
type, wchar_t > OpenGPS::Schemas::ISO5436_-  
2::Record3Type::ListDimension_traits`**

Element traits type.

**9.70.2.17 `typedef ::xml_schema::unsigned_long  
OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension_-  
type`**

Element type.

**9.70.2.18 `typedef ::xml_schema::unsigned_long  
OpenGPS::Schemas::ISO5436_2::Record3Type::ListDimension_-  
type`**

Element type.

**9.70.2.19 `typedef ::xsd::cxx::tree::optional<  
MatrixDimension_type > OpenGPS::Schemas::ISO5436_-  
2::Record3Type::MatrixDimension_optional`**

Element optional container type.

**9.70.2.20** `typedef ::xsd::cxx::tree::optional< MatrixDimension_type > OpenGPS::Schemas::ISO5436_-2::Record3Type::MatrixDimension_optional`

Element optional container type.

**9.70.2.21** `typedef ::xsd::cxx::tree::traits< MatrixDimension_type, wchar_t > OpenGPS::Schemas::ISO5436_-2::Record3Type::MatrixDimension_traits`

Element traits type.

**9.70.2.22** `typedef ::xsd::cxx::tree::traits< MatrixDimension_type, wchar_t > OpenGPS::Schemas::ISO5436_-2::Record3Type::MatrixDimension_traits`

Element traits type.

**9.70.2.23** `typedef ::OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType OpenGPS::Schemas::ISO5436_-2::Record3Type::MatrixDimension_type`

Element type.

**9.70.2.24** `typedef ::OpenGPS::Schemas::ISO5436_-2::MatrixDimensionType OpenGPS::Schemas::ISO5436_-2::Record3Type::MatrixDimension_type`

Element type.

### **9.70.3 Constructor & Destructor Documentation**

**9.70.3.1** `OpenGPS::Schemas::ISO5436_2::Record3Type::Record3Type()`

Construct an instance from the ultimate base and initializers for required elements and attributes.

**9.70.3.2** `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.

**9.70.3.3 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.

**9.70.3.4 OpenGPS::Schemas::ISO5436\_2::Record3Type::Record3Type**  
( )

Construct an instance from the ultimate base and initializers for required elements and attributes.

**9.70.3.5 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.

**9.70.3.6 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.

#### 9.70.4 Member Function Documentation

**9.70.4.1 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.

**9.70.4.2 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.

**9.70.4.3 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.

**9.70.4.4 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.

**9.70.4.5 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.

**9.70.4.6 DataLink\_optional& OpenGPS::Schemas::ISO5436\_2::Record3Type::DataLink ()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.70.4.7 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.

**9.70.4.8 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.

**9.70.4.9 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.

**9.70.4.10 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.

**9.70.4.11 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.

**9.70.4.12 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.

**9.70.4.13 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.

**9.70.4.14 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.

**9.70.4.15 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.

**9.70.4.16 dataList\_optional& OpenGPS::Schemas::ISO5436\_2::Record3Type::DataList ()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.70.4.17 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.

**9.70.4.18 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.

**9.70.4.19 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.

**9.70.4.20 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.

**9.70.4.21 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.

**9.70.4.22 const OpenGPS::Schemas::ISO5436\_2::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.

**9.70.4.23 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.

**9.70.4.24 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.

**9.70.4.25 ListDimension\_optional& OpenGPS::Schemas::ISO5436\_2::Record3Type::ListDimension ()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.70.4.26 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.

**9.70.4.27 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.

**9.70.4.28 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.

**9.70.4.29 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.

**9.70.4.30 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.

**9.70.4.31 void OpenGPS::Schemas::ISO5436\_-  
2::Record3Type::MatrixDimension (const MatrixDimension\_-<  
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.

**9.70.4.32 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.

**9.70.4.33 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.

**9.70.4.34 MatrixDimension\_optional& OpenGPS::Schemas::ISO5436\_-  
2::Record3Type::MatrixDimension ()**

Return a read-write reference to the element container.

**Returns:**

A reference to the optional container.

**9.70.4.35 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.

**9.70.4.36 void OpenGPS::Schemas::ISO5436\_-2::Record3Type::MatrixDimension (const 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.

**9.70.4.37 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.

**9.70.4.38 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.

**9.70.4.39 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.

**9.70.4.40 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.

**9.70.4.41 void OpenGPS::Schemas::ISO5436\_-2::Record3Type::parse (::xsd::cxx::xml::dom::parser< wchar\_t > &, ::xml\_schema::flags) [protected]**

**9.70.4.42 void OpenGPS::Schemas::ISO5436\_-2::Record3Type::parse (::xsd::cxx::xml::dom::parser< wchar\_t > & p, ::xml\_schema::flags f) [protected]**

**9.70.5 Member Data Documentation**

**9.70.5.1 DataLink\_optional OpenGPS::Schemas::ISO5436\_-2::Record3Type::DataLink\_ [private]**

**9.70.5.2 DataList\_optional OpenGPS::Schemas::ISO5436\_-2::Record3Type::DataList\_ [private]**

**9.70.5.3 ListDimension\_optional OpenGPS::Schemas::ISO5436\_-2::Record3Type::ListDimension\_ [private]**

**9.70.5.4 MatrixDimension\_optional OpenGPS::Schemas::ISO5436\_-2::Record3Type::MatrixDimension\_ [private]**

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

## **9.71 OpenGPS::Schemas::ISO5436\_2::Record4Type Class Reference**

```
#include <iso5436_2_xsd.hxx>
```

### **9.71.1 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 containing 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 containing 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\_

### 9.71.2 Member Typedef Documentation

#### 9.71.2.1 typedef ::xsd::cxx::tree::traits< ChecksumFile\_type, wchar\_t > OpenGPS::Schemas::ISO5436\_2::Record4Type::ChecksumFile\_traits

Element traits type.

#### 9.71.2.2 typedef ::xsd::cxx::tree::traits< ChecksumFile\_type, wchar\_t > OpenGPS::Schemas::ISO5436\_2::Record4Type::ChecksumFile\_traits

Element traits type.

#### 9.71.2.3 typedef ::xml\_schema::string OpenGPS::Schemas::ISO5436\_2::Record4Type::ChecksumFile\_type

Element type.

**9.71.2.4 `typedef ::xml_schema::string OpenGPS::Schemas::ISO5436_2::Record4Type::ChecksumFile_type`**

Element type.

**9.71.3 Constructor & Destructor Documentation**

**9.71.3.1 `OpenGPS::Schemas::ISO5436_2::Record4Type::Record4Type(const ChecksumFile_type & ChecksumFile)`**

Construct an instance from the ultimate base and initializers for required elements and attributes.

**9.71.3.2 `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.

**9.71.3.3 `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.

**9.71.3.4 `OpenGPS::Schemas::ISO5436_2::Record4Type::Record4Type(const ChecksumFile_type &)`**

Construct an instance from the ultimate base and initializers for required elements and attributes.

**9.71.3.5 `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.

**9.71.3.6 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.

**9.71.4 Member Function Documentation**

**9.71.4.1 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.

**9.71.4.2 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.

**9.71.4.3 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.

**9.71.4.4 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.

**9.71.4.5 ChecksumFile\_type& OpenGPS::Schemas::ISO5436\_2::Record4Type::ChecksumFile ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.71.4.6 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.

**9.71.4.7 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.

**9.71.4.8 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.

**9.71.4.9 Record4Type::ChecksumFile\_type &**  
**OpenGPS::Schemas::ISO5436\_2::Record4Type::ChecksumFile ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.71.4.10 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.

**9.71.4.11 void** OpenGPS::Schemas::ISO5436\_-  
2::Record4Type::parse (::xsd::cxx::xml::dom::parser< wchar\_t >  
&, ::xml\_schema::flags) [protected]

---

9.71.4.12 void **OpenGPS::Schemas::ISO5436\_-  
2::Record4Type::parse** (::xsd::cxx::xml::dom::parser< wchar\_t >  
& p, ::xml\_schema::flags f) [protected]

### 9.71.5 Member Data Documentation

9.71.5.1 xsd::cxx::tree::one< **ChecksumFile\_type** >  
OpenGPS::Schemas::ISO5436\_2::Record4Type::ChecksumFile\_  
[private]

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

## 9.72 OpenGPS::Schemas::ISO5436\_- 2::RotationMatrixElementType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

### 9.72.1 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.*

### 9.72.2 Constructor & Destructor Documentation

#### 9.72.2.1 OpenGPS::Schemas::ISO5436\_2::RotationMatrixElementType::RotationMatrixElement<sub>2</sub>(const ::xml\_schema::double\_ & double\_ )

Construct an instance from the ultimate base and initializers for required elements and attributes.

---

**9.72.2.2 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.

**9.72.2.3 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.

**9.72.2.4 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.

**9.72.2.5 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.

9.72.2.6 OpenGPS::Schemas::ISO5436\_2::RotationMatrixElementType::RotationMatrixElement  
(const ::xml\_schema::double\_ &)

Construct an instance from the ultimate base and initializers for required elements and attributes.

```
9.72.2.7 OpenGPS::Schemas::ISO5436_2::RotationMatrixElementType::RotationMatrixElem  
(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.

**9.72.2.8 OpenGPS::Schemas::ISO5436\_2::RotationMatrixElementType::RotationMatrixElement**  
(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.

**9.72.2.9 OpenGPS::Schemas::ISO5436\_2::RotationMatrixElementType::RotationMatrixElement**  
(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.

---

**9.72.2.10 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.

### 9.72.3 Member Function Documentation

**9.72.3.1 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.

**9.72.3.2 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.

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

## 9.73 OpenGPS::Schemas::ISO5436\_2::RotationType Class Reference

```
#include <iso5436_2_xsd.hxx>
```

### 9.73.1 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:  $Q = R \cdot P + T$  With Q being 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:  $Q_x = r_{11} \cdot P_x + r_{12} \cdot P_y + r_{13} \cdot P_z + T_x$ . The formula for the y coordinate is:  $Q_y = r_{21} \cdot P_x + r_{22} \cdot P_y + r_{23} \cdot P_z + T_y$ . The formula for the z coordinate is:  $Q_z = r_{31} \cdot P_x + r_{32} \cdot P_y + r_{33} \cdot P_z + T_z$ .

#### 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\\_](#)

**9.73.2 Member Typedef Documentation**

**9.73.2.1** `typedef ::xsd::cxx::tree::traits< r11_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::RotationType::r11\_traits

Element traits type.

**9.73.2.2** `typedef ::xsd::cxx::tree::traits< r11_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::RotationType::r11\_traits

Element traits type.

**9.73.2.3** `typedef ::OpenGPS::Schemas::ISO5436_-2::RotationMatrixElementType ::OpenGPS::Schemas::ISO5436_-2::RotationType::r11_type`

Element type.

**9.73.2.4** `typedef ::OpenGPS::Schemas::ISO5436_-2::RotationMatrixElementType ::OpenGPS::Schemas::ISO5436_-2::RotationType::r11_type`

Element type.

**9.73.2.5** `typedef ::xsd::cxx::tree::traits< r12_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::RotationType::r12\_traits

Element traits type.

**9.73.2.6** `typedef ::xsd::cxx::tree::traits< r12_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::RotationType::r12\_traits

Element traits type.

**9.73.2.7** `typedef ::OpenGPS::Schemas::ISO5436_-  
2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_-  
2::RotationType::r12_type`

Element type.

**9.73.2.8** `typedef ::OpenGPS::Schemas::ISO5436_-  
2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_-  
2::RotationType::r12_type`

Element type.

**9.73.2.9** `typedef ::xsd::cxx::tree::traits< r13_type, wchar_t >  
OpenGPS::Schemas::ISO5436_2::RotationType::r13_traits`

Element traits type.

**9.73.2.10** `typedef ::xsd::cxx::tree::traits< r13_type, wchar_t >  
OpenGPS::Schemas::ISO5436_2::RotationType::r13_traits`

Element traits type.

**9.73.2.11** `typedef ::OpenGPS::Schemas::ISO5436_-  
2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_-  
2::RotationType::r13_type`

Element type.

**9.73.2.12** `typedef ::OpenGPS::Schemas::ISO5436_-  
2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_-  
2::RotationType::r13_type`

Element type.

**9.73.2.13** `typedef ::xsd::cxx::tree::traits< r21_type, wchar_t >  
OpenGPS::Schemas::ISO5436_2::RotationType::r21_traits`

Element traits type.

**9.73.2.14** `typedef ::xsd::cxx::tree::traits< r21_type, wchar_t >  
OpenGPS::Schemas::ISO5436_2::RotationType::r21_traits`

Element traits type.

**9.73.2.15** `typedef ::OpenGPS::Schemas::ISO5436_-  
2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_-  
2::RotationType::r21_type`

Element type.

**9.73.2.16 `typedef ::OpenGPS::Schemas::ISO5436_-  
2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_-  
2::RotationType::r21_type`**

Element type.

**9.73.2.17 `typedef ::xsd::cxx::tree::traits< r22_type, wchar_t >  
OpenGPS::Schemas::ISO5436_2::RotationType::r22_traits`**

Element traits type.

**9.73.2.18 `typedef ::xsd::cxx::tree::traits< r22_type, wchar_t >  
OpenGPS::Schemas::ISO5436_2::RotationType::r22_traits`**

Element traits type.

**9.73.2.19 `typedef ::OpenGPS::Schemas::ISO5436_-  
2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_-  
2::RotationType::r22_type`**

Element type.

**9.73.2.20 `typedef ::OpenGPS::Schemas::ISO5436_-  
2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_-  
2::RotationType::r22_type`**

Element type.

**9.73.2.21 `typedef ::xsd::cxx::tree::traits< r23_type, wchar_t >  
OpenGPS::Schemas::ISO5436_2::RotationType::r23_traits`**

Element traits type.

**9.73.2.22 `typedef ::xsd::cxx::tree::traits< r23_type, wchar_t >  
OpenGPS::Schemas::ISO5436_2::RotationType::r23_traits`**

Element traits type.

**9.73.2.23 `typedef ::OpenGPS::Schemas::ISO5436_-  
2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_-  
2::RotationType::r23_type`**

Element type.

**9.73.2.24 `typedef ::OpenGPS::Schemas::ISO5436_-  
2::RotationMatrixElementType OpenGPS::Schemas::ISO5436_-  
2::RotationType::r23_type`**

Element type.

**9.73.2.25** `typedef ::xsd::cxx::tree::traits< r31_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::RotationType::r31\_traits

Element traits type.

**9.73.2.26** `typedef ::xsd::cxx::tree::traits< r31_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::RotationType::r31\_traits

Element traits type.

**9.73.2.27** `typedef ::OpenGPS::Schemas::ISO5436_-2::RotationMatrixElementType ::OpenGPS::Schemas::ISO5436_-2::RotationType::r31_type`

Element type.

**9.73.2.28** `typedef ::OpenGPS::Schemas::ISO5436_-2::RotationMatrixElementType ::OpenGPS::Schemas::ISO5436_-2::RotationType::r31_type`

Element type.

**9.73.2.29** `typedef ::xsd::cxx::tree::traits< r32_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::RotationType::r32\_traits

Element traits type.

**9.73.2.30** `typedef ::xsd::cxx::tree::traits< r32_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::RotationType::r32\_traits

Element traits type.

**9.73.2.31** `typedef ::OpenGPS::Schemas::ISO5436_-2::RotationMatrixElementType ::OpenGPS::Schemas::ISO5436_-2::RotationType::r32_type`

Element type.

**9.73.2.32** `typedef ::OpenGPS::Schemas::ISO5436_-2::RotationMatrixElementType ::OpenGPS::Schemas::ISO5436_-2::RotationType::r32_type`

Element type.

**9.73.2.33** `typedef ::xsd::cxx::tree::traits< r33_type, wchar_t >`  
OpenGPS::Schemas::ISO5436\_2::RotationType::r33\_traits

Element traits type.

**9.73.2.34** `typedef ::xsd::cxx::tree::traits< r33_type, wchar_t >`  
`OpenGPS::Schemas::ISO5436_2::RotationType::r33_traits`

Element traits type.

**9.73.2.35** `typedef ::OpenGPS::Schemas::ISO5436_-`  
`2::RotationMatrixElementType ::OpenGPS::Schemas::ISO5436_-`  
`2::RotationType::r33_type`

Element type.

**9.73.2.36** `typedef ::OpenGPS::Schemas::ISO5436_-`  
`2::RotationMatrixElementType ::OpenGPS::Schemas::ISO5436_-`  
`2::RotationType::r33_type`

Element type.

### 9.73.3 Constructor & Destructor Documentation

**9.73.3.1** `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.

**9.73.3.2** `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.

**9.73.3.3** `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.

**9.73.3.4 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.

**9.73.3.5 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.

**9.73.3.6 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.

### 9.73.4 Member Function Documentation

**9.73.4.1 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.

**9.73.4.2 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.

**9.73.4.3 void OpenGPS::Schemas::ISO5436\_2::RotationType::parse (::xsd::cxx::xml::dom::parser< wchar\_t > &, ::xml\_schema::flags) [protected]**

**9.73.4.4 void OpenGPS::Schemas::ISO5436\_2::RotationType::parse (::xsd::cxx::xml::dom::parser< wchar\_t > & p, ::xml\_schema::flags f) [protected]**

**9.73.4.5 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.

**9.73.4.6 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.

**9.73.4.7 r11\_type& OpenGPS::Schemas::ISO5436\_2::RotationType::r11 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.8 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.

**9.73.4.9 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.

**9.73.4.10 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.

**9.73.4.11 r11\_type& OpenGPS::Schemas::ISO5436\_2::RotationType::r11 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.12 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.

**9.73.4.13 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.

**9.73.4.14 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.

**9.73.4.15 r12\_type& OpenGPS::Schemas::ISO5436\_2::RotationType::r12 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.16 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.

**9.73.4.17 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.

**9.73.4.18 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.

**9.73.4.19 r12\_type& OpenGPS::Schemas::ISO5436\_2::RotationType::r12 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.20 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.

**9.73.4.21 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.

**9.73.4.22 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.

**9.73.4.23 r13\_type& OpenGPS::Schemas::ISO5436\_-  
2::RotationType::r13 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.24 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.

**9.73.4.25 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.

**9.73.4.26 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.

**9.73.4.27 r13\_type& OpenGPS::Schemas::ISO5436\_2::RotationType::r13 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.28 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.

**9.73.4.29 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.

**9.73.4.30 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.

**9.73.4.31 r21\_type& OpenGPS::Schemas::ISO5436\_2::RotationType::r21 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.32 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.

**9.73.4.33 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.

**9.73.4.34 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.

**9.73.4.35 r21\_type& OpenGPS::Schemas::ISO5436\_2::RotationType::r21 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.36 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.

**9.73.4.37 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.

**9.73.4.38 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.

**9.73.4.39 r22\_type& OpenGPS::Schemas::ISO5436\_2::RotationType::r22 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.40 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.

---

**9.73.4.41 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.

**9.73.4.42 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.

**9.73.4.43 r22\_type& OpenGPS::Schemas::ISO5436\_-  
2::RotationType::r22 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.44 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.

**9.73.4.45 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.

**9.73.4.46 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.

**9.73.4.47 r23\_type& OpenGPS::Schemas::ISO5436\_2::RotationType::r23 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.48 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.

**9.73.4.49 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.

**9.73.4.50 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.

**9.73.4.51 r23\_type& OpenGPS::Schemas::ISO5436\_2::RotationType::r23 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.52 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.

**9.73.4.53 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.

**9.73.4.54 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.

**9.73.4.55 r31\_type& OpenGPS::Schemas::ISO5436\_2::RotationType::r31 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.56 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.

**9.73.4.57 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.

**9.73.4.58 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.

**9.73.4.59 r31\_type& OpenGPS::Schemas::ISO5436\_2::RotationType::r31 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.60 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.

**9.73.4.61 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.

**9.73.4.62 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.

**9.73.4.63 r32\_type& OpenGPS::Schemas::ISO5436\_-  
2::RotationType::r32 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.64 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.

**9.73.4.65 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.

**9.73.4.66 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.

**9.73.4.67 r32\_type& OpenGPS::Schemas::ISO5436\_2::RotationType::r32 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.68 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.

**9.73.4.69 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.

**9.73.4.70 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.

**9.73.4.71 r33\_type& OpenGPS::Schemas::ISO5436\_2::RotationType::r33 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.72 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.

**9.73.4.73 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.

**9.73.4.74 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.

**9.73.4.75 r33\_type& OpenGPS::Schemas::ISO5436\_2::RotationType::r33 ()**

Return a read-write reference to the element.

**Returns:**

A reference to the element.

**9.73.4.76 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.

**9.73.5 Member Data Documentation**

**9.73.5.1 xsd::cxx::tree::one< r11\_type >  
OpenGPS::Schemas::ISO5436\_2::RotationType::r11\_ [private]**

**9.73.5.2 xsd::cxx::tree::one< r12\_type >  
OpenGPS::Schemas::ISO5436\_2::RotationType::r12\_ [private]**

**9.73.5.3 xsd::cxx::tree::one< r13\_type >  
OpenGPS::Schemas::ISO5436\_2::RotationType::r13\_ [private]**

**9.73.5.4 xsd::cxx::tree::one< r21\_type >  
OpenGPS::Schemas::ISO5436\_2::RotationType::r21\_ [private]**

**9.73.5.5 xsd::cxx::tree::one< r22\_type >  
OpenGPS::Schemas::ISO5436\_2::RotationType::r22\_ [private]**

**9.73.5.6 xsd::cxx::tree::one< r23\_type >  
OpenGPS::Schemas::ISO5436\_2::RotationType::r23\_ [private]**

**9.73.5.7 xsd::cxx::tree::one< r31\_type >  
OpenGPS::Schemas::ISO5436\_2::RotationType::r31\_ [private]**

**9.73.5.8 xsd::cxx::tree::one< r32\_type >  
OpenGPS::Schemas::ISO5436\_2::RotationType::r32\_ [private]**

**9.73.5.9 xsd::cxx::tree::one< r33\_type >**  
OpenGPS::Schemas::ISO5436\_2::RotationType::r33\_ [private]

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

## 9.74 OpenGPS::String Class Reference

#include <string.hxx>

Inheritance diagram for OpenGPS::String:

Collaboration diagram for OpenGPS::String:

### 9.74.1 Detailed Description

Stores an [OGPS\\_Character](#) sequence.

#### 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 sequence 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 sequence.*
- `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.*

#### 9.74.2 Member Typedef Documentation

##### 9.74.2.1 `typedef std::string OpenGPS::String::BaseType [private]`

The std::string<::OGPS\_Character> type.

**9.74.2.2 typedef OGPS\_Character OpenGPS::String::ElementType**

The [OGPS\\_Character](#) type.

**9.74.3 Constructor & Destructor Documentation****9.74.3.1 String::String ()**

Creates a new instance.

**9.74.3.2 String::String (const BaseType & s)**

Creates a new instance.

**Parameters:**

*s* Initialize the newly created object with the given character sequence.

**9.74.3.3 String::String (const OGPS\_Character \*const s)**

Creates a new instance.

**Parameters:**

*s* Initialize the newly created object with the given character sequence.

**9.74.3.4 String::~String ()**

Destructs an object.

**9.74.4 Member Function Documentation****9.74.4.1 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 stored in hexadecimal format.

**Returns:**

Returns TRUE on success, FALSE otherwise.

**9.74.4.2 OGPS\_Boolean String::ConvertToMd5  
(OpenGPS::UnsignedByte md5[16]) const**

Converts the current character sequence 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.

**9.74.4.3 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 characters 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.

**9.74.4.4 void String::FromChar (const char \*const s)**

Stores an ANSI character sequence.

**Parameters:**

*s* An ANSI character sequence to store as unicode internally.

**9.74.4.5 const char \* String::ToChar ()**

Converts the unicode character sequence to ANSI character.

**Returns:**

An ANSI character pointer or NULL.

The documentation for this class was generated from the following files:

- [string.hxx](#)
- [string.cxx](#)

## 9.75 OpenGPS::Schemas::ISO5436\_2::Type Class Reference

```
#include <iso5436_2_xsd.hxx>
```

### 9.75.1 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]`
- static const `wchar_t *const _xsd_Type_literals_ [2]`

### Protected Member Functions

- `value _xsd_Type_convert () const`
- `value _xsd_Type_convert () const`

#### 9.75.2 Member Enumeration Documentation

##### 9.75.2.1 enum OpenGPS::Schemas::ISO5436\_2::Type::value

Underlying enum type.

**Enumerator:**

*Contacting*  
*NonContacting*  
*Contacting*  
*NonContacting*

##### 9.75.2.2 enum OpenGPS::Schemas::ISO5436\_2::Type::value

Underlying enum type.

**Enumerator:**

*Contacting*  
*NonContacting*  
*Contacting*  
*NonContacting*

#### 9.75.3 Constructor & Destructor Documentation

##### 9.75.3.1 OpenGPS::Schemas::ISO5436\_2::Type (value *v*)

Construct an instance from the underlying enum value.

**Parameters:**

*v* A enum value.

##### 9.75.3.2 OpenGPS::Schemas::ISO5436\_2::Type::Type (const ::xml\_schema::token & *v*)

Construct an instance from the base value.

**Parameters:**

*v* A base value.

**9.75.3.3 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.

**9.75.3.4 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.

**9.75.3.5 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.

**9.75.3.6 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.

**9.75.3.7 OpenGPS::Schemas::ISO5436\_2::Type::Type (value *v*)**

Construct an instance from the underlying enum value.

**Parameters:**

*v* A enum value.

**9.75.3.8 OpenGPS::Schemas::ISO5436\_2::Type::Type (const ::xml\_schema::token & *v*)**

Construct an instance from the base value.

**Parameters:**

*v* A base value.

**9.75.3.9 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.

**9.75.3.10 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.

**9.75.3.11 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.

**9.75.3.12 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.

**9.75.4 Member Function Documentation****9.75.4.1 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.

**9.75.4.2 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.

**9.75.4.3 value OpenGPS::Schemas::ISO5436\_2::Type::\_xsd\_-  
Type\_convert () const [protected]**

**9.75.4.4 Type::value OpenGPS::Schemas::ISO5436\_2::Type::\_-  
xsd\_Type\_convert () const [protected]**

**9.75.4.5 virtual OpenGPS::Schemas::ISO5436\_2::Type::operator  
value () const [inline, virtual]**

Implicit conversion operator to the underlying enum value.

**Returns:**

A enum value.

**9.75.4.6 virtual OpenGPS::Schemas::ISO5436\_2::Type::operator  
value () const [inline, virtual]**

Implicit conversion operator to the underlying enum value.

**Returns:**

A enum value.

**9.75.4.7 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.

**9.75.4.8 Type& OpenGPS::Schemas::ISO5436\_2::Type::operator=(value *v*)**

Assign the underlying enum value.

**Parameters:**

*v* A enum value.

**Returns:**

A reference to the instance.

**9.75.5 Member Data Documentation****9.75.5.1 static const value OpenGPS::Schemas::ISO5436\_2::Type::\_xsd\_Type\_indexes\_ [static]****9.75.5.2 const wchar\_t\* const OpenGPS::Schemas::ISO5436\_2::Type::\_xsd\_Type\_literals\_[2] [static]****9.75.5.3 const wchar\_t\* const OpenGPS::Schemas::ISO5436\_2::Type::\_xsd\_Type\_literals\_[2] [static]**

The documentation for this class was generated from the following files:

- [src/ISO5436\\_2\\_XML/iso5436\\_2\\_xsd.hxx](#)
- [include/opengps/cxx/iso5436\\_2\\_xsd.hxx](#)
- [iso5436\\_2\\_xsd.cxx](#)

**9.76 OpenGPS::ValidBuffer Class Reference**

```
#include <valid_buffer.hxx>
```

Inheritance diagram for OpenGPS::ValidBuffer:

Collaboration diagram for OpenGPS::ValidBuffer:

### 9.76.1 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 void [Allocate \(\) throw \(...\)](#)  
*Allocates the internal bit array.*
- virtual [OGPS\\_Boolean IsAllocated \(\) const](#)  
*Returns wheter the bit buffer has been allocated already.*
- 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 [AllocateRaw](#) (const unsigned int rawSize) throw (...)  
*Allocates the internal bit array.*
- virtual void [Reset](#) ()  
*Frees allocated resources.*
- [ValidBuffer](#) ([PointBuffer](#) \*const value)  
*Creates a new instance.*

### Private Attributes

- unsigned long [m\\_RawSize](#)  
*Size of the bit array in bytes.*
- [OpenGPS::UnsignedBytePtr](#) [m\\_ValidityBuffer](#)  
*Pointer to the internal bit array.*

## 9.76.2 Constructor & Destructor Documentation

### 9.76.2.1 ValidBuffer::~ValidBuffer () [virtual]

Destroys this instance.

### 9.76.2.2 ValidBuffer::ValidBuffer ([PointBuffer](#) \*const *value*) [protected]

Creates a new instance.

#### Parameters:

*value* The point buffer of the Z axis.

## 9.76.3 Member Function Documentation

### 9.76.3.1 void ValidBuffer::Allocate () throw (...) [virtual]

Allocates the internal bit array.

Initially all point vectors are assumed to be valid.

### 9.76.3.2 void ValidBuffer::AllocateRaw (const unsigned int *rawSize*) throw (...) [protected, 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.

**9.76.3.3 OGPS\_Boolean ValidBuffer::IsAllocated () const [virtual]**

Returns wheter the bit buffer has been allocated already.

**9.76.3.4 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](#).

**9.76.3.5 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.

**9.76.3.6 void ValidBuffer::Reset () [protected, virtual]**

Frees allocated resources.

**9.76.3.7 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](#).

**9.76.3.8 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.

**9.76.4 Member Data Documentation****9.76.4.1 unsigned long OpenGPS::ValidBuffer::m\_RawSize [private]**

Size of the bit array in bytes.

**9.76.4.2 OpenGPS::UnsignedBytePtr OpenGPS::ValidBuffer::m\_ValidityBuffer [private]**

Pointer to the internal bit array.

The documentation for this class was generated from the following files:

- [valid\\_buffer.hxx](#)
- [valid\\_buffer.cxx](#)

**9.77 OpenGPS::VectorBuffer Class Reference**

```
#include <vector_buffer.hxx>
```

Collaboration diagram for OpenGPS::VectorBuffer:

### 9.77.1 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_VisibilityProvider`  
*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.*

### 9.77.2 Constructor & Destructor Documentation

#### 9.77.2.1 VectorBuffer::VectorBuffer ()

Creates a new instance.

#### 9.77.2.2 VectorBuffer::~VectorBuffer ()

Destroys this instance.

### 9.77.3 Member Function Documentation

#### 9.77.3.1 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.

**9.77.3.2 ValidBuffer \* VectorBuffer::GetValidityBuffer () [virtual]**

Gets the object representing an external point validity file.

May return NULL, if point validity information is not acquired through an external file but inline by the usage of special point values for invalid points.

**See also:**

[VectorBuffer::HasValidityBuffer](#)

**9.77.3.3 const ValidBuffer \* VectorBuffer::GetValidityBuffer () const [virtual]**

Gets the object representing an external point validity file.

May return NULL, if point validity information is not acquired through an external file but inline by the usage of special point values for invalid points.

**See also:**

[VectorBuffer::HasValidityBuffer](#)

**9.77.3.4 PointValidityProvider \* VectorBuffer::GetValidityProvider () [virtual]**

Gets the object that tracks the validity of point vectors.

**See also:**

[VectorBuffer::HasValidityBuffer](#)

**9.77.3.5 const PointValidityProvider \* VectorBuffer::GetValidityProvider () const [virtual]**

Gets the object that tracks the validity of point vectors.

**See also:**

[VectorBuffer::HasValidityBuffer](#)

**9.77.3.6 PointBuffer \* VectorBuffer::GetX () [virtual]**

Gets the buffer of point measurement data of the X components of all vectors.

**9.77.3.7 const PointBuffer \* VectorBuffer::GetX () const [virtual]**

Gets the buffer of point measurement data of the X components of all vectors for read-only access.

**9.77.3.8 PointBuffer \* VectorBuffer::GetY () [virtual]**

Gets the buffer of point measurement data of the Y components of all vectors.

**9.77.3.9 const PointBuffer \* VectorBuffer::GetY () const [virtual]**

Gets the buffer of point measurement data of the Y components of all vectors for read-only access.

**9.77.3.10 PointBuffer \* VectorBuffer::GetZ () [virtual]**

Gets the buffer of point measurement data of the Z components of all vectors.

**9.77.3.11 const PointBuffer \* VectorBuffer::GetZ () const [virtual]**

Gets the buffer of point measurement data of the Z components of all vectors for read-only access.

**9.77.3.12 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](#)

**9.77.3.13 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.

**9.77.3.14 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.

**9.77.3.15 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.

**9.77.3.16 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.

## 9.77.4 Member Data Documentation

**9.77.4.1 ValidBuffer\* OpenGPS::VectorBuffer::m\_ValidBuffer [private]**

Buffer for validity information of all point vector indexes.

**9.77.4.2 PointValidityProvider\* OpenGPS::VectorBuffer::m\_-ValidityProvider [private]**

Interface to retrieve/provide point validity information.

**9.77.4.3 PointBuffer\* OpenGPS::VectorBuffer::m\_X [private]**

Buffer for measurements of the X component of all point vectors.

**9.77.4.4 PointBuffer\* OpenGPS::VectorBuffer::m\_Y [private]**

Buffer for measurements of the Y component of all point vectors.

**9.77.4.5 PointBuffer\* OpenGPS::VectorBuffer::m\_Z [private]**

Buffer for measurements of the Z component of all point vectors.

The documentation for this class was generated from the following files:

- [vector\\_buffer.hxx](#)
- [vector\\_buffer.cxx](#)

**9.78 OpenGPS::VectorBufferBuilder Class Reference**

```
#include <vector_buffer_builder.hxx>
```

Collaboration diagram for OpenGPS::VectorBufferBuilder:

**9.78.1 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 unsigned long size)`  
*Connects the appropriate `OpenGPS::PointBuffer` connected with the X axis description.*
- virtual `OGPS_Boolean BuildY (const OGPS_DataPointType dataType, const unsigned long size)`  
*Connects the appropriate `OpenGPS::PointBuffer` connected with the Y axis description.*
- virtual `OGPS_Boolean BuildZ (const OGPS_DataPointType dataType, const unsigned long 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 this instance.*

### Private Member Functions

- `PointBuffer * CreatePointBuffer (const OGPS_DataPointType dataType, const unsigned long 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.*

### 9.78.2 Constructor & Destructor Documentation

#### 9.78.2.1 VectorBufferBuilder::VectorBufferBuilder ()

Creates a new instance.

**9.78.2.2 VectorBufferBuilder::~VectorBufferBuilder ()**

Destroys this instance.

**9.78.3 Member Function Documentation****9.78.3.1 OGPS\_Boolean VectorBufferBuilder::BuildBuffer () [virtual]**

Creates the initial [OpenGPS::VectorBuffer](#) to be assembled.

**Remarks:**

This must precede all other steps of the building process.

**9.78.3.2 OGPS\_Boolean VectorBufferBuilder::BuildValidityProvider () [virtual]**

Connects the appropriate [OpenGPS::PointValidityProvider](#).

**Remarks:**

This is the last step of the building process.

**9.78.3.3 OGPS\_Boolean VectorBufferBuilder::BuildX (const OGPS\_DataPointType dataType, const unsigned long 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.

**9.78.3.4 OGPS\_Boolean VectorBufferBuilder::BuildY (const OGPS\_DataPointType dataType, const unsigned long 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.

**9.78.3.5 OGPS\_Boolean VectorBufferBuilder::BuildZ (const OGPS\_DataPointType dataType, const unsigned long 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 component 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.

**9.78.3.6 PointBuffer \* VectorBufferBuilder::CreatePointBuffer (const OGPS\_DataPointType dataType, const unsigned long 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](#).

**9.78.3.7 VectorBuffer \* VectorBufferBuilder::GetBuffer () [virtual]**

Gets the object built.

This object instance remains managed by the builder, though.

#### 9.78.4 Member Data Documentation

##### 9.78.4.1 VectorBuffer\* OpenGPS::VectorBufferBuilder::m\_Buffer [private]

The vector buffer object to be assembled.

The documentation for this class was generated from the following files:

- [vector\\_buffer\\_builder.hxx](#)
- [vector\\_buffer\\_builder.cxx](#)

## 9.79 OpenGPS::XmlPointVectorReaderContext Class Reference

```
#include <xml_point_vector_reader_context.hxx>
```

Inheritance diagram for OpenGPS::XmlPointVectorReaderContext:

## **9.79 OpenGPS::XmlPointVectorReaderContext Class Reference**

Collaboration diagram for OpenGPS::XmlPointVectorReaderContext:

### **9.79.1 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 separated 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 omitted, 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

- `unsigned long 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.*
- `PointVectorInputStream * m_Stream`  
*The inner stream object which streams the current point vector.*

### 9.79.2 Member Typedef Documentation

**9.79.2.1 typedef Schemas::ISO5436\_2::DataListType::Datum\_-sequence OpenGPS::XmlPointVectorReaderContext::StringList**

### 9.79.3 Constructor & Destructor Documentation

**9.79.3.1 XmlPointVectorReaderContext::XmlPointVectorReaderContext (const StringList \*const *pointVectorList*)**

Creates a new instance.

#### Parameters:

*pointVectorList* The list of point vectors to be streamed herein.

**9.79.3.2 XmlPointVectorReaderContext::~XmlPointVectorReaderContext () [virtual]**

Destroys this instance.

### 9.79.4 Member Function Documentation

**9.79.4.1 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 occurred.

**9.79.4.2 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](#).

**9.79.4.3 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](#).

**9.79.4.4 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.

## **9.79 OpenGPS::XmlPointVectorReaderContext Class Reference**

---

### **Parameters:**

*value* Contains the point value on success.

Implements [OpenGPS::PointVectorReaderContext](#).

**9.79.4.5 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](#).

**9.79.4.6 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](#).

**9.79.4.7 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](#).

**9.79.4.8 void XmlPointVectorReaderContext::Reset () [private]**

Resets the inner stream object.

**9.79.4.9 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.

**9.79.4.10 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](#).

**9.79.5 Member Data Documentation**

**9.79.5.1 unsigned long OpenGPS::XmlPointVectorReaderContext::m\_-Next [private]**

The index of the next point vector to be parsed.

**9.79.5.2 const StringList\* OpenGPS::XmlPointVectorReaderContext::m\_-PointVectorList [private]**

The inner list of all point vectors to be parsed.

**9.79.5.3 PointVectorInputStream\* OpenGPS::XmlPointVectorReaderContext::m\_-Stream [private]**

The inner stream object which streams the current point vector.

## **9.80 OpenGPS::XmlPointVectorWriterContext Class Reference**

The documentation for this class was generated from the following files:

- [xml\\_point\\_vector\\_reader\\_context.hxx](#)
- [xml\\_point\\_vector\\_reader\\_context.cxx](#)

### **9.80 OpenGPS::XmlPointVectorWriterContext Class Reference**

```
#include <xml_point_vector_writer_context.hxx>
```

Inheritance diagram for OpenGPS::XmlPointVectorWriterContext:

Collaboration diagram for OpenGPS::XmlPointVectorWriterContext:

### **9.80.1 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 separated 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 omitted, 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.*

### **9.80.2 Member Typedef Documentation**

#### **9.80.2.1 [typedef Schemas::ISO5436\\_2::DataListType::Datum\\_sequence OpenGPS::XmlPointVectorWriterContext::StringList](#)**

### **9.80.3 Constructor & Destructor Documentation**

#### **9.80.3.1 [XmlPointVectorWriterContext::XmlPointVectorWriterContext \(StringList \\*const pointVectorList\)](#)**

Creates a new instance.

##### **Parameters:**

*pointVectorList* The list of point vectors to be streamed herein.

**9.80.3.2 XmlPointVectorWriterContext::~XmlPointVectorWriterContext () [virtual]**

Destroys this instance.

**9.80.4 Member Function Documentation**

**9.80.4.1 void XmlPointVectorWriterContext::AppendSeparator () [protected, virtual]**

Appends the separator of two data point values.

**9.80.4.2 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 equals the string representation of a single point vector.

**Parameters:**

*value* Returns the content of the stream buffer.

**9.80.4.3 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 occurred.

Implements [OpenGPS::PointVectorWriterContext](#).

**9.80.4.4 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](#).

**9.80.4.5 void XmlPointVectorWriterContext::Reset () [private]**

Resets/empties the inner stream buffer.

## **9.80 OpenGPS::XmlPointVectorWriterContext Class Reference**

---

**9.80.4.6 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](#).

**9.80.4.7 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](#).

**9.80.4.8 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](#).

**9.80.4.9 void XmlPointVectorWriterContext::Write (const OGPS\_-Int32 \*const *value*) throw (...) [virtual]**

## **9.80 OpenGPS::XmlPointVectorWriterContext Class Reference**

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`.

### **9.80.4.10 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`.

## **9.80.5 Member Data Documentation**

### **9.80.5.1 OGPS\_Boolean OpenGPS::XmlPointVectorWriterContext::m\_-NeedsSeparator [private]**

TRUE if a separator needs to be added on the next call to `XmlPointVectorWriterContext::Write`, FALSE otherwise.

### **9.80.5.2 StringList\* OpenGPS::XmlPointVectorWriterContext::m\_-PointVectorList [private]**

The inner string list of point vectors written so far.

### **9.80.5.3 PointVectorOutputStream\* 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.

The documentation for this class was generated from the following files:

- `xml_point_vector_writer_context.hxx`
- `xml_point_vector_writer_context.cxx`

## 9.81 OpenGPS::ZipOutputStream Class Reference

```
#include <zip_stream_buffer.hxx>
```

Inheritance diagram for OpenGPS::ZipOutputStream:



```
graph TD; std::ostream --> ZipOutputStream
```

Collaboration diagram for OpenGPS::ZipOutputStream:



```
graph TD; std::ostream --- ZipOutputStream
```

### 9.81.1 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.*

#### 9.81.2 Member Typedef Documentation

##### 9.81.2.1 [typedef std::ostream OpenGPS::ZipOutputStream::BaseType](#)

Data type of the super class.

#### 9.81.3 Constructor & Destructor Documentation

##### 9.81.3.1 [ZipOutputStream::ZipOutputStream \(ZipStreamBuffer & buffer\)](#)

Creates a new instance.

#### Parameters:

*buffer* The buffer object that is streamed.

##### 9.81.3.2 [ZipOutputStream::~ZipOutputStream \(\)](#)

Destroys this instance.

#### 9.81.4 Member Function Documentation

##### 9.81.4.1 [ZipOutputStream::BaseType & ZipOutputStream::write \(const char \\* s\)](#)

Appends a string to the stream.

#### Parameters:

*s* The string to append.

The documentation for this class was generated from the following files:

- [zip\\_stream\\_buffer.hxx](#)
- [zip\\_stream\\_buffer.cxx](#)

## 9.82 OpenGPS::ZipStreamBuffer Class Reference

```
#include <zip_stream_buffer.hxx>
```

Collaboration diagram for OpenGPS::ZipStreamBuffer:

### 9.82.1 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 [xputn \(const char \\*s, std::streamsize n\)](#)  
*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.*

### 9.82.2 Member Typedef Documentation

#### 9.82.2.1 `typedef std::streambuf OpenGPS::ZipStreamBuffer::BaseType [private]`

Data type of the super class.

### 9.82.3 Constructor & Destructor Documentation

#### 9.82.3.1 `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 checksum data will be generated.

#### 9.82.3.2 `ZipStreamBuffer::~ZipStreamBuffer ()`

Destroys this instance.

### 9.82.4 Member Function Documentation

#### 9.82.4.1 `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.

**9.82.4.2 std::streamsize ZipStreamBuffer::xsputn (const char \* s, std::streamsize n) [protected, virtual]**

Overrides the super class.

### 9.82.5 Member Data Documentation

**9.82.5.1 zipFile OpenGPS::ZipStreamBuffer::m\_Handle [private]**

Handle to the zipFile where buffered data gets written to.

**9.82.5.2 md5\_context\* OpenGPS::ZipStreamBuffer::m\_Md5Context [private]**

The current state of md5 checksum processing.

The documentation for this class was generated from the following files:

- [zip\\_stream\\_buffer.hxx](#)
- [zip\\_stream\\_buffer.cxx](#)

## 10 openGPS ISO 5436-2 XML File Documentation

### 10.1 auto\_ptr\_types.hxx File Reference

#### 10.1.1 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< 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< PointVectorProxyContext >  
[OpenGPS::PointVectorProxyContextAutoPtr](#)
- 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.*

## 10.2 binary\_lsb\_point\_vector\_reader\_context.hxx 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.*

### 10.2.1 Define Documentation

#### 10.2.1.1 #define **\_CHECK\_ISGOOD\_AND\_THROW\_EXCEPTION**

**Value:**

```
if(!IsGood()) \
{ \
    throw OpenGPS::Exception( \
        OGPS_ExInvalidOperationException, \
        _EX_T("The underlying binary stream object became invalid."), \
        _EX_T("A read/write error occurred."), \
        _EX_T("OpenGPS::BinaryLSBPointVectorReaderContext")); \
}
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

**10.2.1.2 #define \_CHECK\_STREAM\_AND\_THROW\_-EXCEPTION**

**Value:**

```
if(!HasStream()) \
{ \
    throw OpenGPS::Exception( \
        OGPS_ExInvalidOperationException, \
        _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.

**10.3 binary\_lsb\_point\_vector\_reader\_context.hxx File**  
**Reference**

**10.3.1 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:

## 10.4 binary\_lsb\_point\_vector\_writer\_context.cxx File Reference

### Namespaces

- namespace [OpenGPS](#)

### Classes

- class [OpenGPS::BinaryLSBPointVectorReaderContext](#)

*Implements [OpenGPS::BinaryPointVectorReaderContext](#) for binary files to be parsed on machines reading in least significant byte order.*

## 10.4 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.*

#### **10.4.1 Define Documentation**

**10.4.1.1 #define \_CHECK\_ISGOOD\_AND\_THROW\_EXCEPTION**

**Value:**

```
if(!IsGood()) \
{ \
    throw OpenGPS::Exception( \
        OGPS_ExInvalidOperationException, \
        _EX_T("The underlying binary stream object became invalid."), \
        _EX_T("A read/write error occurred."), \
        _EX_T("OpenGPS::BinaryLSBPointVectorWriterContext")); \
}
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

**10.4.1.2 #define \_CHECK\_STREAM\_AND\_THROW\_EXCEPTION**

**Value:**

```
if(!HasStream()) \
{ \
    throw OpenGPS::Exception( \
        OGPS_ExInvalidOperationException, \
        _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.

### **10.5 binary\_lsb\_point\_vector\_writer\_context.hxx File** **Reference**

#### **10.5.1 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.*

## **10.6 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 "stdaafx.hxx"
```

Include dependency graph for binary\_msb\_point\_vector\_reader\_context.hxx:

## 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.*

### 10.6.1 Define Documentation

#### 10.6.1.1 #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 occurred."), \
        _EX_T("OpenGPS::BinaryMSBPointVectorReaderContext")); \
}
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

#### 10.6.1.2 #define `_CHECK_STREAM_AND_THROW_EXCEPTION`

**Value:**

```
if(!HasStream()) \
{ \
    throw OpenGPS::Exception( \
        OGPS_ExInvalidOperationException, \
        _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.

## **10.7 binary\_msб\_point\_vector\_reader\_context.hxx File Reference**

### **10.7.1 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\_msб\_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.*

## 10.8 binary\_msб\_point\_vector\_writer\_context.hxx File Reference

```
#include "binary_msб_point_vector_writer_context.hxx"
#include "point_vector_iostream.hxx"
#include <opengps/cxx/exceptions.hxx>
#include "environment.hxx"
#include "stdaafx.hxx"
```

Include dependency graph for binary\_msб\_point\_vector\_writer\_context.hxx:

## 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.*

### 10.8.1 Define Documentation

**10.8.1.1 #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 occurred."), \
        _EX_T("OpenGPS::BinaryMSBPointVectorWriterContext")); \
}
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

**10.8.1.2 #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.

**10.9 binary\_msб\_point\_vector\_writer\_context.hxx  
File Reference**

**10.9.1 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"
```

## 10.10 binary\_point\_vector\_reader\_context.cxx File Reference603

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.*

## **10.10 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 "stdaafx.hxx"
```

## 10.11 binary\_point\_vector\_reader\_context.hxx File Reference

Include dependency graph for binary\_point\_vector\_reader\_context.hxx:

### Defines

- #define `_CHECK_STREAM_AND_THROW_EXCEPTION`  
*Checks whether the underlying stream is valid.*

#### 10.10.1 Define Documentation

##### 10.10.1.1 #define `_CHECK_STREAM_AND_THROW_EXCEPTION`

**Value:**

```
if(!m_Stream) \
{ \
    throw OpenGPS::Exception( \
        OGPS_ExInvalidOperationException, \
        _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.

## 10.11 binary\_point\_vector\_reader\_context.hxx File Reference

### 10.11.1 Detailed Description

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

```
#include "point_vector_reader_context.hxx"
```

## 10.12 binary\_point\_vector\_writer\_context.cxx File Reference605

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.*

## **10.12 binary\_point\_vector\_writer\_context.cxx      File Reference**

```
#include "binary_point_vector_writer_context.hxx"
#include <opengps/cxx/exceptions.hxx>
#include "stdaafx.hxx"
```

## 10.12 binary\_point\_vector\_writer\_context.cxx File Reference606

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.*

#### 10.12.1 Define Documentation

##### 10.12.1.1 #define `_CHECK_ISGOOD_AND_THROW_EXCEPTION`

Value:

```
if(!IsGood()) \
{ \
    throw OpenGPS::Exception( \
        OGPS_ExInvalidOperationException, \
        _EX_T("The underlying binary stream object became invalid."), \
        _EX_T("A read/write error occurred."), \
        _EX_T("OpenGPS::BinaryPointVectorWriterContext")); \
}
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

##### 10.12.1.2 #define `_CHECK_STREAM_AND_THROW_EXCEPTION`

Value:

```
if(!HasStream()) \
```

## 10.13 binary\_point\_vector\_writer\_context.hxx File Reference607

```
{ \
throw OpenGPS::Exception( \
    OGPS_ExInvalidOperationException, \
    _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.

## **10.13 binary\_point\_vector\_writer\_context.hxx      File Reference**

### **10.13.1 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.*

## 10.14 data\_point.h File Reference

### 10.14.1 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.*

### 10.14.2 Typedef Documentation

#### 10.14.2.1 `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`.

### 10.14.3 Function Documentation

#### 10.14.3.1 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.

#### 10.14.3.2 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.

**10.14.3.3 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.

**10.14.3.4 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.

**10.14.3.5 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.

**10.14.3.6 \_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.

**10.14.3.7 \_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.

**10.14.3.8 \_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.

**10.14.3.9 \_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.

## 10.15 data\_point.hxx File Reference

### 10.15.1 Detailed Description

Typesafe representation of a single data point value.

The interface supports getting and setting an encapsulated data value of a pre-determined 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.*

## 10.16 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.hxx:

## 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.*

### 10.16.1 Function Documentation

#### 10.16.1.1 double [ogps\\_GetDouble](#) (const OGPS\_DataPointPtr *dataPoint*) throw ()

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.

#### 10.16.1.2 float [ogps\\_GetFloat](#) (const OGPS\_DataPointPtr *dataPoint*) throw ()

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.

**10.16.1.3 short ogps\_GetInt16 (const OGPS\_DataPointPtr *dataPoint*) throw ()**

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.

**10.16.1.4 int ogps\_GetInt32 (const OGPS\_DataPointPtr *dataPoint*) throw ()**

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.

**10.16.1.5 OGPS\_DataPointType ogps\_GetPointType (const OGPS\_DataPointPtr *dataPoint*) throw ()**

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.

**10.16.1.6 void ogps\_SetDouble (OGPS\_DataPointPtr const *dataPoint*, const OGPS\_Double *value*) throw ()**

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.

**10.16.1.7 void ogps\_SetFloat (OGPS\_DataPointPtr const *dataPoint*, const OGPS\_Float *value*) throw ()**

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.

**10.16.1.8 void ogps\_SetInt16 (OGPS\_DataPointPtr const *dataPoint*, const OGPS\_Int16 *value*) throw ()**

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.

**10.16.1.9 void ogps\_SetInt32 (OGPS\_DataPointPtr const *dataPoint*, const OGPS\_Int32 *value*) throw ()**

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.

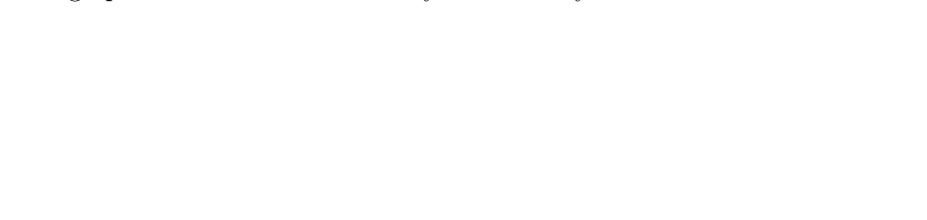
*value* The new value to be stored.

## 10.17 data\_point\_c.hxx File Reference

### 10.17.1 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\\_DATA\\_POINT](#)

*Encapsulates the C++ structure of a data point handle used internally within the C interface.*

### Typedefs

- typedef struct [\\_OGPS\\_DATA\\_POINT](#) [OGPS\\_DataPoint](#)
- typedef struct [\\_OGPS\\_DATA\\_POINT](#) \* [OGPS\\_DataPointPtr](#)

### 10.17.2 Typedef Documentation

#### 10.17.2.1 [typedef struct \\_OGPS\\_DATA\\_POINT OGPS\\_DataPoint](#)

#### 10.17.2.2 [typedef struct \\_OGPS\\_DATA\\_POINT \\* OGPS\\_DataPointPtr](#)

## 10.18 data\_point\_impl.cxx File Reference

```
#include "data_point_impl.hxx"
#include <opengps/cxx/exceptions.hxx>
#include "stdaafx.hxx"

Include dependency graph for data_point_impl.hxx:
```

## 10.19 data\_point\_impl.hxx File Reference

### 10.19.1 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 [OpenGPS::DataPointImpl::\\_OGPS\\_DATA\\_POINT\\_VALUE](#)  
*Typesafe storage for every possible type of point data.*

## 10.20 data\_point\_parser.hxx File Reference

### 10.20.1 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.*

## 10.21 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 "stdaafx.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.*

### 10.21.1 Define Documentation

#### 10.21.1.1 `#define _CHECK_BUFFER_AND_THROW_READ_EXCEPTION`

**Value:**

```
if(!m_Buffer) \
{ \
    throw OpenGPS::Exception( \
        OGPS_ExInvalidOperationException, \
        _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 read") \
        _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.

#### 10.21.1.2 `#define _CHECK_BUFFER_AND_THROW_WRITE_EXCEPTION`

**Value:**

```
if(!m_Buffer) \
{ \
    throw OpenGPS::Exception( \
        OGPS_ExInvalidOperationException, \
        _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 write") \
        _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.

## 10.22 data\_point\_type.h File Reference

### 10.22.1 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:

### TypeDefs

- `typedef enum _OGPS_DATA_POINT_TYPE OGPS_DataPointType`

### Enumerations

- `enum _OGPS_DATA_POINT_TYPE {`  
`OGPS_MissingPointType,       OGPS_Int16PointType,       OGPS_-`  
`Int32PointType, OGPS_FloatPointType,`  
`OGPS_DoublePointType }`

*Possible types of data stored in an `OGPS_DataPointPtr` instance.*

#### 10.22.2 Typedef Documentation

##### 10.22.2.1 `typedef enum _OGPS_DATA_POINT_TYPE OGPS_-DataPointType`

#### 10.22.3 Enumeration Type Documentation

##### 10.22.3.1 `enum _OGPS_DATA_POINT_TYPE`

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`.

### 10.23 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 "stdaafx.hxx"
```

Include dependency graph for double\_data\_point\_parser.hxx:

## 10.24 double\_data\_point\_parser.hxx File Reference

### 10.24.1 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.*

## 10.25 double\_point\_buffer.cxx File Reference

```
#include "double_point_buffer.hxx"
```

```
#include "stdafx.hxx"
```

Include dependency graph for double\_point\_buffer.hxx:

## 10.26 double\_point\_buffer.hxx File Reference

### 10.26.1 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.*

## 10.27 Doxygen.cpp File Reference

### 10.27.1 Detailed Description

Title page of documentation, no source code.

## Namespaces

- namespace [std](#)

## 10.28 environment.cxx File Reference

```
#include "environment.hxx"  
#include "stdafx.hxx"
```

Include dependency graph for environment.hxx:

### Defines

- #define \_OGPS\_DOUBLE\_SIZE 8
- #define \_OGPS\_FLOAT\_SIZE 4
- #define \_OGPS\_INT\_SIZE 4
- #define \_OGPS\_SHORT\_SIZE 2

#### 10.28.1 Define Documentation

10.28.1.1 #define \_OGPS\_DOUBLE\_SIZE 8

10.28.1.2 #define \_OGPS\_FLOAT\_SIZE 4

10.28.1.3 #define \_OGPS\_INT\_SIZE 4

10.28.1.4 #define \_OGPS\_SHORT\_SIZE 2

### 10.29 environment.hxx File Reference

#### 10.29.1 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.*

## 10.30 exceptions.cxx File Reference

```
#include <opengps/cxx/exceptions.hxx>
#include <opengps/cxx/string.hxx>
#include "stdafx.hxx"
```

Include dependency graph for exceptions.hxx:

## 10.31 exceptions.hxx File Reference

### 10.31.1 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.*

## 10.32 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 "stdaafx.hxx"
```

Include dependency graph for float\_data\_point\_parser.cxx:

## 10.33 float\_data\_point\_parser.hxx File Reference

### 10.33.1 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.*

## 10.34 float\_point\_buffer.cxx File Reference

```
#include "float_point_buffer.hxx"
#include "stdaafx.hxx"
```

Include dependency graph for float\_point\_buffer.hxx:

## 10.35 float\_point\_buffer.hxx File Reference

### 10.35.1 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.*

## 10.36 info.cxx File Reference

```
#include <opengps/cxx/info.hxx>
#include <opengps/cxx/string.hxx>
#include <iostream>
#include "stdaafx.hxx"
```

Include dependency graph for info.cxx:

## 10.37 info.h File Reference

### 10.37.1 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.*

### 10.37.2 Function Documentation

#### 10.37.2.1 `_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.

**10.37.2.2 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.

**10.37.2.3 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.

**10.37.2.4 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.

**10.37.2.5 \_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.

**10.37.2.6 \_OPENGPS\_EXPORT void ogps\_PrintCopyrightInfo ()**

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

**10.37.2.7 \_OPENGPS\_EXPORT void ogps\_PrintLicenseInfo ()**

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

**10.37.2.8 \_OPENGPS\_EXPORT void ogps\_PrintVersionInfo ()**

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

## 10.38 info.hxx File Reference

### 10.38.1 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.*

## 10.39 info\_c.hxx 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\_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.*

### 10.39.1 Function Documentation

**10.39.1.1 size\_t ogps\_GetAboutInfo (OGPS\_Character \*const *text*, const size\_t *size*) throw ()**

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.

**10.39.1.2 size\_t ogps\_GetCopyrightInfo (OGPS\_Character \*const *text*, const size\_t *size*) throw ()**

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.

**10.39.1.3 size\_t ogps\_GetLicenseInfo (OGPS\_Character \*const *text*, const size\_t *size*) throw ()**

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.

**10.39.1.4 size\_t ogps\_GetNameInfo (OGPS\_Character \*const *text*, const size\_t *size*) throw ()**

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.

**10.39.1.5 size\_t ogps\_GetVersionInfo (OGPS\_Character \*const *text*, const size\_t *size*) throw ()**

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.

**10.39.1.6 void ogps\_PrintCopyrightInfo () throw ()**

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

**10.39.1.7 void ogps\_PrintLicenseInfo () throw ()**

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

**10.39.1.8 void ogps\_PrintVersionInfo () throw ()**

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

## 10.40 inline\_validity.cxx File Reference

```
#include <limits>
#include "inline_validity.hxx"
```

```
#include "point_buffer.hxx"
#include "stdafx.hxx"

Include dependency graph for inline_validity.hxx:
```

## 10.41 inline\_validity.hxx File Reference

### 10.41.1 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.*

## 10.42 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 "stdaafx.hxx"
```

Include dependency graph for int16\_data\_point\_parser.cxx:

## 10.43 int16\_data\_point\_parser.hxx File Reference

### 10.43.1 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.*

## 10.44 int16\_point\_buffer.cxx File Reference

```
#include "int16_point_buffer.hxx"
#include "stdafx.hxx"
```

Include dependency graph for int16\_point\_buffer.hxx:

## 10.45 int16\_point\_buffer.hxx File Reference

### 10.45.1 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.*

## 10.46 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 "stdaafx.hxx"
```

Include dependency graph for int32\_data\_point\_parser.cxx:

## 10.47 int32\_data\_point\_parser.hxx File Reference

### 10.47.1 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.*

## 10.48 int32\_point\_buffer.cxx File Reference

```
#include "int32_point_buffer.hxx"
#include "stdafx.hxx"
```

Include dependency graph for int32\_point\_buffer.hxx:

## 10.49 int32\_point\_buffer.hxx File Reference

### 10.49.1 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.*

## 10.50 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:

## 10.51 iso5436\_2.h File Reference

### 10.51.1 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 unsigned long 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 void ogps_GetListPoint (const OGPS_ISO5436_2Handle handle, const unsigned long 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 unsigned long u, const unsigned long v, const unsigned long 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_GetMatrixPoint (const OGPS_ISO5436_2Handle handle, const unsigned long u, const unsigned long v, const unsigned long 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 vendor-specific data from the current archive to a given file location.*
- `_OPENGPS_EXPORT OGPS_Boolean ogps_IsMatrixCoordValid (const OGPS_ISO5436_2Handle handle, const unsigned long u, const unsigned long v, const unsigned long 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 unsigned long 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 unsigned long u, const unsigned long v, const unsigned long 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.*

### 10.51.2 Typedef Documentation

#### 10.51.2.1 `typedef struct _OPGPS_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`.

### 10.51.3 Function Documentation

#### 10.51.3.1 `_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.

**10.51.3.2 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.

**10.51.3.3 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.

**10.51.3.4 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.

**10.51.3.5 OPENGPS\_EXPORT void ogps\_GetListCoord (const OGPS\_ISO5436\_2Handle handle, const unsigned long 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.

**10.51.3.6 OPENGPS\_EXPORT void ogps\_GetListPoint (const OGPS\_ISO5436\_2Handle *handle*, const unsigned long *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.

**10.51.3.7 OPENGPS\_EXPORT void ogps\_GetMatrixCoord (const OGPS\_ISO5436\_2Handle *handle*, const unsigned long *u*, const unsigned long *v*, const unsigned long *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.

---

**10.51.3.8 OPENGPS\_EXPORT void ogps\_GetMatrixPoint  
(const OGPS\_ISO5436\_2Handle *handle*, const unsigned long *u*,  
const unsigned long *v*, const unsigned long *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.

**10.51.3.9 OPENGPS\_EXPORT OGPS\_Boolean ogps\_-GetVendorSpecific (const OGPS\_ISO5436\_2Handle *handle*, const OGPS\_Character \* *vendorURI*, const OGPS\_Character \* *fileName*, const OGPS\_Character \* *targetPath*)**

Extracts vendor-specific data from the current archive to a given file location.

If the current X3P archive contains vendor-specific 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.

**10.51.3.10 OPENGPS\_EXPORT OGPS\_Boolean ogps\_-IsMatrixCoordValid (const OGPS\_ISO5436\_2Handle *handle*, const unsigned long *u*, const unsigned long *v*, const unsigned long *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.

**10.51.3.11 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.

**10.51.3.12 OPENGPS\_EXPORT void ogps\_SetListPoint (const OGPS\_ISO5436\_2Handle *handle*, const unsigned long *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.

**10.51.3.13 OPENGPS\_EXPORT void ogps\_SetMatrixPoint (const OGPS\_ISO5436\_2Handle *handle*, const unsigned long *u*, const unsigned long *v*, const unsigned long *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.

---

**10.51.3.14 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.

## 10.52 iso5436\_2.hxx File Reference

### 10.52.1 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.*

## 10.53 iso5436\_2\_c.hxx 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.hxx:

## 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 unsigned long 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 unsigned long 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.*

- void `ogps_GetListPoint` (const `OGPS_ISO5436_2Handle` handle, const unsigned long 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 unsigned long u, const unsigned long v, const unsigned long 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_GetMatrixPoint` (const `OGPS_ISO5436_2Handle` handle, const unsigned long u, const unsigned long v, const unsigned long 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_IsMatrixCoordValid` (const `OGPS_ISO5436_2Handle` handle, const unsigned long u, const unsigned long v, const unsigned long 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 unsigned long 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 unsigned long u, const unsigned long v, const unsigned long 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.*

### 10.53.1 Function Documentation

**10.53.1.1 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.

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.

**10.53.1.2 void ogps\_CloseISO5436\_2 (OGPS\_ISO5436\_2Handle \* *handle*) throw ()**

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.

**10.53.1.3 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 unsigned long *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. The given object instance must be valid.

*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.

```
10.53.1.4 OGPS_ISO5436_2Handle
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 & ma-
trixDimension, 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.  
The given object instance must be valid.

***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.

**10.53.1.5 OGPS\_PointIteratorPtr ogps\_CreateNextPointIterator  
(const OGPS\_ISO5436\_2Handle *handle*) throw ()**

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.

**10.53.1.6 OGPS\_PointIteratorPtr ogps\_CreatePrevPointIterator  
(const OGPS\_ISO5436\_2Handle *handle*) throw ()**

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.

**10.53.1.7 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.

**10.53.1.8 void ogps\_GetListCoord (const OGPS\_ISO5436\_2Handle handle, const unsigned long 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.

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.

#### 10.53.1.9 void ogps\_GetListPoint (const OGPS\_ISO5436\_2Handle *handle*, const unsigned long *index*, OGPS\_PointVectorPtr *vector*) throw ()

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.

#### 10.53.1.10 void ogps\_GetMatrixCoord (const OGPS\_ISO5436\_2Handle *handle*, const unsigned long *u*, const unsigned long *v*, const unsigned long *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.

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.

**10.53.1.11 void ogps\_GetMatrixPoint (const OGPS\_ISO5436\_-2Handle *handle*, const unsigned long *u*, const unsigned long *v*, const unsigned long *w*, OGPS\_PointVectorPtr *vector*) throw ()**

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.

```
10.53.1.12 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.

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.

```
10.53.1.13 OGPS_Boolean ogps_IsMatrixCoordValid (const
OGPS_ISO5436_2Handle handle, const unsigned long u, const
unsigned long v, const unsigned long w) throw ()
```

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.

**10.53.1.14 OGPS\_ISO5436\_2Handle      [ogps\\_OpenISO5436\\_2](#)**  
**(const OGPS\_Character \*const *file*, const OGPS\_Character \*const *temp* = NULL, const OGPS\_Boolean *readOnly* = FALSE) throw ()**

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 subsequent 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.

```
10.53.1.15 void ogps_SetListPoint (const OGPS_ISO5436_-
2Handle handle, const unsigned long index, const OGPS_-
PointVectorPtr vector) throw ()
```

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.

```
10.53.1.16 void ogps_SetMatrixPoint (const OGPS_ISO5436_-
2Handle handle, const unsigned long u, const unsigned long v, const
unsigned long w, const OGPS_PointVectorPtr vector) throw ()
```

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.

**10.53.1.17 void ogps\_WriteISO5436\_2 (const OGPS\_ISO5436\_2Handle *handle*, const int *compressionLevel* = -1) throw ()**

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.

## 10.54 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 "stdaafx.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

**10.54.1 Define Documentation**

10.54.1.1 #define `_OPENGPS_FILE_URI_PREF` \_T("file:///")

10.54.1.2 #define `_OPENGPS_ZIP_CHUNK_MAX` (256\*1024)

**10.54.2 Typedef Documentation**

10.54.2.1 `typedef std::auto_ptr<PointVectorReaderContext> PointVectorReaderContextAutoPtr`

10.54.2.2 `typedef std::auto_ptr<PointVectorWriterContext> PointVectorWriterContextAutoPtr`

**10.55 iso5436\_2\_container.hxx File Reference****10.55.1 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.*

## 10.56 iso5436\_2\_handle.hxx File Reference

### 10.56.1 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 unsigned long 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.*

### 10.56.2 Function Documentation

```
10.56.2.1 _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
unsigned long 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. The given object instance must be valid.

***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.

```
10.56.2.2 _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. The given object instance must be valid.

***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.

---

```
10.56.2.3 _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:

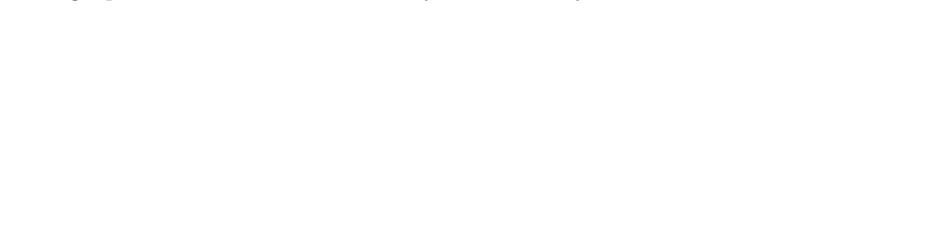
Returns ISO5436\_2 XML document handle or NULL on failure.

## 10.57 iso5436\_2\_handle\_c.hxx File Reference

### 10.57.1 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\\_HANDLE](#)

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

#### Typedefs

- typedef struct [\\_OGPS\\_ISO5436\\_2\\_HANDLE](#) [OGPS\\_ISO5436\\_2](#)
- typedef struct [\\_OGPS\\_ISO5436\\_2\\_HANDLE](#) \* [OGPS\\_ISO5436\\_2Handle](#)

### 10.57.2 Typedef Documentation

#### 10.57.2.1 `typedef struct _OGPS_ISO5436_2_HANDLE OGPS_ISO5436_2`

#### 10.57.2.2 `typedef struct _OGPS_ISO5436_2_HANDLE * OGPS_ISO5436_2Handle`

## 10.58 ISO5436\_2\_XML\_Demo.cxx File Reference

```
#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 <string>
#include <iostream>
#include <ostream>
#include <fstream>
#include <cstdlib>
#include <ctime>
#include <limits>
#include <tchar.h>
```

Include dependency graph for ISO5436\_2\_XML\_Demo.hxx:

## Functions

- int \_cdecl \_tmain (int argc, \_TCHAR \*argv[])
- void mediumComplexExample (OpenGPS::String fileName)
- void performanceDouble (OpenGPS::String fileName, unsigned long dimension, OGPS\_Boolean binary)
- void performanceInt16 (OpenGPS::String fileName, unsigned long dimension, OGPS\_Boolean binary)
- void readonlyExample (OpenGPS::String fileName)
- void readonlyExample2 (OpenGPS::String fileName)
- void readonlyExample3 (OpenGPS::String fileName)
- void readonlyExample4 (OpenGPS::String fileName)
- void simpleExample (OpenGPS::String fileName)

### 10.58.1 Function Documentation

#### 10.58.1.1 int \_cdecl \_tmain (int *argc*, \_TCHAR \* *argv*[])

#### 10.58.1.2 void mediumComplexExample (OpenGPS::String *fileName*)

10.58.1.3 void performanceDouble (OpenGPS::String *fileName*, unsigned long *dimension*, OGPS\_Boolean *binary*)

10.58.1.4 void performanceInt16 (OpenGPS::String *fileName*, unsigned long *dimension*, OGPS\_Boolean *binary*)

10.58.1.5 void readonlyExample (OpenGPS::String *fileName*)

10.58.1.6 void readonlyExample2 (OpenGPS::String *fileName*)

10.58.1.7 void readonlyExample3 (OpenGPS::String *fileName*)

10.58.1.8 void readonlyExample4 (OpenGPS::String *fileName*)

10.58.1.9 void simpleExample (OpenGPS::String *fileName*)

## 10.59 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 <iostream>
#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

- bool `OpenGPS::Schemas::ISO5436_2::operator!= (const RotationType &x, const RotationType &y)`
- bool `OpenGPS::Schemas::ISO5436_2::operator!= (const MatrixDimensionType &x, const MatrixDimensionType &y)`
- bool `OpenGPS::Schemas::ISO5436_2::operator!= (const DataLinkType &x, const DataLinkType &y)`
- bool `OpenGPS::Schemas::ISO5436_2::operator!= (const DataListType &x, const DataListType &y)`
- bool `OpenGPS::Schemas::ISO5436_2::operator!= (const ProbingSystemType &x, const ProbingSystemType &y)`
- bool `OpenGPS::Schemas::ISO5436_2::operator!= (const InstrumentType &x, const InstrumentType &y)`
- bool `OpenGPS::Schemas::ISO5436_2::operator!= (const AxisDescriptionType &x, const AxisDescriptionType &y)`
- bool `OpenGPS::Schemas::ISO5436_2::operator!= (const AxesType &x, const AxesType &y)`
- bool `OpenGPS::Schemas::ISO5436_2::operator!= (const Record4Type &x, const Record4Type &y)`
- bool `OpenGPS::Schemas::ISO5436_2::operator!= (const Record3Type &x, const Record3Type &y)`
- bool `OpenGPS::Schemas::ISO5436_2::operator!= (const Record2Type &x, const Record2Type &y)`
- bool `OpenGPS::Schemas::ISO5436_2::operator!= (const ISO5436_2Type &x, const ISO5436_2Type &y)`
- bool `OpenGPS::Schemas::ISO5436_2::operator!= (const Record1Type &x, const Record1Type &y)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &l, const Datum &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &a, const Datum &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const Datum &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &l, const Type &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &a, const Type &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const Type &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &l, const DataType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &a, const DataType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const DataType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &l, const AxisType &i)`

- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &a, const AxisType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const AxisType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_- stream< wchar_t > &l, const FeatureType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &a, const FeatureType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const FeatureType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_- stream< wchar_t > &l, const RotationMatrixElementType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &a, const RotationMatrixElementType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const RotationMatrixElementType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const RotationType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const MatrixDimensionType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const DataLinkType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const DataListType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const ProbingSystemType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const InstrumentType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const AxisDescriptionType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const AxesType &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const Record4Type &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const Record3Type &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const Record2Type &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const ISO5436_2Type &i)`
- void `OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const Record1Type &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const Datum &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const Type &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, Type::value i)`

- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const DataType &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, DataType::value i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const AxisType &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, AxisType::value i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const FeatureType &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const RotationMatrixElementType &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const RotationType &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const MatrixDimensionType &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const DataLinkType &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const DataListType &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const ProbingSystemType &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const InstrumentType &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const AxisDescriptionType &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const AxesType &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const Record4Type &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const Record3Type &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const Record2Type &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const ISO5436_2Type &i)`
- ::std::wostream & `OpenGPS::Schemas::ISO5436_2::operator<<`  
    `(::std::wostream &o, const Record1Type &i)`
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const RotationType  
    &x, const RotationType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const MatrixDimensionType  
    &x, const MatrixDimensionType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const DataLinkType  
    &x, const DataLinkType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const DataListType  
    &x, const DataListType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const ProbingSystemType  
    &x, const ProbingSystemType &y)

- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const InstrumentType &x, const InstrumentType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const AxisDescriptionType &x, const AxisDescriptionType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const AxesType &x, const AxesType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const Record4Type &x, const Record4Type &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const Record3Type &x, const Record3Type &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const Record2Type &x, const Record2Type &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const ISO5436\_2Type &x, const ISO5436\_2Type &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const Record1Type &x, const Record1Type &y)

#### Serialization functions for the ISO5436\_2 document root.

*The only global element: The root node*

- ::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.*
- 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.*
- 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.*
- 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.*
- 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.*

- 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.*
- 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.*
- 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*

- ::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.*
- ::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.*
- ::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.*
- ::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.*

- `::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.*

- `::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.*

- `::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.*

- `::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.*

- `::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.*

- `::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.*

- `::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.*

- `::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.*

- `::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.*

- `::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.*

## 10.60 iso5436\_2\_xsd.hxx File Reference

### 10.60.1 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 <iostream>
#include <xercesc/dom/DOMDocument.hpp>
#include <xercesc/dom/DOMInputSource.hpp>
#include <xercesc/dom/DOMEErrorHandler.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.*

## 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

- `bool OpenGPS::Schemas::ISO5436_2::operator!= (const RotationType &x, const RotationType &y)`
- `bool OpenGPS::Schemas::ISO5436_2::operator!= (const MatrixDimensionType &x, const MatrixDimensionType &y)`
- `bool OpenGPS::Schemas::ISO5436_2::operator!= (const DataLinkType &x, const DataLinkType &y)`
- `bool OpenGPS::Schemas::ISO5436_2::operator!= (const DataListType &x, const DataListType &y)`
- `bool OpenGPS::Schemas::ISO5436_2::operator!= (const ProbingSystemType &x, const ProbingSystemType &y)`
- `bool OpenGPS::Schemas::ISO5436_2::operator!= (const InstrumentType &x, const InstrumentType &y)`
- `bool OpenGPS::Schemas::ISO5436_2::operator!= (const AxisDescriptionType &x, const AxisDescriptionType &y)`
- `bool OpenGPS::Schemas::ISO5436_2::operator!= (const AxesType &x, const AxesType &y)`
- `bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record4Type &x, const Record4Type &y)`
- `bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record3Type &x, const Record3Type &y)`
- `bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record2Type &x, const Record2Type &y)`
- `bool OpenGPS::Schemas::ISO5436_2::operator!= (const ISO5436_2Type &x, const ISO5436_2Type &y)`
- `bool OpenGPS::Schemas::ISO5436_2::operator!= (const Record1Type &x, const Record1Type &y)`
- `void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &l, const Datum &i)`
- `void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &a, const Datum &i)`
- `void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const Datum &i)`
- `void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &l, const Type &i)`
- `void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMAttr &a, const Type &i)`
- `void OpenGPS::Schemas::ISO5436_2::operator<< (::xercesc::DOMElement &e, const Type &i)`
- `void OpenGPS::Schemas::ISO5436_2::operator<< (::xsd::cxx::tree::list_stream< wchar_t > &l, const DataType &i)`

- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMAttr &a, const DataType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const DataType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xsd::cxx::tree::list\_- stream< wchar\_t > &l, const AxisType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMAttr &a, const AxisType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const AxisType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xsd::cxx::tree::list\_- stream< wchar\_t > &l, const FeatureType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMAttr &a, const FeatureType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const FeatureType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xsd::cxx::tree::list\_- stream< wchar\_t > &l, const RotationMatrixElementType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMAttr &a, const RotationMatrixElementType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const RotationMatrixElementType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const RotationType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const MatrixDimensionType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const DataLinkType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const DataListType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const ProbingSystemType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const InstrumentType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const AxisDescriptionType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const AxesType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const Record4Type &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const Record3Type &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const Record2Type &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const ISO5436\_2Type &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const Record1Type &i)

- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const Datum &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const Type &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, Type::value i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const DataType &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, DataType::value i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const AxisType &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, AxisType::value i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const FeatureType &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const RotationMatrixElementType &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const RotationType &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const MatrixDimensionType &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const DataLinkType &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const DataListType &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const ProbingSystemType &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const InstrumentType &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const AxisDescriptionType &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const AxesType &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const Record4Type &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const Record3Type &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const Record2Type &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const ISO5436_2Type &i)`
- `::std::wostream & OpenGPS::Schemas::ISO5436_2::operator<< (::std::wostream &o, const Record1Type &i)`
- `bool OpenGPS::Schemas::ISO5436_2::operator== (const RotationType &x, const RotationType &y)`
- `bool OpenGPS::Schemas::ISO5436_2::operator== (const MatrixDimensionType &x, const MatrixDimensionType &y)`

- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const DataLinkType &x, const DataLinkType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const DataListType &x, const DataListType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const ProbingSystemType &x, const ProbingSystemType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const InstrumentType &x, const InstrumentType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const AxisDescriptionType &x, const AxisDescriptionType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const AxesType &x, const AxesType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const Record4Type &x, const Record4Type &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const Record3Type &x, const Record3Type &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const Record2Type &x, const Record2Type &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const ISO5436\_2Type &x, const ISO5436\_2Type &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const Record1Type &x, const Record1Type &y)

#### Serialization functions for the ISO5436\_2 document root.

*The only global element: The root node*

- ::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.*
- 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.*
- 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.*
- 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.*

- 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.*
- 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.*
- 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.*
- 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*

- `::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.*
- `::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.*
- `::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.*

- `::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.*
- `::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.*
- `::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.*
- `::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.*
- `::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.*
- `::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.*
- `::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.*
- `::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 with properties.*

*Parse a standard input stream.*

- `::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.*

- `::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.*

- `::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`

## 10.61 iso5436\_2\_xsd.hxx File Reference

### 10.61.1 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 <iostream>
#include <xercesc/dom/DOMDocument.hpp>
#include <xercesc/dom/DOMInputSource.hpp>
#include <xercesc/dom/DOMEErrorHandler.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.*

## Functions

- bool `OpenGPS::Schemas::ISO5436_2::operator!=` (const RotationType &x, const RotationType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator!=` (const MatrixDimensionType &x, const MatrixDimensionType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator!=` (const DataLinkType &x, const DataLinkType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator!=` (const DataListType &x, const DataListType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator!=` (const ProbingSystemType &x, const ProbingSystemType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator!=` (const InstrumentType &x, const InstrumentType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator!=` (const AxisDescriptionType &x, const AxisDescriptionType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator!=` (const AxesType &x, const AxesType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator!=` (const Record4Type &x, const Record4Type &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator!=` (const Record3Type &x, const Record3Type &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator!=` (const Record2Type &x, const Record2Type &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator!=` (const ISO5436\_2Type &x, const ISO5436\_2Type &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator!=` (const Record1Type &x, const Record1Type &y)
- void `OpenGPS::Schemas::ISO5436_2::operator<<` (::xsd::cxx::tree::list\_-stream< wchar\_t > &l, const Datum &i)
- void `OpenGPS::Schemas::ISO5436_2::operator<<` (::xercesc::DOMAttr &a, const Datum &i)
- void `OpenGPS::Schemas::ISO5436_2::operator<<` (::xercesc::DOMElement &e, const Datum &i)
- void `OpenGPS::Schemas::ISO5436_2::operator<<` (::xsd::cxx::tree::list\_-stream< wchar\_t > &l, const Type &i)
- void `OpenGPS::Schemas::ISO5436_2::operator<<` (::xercesc::DOMAttr &a, const Type &i)
- void `OpenGPS::Schemas::ISO5436_2::operator<<` (::xercesc::DOMElement &e, const Type &i)
- void `OpenGPS::Schemas::ISO5436_2::operator<<` (::xsd::cxx::tree::list\_-stream< wchar\_t > &l, const DataType &i)

- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMAttr &a, const DataType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const DataType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xsd::cxx::tree::list\_- stream< wchar\_t > &l, const AxisType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMAttr &a, const AxisType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const AxisType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xsd::cxx::tree::list\_- stream< wchar\_t > &l, const FeatureType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMAttr &a, const FeatureType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const FeatureType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xsd::cxx::tree::list\_- stream< wchar\_t > &l, const RotationMatrixElementType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMAttr &a, const RotationMatrixElementType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const RotationMatrixElementType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const RotationType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const MatrixDimensionType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const DataLinkType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const DataListType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const ProbingSystemType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const InstrumentType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const AxisDescriptionType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const AxesType &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const Record4Type &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const Record3Type &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const Record2Type &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const ISO5436\_2Type &i)
- void OpenGPS::Schemas::ISO5436\_2::operator<< (::xercesc::DOMElement &e, const Record1Type &i)

- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const Datum &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const Type &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, Type::value i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const DataType &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, DataType::value i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const AxisType &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, AxisType::value i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const FeatureType &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const RotationMatrixElementType &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const RotationType &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const MatrixDimensionType &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const DataLinkType &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const DataListType &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const ProbingSystemType &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const InstrumentType &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const AxisDescriptionType &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const AxesType &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const Record4Type &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const Record3Type &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const Record2Type &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const ISO5436\_2Type &i)
- ::std::wostream & OpenGPS::Schemas::ISO5436\_2::operator<< (::std::wostream &o, const Record1Type &i)
- bool OpenGPS::Schemas::ISO5436\_2::operator== (const RotationType &x, const RotationType &y)
- bool OpenGPS::Schemas::ISO5436\_2::operator== (const MatrixDimensionType &x, const MatrixDimensionType &y)

- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const DataLinkType &x, const DataLinkType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const DataListType &x, const DataListType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const ProbingSystemType &x, const ProbingSystemType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const InstrumentType &x, const InstrumentType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const AxisDescriptionType &x, const AxisDescriptionType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const AxesType &x, const AxesType &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const Record4Type &x, const Record4Type &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const Record3Type &x, const Record3Type &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const Record2Type &x, const Record2Type &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const ISO5436\_2Type &x, const ISO5436\_2Type &y)
- bool `OpenGPS::Schemas::ISO5436_2::operator==` (const Record1Type &x, const Record1Type &y)

#### Serialization functions for the ISO5436\_2 document root.

*The only global element: The root node*

- ::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.*
- 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.*
- 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.*
- 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.*

- 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.*
- 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.*
- 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.*
- 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*

- `::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.*
- `::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.*
- `::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.*

- `::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.*
- `::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.*
- `::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.*
- `::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.*
- `::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.*
- `::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.*
- `::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.*
- `::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.*

*Parse a standard input stream.*

- `::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.*

- `::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.*

- `::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.*

## 10.62 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])  
 $Output = MD5(\text{ }input \text{ buffer} \text{ }).$
- int `md5_file` (const char \*path, unsigned char output[16])  
 $Output = MD5(\text{ }file \text{ contents} \text{ }).$
- void `md5_finish` (`md5_context` \*ctx, unsigned char output[16])  
 $MD5 \text{ final digest}.$
- void `md5_hmac` (unsigned char \*key, int keylen, unsigned char \*input, int ilen, unsigned char output[16])  
 $Output = HMAC-MD5(\text{ }hmac \text{ key}, \text{ }input \text{ buffer} \text{ }).$
- void `md5_hmac_finish` (`md5_context` \*ctx, unsigned char output[16])  
 $MD5 \text{ HMAC final digest}.$
- void `md5_hmac_starts` (`md5_context` \*ctx, unsigned char \*key, int keylen)  
 $MD5 \text{ HMAC context setup}.$
- void `md5_hmac_update` (`md5_context` \*ctx, unsigned char \*input, int ilen)  
 $MD5 \text{ HMAC process buffer}.$
- static void `md5_process` (`md5_context` \*ctx, const unsigned char data[64])
- void `md5_starts` (`md5_context` \*ctx)  
 $MD5 \text{ context setup}.$
- void `md5_update` (`md5_context` \*ctx, const unsigned char \*input, int ilen)  
 $MD5 \text{ process buffer}.$

## Variables

- static const unsigned char `md5_padding` [64]

### 10.62.1 Define Documentation

**10.62.1.1 #define F(x, y, z) (y ^ (x | ~z))**

**10.62.1.2 #define F(x, y, z) (x ^ y ^ z)**

10.62.1.3 `#define F(x, y, z) (y ^ (z & (x ^ y)))`

10.62.1.4 `#define F(x, y, z) (z ^ (x & (y ^ z)))`

10.62.1.5 `#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 );      \
}
```

10.62.1.6 `#define P(a, b, c, d, k, s, t)`

**Value:**

```
{
    a += F(b,c,d) + X[k] + t; a = S(a,s) + b;      \
}
```

10.62.1.7 `#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 );     \
}
```

10.62.1.8 `#define S(x, n) ((x << n) | ((x & 0xFFFFFFFF) >> (32 - n)))`

10.62.1.9 `#define XYSSL MD5 C`

## 10.62.2 Function Documentation

10.62.2.1 `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

#### 10.62.2.2 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

#### 10.62.2.3 void md5\_finish (md5\_context \* *ctx*, unsigned char *output*[16])

MD5 final digest.

**Parameters:**

*ctx* MD5 context  
*output* MD5 checksum result

#### 10.62.2.4 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

#### 10.62.2.5 void md5\_hmac\_finish (md5\_context \* *ctx*, unsigned char *output*[16])

MD5 HMAC final digest.

**Parameters:**

*ctx* HMAC context  
*output* MD5 HMAC checksum result

**10.62.2.6 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

**10.62.2.7 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

**10.62.2.8 static void md5\_process (md5\_context \* *ctx*, const unsigned char *data*[64]) [static]**

**10.62.2.9 void md5\_starts (md5\_context \* *ctx*)**

MD5 context setup.

**Parameters:**

*ctx* context to be initialized

**10.62.2.10 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

### 10.62.3 Variable Documentation

#### 10.62.3.1 const unsigned char md5\_padding[64] [static]

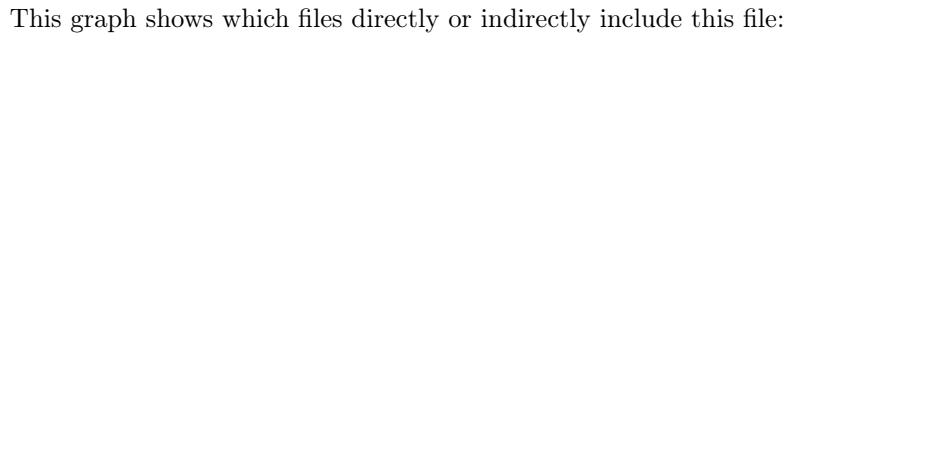
Initial value:

```
{  
    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  
}
```

## 10.63 md5.h File Reference

### 10.63.1 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.*

### 10.63.2 Function Documentation

**10.63.2.1 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

**10.63.2.2 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

**10.63.2.3 void md5\_finish (md5\_context \* *ctx*, unsigned char *output*[16])**

MD5 final digest.

**Parameters:**

*ctx* MD5 context

*output* MD5 checksum result

**10.63.2.4 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

**10.63.2.5 void md5\_hmac\_finish (md5\_context \* *ctx*, unsigned char *output*[16])**

MD5 HMAC final digest.

**Parameters:**

*ctx* HMAC context

*output* MD5 HMAC checksum result

**10.63.2.6 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

**10.63.2.7 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

**10.63.2.8 int md5\_self\_test (int *verbose*)**

Checkup routine.

**Returns:**

0 if successful, or 1 if the test failed

**10.63.2.9 void md5\_starts (md5\_context \* *ctx*)**

MD5 context setup.

**Parameters:**

*ctx* context to be initialized

**10.63.2.10 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

## 10.64 messages.h File Reference

### 10.64.1 Detailed Description

Global handling of error and warning messages.

This wraps 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:

### Typedefs

- `typedef enum _OPENGPS_EXCEPTION_ID OGPS_ExceptionId`

### Enumerations

- `enum _OPENGPS_EXCEPTION_ID {  
 OGPS_ExNone, OGPS_ExGeneral, OGPS_ExInvalidArgument,  
 OGPS_ExInvalidOperationException,  
 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.*

### 10.64.2 Typedef Documentation

#### 10.64.2.1 `typedef enum _OPENGPS_EXCEPTION_ID OGPS_ExceptionId`

### 10.64.3 Enumeration Type Documentation

#### 10.64.3.1 `enum _OPENGPS_EXCEPTION_ID`

Possible failure conditions.

##### Enumerator:

***OGPS\_ExNone*** No failure condition trapped.

This serves as some kind of default value.

***OGPS\_ExGeneral*** A failure condition occurred, 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\_ExInvalidOperationException*** 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 occurred.

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.

### 10.64.4 Function Documentation

#### 10.64.4.1 `_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 occurred. This is because the description of a particular error condition may not be available.

##### See also:

[ogps\\_GetErrorId](#)

#### 10.64.4.2 `_OPENGPS_EXPORT OGPS_ExceptionId ogps_GetErrorId ()`

Gets the type of the last handled error condition.

**See also:**

[ogps\\_GetErrorMessage](#)

**10.64.4.3 \_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 occurred. 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](#)

**10.64.4.4 \_OPENGPS\_EXPORT OGPS\_Boolean ogps\_HasError ()**

Returns TRUE when the statement executed previously failed.

Use [ogps\\_GetErrorId](#) to obtain more information about the failure.

**10.65 messages\_c.hxx File Reference**

```
#include <opengps/messages.h>
#include "messages_c.hxx"
#include "../cxx/stdafx.hxx"
```

Include dependency graph for messages\_c.hxx:

## 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.*

### 10.65.1 Function Documentation

#### 10.65.1.1 const OGPS\_Character\* ogps\_GetErrorDescription ()

Gets the last handled detailed error description or NULL.

#### Remarks:

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

#### See also:

[ogps\\_GetErrorId](#)

**10.65.1.2 OGPS\_ExceptionId ogps\_GetErrorId ()**

Gets the type of the last handled error condition.

**See also:**

[ogps\\_GetErrorMessage](#)

**10.65.1.3 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 occurred. 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](#)

**10.65.1.4 OGPS\_Boolean ogps\_HasError ()**

Returns TRUE when the statement executed previously failed.

Use [ogps\\_GetErrorId](#) to obtain more information about the failure.

**10.66 messages\_c.hxx File Reference****10.66.1 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.*

### 10.66.2 Define Documentation

**10.66.2.1 #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.

**10.66.2.2 #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.

### 10.66.2.3 #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.

## 10.67 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 "stdaafx.hxx"
```

Include dependency graph for missing\_data\_point\_parser.cxx:

## 10.68 missing\_data\_point\_parser.hxx File Reference

### 10.68.1 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.*

## 10.69 opengps.h File Reference

### 10.69.1 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 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.*

### 10.69.2 Define Documentation

#### 10.69.2.1 `#define _EX_T(x) (x)`

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

#### 10.69.2.2 `#define _OPENGPS_EXPORT`

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

#### Remarks:

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

**10.69.2.3 #define FALSE 0**

The FALSE boolean value.

**See also:**

[OGPS\\_Boolean](#)

**10.69.2.4 #define NULL 0**

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

**10.69.2.5 #define TRUE 1**

The TRUE boolean value.

**See also:**

[OGPS\\_Boolean](#)

**10.69.3 Typedef Documentation****10.69.3.1 typedef int OGPS\_Boolean**

Holds a boolean value.

**See also:**

[FALSE, TRUE](#)

**10.69.3.2 typedef char OGPS\_Character**

The current type of characters.

This is either unicode (wchar\_t) or char.

**10.69.3.3 typedef double OGPS\_Double**

Represents measurement data of type double.

**10.69.3.4 typedef float OGPS\_Float**

Represents measurement data of type float.

**10.69.3.5 typedef short OGPS\_Int16**

Represents measurement data of type short.

**10.69.3.6 typedef int OGPS\_Int32**

Represents measurement data of type int.

## 10.70 opengps.hxx File Reference

### 10.70.1 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.*

## 10.71 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:

## 10.72 point\_buffer.hxx File Reference

### 10.72.1 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.*

## 10.73 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:

## 10.74 point\_iterator.h File Reference

### 10.74.1 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, unsigned long *index)`  
*Gets the current position of the iterator.*
- `_OPENGPS_EXPORT OGPS_Boolean ogps_GetMatrixPosition (const OGPS_PointIteratorPtr iterator, unsigned long *u, unsigned long *v, unsigned long *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.*

#### 10.74.2 Typedef Documentation

##### 10.74.2.1 `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](#).

#### 10.74.3 Function Documentation

##### 10.74.3.1 **\_OPENGPS\_EXPORT void ogps\_FreePointIterator (OGPS\_PointIteratorPtr \*const iterator)**

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.

##### 10.74.3.2 **\_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.

##### 10.74.3.3 **\_OPENGPS\_EXPORT OGPS\_Boolean ogps\_GetListPosition (const OGPS\_PointIteratorPtr iterator, unsigned long \* 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.

**10.74.3.4 OPENGPS\_EXPORT OGPS\_Boolean ogps\_-  
GetMatrixPosition (const OGPS\_PointIteratorPtr iterator, un-  
signed long \* u, unsigned long \* v, unsigned long \* 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.

**10.74.3.5 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.

**10.74.3.6 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.

**10.74.3.7 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.

---

**10.74.3.8 \_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.

---

**10.74.3.9 \_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.

---

**10.74.3.10 \_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.

---

**10.74.3.11 \_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.

## 10.75 point\_iterator.hxx File Reference

### 10.75.1 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.*

## 10.76 point\_iterator\_c.hxx 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.hxx:

## 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, unsigned long \*index) throw ()  
*Gets the current position of the iterator.*

- `OGPS_Boolean ogps_GetMatrixPosition (const OGPS_PointIteratorPtr iterator, unsigned long *u, unsigned long *v, unsigned long *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.*

### 10.76.1 Function Documentation

#### 10.76.1.1 void ogps\_FreePointIterator (OGPS\_PointIteratorPtr \*const iterator) throw ()

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.

```
10.76.1.2 void ogps_GetCurrentPoint (const OGPS_-  
PointIteratorPtr iterator, OGPS_PointVectorPtr const vector)  
throw ()
```

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.

```
10.76.1.3 OGPS_Boolean ogps_GetListPosition (const OGPS_-  
PointIteratorPtr iterator, unsigned long * index) throw ()
```

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.

```
10.76.1.4 OGPS_Boolean ogps_GetMatrixPosition (const  
OGPS_PointIteratorPtr iterator, unsigned long * u, unsigned  
long * v, unsigned long * w) throw ()
```

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.

**10.76.1.5 OGPS\_Boolean ogps\_HasNextPoint (const OGPS\_PointIteratorPtr *iterator*) throw ()**

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.

**10.76.1.6 OGPS\_Boolean ogps\_HasPrevPoint (const OGPS\_PointIteratorPtr *iterator*) throw ()**

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.

#### 10.76.1.7 OGPS\_Boolean ogps\_MoveNextPoint (OGPS\_- PointIteratorPtr const *iterator*) throw ()

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.

#### 10.76.1.8 OGPS\_Boolean ogps\_MovePrevPoint (OGPS\_- PointIteratorPtr const *iterator*) throw ()

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.

**10.76.1.9 void ogps\_ResetNextPointIterator (OGPS\_-  
PointIteratorPtr const *iterator*) throw ()**

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

**Parameters:**

*iterator* Operate on this iterator handle.

**10.76.1.10 void ogps\_ResetPrevPointIterator (OGPS\_-  
PointIteratorPtr const *iterator*) throw ()**

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

**Parameters:**

*iterator* Operate on this iterator handle.

**10.76.1.11 void ogps\_SetCurrentPoint (const OGPS\_-  
PointIteratorPtr *iterator*, const OGPS\_PointVectorPtr *vector*)  
throw ()**

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.

**10.77 point\_iterator\_c.hxx File Reference****10.77.1 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\\_POINT\\_ITERATOR](#)

*Encapsulates the internal C++ structure of a point iterator handle used within the C interface.*

## TypeDefs

- typedef struct [\\_OGPS\\_POINT\\_ITERATOR](#) OGPS\_PointIterator
- typedef struct [\\_OGPS\\_POINT\\_ITERATOR](#) \* OGPS\_PointIteratorPtr

### 10.77.2 Typedef Documentation

#### 10.77.2.1 [typedef struct \\_OGPS\\_POINT\\_ITERATOR OGPS\\_PointIterator](#)

#### 10.77.2.2 [typedef struct \\_OGPS\\_POINT\\_ITERATOR \\* OGPS\\_PointIteratorPtr](#)

## 10.78 point\_validity\_provider.hxx File Reference

```
#include "point_validity_provider.hxx"
#include "point_buffer.hxx"
#include "stdafx.hxx"
```

Include dependency graph for point\_validity\_provider.hxx:

## 10.79 point\_validity\_provider.hxx File Reference

### 10.79.1 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.*

## 10.80 point\_vector.cxx File Reference

```
#include "data_point_impl.hxx"
#include <opengps/cxx/point_vector.hxx>
#include "stdaafx.hxx"
```

Include dependency graph for point\_vector.cxx:

## 10.81 point\_vector.h File Reference

### 10.81.1 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.*

*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.*

## 10.81.2 Typedef Documentation

### 10.81.2.1 `typedef struct _OGPS_POINT_VECTOR* OGPS_PointVectorPtr`

Typesafe representation of three-dimensional point measurement data.

The corresponding C++ implementation is provided by [OpenGPS::PointVector](#).

## 10.81.3 Function Documentation

### 10.81.3.1 `_OPENGPS_EXPORT OGPS_PointVectorPtr ogps_CreatePointVector (void)`

Creates an instance of type [OGPS\\_PointVectorPtr](#).

#### Remarks:

If you 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](#).

### 10.81.3.2 `_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.

**10.81.3.3 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.

**10.81.3.4 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.

**10.81.3.5 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.

#### 10.81.3.6 **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.

#### 10.81.3.7 **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.

#### 10.81.3.8 **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.

#### 10.81.3.9 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.

#### 10.81.3.10 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.

#### 10.81.3.11 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.

**10.81.3.12 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.

**10.81.3.13 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.

**10.81.3.14 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.

#### 10.81.3.15 \_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.

#### 10.81.3.16 \_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.

#### 10.81.3.17 \_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.

#### 10.81.3.18 \_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.

---

**10.81.3.19 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.

**10.81.3.20 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.

**10.81.3.21 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.

**10.81.3.22 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.

**10.81.3.23 \_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.

**10.81.3.24 \_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.

**10.81.3.25 \_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.

**10.81.3.26 \_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.

**10.81.3.27 \_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.

**10.81.3.28 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.

**10.81.3.29 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.

**10.81.3.30 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.

**10.81.3.31 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.

**10.81.3.32 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.

**10.81.3.33 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.

**10.81.3.34 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.

## 10.82 point\_vector.hxx File Reference

### 10.82.1 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.*

## 10.83 point\_vector\_base.hxx File Reference

### 10.83.1 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.*

## 10.84 point\_vector\_c.hxx 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.hxx:

## 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.*

#### 10.84.1 Function Documentation

##### 10.84.1.1 OGPS\_PointVectorPtr ogps\_CreatePointVector (void) throw ()

Creates an instance of type `OGPS_PointVectorPtr`.

###### Remarks:

If you 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`.

##### 10.84.1.2 void ogps\_FreePointVector (OGPS\_PointVectorPtr \*const vector) throw ()

Frees an instance of a point vector.

###### See also:

`ogps_CreatePointVector`

###### Parameters:

`vector` Operate on this point vector instance.

##### 10.84.1.3 OGPS\_Double ogps\_GetDoubleX (const OGPS\_- PointVectorPtr vector) throw ()

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.

#### 10.84.1.4 OGPS\_Double ogps\_GetDoubleY (const OGPS\_- PointVectorPtr *vector*) throw ()

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.

#### 10.84.1.5 OGPS\_Double ogps\_GetDoubleZ (const OGPS\_- PointVectorPtr *vector*) throw ()

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.

#### 10.84.1.6 OGPS\_Float ogps\_GetFloatX (const OGPS\_- PointVectorPtr *vector*) throw ()

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.

**10.84.1.7 OGPS\_Float ogps\_GetFloatY (const OGPS\_-  
PointVectorPtr *vector*) throw ()**

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.

**10.84.1.8 OGPS\_Float ogps\_GetFloatZ (const OGPS\_-  
PointVectorPtr *vector*) throw ()**

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.

**10.84.1.9 OGPS\_Int16 ogps.GetInt16X (const OGPS\_-  
PointVectorPtr *vector*) throw ()**

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.

**10.84.1.10 OGPS\_Int16 ogps\_GetInt16Y (const OGPS\_-  
PointVectorPtr *vector*) throw ()**

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.

**10.84.1.11 OGPS\_Int16 ogps\_GetInt16Z (const OGPS\_-  
PointVectorPtr *vector*) throw ()**

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.

**10.84.1.12 OGPS\_Int32 ogps\_GetInt32X (const OGPS\_-  
PointVectorPtr *vector*) throw ()**

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.

**10.84.1.13 OGPS\_Int32 ogps\_GetInt32Y (const OGPS\_-  
PointVectorPtr *vector*) throw ()**

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.

**10.84.1.14 OGPS\_Int32 ogps\_GetInt32Z (const OGPS\_-  
PointVectorPtr *vector*) throw ()**

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.

**10.84.1.15 OGPS\_DataPointType ogps\_GetPointTypeX (const  
OGPS\_PointVectorPtr *vector*) throw ()**

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

**Parameters:**

*vector* Operate on this point vector instance.

**10.84.1.16 OGPS\_DataPointType ogps\_GetPointTypeY (const OGPS\_PointVectorPtr *vector*) throw ()**

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

**Parameters:**

*vector* Operate on this point vector instance.

**10.84.1.17 OGPS\_DataPointType ogps\_GetPointTypeZ (const OGPS\_PointVectorPtr *vector*) throw ()**

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

**Parameters:**

*vector* Operate on this point vector instance.

**10.84.1.18 OGPS\_DataPointPtr const ogps\_GetX (OGPS\_PointVectorPtr const *vector*) throw ()**

Gets the x component of the given vector.

**Parameters:**

*vector* Operate on this point vector instance.

**10.84.1.19 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.

**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.

**10.84.1.20 OGPS\_DataPointPtr const ogps\_GetY (OGPS\_PointVectorPtr const *vector*) throw ()**

Gets the y component of the given vector.

**Parameters:**

*vector* Operate on this point vector instance.

**10.84.1.21 OGPS\_DataPointPtr const ogps\_GetZ (OGPS\_-  
PointVectorPtr const *vector*) throw ()**

Gets the z component of the given vector.

**Parameters:**

*vector* Operate on this point vector instance.

**10.84.1.22 OGPS\_Boolean ogps\_IsValidPoint (const OGPS\_-  
PointVectorPtr *vector*) throw ()**

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.

**10.84.1.23 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.

**Parameters:**

*vector* Operate on this point vector instance.

*value* The new value.

**10.84.1.24 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.

**Parameters:**

*vector* Operate on this point vector instance.

*value* The new value.

**10.84.1.25 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.

**Parameters:**

*vector* Operate on this point vector instance.

*value* The new value.

**10.84.1.26 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.

**Parameters:**

*vector* Operate on this point vector instance.

*value* The new value.

**10.84.1.27 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.

**Parameters:**

*vector* Operate on this point vector instance.

*value* The new value.

**10.84.1.28 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.

**Parameters:**

*vector* Operate on this point vector instance.

*value* The new value.

**10.84.1.29 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.

**Parameters:**

*vector* Operate on this point vector instance.

*value* The new value.

**10.84.1.30 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.

**Parameters:**

*vector* Operate on this point vector instance.

*value* The new value.

**10.84.1.31 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.

**Parameters:**

*vector* Operate on this point vector instance.

*value* The new value.

**10.84.1.32 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.

**Parameters:**

*vector* Operate on this point vector instance.

*value* The new value.

**10.84.1.33 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.

**Parameters:**

*vector* Operate on this point vector instance.

*value* The new value.

**10.84.1.34 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.

**Parameters:**

*vector* Operate on this point vector instance.

*value* The new value.

## 10.85 point\_vector\_c.hxx File Reference

### 10.85.1 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_POINT_VECTOR`

*Encapsulates the internal C++ structure of a point vector handle used within the C interface.*

### Typedefs

- typedef struct `_OGPS_POINT_VECTOR` `OGPS_PointVector`
- typedef struct `_OGPS_POINT_VECTOR *` `OGPS_PointVectorPtr`

### 10.85.2 Typedef Documentation

#### 10.85.2.1 `typedef struct _OGPS_POINT_VECTOR OGPS_PointVector`

10.85.2.2 `typedef struct _OGPS_POINT_VECTOR * OGPS_-PointVectorPtr`

## 10.86 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:

## 10.87 point\_vector\_iostream.hxx File Reference

### 10.87.1 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::PointVectorInputStream](#)  
*A string stream used for parsing a point vector as defined in ISO5436-2 XML.*
- class [OpenGPS::PointVectorInvariantLocale](#)  
*The culture invariant locale.*
- class [OpenGPS::PointVectorOutputStream](#)  
*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.*

## 10.88 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.hxx:

## 10.89 point\_vector\_parser.hxx File Reference

### 10.89.1 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.*

## 10.90 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:

## 10.91 point\_vector\_parser\_builder.hxx File Reference

### 10.91.1 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.*

## 10.92 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:

## 10.93 point\_vector\_proxy.hxx File Reference

### 10.93.1 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.*

## 10.94 point\_vector\_proxy\_context.cxx File Reference

```
#include "point_vector_proxy_context.hxx"
#include "stdafx.hxx"
```

Include dependency graph for point\_vector\_proxy\_context.hxx:

## 10.95 point\_vector\_proxy\_context.hxx File Reference

### 10.95.1 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.*

**10.96 point\_vector\_proxy\_context\_list.cxx File Reference**

```
#include "point_vector_proxy_context_list.hxx"
#include <opengps/cxx/exceptions.hxx>
#include "stdaafx.hxx"
```

Include dependency graph for point\_vector\_proxy\_context\_list.hxx:

**10.97 point\_vector\_proxy\_context\_list.hxx File Reference**

**10.97.1 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:

## **10.98 point\_vector\_proxy\_context\_matrix.cxx File Reference**

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](#).*

## **10.98 point\_vector\_proxy\_context\_matrix.cxx File Reference**

```
#include "point_vector_proxy_context_matrix.hxx"
#include <opengps/cxx/exceptions.hxx>
#include "stdaafx.hxx"
```

Include dependency graph for point\_vector\_proxy\_context\_matrix.cxx:

## **10.99 point\_vector\_proxy\_context\_matrix.hxx File Reference**

### **10.99.1 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](#).*

## 10.100 point\_vector\_reader\_context.hxx File Reference

### 10.100.1 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.*

---

**10.101 point\_vector\_writer\_context.hxx File Reference**

### 10.101.1 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.*

## 10.102 stdafx.hxx File Reference

### 10.102.1 Detailed Description

Standard includes and defines.

```
#include <tchar.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_BUILD 1`  
*Build number, automatically extracted from SVN repository.*
- `#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_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_MINVERSION 1`  
*Minor program revision. This is increased manually in the release process.*
- `#define _OPENGPS_MSTR(x) #x`  
*Build a version string from version numbers.*
- `#define _OPENGPS_NAME _T("openGPS ISO 5436-2 XML")`  
*Version definition for the openGPS library.*
- `#define _OPENGPS_REVISION 0`  
*openGPS revision (currently unused)*
- `#define _OPENGPS_VERSION 0`  
*Major program version. This is increased manually in the release process.*
- `#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(".") _OPENGPS_MSTR(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

### 10.102.2 Define Documentation

10.102.2.1 #define \_ASSERT(x) ((void)0)

10.102.2.2 #define \_OPENGPS\_BINFORMAT\_DOUBLE --  
SIZE 8

10.102.2.3 #define \_OPENGPS\_BINFORMAT\_FLOAT\_SIZE 4

10.102.2.4 #define \_OPENGPS\_BINFORMAT\_INT16\_SIZE 2

10.102.2.5 #define \_OPENGPS\_BINFORMAT\_INT32\_SIZE 4

10.102.2.6 #define \_OPENGPS\_BUILD 1

Build number, automatically extracted from SVN repository.

10.102.2.7 #define \_OPENGPS\_DELETE(x) if((x) != NULL) {  
delete (x); (x) = NULL; }

10.102.2.8 #define \_OPENGPS\_DELETE\_ARRAY(x) if((x) !=  
NULL) { delete[] (x); (x) = NULL; }

10.102.2.9 #define \_OPENGPS\_DESCRIPTION \_T("openGPS  
class library implementing an xml-version of ISO 5436-2 file format.")

Short description of the library.

10.102.2.10 #define \_OPENGPS\_ENV\_OPENGPS --  
LOCATION \_T("OPENGPS\_LOCATION")

---

**10.102.2.11** #define \_OPENGPS\_EXCEPTION\_MESG(x) x

**10.102.2.12** #define \_OPENGPS\_FREE(x) if((x) != NULL) { free(x); (x) = NULL; }

**10.102.2.13** #define \_OPENGPS\_ID \_OPENGPS\_NAME \_T("V") \_OPENGPS\_VERSIONSTRING \_T(")")

Define ID string with programm name and Version.

**10.102.2.14** #define \_OPENGPS\_ISO5436\_LOCATION \_T("iso5436\_2.xsd")

**10.102.2.15** #define \_OPENGPS\_MINVERSION 1

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

**10.102.2.16** #define \_OPENGPS\_MSTR(x) #x

Build a version string from version numbers.

**10.102.2.17** #define \_OPENGPS\_NAME \_T("openGPS ISO 5436-2 XML")

Version definition for the openGPS library.

Name of the program

**10.102.2.18** #define \_OPENGPS\_REVISION 0

openGPS revision (currently unused)

**10.102.2.19** #define \_OPENGPS\_VERSION 0

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

**10.102.2.20** #define \_OPENGPS\_VERSIONSTRING \_OPENGPS\_VERSIONSTRING\_M(\_OPENGPS\_VERSION, \_OPENGPS\_MINVERSION, \_OPENGPS\_BUILD, \_OPENGPS\_REVISION)

Build a version string.

**10.102.2.21** #define \_OPENGPS\_VERSIONSTRING\_M(ver, mver, build, rev) \_OPENGPS\_MSTR(ver) \_T(".") \_OPENGPS\_MSTR(mver) \_T(".") \_OPENGPS\_MSTR(build) \_T(".") \_OPENGPS\_MSTR(rev)

```
10.102.2.22 #define _OPENGPS_XSD_ISO5436_DATALINK_-
PATH _T("bindata/data.bin")
```

```
10.102.2.23 #define _OPENGPS_XSD_ISO5436_LOCATION_-
T("http://www.opengps.eu/2008/ISO5436_2/ISO5436_2.xsd")
```

```
10.102.2.24 #define _OPENGPS_XSD_ISO5436_MAIN_-
CHECKSUM_PATH _T("md5checksum.hex")
```

```
10.102.2.25 #define _OPENGPS_XSD_ISO5436_MAIN_-
PATH _T("main.xml")
```

```
10.102.2.26 #define _OPENGPS_XSD_ISO5436_-
NAMESPACE _T("http://www.opengps.eu/2008/ISO5436_2")
```

```
10.102.2.27 #define _OPENGPS_XSD_ISO5436_-
VALIDPOINTSLINK_PATH _T("bindata/valid.bin")
```

```
10.102.2.28 #define _VERIFY(func, retval) func
```

## 10.103 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:

## 10.104 string.hxx File Reference

### 10.104.1 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.*

## 10.105 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.hxx:

## 10.106 valid\_buffer.hxx File Reference

### 10.106.1 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.*

## 10.107 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:

## 10.108 `vector_buffer.hxx` File Reference

### 10.108.1 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.*

## 10.109 `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 "stdaafx.hxx"
```

Include dependency graph for vector\_buffer\_builder.hxx:

## 10.110 vector\_buffer\_builder.hxx File Reference

### 10.110.1 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.*

## 10.111 win32\_environment.cxx File Reference

## 10.112 win32\_environment.hxx File Reference

### 10.112.1 Detailed Description

The environment of Microsoft Windows operating systems.

## 10.113 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](#)

## **10.114 xml\_point\_vector\_reader\_context.hxx File Reference 799**

---

*Checks whether the underlying stream is valid.*

- #define **\_CHECK\_STREAM\_AND\_THROW\_EXCEPTION**

*Checks whether the underlying stream is valid.*

### **10.113.1 Define Documentation**

#### **10.113.1.1 #define \_CHECK\_ISGOOD\_AND\_THROW\_EXCEPTION**

**Value:**

```
if(!IsGood()) \
{ \
    throw OpenGPS::Exception( \
        OGPS_ExInvalidOperationException, \
        _EX_T("The underlying stream object became invalid."), \
        _EX_T("A read/write error occurred."), \
        _EX_T("OpenGPS::XmlPointVectorReaderContext")); \
}
```

*Checks whether the underlying stream is valid.*

*Throws an exception if this is not the case.*

#### **10.113.1.2 #define \_CHECK\_STREAM\_AND\_THROW\_EXCEPTION**

**Value:**

```
if(!m_Stream) \
{ \
    throw OpenGPS::Exception( \
        OGPS_ExInvalidOperationException, \
        _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.*

## **10.114 xml\_point\_vector\_reader\_context.hxx File Reference**

### **10.114.1 Detailed Description**

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

## **10.115 xml\_point\_vector\_writer\_context.cxx File Reference 800**

```
#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.*

## **10.115 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.hxx:

**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.*

**10.115.1 Define Documentation**

**10.115.1.1 #define \_CHECK\_ISGOOD\_AND\_THROW\_EXCEPTION**

**Value:**

```
if(!IsGood()) \
{ \
    throw OpenGPS::Exception( \
        OGPS_ExInvalidOperationException, \
        _EX_T("The underlying stream object became invalid."), \
        _EX_T("A read/write error occurred."), \
        _EX_T("OpenGPS::XmlPointVectorWriterContext")); \
}
```

Checks whether the underlying stream is valid.

Throws an exception if this is not the case.

**10.115.1.2 #define \_CHECK\_STREAM\_AND\_THROW\_EXCEPTION**

**Value:**

```
if(!m_Stream) \
{ \
    throw OpenGPS::Exception( \
        OGPS_ExInvalidOperationException, \
        _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.

## **10.116 xml\_point\_vector\_writer\_context.hxx File Reference 802**

### **10.116 xml\_point\_vector\_writer\_context.hxx File Reference**

#### **10.116.1 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.*

## **10.117 xsd\_Licence\_Header.c File Reference**

```
#include <opengps/opengps.h>
```

Include dependency graph for xsd\_Licence\_Header.c:

## 10.118 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:

## 10.119 zip\_stream\_buffer.hxx File Reference

### 10.119.1 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::FloatInlineValidity,  
    OpenGPS::BinaryLSBPointVectorReaderContext, [288](#)  
    [117](#)                                  ~FloatPointBuffer
- ~BinaryLSBPointVectorWriterContext      OpenGPS::FloatPointBuffer, [230](#)  
    OpenGPS::BinaryLSBPointVectorWriter, [120](#)                                  OpenGPS::ISO5436\_2  
    [120](#)                                  OpenGPS::ISO5436\_2, [269](#)
- ~BinaryMSBPointVectorReaderContext      ~ISO5436\_2Container  
    OpenGPS::BinaryMSBPointVectorReader, [123](#)                                  OpenGPS::ISO5436\_2Container,  
    [123](#)                                  [289](#)
- ~BinaryMSBPointVectorWriterContext      ~Info  
    OpenGPS::BinaryMSBPointVectorWriter, [126](#)                                  OpenGPS::Info, [232](#)  
    [126](#)                                  ~InputBinaryFileStream
- ~BinaryPointVectorReaderContext      OpenGPS::InputBinaryFileStream,  
    OpenGPS::BinaryPointVectorReaderContext, [235](#)  
    [129](#)                                  ~Int16DataPointParser
- ~BinaryPointVectorWriterContext      OpenGPS::Int16DataPointParser,  
    OpenGPS::BinaryPointVectorWriterContext, [254](#)  
    [133](#)                                  ~Int16PointBuffer  
    OpenGPS::DataPoint, [166](#)                                  OpenGPS::Int16PointBuffer, [256](#)
- ~DataPointImpl      OpenGPS::Int16ValidBuffer  
    OpenGPS::DataPointImpl, [173](#)                                  OpenGPS::Int16ValidBuffer, [259](#)
- ~DataPointParser      ~Int32DataPointParser  
    OpenGPS::DataPointParser, [180](#)                                  OpenGPS::Int32DataPointParser,  
    [196](#)                                  [260](#)
- ~DataPointProxy      ~Int32PointBuffer  
    OpenGPS::PointVectorProxy::DataPoint, [409](#)                                  OpenGPS::Int32PointBuffer, [262](#)  
    [409](#)                                  ~Int32ValidBuffer  
    OpenGPS::PointVectorProxy::DataPoint, [409](#)                                  OpenGPS::Int32ValidBuffer, [265](#)
- ~DoubleDataPointParser      OpenGPS::MissingDataPointParser  
    OpenGPS::DoubleDataPointParser, [196](#)                                  OpenGPS::MissingDataPointParser,  
    [196](#)                                  [358](#)
- ~DoubleInlineValidity      ~OutputBinaryFileStream  
    OpenGPS::DoubleInlineValidity, [198](#)                                  OpenGPS::OutputBinaryFileStream,  
    [198](#)                                  [361](#)
- ~DoublePointBuffer      ~PointBuffer  
    OpenGPS::DoublePointBuffer, [200](#)                                  OpenGPS::PointBuffer, [364](#)
- ~Environment      ~PointIterator  
    OpenGPS::Environment, [205](#)                                  OpenGPS::PointIterator, [369](#)
- ~Exception      ~PointIteratorImpl  
    OpenGPS::Exception, [215](#)                                  OpenGPS::ISO5436\_-  
    [215](#)                                  2Container::PointIteratorImpl,  
    [317](#)
- ~ExceptionHistory      ~PointValidityProvider  
    OpenGPS::ExceptionHistory, [218](#)                                  OpenGPS::PointValidityProvider,  
    [218](#)                                  [374](#)
- ~FloatDataPointParser      ~PointVector  
    OpenGPS::FloatDataPointParser, [225](#)                                  OpenGPS::PointVector, [379](#)
- ~FloatInlineValidity

~PointVectorBase	OpenGPS::PointVectorBase, 388	OpenGPS::XmlPointVectorWriterContext, 584
~PointVectorInputStream	OpenGPS::PointVectorInputStream, 392	~ZipOutputStream OpenGPS::ZipOutputStream, 589
~PointVectorInvariantLocale	OpenGPS::PointVectorInvariantLocale, 393	~ZipStreamBuffer OpenGPS::ZipStreamBuffer, 591
		ASSERT stlafx.hxx, 790
~PointVectorOutputStream	OpenGPS::PointVectorOutputStream, 395	_CHECK_BUFFER_AND_- THROW_READ_- EXCEPTION
~PointVectorParser	OpenGPS::PointVectorParser, 397	_CHECK_BUFFER_AND_- THROW_WRITE_- EXCEPTION
~PointVectorParserBuilder	OpenGPS::PointVectorParserBuilder, 401	data_point_proxy.cxx, 622 _CHECK_ISGOOD_AND_- THROW_EXCEPTION
~PointVectorProxy	OpenGPS::PointVectorProxy, 405	binary_lsb_point_vector_- reader_context.cxx, 594
~PointVectorProxyContext	OpenGPS::PointVectorProxyContext, 415	binary_lsb_point_vector_- writer_context.cxx, 597
~PointVectorProxyContextList	OpenGPS::PointVectorProxyContextList, 418	binary_msb_point_vector_- reader_context.cxx, 599
~PointVectorProxyContextMatrix	OpenGPS::PointVectorProxyContextMatrix, 421	binary_msb_point_vector_- writer_context.cxx, 601
~PointVectorReaderContext	OpenGPS::PointVectorReaderContext, 424	binary_point_vector_writer_- context.cxx, 606
~PointVectorWhitespaceFacet	OpenGPS::PointVectorWhitespaceFacet, 428	xml_point_vector_reader_- context.cxx, 799
~PointVectorWriterContext	OpenGPS::PointVectorWriterContext, 430	xml_point_vector_writer_- context.cxx, 801
~String	OpenGPS::String, 551	CHECK_STREAM_AND_- THROW_EXCEPTION
~ValidBuffer	OpenGPS::ValidBuffer, 562	binary_lsb_point_vector_- reader_context.cxx, 594
~VectorBuffer	OpenGPS::VectorBuffer, 567	binary_lsb_point_vector_- writer_context.cxx, 602
~VectorBufferBuilder	OpenGPS::VectorBufferBuilder, 572	binary_point_vector_reader_- context.cxx, 604
~XmlPointVectorReaderContext	OpenGPS::XmlPointVectorReaderContext, 578	binary_point_vector_writer_- context.cxx, 606
~XmlPointVectorWriterContext		xml_point_vector_reader_- context.cxx, 799
		xml_point_vector_writer_- context.cxx, 801

\_EX\_T  
    opengps.h, 728  
\_OGPS\_DATA\_POINT, 66  
    instance, 67  
\_OGPS\_DATA\_POINT\_TYPE  
    data\_point\_type.h, 623  
\_OGPS\_DOUBLE\_SIZE  
    environment.hxx, 627  
\_OGPS\_FLOAT\_SIZE  
    environment.hxx, 627  
\_OGPS\_INT\_SIZE  
    environment.hxx, 627  
\_OGPS\_ISO5436\_2\_HANDLE, 67  
    instance, 68  
\_OGPS\_POINT\_ITERATOR, 68  
    instance, 69  
\_OGPS\_POINT\_VECTOR, 69  
    instance, 70  
        x, 70  
        y, 70  
        z, 70  
\_OGPS\_SHORT\_SIZE  
    environment.hxx, 627  
\_OPENGPS\_BINFORMAT\_-  
    DOUBLE\_SIZE  
    stdafx.hxx, 790  
\_OPENGPS\_BINFORMAT\_-  
    FLOAT\_SIZE  
    stdafx.hxx, 790  
\_OPENGPS\_BINFORMAT\_-  
    INT16\_SIZE  
    stdafx.hxx, 790  
\_OPENGPS\_BINFORMAT\_-  
    INT32\_SIZE  
    stdafx.hxx, 790  
\_OPENGPS\_BUILD  
    stdafx.hxx, 790  
\_OPENGPS\_DELETE  
    stdafx.hxx, 790  
\_OPENGPS\_DELETE\_ARRAY  
    stdafx.hxx, 790  
\_OPENGPS\_DESCRIPTION  
    stdafx.hxx, 790  
\_OPENGPS\_ENV\_OPENGPS\_-  
    LOCATION  
    stdafx.hxx, 790  
\_OPENGPS\_EXCEPTION\_ID  
    messages.h, 719  
\_OPENGPS\_EXCEPTION\_MSG  
    stdafx.hxx, 790  
\_OPENGPS\_EXPORT  
    opengps.h, 728  
\_OPENGPS\_FILE\_URI\_PREF  
    iso5436\_2\_container.hxx, 675  
\_OPENGPS\_FREE  
    stdafx.hxx, 791  
\_OPENGPS\_GENERIC\_-  
    EXCEPTION\_HANDLER  
    messages\_c.hxx, 724  
\_OPENGPS\_GENERIC\_-  
    EXCEPTION\_-  
    HANDLER\_CLEANUP  
    messages\_c.hxx, 724  
\_OPENGPS\_GENERIC\_-  
    EXCEPTION\_-  
    HANDLER\_-  
    RETVALBOOL  
    messages\_c.hxx, 725  
\_OPENGPS\_ID  
    stdafx.hxx, 791  
\_OPENGPS\_ISO5436\_LOCATION  
    stdafx.hxx, 791  
\_OPENGPS\_MINVERSION  
    stdafx.hxx, 791  
\_OPENGPS\_MSTR  
    stdafx.hxx, 791  
\_OPENGPS\_NAME  
    stdafx.hxx, 791  
\_OPENGPS\_REVISION  
    stdafx.hxx, 791  
\_OPENGPS\_VERSION  
    stdafx.hxx, 791  
\_OPENGPS\_VERSIONSTRING  
    stdafx.hxx, 791  
\_OPENGPS\_VERSIONSTRING\_M  
    stdafx.hxx, 791  
\_OPENGPS\_XSD\_ISO5436\_-  
    DATALINK\_PATH  
    stdafx.hxx, 791  
\_OPENGPS\_XSD\_ISO5436\_-  
    LOCATION  
    stdafx.hxx, 792  
\_OPENGPS\_XSD\_ISO5436\_-  
    MAIN\_CHECKSUM\_-  
    PATH  
    stdafx.hxx, 792  
\_OPENGPS\_XSD\_ISO5436\_-  
    MAIN\_PATH  
    stdafx.hxx, 792

\_OPENGPS\_XSD\_ISO5436\_-  
    NAMESPACE  
        stdafx.hxx, 792

\_OPENGPS\_XSD\_ISO5436\_-  
    VALIDPOINTSLINK\_-  
        PATH  
            stdafx.hxx, 792

\_OPENGPS\_ZIP\_CHUNK\_MAX  
    iso5436\_2\_container.hxx, 675

\_VERIFY  
    stdafx.hxx, 792

\_clone  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxesType, 80

    OpenGPS::Schemas::ISO5436\_-  
        2::AxisDescriptionType,  
            99

    OpenGPS::Schemas::ISO5436\_-  
        2::AxisType, 113

    OpenGPS::Schemas::ISO5436\_-  
        2::DataLinkType, 147

    OpenGPS::Schemas::ISO5436\_-  
        2::DataListType, 161, 162

    OpenGPS::Schemas::ISO5436\_-  
        2::DataType, 188

    OpenGPS::Schemas::ISO5436\_-  
        2::Datum, 194

    OpenGPS::Schemas::ISO5436\_-  
        2::FeatureType, 224

    OpenGPS::Schemas::ISO5436\_-  
        2::InstrumentType, 245

    OpenGPS::Schemas::ISO5436\_-  
        2::ISO5436\_2Type, 333

    OpenGPS::Schemas::ISO5436\_-  
        2::MatrixDimensionType,  
            351

    OpenGPS::Schemas::ISO5436\_-  
        2::ProbingSystemType, 438

    OpenGPS::Schemas::ISO5436\_-  
        2::Record1Type, 450, 451

    OpenGPS::Schemas::ISO5436\_-  
        2::Record2Type, 470

    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 494

    OpenGPS::Schemas::ISO5436\_-  
        2::Record4Type, 508

    OpenGPS::Schemas::ISO5436\_-  
        2::RotationMatrixElementType  
            515

    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 532, 533

OpenGPS::Schemas::ISO5436\_-  
    2::Type, 558

\_tmain  
    ISO5436\_2\_XML\_Demo.hxx,  
        681

\_xsd\_AxisType\_convert  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisType, 113, 114

\_xsd\_AxisType\_indexes\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisType, 115

\_xsd\_AxisType\_literals\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisType, 115

\_xsd\_DataType\_convert  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataType, 188

\_xsd\_DataType\_indexes\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataType, 189

\_xsd\_DataType\_literals\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataType, 189

\_xsd\_Type\_convert  
    OpenGPS::Schemas::ISO5436\_-  
        2::Type, 559

\_xsd\_Type\_indexes\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::Type, 560

\_xsd\_Type\_literals\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::Type, 560

A

    OpenGPS::Schemas::ISO5436\_-  
        2::AxisType, 110

Allocate  
    OpenGPS::DoublePointBuffer,  
        200

    OpenGPS::FloatPointBuffer, 230

    OpenGPS::Int16PointBuffer, 256

    OpenGPS::Int32PointBuffer, 263

    OpenGPS::PointBuffer, 364

    OpenGPS::ValidBuffer, 562

AllocateRaw  
    OpenGPS::ValidBuffer, 562

AppendSeparator  
    OpenGPS::XmlPointVectorWriterContext,  
        585

AppendVendorSpecific  
    OpenGPS::ISO5436\_2, 270  
    OpenGPS::ISO5436\_2Container,  
        290  
auto\_ptr\_types.hxx, 592  
Axes  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record1Type, 451, 452  
Axes\_-  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record1Type, 456  
Axes\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record1Type, 448  
Axes\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record1Type, 448  
AxesType  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxesType, 78, 79  
AxisDescriptionType  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisDescriptionType, 97,  
            98  
AxisType  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisDescriptionType,  
            99–101  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisType, 110–112  
AxisType\_-  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisDescriptionType,  
            107  
AxisType\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisDescriptionType,  
            95  
AxisType\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisDescriptionType,  
            95  
base64\_binary  
    xml\_schema, 62  
BaseType  
    OpenGPS::InputBinaryFileStream,  
        235  
    OpenGPS::OutputBinaryFileStream,  
        361  
OpenGPS::PointVectorInputStream, 392  
OpenGPS::PointVectorInvariantLocale, 393  
OpenGPS::PointVectorOutputStream, 395  
OpenGPS::PointVectorWhitespaceFacet, 428  
OpenGPS::String, 550  
OpenGPS::ZipOutputStream, 589  
OpenGPS::ZipStreamBuffer, 591  
binary\_lsb\_point\_vector\_reader\_-  
    context.hxx, 593  
    \_CHECK\_ISGOOD\_AND\_-  
        THROW\_EXCEPTION,  
            594  
    \_CHECK\_STREAM\_AND\_-  
        THROW\_EXCEPTION,  
            594  
binary\_lsb\_point\_vector\_reader\_-  
    context.hxx, 595  
binary\_lsb\_point\_vector\_writer\_-  
    context.hxx, 596  
    \_CHECK\_ISGOOD\_AND\_-  
        THROW\_EXCEPTION,  
            597  
    \_CHECK\_STREAM\_AND\_-  
        THROW\_EXCEPTION,  
            597  
binary\_lsb\_point\_vector\_writer\_-  
    context.hxx, 597  
binary\_msb\_point\_vector\_reader\_-  
    context.hxx, 598  
    \_CHECK\_ISGOOD\_AND\_-  
        THROW\_EXCEPTION,  
            599  
    \_CHECK\_STREAM\_AND\_-  
        THROW\_EXCEPTION,  
            599  
binary\_msb\_point\_vector\_reader\_-  
    context.hxx, 600  
binary\_msb\_point\_vector\_writer\_-  
    context.hxx, 601  
    \_CHECK\_ISGOOD\_AND\_-  
        THROW\_EXCEPTION,  
            601  
    \_CHECK\_STREAM\_AND\_-  
        THROW\_EXCEPTION,  
            602  
    binary\_msb\_point\_vector\_writer\_-

context.hxx, 602  
binary\_point\_vector\_reader\_-  
    context.cxx, 603  
    \_CHECK\_STREAM\_AND\_-  
        THROW\_EXCEPTION,  
            604  
binary\_point\_vector\_reader\_-  
    context.hxx, 604  
binary\_point\_vector\_writer\_-  
    context.cxx, 605  
    \_CHECK\_ISGOOD\_AND\_-  
        THROW\_EXCEPTION,  
            606  
    \_CHECK\_STREAM\_AND\_-  
        THROW\_EXCEPTION,  
            606  
binary\_point\_vector\_writer\_-  
    context.hxx, 607  
BinaryLSBPointVectorReaderContext BuildZ  
    OpenGPS::BinaryLSBPointVectorReaderContext  
        117  
BinaryLSBPointVectorWriterContext OpenGPS::PointVectorParserBuilder,  
    OpenGPS::BinaryLSBPointVectorWriterContext  
        574  
        120  
BinaryMSBPointVectorReaderContext OpenGPS::PointVectorParserBuilder,  
    OpenGPS::BinaryMSBPointVectorReaderContext  
        123  
BinaryMSBPointVectorWriterContext Byte  
    OpenGPS::BinaryMSBPointVectorWriterContext  
        126  
BinaryPointVectorReaderContext BytePtr  
    OpenGPS::BinaryPointVectorReaderContext  
        129  
BinaryPointVectorWriterContext ByteSwap  
    OpenGPS::BinaryPointVectorWriterContext  
        133  
boolean ByteSwap16  
    xml\_schema, 62  
bounds ByteSwap  
    xml\_schema, 62  
buffer ByteSwap32  
    md5\_context, 357  
    xml\_schema, 62  
BuildBuffer ByteSwap64  
    OpenGPS::VectorBufferBuilder,  
        573  
BuildParser CalibrationDate  
    OpenGPS::PointVectorParserBuilder  
        401  
BuildPointVectorParser CalibrationDate\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record2Type, 465  
        2::Record2Type, 466

CanIncrementIndex  
 OpenGPS::PointVectorProxyContext, 415

OpenGPS::PointVectorProxyContext  
 418

OpenGPS::PointVectorProxyContext  
 421

ChecksumFile  
 OpenGPS::Schemas::ISO5436\_-  
 2::Record4Type, 509, 510

ChecksumFile\_  
 OpenGPS::Schemas::ISO5436\_-  
 2::Record4Type, 511

ChecksumFile\_traits  
 OpenGPS::Schemas::ISO5436\_-  
 2::Record4Type, 506

ChecksumFile\_type  
 OpenGPS::Schemas::ISO5436\_-  
 2::Record4Type, 506

Close  
 OpenGPS::BinaryPointVectorReader  
 130

OpenGPS::BinaryPointVectorWriter  
 133

OpenGPS::ISO5436\_2, 270

OpenGPS::ISO5436\_2Container,  
 291

Comment  
 OpenGPS::Schemas::ISO5436\_-  
 2::Record2Type, 472–474

Comment\_  
 OpenGPS::Schemas::ISO5436\_-  
 2::Record2Type, 482

Comment\_optional  
 OpenGPS::Schemas::ISO5436\_-  
 2::Record2Type, 466

Comment\_traits  
 OpenGPS::Schemas::ISO5436\_-  
 2::Record2Type, 466

Comment\_type  
 OpenGPS::Schemas::ISO5436\_-  
 2::Record2Type, 466

Compress  
 OpenGPS::ISO5436\_2Container,  
 291

ConcatPathes  
 OpenGPS::Environment, 209

ConfigureNamespaceMap  
 OpenGPS::ISO5436\_2Container,  
 291

Contacting  
 OpenGPS::Schemas::ISO5436\_-  
 2::Type, 555

ConvertFromMd5  
 OpenGPS::String, 551

ConvertPointToCoord  
 OpenGPS::ISO5436\_2Container,  
 291

ConvertToMd5  
 OpenGPS::String, 551

ConvertULongLongToULong  
 OpenGPS::ISO5436\_2Container,  
 292

ConvertULongToInt32  
 OpenGPS::ISO5436\_2Container,  
 292

CopyTo  
 OpenGPS::String, 552

Create  
 OpenGPS::ISO5436\_2, 270, 271

OpenGPS::ISO5436\_2Container,  
 292, 293

ContainerTempFilePath  
 OpenGPS::ISO5436\_2Container,  
 294

CreateDataPointParser  
 OpenGPS::PointVectorParser,  
 397

CreateDir  
 OpenGPS::Environment, 210

CreateDocument  
 OpenGPS::ISO5436\_2Container,  
 294

CreateInstance  
 OpenGPS::Environment, 210

CreateNextPointIterator  
 OpenGPS::ISO5436\_2, 271

OpenGPS::ISO5436\_2Container,  
 294

CreatePointBuffer  
 OpenGPS::ISO5436\_2Container,  
 294

OpenGPS::VectorBufferBuilder,  
 574

CreatePointVectorProxyContext  
 OpenGPS::ISO5436\_2Container,  
 295

CreatePointVectorReaderContext  
 OpenGPS::ISO5436\_2Container,  
 295

CreatePointVectorWriterContext  
    OpenGPS::ISO5436\_2Container, 295

CreatePrevPointIterator  
    OpenGPS::ISO5436\_2, 272  
    OpenGPS::ISO5436\_2Container, 295

CreateTempDir  
    OpenGPS::ISO5436\_2Container, 296

Creator  
    OpenGPS::Schemas::ISO5436\_-2::Record2Type, 474–476

Creator\_-  
    OpenGPS::Schemas::ISO5436\_-2::Record2Type, 482

Creator\_optional  
    OpenGPS::Schemas::ISO5436\_-2::Record2Type, 466

Creator\_traits  
    OpenGPS::Schemas::ISO5436\_-2::Record2Type, 467

Creator\_type  
    OpenGPS::Schemas::ISO5436\_-2::Record2Type, 467

CX  
    OpenGPS::Schemas::ISO5436\_-2::AxesType, 80–82

CX\_-  
    OpenGPS::Schemas::ISO5436\_-2::AxesType, 88

CX\_traits  
    OpenGPS::Schemas::ISO5436\_-2::AxesType, 76

CX\_type  
    OpenGPS::Schemas::ISO5436\_-2::AxesType, 77

CY  
    OpenGPS::Schemas::ISO5436\_-2::AxesType, 82, 83

CY\_-  
    OpenGPS::Schemas::ISO5436\_-2::AxesType, 88

CY\_traits  
    OpenGPS::Schemas::ISO5436\_-2::AxesType, 77

CY\_type  
    OpenGPS::Schemas::ISO5436\_-2::AxesType, 77

CZ

    OpenGPS::Schemas::ISO5436\_-2::AxesType, 84, 85

    CZ\_-  
        OpenGPS::Schemas::ISO5436\_-2::AxesType, 88

    CZ\_traits  
        OpenGPS::Schemas::ISO5436\_-2::AxesType, 77

    CZ\_type  
        OpenGPS::Schemas::ISO5436\_-2::AxesType, 77, 78

D

    OpenGPS::Schemas::ISO5436\_-2::DataType, 184

data\_point\_type.h  
    OGPS\_DoublePointType, 623  
    OGPS\_FloatPointType, 623  
    OGPS\_Int16PointType, 623  
    OGPS\_Int32PointType, 623  
    OGPS\_MissingPointType, 623

data\_point.h, 608  
    OGPS\_DataPointPtr, 609  
    ogps\_GetDouble, 610  
    ogps\_GetFloat, 610  
    ogps.GetInt16, 610  
    ogps.GetInt32, 611  
    ogps\_GetPointType, 611  
    ogps\_SetDouble, 611  
    ogps\_SetFloat, 612  
    ogps\_SetInt16, 612  
    ogps\_SetInt32, 612

data\_point.hxx, 613

data\_point\_c.cxx, 613  
    ogps\_GetDouble, 615  
    ogps\_GetFloat, 615  
    ogps.GetInt16, 615  
    ogps.GetInt32, 616  
    ogps\_GetPointType, 616  
    ogps\_SetDouble, 616  
    ogps\_SetFloat, 617  
    ogps\_SetInt16, 617  
    ogps\_SetInt32, 617

data\_point\_c.hxx, 618  
    OGPS\_DataPoint, 618  
    OGPS\_DataPointPtr, 618

data\_point\_impl.cxx, 619

data\_point\_impl.hxx, 619

data\_point\_parser.hxx, 620

data\_point\_proxy.cxx, 621

\_CHECK\_BUFFER\_AND\_-  
THROW\_READ\_-  
EXCEPTION, 622  
\_CHECK\_BUFFER\_AND\_-  
THROW\_WRITE\_-  
EXCEPTION, 622  
data\_point\_type.h, 622  
\_OGPS\_DATA\_POINT\_-  
TYPE, 623  
OGPS\_DataPointType, 623  
DataLink  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 494–496  
DataLink\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 503  
DataLink\_optional  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 489  
DataLink\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 490  
DataLink\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 490  
DataLinkType  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataLinkType, 145, 146  
DataList  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 496–498  
DataList\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 503  
DataList\_optional  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 490  
DataList\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 490  
DataList\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 490, 491  
DataListType  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataListType, 160, 161  
DataPoint  
    OpenGPS::DataPoint, 166  
DataPointImpl  
    OpenGPS::DataPointImpl, 173  
DataPointParser  
    OpenGPS::DataPointParser, 180  
DataPointProxy  
    OpenGPS::PointVectorProxy::DataPointProxy,  
        409  
DataType  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisDescriptionType,  
            101–103  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataType, 185–187  
DataType\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisDescriptionType,  
            107  
DataType\_optional  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisDescriptionType,  
            95  
DataType\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisDescriptionType,  
            95  
DataType\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisDescriptionType, 95,  
            96  
Date  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record2Type, 477, 478  
date  
    xml\_schema, 62  
Date\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record2Type, 482  
date\_time  
    xml\_schema, 62  
Date\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record2Type, 467  
Date\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record2Type, 467  
Datum  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataListType, 162, 163  
    OpenGPS::Schemas::ISO5436\_-  
        2::Datum, 191–193  
Datum\_

OpenGPS::Schemas::ISO5436\_-  
2::DataListType, 164  
Datum\_const\_iterator  
  OpenGPS::Schemas::ISO5436\_-  
2::DataListType, 159  
Datum\_iterator  
  OpenGPS::Schemas::ISO5436\_-  
2::DataListType, 159  
Datum\_sequence  
  OpenGPS::Schemas::ISO5436\_-  
2::DataListType, 159  
Datum\_traits  
  OpenGPS::Schemas::ISO5436\_-  
2::DataListType, 159, 160  
Datum\_type  
  OpenGPS::Schemas::ISO5436\_-  
2::DataListType, 160  
day  
  xml\_schema, 62  
decimal  
  xml\_schema, 62  
Decompress  
  OpenGPS::ISO5436\_2Container,  
296  
DecompressChecksum  
  OpenGPS::ISO5436\_2Container,  
296  
DecompressDataBin  
  OpenGPS::ISO5436\_2Container,  
297  
DecompressMain  
  OpenGPS::ISO5436\_2Container,  
297  
details  
  OpenGPS::Exception, 215  
diagnostics  
  xml\_schema, 62  
do\_is  
  OpenGPS::PointVectorWhitespaceFacet,xml\_schema, 63  
  428  
double  
  xml\_schema, 62  
double\_data\_point\_parser.cxx, 623  
double\_data\_point\_parser.hxx, 624  
double\_point\_buffer.cxx, 625  
double\_point\_buffer.hxx, 625  
DoubleDataPointParser  
  OpenGPS::DoubleDataPointParser, expected\_text\_content  
  196  
DoubleInlineValidity  
  OpenGPS::DoubleInlineValidity,  
198  
DoublePointBuffer  
  OpenGPS::DoublePointBuffer,  
200  
doubleValue  
  OpenGPS::DataPointImpl::-  
OGPS\_DATA\_POINT\_-  
VALUE, 179  
Doxygen.cpp, 626  
DumpIt  
  OpenGPS::ExceptionHistory, 218  
duplicate\_id  
  xml\_schema, 62  
duration  
  xml\_schema, 62  
ElementType  
  OpenGPS::String, 550  
entities  
  xml\_schema, 62  
entity  
  xml\_schema, 63  
Environment  
  OpenGPS::Environment, 205  
environment.cxx, 626  
  OGPS\_DOUBLE\_SIZE, 627  
  OGPS\_FLOAT\_SIZE, 627  
  OGPS\_INT\_SIZE, 627  
  OGPS\_SHORT\_SIZE, 627  
environment.hxx, 627  
error  
  xml\_schema, 63  
error\_handler  
  xml\_schema, 63  
Exception  
  OpenGPS::Exception, 215  
exception  
  xml\_schema, 63  
ExceptionHistory  
  OpenGPS::ExceptionHistory, 218  
exceptions.cxx, 628  
exceptions.hxx, 629  
expected\_attribute  
  xml\_schema, 63  
expected\_element  
  xml\_schema, 63  
  xml\_schema, 63

F  
    md5.c, 710, 711  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataType, 184  
    FALSE  
        opengps.h, 728  
    FeatureType  
        OpenGPS::Schemas::ISO5436\_-  
            2::FeatureType, 221–223  
        OpenGPS::Schemas::ISO5436\_-  
            2::Record1Type, 453, 454  
    FeatureType\_  
        OpenGPS::Schemas::ISO5436\_-  
            2::Record1Type, 456  
    FeatureType\_traits  
        OpenGPS::Schemas::ISO5436\_-  
            2::Record1Type, 448  
    FeatureType\_type  
        OpenGPS::Schemas::ISO5436\_-  
            2::Record1Type, 448  
    flags  
        xml\_schema, 63  
    float\_  
        xml\_schema, 63  
    float\_data\_point\_parser.cxx, 630  
    float\_data\_point\_parser.hxx, 630  
    float\_point\_buffer.cxx, 631  
    float\_point\_buffer.hxx, 632  
    FloatDataPointParser  
        OpenGPS::FloatDataPointParser,  
            225  
    FloatInlineValidity  
        OpenGPS::FloatInlineValidity,  
            228  
    FloatPointBuffer  
        OpenGPS::FloatPointBuffer, 230  
    floatValue  
        OpenGPS::DataPointImpl::-\_  
            OGPS\_DATA\_POINT\_-  
            VALUE, 179  
    Free  
        OpenGPS::PointBuffer, 365  
    FromChar  
        OpenGPS::String, 552  
    Get  
        OpenGPS::DataPoint, 166–168  
        OpenGPS::DataPointImpl, 174,  
            175  
    OpenGPS::DoublePointBuffer,  
        200  
    OpenGPS::FloatPointBuffer, 230  
    OpenGPS::Int16PointBuffer, 257  
    OpenGPS::Int32PointBuffer, 263  
    OpenGPS::PointBuffer, 365, 366  
    OpenGPS::PointVector, 379  
    OpenGPS::PointVectorBase, 389  
    OpenGPS::PointVectorProxy, 405  
    OpenGPS::PointVectorProxy::DataPointProxy,  
        409–411  
    OpenGPS::XmlPointVectorWriterContext,  
        585  
GET ULONG \_LE  
    md5.c, 711  
GetAbout  
    OpenGPS::Info, 232  
GetAltDirectorySeparator  
    OpenGPS::Environment, 210  
GetAxisDataType  
    OpenGPS::ISO5436\_2Container,  
        297  
GetBuffer  
    OpenGPS::VectorBufferBuilder,  
        574  
GetChecksumArchiveName  
    OpenGPS::ISO5436\_2Container,  
        297  
GetChecksumFileName  
    OpenGPS::ISO5436\_2Container,  
        298  
GetCopyright  
    OpenGPS::Info, 232  
GetCurrent  
    OpenGPS::ISO5436\_-  
        2Container::PointIteratorImpl,  
            317  
        OpenGPS::PointIterator, 370  
GetDirectorySeparator  
    OpenGPS::Environment, 210  
GetDocument  
    OpenGPS::ISO5436\_2, 272  
    OpenGPS::ISO5436\_2Container,  
        298  
GetFileName  
    OpenGPS::Environment, 210  
GetFilePath  
    OpenGPS::ISO5436\_2Container,  
        298  
GetFullPath

OpenGPS::ISO5436\_2Container, 298  
GetIncrementX  
    OpenGPS::ISO5436\_2Container, 298  
GetIncrementY  
    OpenGPS::ISO5436\_2Container, 298  
GetIndex  
    OpenGPS::PointVectorProxyContext, 415  
    OpenGPS::PointVectorProxyContextListOpenGPS::Info, 418  
    OpenGPS::PointVectorProxyContextMatrixOpenGPS::ISO5436\_2Container, 421  
GetInstance  
    OpenGPS::Environment, 211  
GetLastErrorDescription  
    OpenGPS::ExceptionHistory, 218  
GetLastErrorId  
    OpenGPS::ExceptionHistory, 218  
GetLastErrorMessage  
    OpenGPS::ExceptionHistory, 218  
GetLicense  
    OpenGPS::Info, 233  
GetListCoord  
    OpenGPS::ISO5436\_2, 272  
    OpenGPS::ISO5436\_2Container, 298  
GetListPoint  
    OpenGPS::ISO5436\_2, 273  
    OpenGPS::ISO5436\_2Container, 299  
GetMainArchiveName  
    OpenGPS::ISO5436\_2Container, 300  
GetMainFileName  
    OpenGPS::ISO5436\_2Container, 300  
GetMatrixCoord  
    OpenGPS::ISO5436\_2, 273  
    OpenGPS::ISO5436\_2Container, 300  
GetMatrixPoint  
    OpenGPS::ISO5436\_2, 274  
    OpenGPS::ISO5436\_2Container, 301  
GetMaxU  
    OpenGPS::ISO5436\_2Container, 301  
    GetMaxV  
        OpenGPS::ISO5436\_2Container, 302  
    GetMaxW  
        OpenGPS::ISO5436\_2Container, 302  
    GetMd5  
        OpenGPS::BinaryPointVectorWriterContext, 133  
    GetName  
        OpenGPS::ZipStreamBuffer, 591  
    GetOffsetX  
        OpenGPS::ISO5436\_2Container, 302  
    GetOffsetY  
        OpenGPS::ISO5436\_2Container, 303  
    GetOffsetZ  
        OpenGPS::ISO5436\_2Container, 303  
    GetParser  
        OpenGPS::PointVectorParserBuilder, 402  
    GetPathName  
        OpenGPS::Environment, 211  
    GetPointBuffer  
        OpenGPS::PointValidityProvider, 375  
    GetPointCount  
        OpenGPS::ISO5436\_2Container, 303  
    GetPointDataArchiveName  
        OpenGPS::ISO5436\_2Container, 303  
    GetPointDataFileName  
        OpenGPS::ISO5436\_2Container, 303  
    GetPointType  
        OpenGPS::DataPoint, 168  
        OpenGPS::DataPointImpl, 175  
        OpenGPS::DoublePointBuffer, 201  
        OpenGPS::FloatPointBuffer, 230  
        OpenGPS::Int16PointBuffer, 257  
        OpenGPS::Int32PointBuffer, 263  
        OpenGPS::PointBuffer, 366  
        OpenGPS::PointVectorProxy::DataPointProxy, 411  
    GetPointVectorProxy

OpenGPS::VectorBuffer, 567  
GetPosition  
    OpenGPS::ISO5436\_-  
        2Container::PointIteratorImpl,  
            317, 318  
    OpenGPS::PointIterator, 370  
GetSize  
    OpenGPS::PointBuffer, 366  
GetStream  
    OpenGPS::BinaryPointVectorReaderContext  
        130  
    OpenGPS::BinaryPointVectorWriterContext  
        134  
GetTempDir  
    OpenGPS::Environment, 211  
    OpenGPS::ISO5436\_2Container,  
        303  
GetUniqueName  
    OpenGPS::Environment, 211  
GetValidityBuffer  
    OpenGPS::VectorBuffer, 567, 568  
GetValidityProvider  
    OpenGPS::VectorBuffer, 568  
GetValidPointsArchiveName  
    OpenGPS::ISO5436\_2Container,  
        303  
GetValidPointsFileName  
    OpenGPS::ISO5436\_2Container,  
        303  
GetVariable  
    OpenGPS::Environment, 211  
GetVectorBuffer  
    OpenGPS::ISO5436\_2Container,  
        303  
GetVendorSpecific  
    OpenGPS::ISO5436\_2, 275  
    OpenGPS::ISO5436\_2Container,  
        303  
GetVersion  
    OpenGPS::Info, 233  
GetX  
    OpenGPS::PointVector, 379–381  
    OpenGPS::PointVectorBase, 389  
    OpenGPS::PointVectorProxy, 405  
    OpenGPS::VectorBuffer, 568  
GetXaxisDataType  
    OpenGPS::ISO5436\_2Container,  
        304  
GetXYZ  
    OpenGPS::PointVector, 381  
GetY  
    OpenGPS::PointVector, 381, 382  
    OpenGPS::PointVectorBase, 389  
    OpenGPS::PointVectorProxy, 405  
    OpenGPS::VectorBuffer, 568, 569  
GetYaxisDataType  
    OpenGPS::ISO5436\_2Container,  
        304  
GetZ  
    OpenGPS::PointVector, 383, 384  
    OpenGPS::PointVectorBase, 389  
    OpenGPS::PointVectorProxy, 406  
    OpenGPS::VectorBuffer, 569  
GetZaxisDataType  
    OpenGPS::ISO5436\_2Container,  
        304  
HasDocument  
    OpenGPS::ISO5436\_2Container,  
        304  
HasNext  
    OpenGPS::ISO5436\_-  
        2Container::PointIteratorImpl,  
            318  
    OpenGPS::PointIterator, 371  
HasPrev  
    OpenGPS::ISO5436\_-  
        2Container::PointIteratorImpl,  
            318  
    OpenGPS::PointIterator, 371  
HasStream  
    OpenGPS::BinaryPointVectorReaderContext,  
        130  
    OpenGPS::BinaryPointVectorWriterContext,  
        134  
HasTempDir  
    OpenGPS::ISO5436\_2Container,  
        304  
HasValidityBuffer  
    OpenGPS::VectorBuffer, 569  
HasValidPointsLink  
    OpenGPS::ISO5436\_2Container,  
        305  
HasVectorBuffer  
    OpenGPS::ISO5436\_2Container,  
        305  
hex\_binary  
    xml\_schema, 63

OpenGPS::Schemas::ISO5436\_-  
2::AxisType, 110  
OpenGPS::Schemas::ISO5436\_-  
2::DataType, 184  
id  
    OpenGPS::Exception, 215  
    xml\_schema, 63  
Identification  
    OpenGPS::Schemas::ISO5436\_-  
        2::ProbingSystemType, 439,  
            440  
Identification\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::ProbingSystemType, 442  
Identification\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::ProbingSystemType, 436  
Identification\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::ProbingSystemType, 436  
idref  
    xml\_schema, 63  
idrefs  
    xml\_schema, 63  
Increment  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisDescriptionType,  
            103–105  
Increment\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisDescriptionType,  
            107  
Increment\_optional  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisDescriptionType,  
            96  
Increment\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisDescriptionType,  
            96  
Increment\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxisDescriptionType,  
            96  
IncrementIndex  
    OpenGPS::PointVectorProxyContext  
        416  
    OpenGPS::PointVectorProxyContextList,  
        2::Record2Type, 418  
            InstrumentType  
OpenGPS::PointVectorProxyContextMatrix,  
    421  
Info  
    OpenGPS::Info, 232  
info.cxx, 633  
info.h, 633  
    ops\_GetAboutInfo, 634  
    ops\_GetCopyrightInfo, 635  
    ops\_GetLicenseInfo, 635  
    ops.GetNameInfo, 635  
    ops\_GetVersionInfo, 636  
    ops\_PrintCopyrightInfo, 636  
    ops\_PrintLicenseInfo, 636  
    ops\_PrintVersionInfo, 636  
info.hxx, 636  
info\_c.cxx, 637  
    ops\_GetAboutInfo, 639  
    ops\_GetCopyrightInfo, 639  
    ops\_GetLicenseInfo, 639  
    ops.GetNameInfo, 639  
    ops\_GetVersionInfo, 640  
    ops\_PrintCopyrightInfo, 640  
    ops\_PrintLicenseInfo, 640  
    ops\_PrintVersionInfo, 640  
inline\_validity.cxx, 640  
inline\_validity.hxx, 641  
InputBinaryFileStream  
    OpenGPS::InputBinaryFileStream,  
        235  
instance  
    \_OGPS\_DATA\_POINT, 67  
    \_OGPS\_ISO5436\_2\_HANDLE,  
        68  
    \_OGPS\_POINT\_ITERATOR,  
        69  
    \_OGPS\_POINT\_VECTOR, 70  
Instrument  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record2Type, 478–480  
Instrument\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record2Type, 482  
Instrument\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record2Type, 467  
            InstrumentType  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record2Type, 468

OpenGPS::Schemas::ISO5436\_-  
2::InstrumentType, [243](#),  
[244](#)  
int16\_data\_point\_parser.cxx, [642](#)  
int16\_data\_point\_parser.hxx, [642](#)  
int16\_point\_buffer.cxx, [643](#)  
int16\_point\_buffer.hxx, [644](#)  
Int16DataPointParser  
    OpenGPS::Int16DataPointParser,  
        [254](#)  
Int16PointBuffer  
    OpenGPS::Int16PointBuffer, [256](#)  
Int16ValidBuffer  
    OpenGPS::Int16ValidBuffer, [259](#)  
int16Value  
    OpenGPS::DataPointImpl::-\_  
        OGPS\_DATA\_POINT\_-  
            VALUE, [179](#)  
int32\_data\_point\_parser.cxx, [645](#)  
int32\_data\_point\_parser.hxx, [645](#)  
int32\_point\_buffer.cxx, [646](#)  
int32\_point\_buffer.hxx, [647](#)  
Int32DataPointParser  
    OpenGPS::Int32DataPointParser,  
        [260](#)  
Int32PointBuffer  
    OpenGPS::Int32PointBuffer, [262](#)  
Int32ValidBuffer  
    OpenGPS::Int32ValidBuffer, [265](#)  
int32Value  
    OpenGPS::DataPointImpl::-\_  
        OGPS\_DATA\_POINT\_-  
            VALUE, [179](#)  
int\_-  
    xml\_schema, [63](#)  
integer  
    xml\_schema, [63](#)  
ipad  
    md5\_context, [357](#)  
IsAllocated  
    OpenGPS::ValidBuffer, [563](#)  
IsBinary  
    OpenGPS::ISO5436\_2Container,  
        [305](#)  
IsGood  
    OpenGPS::BinaryPointVectorReaderContext  
        [130](#)  
    OpenGPS::BinaryPointVectorWriterContext  
        [134](#)  
OpenGPS::PointVectorWriterContext,  
    [430](#)  
OpenGPS::XmlPointVectorReaderContext,  
    [578](#)  
OpenGPS::XmlPointVectorWriterContext,  
    [585](#)  
IsIncrementalX  
    OpenGPS::ISO5436\_2Container,  
        [305](#)  
IsIncrementalY  
    OpenGPS::ISO5436\_2Container,  
        [305](#)  
IsLittleEndian  
    OpenGPS::Environment, [212](#)  
IsMatrix  
    OpenGPS::ISO5436\_2Container,  
        [305](#)  
    OpenGPS::PointVectorProxyContext,  
        [416](#)  
    OpenGPS::PointVectorProxyContextList,  
        [418](#)  
    OpenGPS::PointVectorProxyContextMatrix,  
        [421](#)  
IsMatrixCoordValid  
    OpenGPS::ISO5436\_2, [276](#)  
    OpenGPS::ISO5436\_2Container,  
        [306](#)  
ISO5436\_2  
    OpenGPS::ISO5436\_2, [268](#), [269](#)  
    OpenGPS::Schemas::ISO5436\_2,  
        [39-48](#)  
iso5436\_2.cxx, [648](#)  
iso5436\_2.h, [648](#)  
    ogps\_AppendVendorSpecific, [651](#)  
    ogps\_CloseISO5436\_2, [652](#)  
    ogps\_CreateNextPointIterator,  
        [652](#)  
    ogps\_CreatePrevPointIterator,  
        [652](#)  
    ogps\_GetListCoord, [653](#)  
    ogps\_GetListPoint, [654](#)  
    ogps\_GetMatrixCoord, [654](#)  
    ogps\_GetMatrixPoint, [655](#)  
    ogps\_GetVendorSpecific, [656](#)  
    ogps\_IsMatrixCoordValid, [656](#)  
    OGPS\_ISO5436\_2Handle, [651](#)  
    ogps\_OpenISO5436\_2, [657](#)  
    ogps\_SetListPoint, [658](#)  
    ogps\_SetMatrixPoint, [658](#)  
    ogps\_WriteISO5436\_2, [659](#)

iso5436\_2.hxx, 660  
iso5436\_2\_c.hxx, 661  
    ogps\_AppendVendorSpecific, 664  
    ogps\_CloseISO5436\_2, 664  
    ogps\_CreateListISO5436\_2, 664  
    ogps\_CreateMatrixISO5436\_2,  
        665  
    ogps\_CreateNextPointIterator,  
        666  
    ogps\_CreatePrevPointIterator,  
        666  
    ogps\_GetDocument, 667  
    ogps\_GetListCoord, 667  
    ogps\_GetListPoint, 668  
    ogps\_GetMatrixCoord, 668  
    ogps\_GetMatrixPoint, 669  
    ogps\_GetVendorSpecific, 670  
    ogps\_IsMatrixCoordValid, 670  
    ogps\_OpenISO5436\_2, 671  
    ogps\_SetListPoint, 671  
    ogps\_SetMatrixPoint, 672  
    ogps\_WriteISO5436\_2, 673  
iso5436\_2\_container.hxx, 674  
    \_OPENGPS\_FILE\_URI\_-  
        PREF, 675  
    \_OPENGPS\_ZIP\_CHUNK\_-  
        MAX, 675  
PointVectorReaderContextAu-  
    toPtr, 675  
PointVectorWriterContextAu-  
    toPtr, 675  
iso5436\_2\_container.hxx, 675  
iso5436\_2\_handle.hxx, 676  
    ogps\_CreateListISO5436\_2, 678  
    ogps\_CreateMatrixISO5436\_2,  
        678  
    ogps\_GetDocument, 679  
iso5436\_2\_handle\_c.hxx, 680  
    OGPS\_ISO5436\_2, 680  
    OGPS\_ISO5436\_2Handle, 680  
ISO5436\_2\_XML/ Directory Refer-  
    ence, 24  
ISO5436\_2\_XML/c/ Directory Refer-  
    ence, 18  
ISO5436\_2\_XML/cxx/ Directory  
    Reference, 20  
ISO5436\_2\_XML/xyssl/ Directory  
    Reference, 26  
ISO5436\_2\_XML\_Demo.cxx, 680  
    \_tmain, 681  
mediumComplexExample, 681  
performanceDouble, 681  
performanceInt16, 682  
readonlyExample, 682  
readonlyExample2, 682  
readonlyExample3, 682  
readonlyExample4, 682  
simpleExample, 682  
ISO5436\_2\_XML\_Demo/ Directory  
    Reference, 24  
iso5436\_2\_xsd.hxx, 682  
iso5436\_2\_xsd.hxx, 689, 700  
ISO5436\_2Container  
    OpenGPS::ISO5436\_2Container,  
        289  
ISO5436\_2Type  
    OpenGPS::Schemas::ISO5436\_-  
        2::ISO5436\_2Type, 332,  
        333  
IsValid  
    OpenGPS::BinaryPointVectorReaderContext,  
        130  
    OpenGPS::DataPoint, 168  
    OpenGPS::DataPointImpl, 176  
    OpenGPS::DoubleInlineValidity,  
        198  
    OpenGPS::FloatInlineValidity,  
        228  
    OpenGPS::PointValidityProvider,  
        375  
    OpenGPS::PointVector, 384  
    OpenGPS::PointVectorProxy::DataPointProxy,  
        412  
    OpenGPS::PointVectorReaderContext,  
        424  
    OpenGPS::ValidBuffer, 563  
    OpenGPS::XmlPointVectorReaderContext,  
        578  
L  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataType, 184  
language  
    xml\_schema, 64  
ListDimension  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 499, 500  
ListDimension\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 503

ListDimension\_optional  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 491

ListDimension\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 491

ListDimension\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 491

long\_  
    xml\_schema, 64

m\_Buffer  
    OpenGPS::BinaryPointVectorWriterContext  
        135

    OpenGPS::DoublePointBuffer,  
        201

    OpenGPS::FloatPointBuffer, 231

    OpenGPS::Int16PointBuffer, 257

    OpenGPS::Int32PointBuffer, 264

    OpenGPS::PointVectorProxy, 406

    OpenGPS::PointVectorProxy::DataPointProxy  
        414

    OpenGPS::VectorBufferBuilder,  
        575

m\_CompressionLevel  
    OpenGPS::ISO5436\_2Container,  
        312

m\_Context  
    OpenGPS::PointVectorProxy, 406

    OpenGPS::PointVectorProxy::DataPointProxy  
        414

m\_DataBinChecksum  
    OpenGPS::ISO5436\_2Container,  
        312

m\_Details  
    OpenGPS::Exception, 216

m\_Document  
    OpenGPS::ISO5436\_2Container,  
        312

m\_FilePath  
    OpenGPS::ISO5436\_2Container,  
        312

m\_Handle  
    OpenGPS::ISO5436\_-  
        2Container::PointIteratorImpl,  
        321

    OpenGPS::ZipStreamBuffer, 592

m\_Id  
    OpenGPS::Exception, 216

m\_Index  
    OpenGPS::PointVectorProxyContextList,  
        419

m\_Instance  
    OpenGPS::Environment, 213

    OpenGPS::ISO5436\_2, 279

m\_IsCreating  
    OpenGPS::ISO5436\_2Container,  
        312

m\_IsForward  
    OpenGPS::ISO5436\_-  
        2Container::PointIteratorImpl,  
        321

m\_IsMatrix  
    OpenGPS::ISO5436\_-  
        2Container::PointIteratorImpl,  
        321

m\_IsProtected  
    OpenGPS::ISO5436\_2, 279

m\_IsReadOnly  
    OpenGPS::ISO5436\_2Container,  
        312

m\_IsReset  
    OpenGPS::ISO5436\_-  
        2Container::PointIteratorImpl,  
        321

m\_LastErrorDescription  
    OpenGPS::ExceptionHistory, 219

m\_LastErrorId  
    OpenGPS::ExceptionHistory, 219

m\_LastErrorMessage  
    OpenGPS::ExceptionHistory, 219

m\_LastErrorSource  
    OpenGPS::ExceptionHistory, 219

m\_Locale  
    OpenGPS::InputBinaryFileStream,  
        236

    OpenGPS::OutputBinaryFileStream,  
        362

    OpenGPS::PointVectorInputStream, 392

    OpenGPS::PointVectorOutputStream, 395

m\_MainChecksum  
    OpenGPS::ISO5436\_2Container,  
        312

m\_MaxIndex  
    OpenGPS::PointVectorProxyContextList,  
        419

m\_MaxU

OpenGPS::PointVectorProxyContextMatrix, OpenGPS::XmlPointVectorWriterContext, 587  
m\_MaxV m\_TempBasePath  
OpenGPS::PointVectorProxyContextMatrix, OpenGPS::ISO5436\_2Container, 313  
m\_MaxW m\_TempPath  
OpenGPS::PointVectorProxyContextMatrix, OpenGPS::ISO5436\_2Container, 313  
m\_Md5Context m\_Type  
OpenGPS::ZipStreamBuffer, 592 OpenGPS::DataPointImpl, 178  
m\_Method m\_U  
OpenGPS::Exception, 216 OpenGPS::ISO5436\_-  
m\_NeedsSeparator 2Container::PointIteratorImpl,  
OpenGPS::XmlPointVectorWriterContext, 321  
587 OpenGPS::PointVectorProxyContextMatrix, 422  
m\_Next OpenGPS::XmlPointVectorReaderContext, 581  
581 OpenGPS::ISO5436\_-  
m\_Parser 2Container::PointIteratorImpl,  
OpenGPS::PointVectorParserBuilder, 402 OpenGPS::PointVectorProxyContextMatrix, 422  
m\_PointBuffer OpenGPS::PointValidityProvider, m\_ValidBinChecksum  
OpenGPS::ISO5436\_2Container, 313 OpenGPS::ISO5436\_2Container, 313  
m\_PointDataFileName m\_ValidBuffer  
OpenGPS::ISO5436\_2Container, 313 OpenGPS::VectorBuffer, 570  
m\_PointVector OpenGPS::ValidityBuffer  
OpenGPS::ISO5436\_2Container, 313 OpenGPS::ValidBuffer, 564  
m\_PointVectorList m\_ValidityProvider  
OpenGPS::XmlPointVectorReaderContext, 581 OpenGPS::VectorBuffer, 570  
581 OpenGPS::ISO5436\_2Container, 313  
OpenGPS::XmlPointVectorWriterContext, 587 m\_Value  
OpenGPS::DataPointImpl, 178  
m\_ProxyContext OpenGPS::VectorBufferBuilder  
OpenGPS::ISO5436\_2Container, 313 OpenGPS::ISO5436\_2Container, 313  
m\_RawSize m\_VendorSpecific  
OpenGPS::ValidBuffer, 564 OpenGPS::ISO5436\_2Container, 313  
m\_Size m\_VendorURI  
OpenGPS::PointBuffer, 368 OpenGPS::BinaryPointVectorReaderConfig, OpenGPS::ISO5436\_2Container, 314  
m\_Stream OpenGPS::ISO5436\_2Container, 314  
OpenGPS::BinaryPointVectorWriterConfig, OpenGPS::ISO5436\_-  
OpenGPS::XmlPointVectorReaderContext, 581 2Container::PointIteratorImpl, 321

OpenGPS::PointVectorProxyContextMatrix, 350, 351  
    422  
m\_X  
    OpenGPS::PointVector, 387  
    OpenGPS::PointVectorParser,  
        399  
    OpenGPS::PointVectorProxy, 406  
    OpenGPS::VectorBuffer, 570  
m\_Y  
    OpenGPS::PointVector, 387  
    OpenGPS::PointVectorParser,  
        399  
    OpenGPS::PointVectorProxy, 407  
    OpenGPS::VectorBuffer, 571  
m\_Z  
    OpenGPS::PointVector, 387  
    OpenGPS::PointVectorParser,  
        399  
    OpenGPS::PointVectorProxy, 407  
    OpenGPS::VectorBuffer, 571  
Manufacturer  
    OpenGPS::Schemas::ISO5436\_-  
        2::InstrumentType, 245–247  
Manufacturer\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::InstrumentType, 252  
Manufacturer\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::InstrumentType, 242  
Manufacturer\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::InstrumentType, 242  
MatrixDimension  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 500–503  
MatrixDimension\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 503  
MatrixDimension\_optional  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 491, 492  
MatrixDimension\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 492  
MatrixDimension\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record3Type, 492  
MatrixDimensionType  
    OpenGPS::Schemas::ISO5436\_-  
        2::MatrixDimensionType,  
            md5  
            md5.c, 711  
            md5.h, 715  
            md5.c, 709  
            F, 710, 711  
            GET ULONG LE, 711  
            md5, 711  
            md5\_file, 712  
            md5\_finish, 712  
            md5\_hmac, 712  
            md5\_hmac\_finish, 712  
            md5\_hmac\_starts, 712  
            md5\_hmac\_update, 713  
            md5\_padding, 714  
            md5\_process, 713  
            md5\_starts, 713  
            md5\_update, 713  
            P, 711  
            PUT ULONG LE, 711  
            S, 711  
            XYSSL\_MD5\_C, 711  
md5.h, 714  
    md5, 715  
    md5\_file, 715  
    md5\_finish, 716  
    md5\_hmac, 716  
    md5\_hmac\_finish, 716  
    md5\_hmac\_starts, 716  
    md5\_hmac\_update, 716  
    md5\_self\_test, 717  
    md5\_starts, 717  
    md5\_update, 717  
md5\_context, 356  
    buffer, 357  
    ipad, 357  
    opad, 357  
    state, 357  
    total, 357  
md5\_file  
    md5.c, 712  
    md5.h, 715  
md5\_finish  
    md5.c, 712  
    md5.h, 716  
md5\_hmac  
    md5.c, 712  
    md5.h, 716  
md5\_hmac\_finish  
    md5.c, 712

md5.h, 716  
    md5\_hmac\_starts  
        md5.c, 712  
        md5.h, 716  
    md5\_hmac\_update  
        md5.c, 713  
        md5.h, 716  
    md5\_padding  
        md5.c, 714  
    md5\_process  
        md5.c, 713  
    md5\_self\_test  
        md5.h, 717  
    md5\_starts  
        md5.c, 713  
        md5.h, 717  
    md5\_update  
        md5.c, 713  
        md5.h, 717  
MD5ChecksumPointData  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataLinkType, 147–149  
MD5ChecksumPointData  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataLinkType, 156  
MD5ChecksumPointData\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataLinkType, 143  
MD5ChecksumPointData\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataLinkType, 143  
MD5ChecksumValidPoints  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataLinkType, 149–151  
MD5ChecksumValidPoints  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataLinkType, 156  
MD5ChecksumValidPoints\_optional  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataLinkType, 143  
MD5ChecksumValidPoints\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataLinkType, 143, 144  
MD5ChecksumValidPoints\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataLinkType, 144  
mediumComplexExample  
    ISO5436\_2\_XML\_Demo.cxx,  
        681  
messages.h, 717  
\_OPENGPS\_EXCEPTION\_ID,  
    719  
OGPS\_ExGeneral, 719  
OGPS\_ExInvalidArgument, 719  
OGPS\_ExInvalidOperationException, 719  
OGPS\_ExNone, 719  
OGPS\_ExNotImplemented, 719  
OGPS\_ExOverflow, 719  
OGPS\_ExWarning, 719  
OGPS\_ExceptionId, 719  
ogs\_GetErrorDescription, 719  
ogs\_GetErrorId, 719  
ogs\_GetErrorMessage, 720  
ogs\_HasError, 720  
messages\_c.hxx, 720  
    ogs\_GetErrorDescription, 721  
    ogs\_GetErrorId, 721  
    ogs\_GetErrorMessage, 722  
    ogs\_HasError, 722  
messages\_c.hxx, 722  
    \_OPENGPS\_GENERIC\_-  
        EXCEPTION\_HANDLER,  
        724  
    \_OPENGPS\_GENERIC\_-  
        EXCEPTION\_-  
        HANDLER\_CLEANUP,  
        724  
    \_OPENGPS\_GENERIC\_-  
        EXCEPTION\_-  
        HANDLER\_-  
        RETVALBOOL, 725  
method  
    OpenGPS::Exception, 215  
missing\_data\_point\_parser.cxx, 726  
missing\_data\_point\_parser.hxx, 726  
MissingDataPointParser  
    OpenGPS::MissingDataPointParser,  
        358  
Model  
    OpenGPS::Schemas::ISO5436\_-  
        2::InstrumentType, 247–249  
Model\_-  
    OpenGPS::Schemas::ISO5436\_-  
        2::InstrumentType, 252  
Model\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::InstrumentType, 242  
Model\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::InstrumentType, 242

month  
    xml\_schema, 64

month\_day  
    xml\_schema, 64

MoveNext  
    OpenGPS::BinaryPointVectorReaderContext, schema, 65  
        130  
    not\_derived

OpenGPS::BinaryPointVectorWriterContext, schema, 65  
        134  
    NULL

OpenGPS::ISO5436\_-  
    2Container::PointIteratorImpl,  
        319  
    Offset

OpenGPS::PointIterator, 371  
    OpenGPS::Schemas::ISO5436\_-  
    2::AxisDescriptionType, 105,

OpenGPS::PointVectorReaderContext,  
    424  
    Offset\_

OpenGPS::PointVectorWriterContext,  
    430  
    Offset\_ optional

OpenGPS::XmlPointVectorReaderContext,  
    579  
    Offset\_

OpenGPS::XmlPointVectorWriterContext,  
    585  
    Offset\_ optional

MovePrev  
    OpenGPS::ISO5436\_-  
        2Container::PointIteratorImpl, Offset\_traits  
            319  
    Offset\_

OpenGPS::PointIterator, 372  
    2::AxisDescriptionType,  
        97  
    Offset\_type

name  
    xml\_schema, 64

namespace\_info  
    xml\_schema, 64

namespace\_infomap  
    xml\_schema, 64

ncname  
    xml\_schema, 64

negative\_integer  
    xml\_schema, 64

nmtoken  
    xml\_schema, 64

nmtokens  
    xml\_schema, 64

no\_namespace\_mapping  
    xml\_schema, 64

no\_prefix\_mapping  
    xml\_schema, 64

no\_type\_info  
    xml\_schema, 64

non\_negative\_integer  
    xml\_schema, 64

non\_positive\_integer

NonContacting  
    OpenGPS::Schemas::ISO5436\_-  
        2::Type, 555  
    normalized\_string

normalize  
    OpenGPS::Schemas::ISO5436\_-  
        2::Type, 555  
    normalized\_string

OpenGPS::Schemas::ISO5436\_-  
    2::AxisDescriptionType, 106

OpenGPS::Schemas::ISO5436\_-  
    2::AxisDescriptionType, 107

OpenGPS::Schemas::ISO5436\_-  
    2::AxisDescriptionType, 96,

OpenGPS::Schemas::ISO5436\_-  
    97

OGPS\_DoublePointType  
    data\_point\_type.h, 623

OGPS\_ExGeneral  
    messages.h, 719

OGPS\_ExInvalidArgument  
    messages.h, 719

OGPS\_ExInvalidOperationException  
    messages.h, 719

OGPS\_ExNone  
    messages.h, 719

OGPS\_ExNotImplemented  
    messages.h, 719

OGPS\_ExOverflow  
    messages.h, 719

OGPS\_ExWarning  
    messages.h, 719

OGPS\_FloatPointType  
    data\_point\_type.h, 623

OGPS\_Int16PointType

data\_point\_type.h, 623  
OGPS\_Int32PointType  
    data\_point\_type.h, 623  
OGPS\_MissingPointType  
    data\_point\_type.h, 623  
ogps\_AppendVendorSpecific  
    iso5436\_2.h, 651  
    iso5436\_2\_c.hxx, 664  
OGPS\_Boolean  
    opengps.h, 729  
OGPS\_Character  
    opengps.h, 729  
ogps\_CloseISO5436\_2  
    iso5436\_2.h, 652  
    iso5436\_2\_c.hxx, 664  
ogps\_CreateListISO5436\_2  
    iso5436\_2\_c.hxx, 664  
    iso5436\_2\_handle.hxx, 678  
ogps\_CreateMatrixISO5436\_2  
    iso5436\_2\_c.hxx, 665  
    iso5436\_2\_handle.hxx, 678  
ogps\_CreateNextPointIterator  
    iso5436\_2.h, 652  
    iso5436\_2\_c.hxx, 666  
ogps\_CreatePointVector  
    point\_vector.h, 752  
    point\_vector\_c.hxx, 767  
ogps\_CreatePrevPointIterator  
    iso5436\_2.h, 652  
    iso5436\_2\_c.hxx, 666  
OGPS\_DataPoint  
    data\_point\_c.hxx, 618  
OGPS\_DataPointPtr  
    data\_point.h, 609  
    data\_point\_c.hxx, 618  
OGPS\_DataPointType  
    data\_point\_type.h, 623  
OGPS\_DataPointValue  
    OpenGPS::DataPointImpl, 173  
OGPS\_Double  
    opengps.h, 729  
OGPS\_ExceptionChar  
    OpenGPS, 31  
OGPS\_ExceptionId  
    messages.h, 719  
OGPS\_Float  
    opengps.h, 729  
ogps\_FreePointIterator  
    point\_iterator.h, 735  
    point\_iterator\_c.hxx, 741  
ogps\_FreePointVector  
    point\_vector.h, 752  
    point\_vector\_c.hxx, 767  
ogps\_GetAboutInfo  
    info.h, 634  
    info\_c.hxx, 639  
ogps\_GetCopyrightInfo  
    info.h, 635  
    info\_c.hxx, 639  
ogps\_GetCurrentPoint  
    point\_iterator.h, 735  
    point\_iterator\_c.hxx, 742  
ogps\_GetDocument  
    iso5436\_2\_c.hxx, 667  
    iso5436\_2\_handle.hxx, 679  
ogps\_GetDouble  
    data\_point.h, 610  
    data\_point\_c.hxx, 615  
ogps\_GetDoubleX  
    point\_vector.h, 753  
    point\_vector\_c.hxx, 767  
ogps\_GetDoubleY  
    point\_vector.h, 753  
    point\_vector\_c.hxx, 768  
ogps\_GetDoubleZ  
    point\_vector.h, 753  
    point\_vector\_c.hxx, 768  
ogps\_GetErrorDescription  
    messages.h, 719  
    messages\_c.hxx, 721  
ogps\_GetErrorId  
    messages.h, 719  
    messages\_c.hxx, 721  
ogps\_GetErrorMessage  
    messages.h, 720  
    messages\_c.hxx, 722  
ogps\_GetFloat  
    data\_point.h, 610  
    data\_point\_c.hxx, 615  
ogps\_GetFloatX  
    point\_vector.h, 754  
    point\_vector\_c.hxx, 768  
ogps\_GetFloatY  
    point\_vector.h, 754  
    point\_vector\_c.hxx, 769  
ogps\_GetFloatZ  
    point\_vector.h, 754  
    point\_vector\_c.hxx, 769  
ogps.GetInt16  
    data\_point.h, 610

data\_point\_c.hxx, 615  
ogps\_GetInt16X  
    point\_vector.h, 755  
    point\_vector\_c.hxx, 769  
ogps\_GetInt16Y  
    point\_vector.h, 755  
    point\_vector\_c.hxx, 770  
ogps\_GetInt16Z  
    point\_vector.h, 755  
    point\_vector\_c.hxx, 770  
ogps\_GetInt32  
    data\_point.h, 611  
    data\_point\_c.hxx, 616  
ogps\_GetInt32X  
    point\_vector.h, 756  
    point\_vector\_c.hxx, 770  
ogps\_GetInt32Y  
    point\_vector.h, 756  
    point\_vector\_c.hxx, 771  
ogps\_GetInt32Z  
    point\_vector.h, 756  
    point\_vector\_c.hxx, 771  
ogps\_GetLicenseInfo  
    info.h, 635  
    info\_c.hxx, 639  
ogps\_GetListCoord  
    iso5436\_2.h, 653  
    iso5436\_2\_c.hxx, 667  
ogps\_GetListPoint  
    iso5436\_2.h, 654  
    iso5436\_2\_c.hxx, 668  
ogps\_GetListPosition  
    point\_iterator.h, 735  
    point\_iterator\_c.hxx, 742  
ogps\_GetMatrixCoord  
    iso5436\_2.h, 654  
    iso5436\_2\_c.hxx, 668  
ogps\_GetMatrixPoint  
    iso5436\_2.h, 655  
    iso5436\_2\_c.hxx, 669  
ogps\_GetMatrixPosition  
    point\_iterator.h, 736  
    point\_iterator\_c.hxx, 742  
ogps.GetNameInfo  
    info.h, 635  
    info\_c.hxx, 639  
ogps\_GetPointType  
    data\_point.h, 611  
    data\_point\_c.hxx, 616  
ogps\_GetPointTypeX  
    point\_vector.h, 757  
    point\_vector\_c.hxx, 771  
ogps\_GetPointTypeY  
    point\_vector.h, 757  
    point\_vector\_c.hxx, 771  
ogps\_GetPointTypeZ  
    point\_vector.h, 757  
    point\_vector\_c.hxx, 772  
ogps\_GetVendorSpecific  
    iso5436\_2.h, 656  
    iso5436\_2\_c.hxx, 670  
ogps\_GetVersionInfo  
    info.h, 636  
    info\_c.hxx, 640  
ogpsGetX  
    point\_vector.h, 757  
    point\_vector\_c.hxx, 772  
ogpsGetXYZ  
    point\_vector.h, 757  
    point\_vector\_c.hxx, 772  
ogpsGetY  
    point\_vector.h, 758  
    point\_vector\_c.hxx, 772  
ogpsGetZ  
    point\_vector.h, 758  
    point\_vector\_c.hxx, 772  
ogps\_HasError  
    messages.h, 720  
    messages\_c.hxx, 722  
ogps\_HasNextPoint  
    point\_iterator.h, 736  
    point\_iterator\_c.hxx, 743  
ogps\_HasPrevPoint  
    point\_iterator.h, 737  
    point\_iterator\_c.hxx, 743  
OGPS\_Int16  
    opengps.h, 729  
OGPS\_Int32  
    opengps.h, 729  
ogps\_IsMatrixCoordValid  
    iso5436\_2.h, 656  
    iso5436\_2\_c.hxx, 670  
OGPS\_ISO5436\_2  
    iso5436\_2\_handle\_c.hxx, 680  
OGPS\_ISO5436\_2Handle  
    iso5436\_2.h, 651  
    iso5436\_2\_handle\_c.hxx, 680  
ogps\_IsValidPoint  
    point\_vector.h, 758  
    point\_vector\_c.hxx, 773

ogps\_MoveNextPoint  
    point\_iterator.h, 737  
    point\_iterator\_c.hxx, 744  
ogps\_MovePrevPoint  
    point\_iterator.h, 737  
    point\_iterator\_c.hxx, 744  
ogps\_OpenISO5436\_2  
    iso5436\_2.h, 657  
    iso5436\_2\_c.hxx, 671  
OGPS\_PointIterator  
    point\_iterator\_c.hxx, 746  
OGPS\_PointIteratorPtr  
    point\_iterator.h, 734  
    point\_iterator\_c.hxx, 746  
OGPS\_PointVector  
    point\_vector\_c.hxx, 776  
OGPS\_PointVectorPtr  
    point\_vector.h, 752  
    point\_vector\_c.hxx, 776  
ogps\_PrintCopyrightInfo  
    info.h, 636  
    info\_c.hxx, 640  
ogps\_PrintLicenseInfo  
    info.h, 636  
    info\_c.hxx, 640  
ogps\_PrintVersionInfo  
    info.h, 636  
    info\_c.hxx, 640  
ogps\_ResetNextPointIterator  
    point\_iterator.h, 738  
    point\_iterator\_c.hxx, 744  
ogps\_ResetPrevPointIterator  
    point\_iterator.h, 738  
    point\_iterator\_c.hxx, 745  
ogps\_SetCurrentPoint  
    point\_iterator.h, 738  
    point\_iterator\_c.hxx, 745  
ogps\_SetDouble  
    data\_point.h, 611  
    data\_point\_c.hxx, 616  
ogps\_SetDoubleX  
    point\_vector.h, 758  
    point\_vector\_c.hxx, 773  
ogps\_SetDoubleY  
    point\_vector.h, 759  
    point\_vector\_c.hxx, 773  
ogps\_SetDoubleZ  
    point\_vector.h, 759  
    point\_vector\_c.hxx, 773  
ogps\_SetFloat  
    data\_point.h, 612  
    data\_point\_c.hxx, 617  
ogps\_SetFloatX  
    point\_vector.h, 759  
    point\_vector\_c.hxx, 774  
ogps\_SetFloatY  
    point\_vector.h, 759  
    point\_vector\_c.hxx, 774  
ogps\_SetFloatZ  
    point\_vector.h, 759  
    point\_vector\_c.hxx, 774  
ogps\_SetInt16  
    data\_point.h, 612  
    data\_point\_c.hxx, 617  
ogps\_SetInt16X  
    point\_vector.h, 760  
    point\_vector\_c.hxx, 774  
ogps\_SetInt16Y  
    point\_vector.h, 760  
    point\_vector\_c.hxx, 774  
ogps\_SetInt16Z  
    point\_vector.h, 760  
    point\_vector\_c.hxx, 775  
ogps\_SetInt32  
    data\_point.h, 612  
    data\_point\_c.hxx, 617  
ogps\_SetInt32X  
    point\_vector.h, 760  
    point\_vector\_c.hxx, 775  
ogps\_SetInt32Y  
    point\_vector.h, 760  
    point\_vector\_c.hxx, 775  
ogps\_SetInt32Z  
    point\_vector.h, 761  
    point\_vector\_c.hxx, 775  
ogps\_SetListPoint  
    iso5436\_2.h, 658  
    iso5436\_2\_c.hxx, 671  
ogps\_SetMatrixPoint  
    iso5436\_2.h, 658  
    iso5436\_2\_c.hxx, 672  
ogps\_WriteISO5436\_2  
    iso5436\_2.h, 659  
    iso5436\_2\_c.hxx, 673  
opad  
    md5\_context, 357  
Open  
    OpenGPS::ISO5436\_2, 276  
    OpenGPS::ISO5436\_2Container,  
        306

OpenGPS, 26  
    Byte, 31  
    BytePtr, 31  
    OGPS\_ExceptionChar, 31  
    PointBufferAutoPtr, 31  
    PointIteratorAutoPtr, 31  
    PointVectorAutoPtr, 32  
    PointVectorParserBuilder-  
        AutoPtr, 32  
    PointVectorProxyContextAu-  
        toPtr, 32  
    UnsignedByte, 32  
    UnsignedBytePtr, 32  
    VectorBufferAutoPtr, 32  
    VectorBufferBuilderAutoPtr, 32  
opengps.h, 727  
    \_EX\_T, 728  
    \_OPENGPS\_EXPORT, 728  
FALSE, 728  
NULL, 729  
    OGPS\_Boolean, 729  
    OGPS\_Character, 729  
    OGPS\_Double, 729  
    OGPS\_Float, 729  
    OGPS\_Int16, 729  
    OGPS\_Int32, 729  
    TRUE, 729  
opengps.hxx, 730  
opengps/ Directory Reference, 25  
opengps/cxx/ Directory Reference, 19  
OpenGPS::BinaryLSBPointVectorReaderCon-  
    115  
    ~BinaryLSBPointVectorReaderContext, 117  
    BinaryLSBPointVectorReader-  
        Context, 117  
        Read, 117, 118  
OpenGPS::BinaryLSBPointVectorWriter  
    119  
    ~BinaryLSBPointVectorWriterContext, 120  
    BinaryLSBPointVectorWriter-  
        Context, 120  
        Write, 120, 121  
OpenGPS::BinaryMSBPointVectorReaderCon-  
    121  
    ~BinaryMSBPointVectorReaderCon-  
        123  
    BinaryMSBPointVectorReader-  
        Context, 123  
            Read, 123, 124  
OpenGPS::BinaryMSBPointVectorWriterContext,  
    125  
    ~BinaryMSBPointVectorWriterContext,  
        126  
    BinaryMSBPointVectorWriter-  
        Context, 126  
        Write, 126, 127  
OpenGPS::BinaryPointVectorReaderContext,  
    127  
    ~BinaryPointVectorReaderContext,  
        129  
    BinaryPointVectorReaderCon-  
        text, 129  
    Close, 130  
    GetStream, 130  
    HasStream, 130  
    IsGood, 130  
    IsValid, 130  
    m\_Stream, 131  
    MoveNext, 130  
    Skip, 131  
OpenGPS::BinaryPointVectorWriterContext,  
    132  
    ~BinaryPointVectorWriterContext,  
        133  
    BinaryPointVectorWriterCon-  
        text, 133  
    Close, 133  
    GetMd5, 133  
    GetStream, 134  
    HasStream, 134  
    IsGood, 134  
    m\_Buffer, 135  
    m\_Stream, 135  
    MoveNext, 134  
    Skip, 135  
OpenGPS::DataPoint, 164  
    ~DataPoint, 166  
    DataPoint, 166  
        Get, 166–168  
        GetPointType, 168  
        IsValid, 168  
        operator=, 169  
    Set, 169, 170  
OpenGPS::DataPointImpl, 171  
    ~DataPointImpl, 173  
    DataPointImpl, 173  
        Get, 174, 175

GetPointType, 175  
IsValid, 176  
m\_Type, 178  
m\_Value, 178  
OGPS\_DataPointValue, 173  
Reset, 176  
Set, 176, 177  
OpenGPS::DataPointImpl:: OGPS\_-  
    DATA\_POINT\_VALUE,  
    178  
    doubleValue, 179  
    floatValue, 179  
    int16Value, 179  
    int32Value, 179  
OpenGPS::DataPointParser, 180  
    ~DataPointParser, 180  
    DataPointParser, 180  
    Read, 181  
    Write, 181  
OpenGPS::DoubleDataPointParser,  
    195  
    ~DoubleDataPointParser, 196  
    DoubleDataPointParser, 196  
    Read, 196  
    Write, 196  
OpenGPS::DoubleInlineValidity, 197  
    ~DoubleInlineValidity, 198  
    DoubleInlineValidity, 198  
    IsValid, 198  
    SetValid, 198  
OpenGPS::DoublePointBuffer, 199  
    ~DoublePointBuffer, 200  
    Allocate, 200  
    DoublePointBuffer, 200  
    Get, 200  
    GetPointType, 201  
    m\_Buffer, 201  
    Set, 201  
OpenGPS::Environment, 202  
    ~Environment, 205  
    ByteSwap, 205–207  
    ByteSwap16, 207  
    ByteSwap32, 208, 209  
    ByteSwap64, 209  
    ConcatPathes, 209  
    CreateDir, 210  
    CreateInstance, 210  
    Environment, 205  
    GetAltDirectorySeparator, 210  
    GetDirectorySeparator, 210  
    GetFileName, 210  
    GetInstance, 211  
    GetPathName, 211  
    GetTempDir, 211  
    GetUniqueName, 211  
    GetVariable, 211  
    IsLittleEndian, 212  
    m\_Instance, 213  
    PathExists, 212  
    RemoveDir, 212  
    RemoveFile, 212  
    RenameFile, 212  
    Reset, 213  
OpenGPS::Exception, 213  
    ~Exception, 215  
    details, 215  
    Exception, 215  
    id, 215  
    m\_Details, 216  
    m\_Id, 216  
    m\_Method, 216  
    method, 215  
OpenGPS::ExceptionHistory, 216  
    ~ExceptionHistory, 218  
    DumpIt, 218  
    ExceptionHistory, 218  
    GetLastErrorDescription, 218  
    GetLastErrorId, 218  
    GetLastErrorMessage, 218  
    m\_LastErrorDescription, 219  
    m\_LastErrorId, 219  
    m\_LastErrorMessage, 219  
    m\_LastErrorSource, 219  
    Reset, 218  
    SetLastException, 218, 219  
OpenGPS::FloatDataPointParser, 224  
    ~FloatDataPointParser, 225  
    FloatDataPointParser, 225  
    Read, 226  
    Write, 226  
OpenGPS::FloatInlineValidity, 226  
    ~FloatInlineValidity, 228  
    FloatInlineValidity, 228  
    IsValid, 228  
    SetValid, 228  
OpenGPS::FloatPointBuffer, 229  
    ~FloatPointBuffer, 230  
    Allocate, 230  
    FloatPointBuffer, 230  
    Get, 230

GetPointType, 230  
m\_Buffer, 231  
Set, 231  
OpenGPS::Info, 231  
    ~Info, 232  
    GetAbout, 232  
    GetCopyright, 232  
    GetLicense, 233  
    GetName, 233  
    GetVersion, 233  
    Info, 232  
    PrintCopyright, 233  
    PrintLicense, 233  
    PrintVersion, 233  
OpenGPS::InputBinaryFileStream,  
    234  
    ~InputBinaryFileStream, 235  
    BaseType, 235  
    InputBinaryFileStream, 235  
    m\_Locale, 236  
OpenGPS::Int16DataPointParser, 253  
    ~Int16DataPointParser, 254  
    Int16DataPointParser, 254  
    Read, 254  
    Write, 254  
OpenGPS::Int16PointBuffer, 255  
    ~Int16PointBuffer, 256  
    Allocate, 256  
    Get, 257  
    GetPointType, 257  
    Int16PointBuffer, 256  
    m\_Buffer, 257  
    Set, 257  
OpenGPS::Int16ValidBuffer, 258  
    ~Int16ValidBuffer, 259  
    Int16ValidBuffer, 259  
    SetValid, 259  
OpenGPS::Int32DataPointParser, 259  
    ~Int32DataPointParser, 260  
    Int32DataPointParser, 260  
    Read, 260  
    Write, 261  
OpenGPS::Int32PointBuffer, 261  
    ~Int32PointBuffer, 262  
    Allocate, 263  
    Get, 263  
    GetPointType, 263  
    Int32PointBuffer, 262  
    m\_Buffer, 264  
    Set, 263  
OpenGPS::Int32ValidBuffer, 264  
    ~Int32ValidBuffer, 265  
    Int32ValidBuffer, 265  
    SetValid, 265  
OpenGPS::ISO5436\_2, 266  
    ~ISO5436\_2, 269  
    AppendVendorSpecific, 270  
    Close, 270  
    Create, 270, 271  
    CreateNextPointIterator, 271  
    CreatePrevPointIterator, 272  
    GetDocument, 272  
    GetListCoord, 272  
    GetListPoint, 273  
    GetMatrixCoord, 273  
    GetMatrixPoint, 274  
    GetVendorSpecific, 275  
    IsMatrixCoordValid, 276  
    ISO5436\_2, 268, 269  
    m\_Instance, 279  
    m\_IsProtected, 279  
    Open, 276  
    operator=, 277  
    SetListPoint, 277  
    SetMatrixPoint, 277  
    Write, 278  
OpenGPS::ISO5436\_2Container, 279  
    ~ISO5436\_2Container, 289  
    AppendVendorSpecific, 290  
    BuildPointVectorParser, 290  
    BuildVectorBuffer, 290  
    Close, 291  
    Compress, 291  
    ConfigureNamespaceMap, 291  
    ConvertPointToCoord, 291  
    ConvertULongLongToULong, 292  
    ConvertULongToInt32, 292  
    Create, 292, 293  
    CreateContainerTempFilePath,  
        294  
    CreateDocument, 294  
    CreateNextPointIterator, 294  
    CreatePointBuffer, 294  
    CreatePointVectorProxyContext,  
        295  
    CreatePointVectorReaderCon-  
        text, 295  
    CreatePointVectorWriterContext,  
        295  
    CreatePrevPointIterator, 295

CreateTempDir, 296  
Decompress, 296  
DecompressChecksum, 296  
DecompressDataBin, 297  
DecompressMain, 297  
GetAxisDataType, 297  
GetChecksumArchiveName, 297  
GetChecksumFileName, 298  
GetDocument, 298  
GetFilePath, 298  
GetFullPath, 298  
GetIncrementX, 298  
GetIncrementY, 298  
GetListCoord, 298  
GetListPoint, 299  
GetMainArchiveName, 300  
GetMainFileName, 300  
GetMatrixCoord, 300  
GetMatrixPoint, 301  
GetMaxU, 301  
GetMaxV, 302  
GetMaxW, 302  
GetOffsetX, 302  
GetOffsetY, 303  
GetOffsetZ, 303  
GetPointCount, 303  
GetPointDataArchiveName, 303  
GetPointDataFileName, 303  
GetTempDir, 303  
GetValidPointsArchiveName, 303  
GetValidPointsFileName, 303  
GetVectorBuffer, 303  
GetVendorSpecific, 303  
GetXaxisDataType, 304  
GetYaxisDataType, 304  
GetZaxisDataType, 304  
HasDocument, 304  
HasTempDir, 304  
HasValidPointsLink, 305  
HasVectorBuffer, 305  
IsBinary, 305  
IsIncrementalX, 305  
IsIncrementalY, 305  
IsMatrix, 305  
IsMatrixCoordValid, 306  
ISO5436\_2Container, 289  
m\_CompressionLevel, 312  
m\_DataBinChecksum, 312  
m\_Document, 312  
m\_FilePath, 312  
m\_IsCreating, 312  
m\_IsReadOnly, 312  
m\_MainChecksum, 312  
m\_PointDataFileName, 313  
m\_PointVector, 313  
m\_ProxyContext, 313  
m\_TempBasePath, 313  
m\_TempPath, 313  
m\_ValidBinChecksum, 313  
m\_ValidPointsFileName, 313  
m\_VectorBufferBuilder, 313  
m\_VendorSpecific, 313  
m\_VendorURI, 314  
Open, 306  
ReadDocument, 307  
ReadMd5FromFile, 307  
ReadXmlDocument, 307  
RemoveTempDir, 307  
Reset, 307  
ResetValidPointsLink, 307  
ResetXmlPointList, 308  
SafeMultiplication, 308  
SaveChecksumFile, 308  
SavePointBuffer, 308  
SaveValidPointsLink, 308  
SaveXmlDocument, 309  
SetListPoint, 309  
SetMatrixPoint, 309  
StringList, 289  
TestChecksums, 310  
VerifyChecksum, 310  
VerifyDataBinChecksum, 311  
VerifyMainChecksum, 311  
VerifyValidBinChecksum, 311  
Write, 311  
WriteVendorSpecific, 311  
OpenGPS::ISO5436\_-  
    2Container::PointIteratorImpl,  
        314  
    ~PointIteratorImpl, 317  
    GetCurrent, 317  
    GetPosition, 317, 318  
    HasNext, 318  
    HasPrev, 318  
    m\_Handle, 321  
    m\_IsForward, 321  
    m\_IsMatrix, 321  
    m\_IsReset, 321  
    m\_U, 321  
    m\_V, 321

m\_W, 321  
MoveNext, 319  
MovePrev, 319  
operator=, 320  
PointIteratorImpl, 316, 317  
ResetNext, 320  
ResetPrev, 320  
SetCurrent, 320  
OpenGPS::MissingDataPointParser,  
    357  
    ~MissingDataPointParser, 358  
    MissingDataPointParser, 358  
    Read, 358  
    Write, 359  
OpenGPS::OutputBinaryFileStream,  
    359  
    ~OutputBinaryFileStream, 361  
    BaseType, 361  
    m\_Locale, 362  
    OutputBinaryFileStream, 361  
OpenGPS::PointBuffer, 362  
    ~PointBuffer, 364  
    Allocate, 364  
    Free, 365  
    Get, 365, 366  
    GetPointType, 366  
    GetSize, 366  
    m\_Size, 368  
    PointBuffer, 364  
    Set, 366, 367  
OpenGPS::PointIterator, 368  
    ~PointIterator, 369  
    GetCurrent, 370  
    GetPosition, 370  
    HasNext, 371  
    HasPrev, 371  
    MoveNext, 371  
    MovePrev, 372  
    PointIterator, 369  
    ResetNext, 372  
    ResetPrev, 372  
    SetCurrent, 372  
OpenGPS::PointValidityProvider, 373  
    ~PointValidityProvider, 374  
    GetPointBuffer, 375  
    IsValid, 375  
    m\_PointBuffer, 376  
    PointValidityProvider, 374  
    SetValid, 375  
OpenGPS::PointVector, 376  
    ~PointVector, 379  
    Get, 379  
    GetX, 379–381  
    GetXYZ, 381  
    GetY, 381, 382  
    GetZ, 383, 384  
    IsValid, 384  
    m\_X, 387  
    m\_Y, 387  
    m\_Z, 387  
    operator=, 384  
    PointVector, 379  
    Set, 384  
    SetX, 385  
    SetY, 385, 386  
    SetZ, 386, 387  
OpenGPS::PointVectorBase, 387  
    ~PointVectorBase, 388  
    Get, 389  
    GetX, 389  
    GetY, 389  
    GetZ, 389  
    PointVectorBase, 388  
    Set, 390  
OpenGPS::PointVectorInputStream, 390  
    ~PointVectorInputStream, 392  
    BaseType, 392  
    m\_Locale, 392  
    PointVectorInputStream, 392  
OpenGPS::PointVectorInvariantLocale, 392  
    ~PointVectorInvariantLocale, 393  
    BaseType, 393  
    PointVectorInvariantLocale, 393  
OpenGPS::PointVectorOutputStream, 393  
    ~PointVectorOutputStream, 395  
    BaseType, 395  
    m\_Locale, 395  
    PointVectorOutputStream, 395  
OpenGPS::PointVectorParser, 396  
    ~PointVectorParser, 397  
    CreateDataPointParser, 397  
    m\_X, 399  
    m\_Y, 399

m\_Z, 399  
PointVectorParser, 397  
Read, 398  
SetX, 398  
SetY, 398  
SetZ, 398  
Write, 399  
OpenGPS::PointVectorParserBuilder, 400  
~PointVectorParserBuilder, 401  
BuildParser, 401  
BuildX, 401  
BuildY, 401  
BuildZ, 402  
GetParser, 402  
m\_Parser, 402  
PointVectorParserBuilder, 401  
OpenGPS::PointVectorProxy, 402  
~PointVectorProxy, 405  
Get, 405  
GetX, 405  
GetY, 405  
GetZ, 406  
m\_Buffer, 406  
m\_Context, 406  
m\_X, 406  
m\_Y, 407  
m\_Z, 407  
operator=, 406  
PointVectorProxy, 404, 405  
Set, 406  
OpenGPS::PointVectorProxy::DataPoint  
OpenGPS::PointVectorReaderContext, 407  
~DataPointProxy, 409  
DataPointProxy, 409  
Get, 409–411  
GetPointType, 411  
IsValid, 412  
m\_Buffer, 414  
m\_Context, 414  
operator=, 412  
Reset, 412  
Set, 412, 413  
OpenGPS::PointVectorProxyContext, 414  
~PointVectorProxyContext, 415  
CanIncrementIndex, 415  
GetIndex, 415  
IncrementIndex, 416  
IsMatrix, 416  
PointVectorProxyContext, 415  
OpenGPS::PointVectorProxyContextList, 416  
~PointVectorProxyContextList, 418  
CanIncrementIndex, 418  
GetIndex, 418  
IncrementIndex, 418  
IsMatrix, 418  
m\_Index, 419  
m\_MaxIndex, 419  
PointVectorProxyContextList, 417  
SetIndex, 418  
OpenGPS::PointVectorProxyContextMatrix, 419  
~PointVectorProxyContextMatrix, 421  
CanIncrementIndex, 421  
GetIndex, 421  
IncrementIndex, 421  
IsMatrix, 421  
m\_MaxU, 422  
m\_MaxV, 422  
m\_MaxW, 422  
m\_U, 422  
m\_V, 422  
m\_W, 422  
PointVectorProxyContextMatrix, 421  
SetIndex, 421  
OpenGPS::PointVectorReaderContext, 423  
~PointVectorReaderContext, 424  
IsValid, 424  
MoveNext, 424  
PointVectorReaderContext, 424  
Read, 425, 426  
Skip, 426  
OpenGPS::PointVectorWhitespaceFacet, 427  
~PointVectorWhitespaceFacet, 428  
BaseType, 428  
do\_is, 428  
PointVectorWhitespaceFacet, 428  
OpenGPS::PointVectorWriterContext, 428  
~PointVectorWriterContext, 430  
IsGood, 430

MoveNext, 430  
PointVectorWriterContext, 430  
Skip, 430  
Write, 431, 432  
OpenGPS::Schemas, 32  
OpenGPS::Schemas::ISO5436\_-  
    2::AxisType  
        A, 110  
        I, 110  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataType  
        D, 184  
        F, 184  
        I, 184  
        L, 184  
    OpenGPS::Schemas::ISO5436\_-  
        2::Type  
        Contacting, 555  
        NonContacting, 555  
OpenGPS::Schemas::ISO5436\_2, 33  
    ISO5436\_2, 39–48  
    operator!=, 48, 49  
    operator<<, 49–54  
    operator==, 54, 55  
OpenGPS::Schemas::ISO5436\_-  
    2::AxesType, 70  
    \_clone, 80  
AxesType, 78, 79  
CX, 80–82  
CX\_, 88  
CX\_traits, 76  
CX\_type, 77  
CY, 82, 83  
CY\_, 88  
CY\_traits, 77  
CY\_type, 77  
CZ, 84, 85  
CZ\_, 88  
CZ\_traits, 77  
CZ\_type, 77, 78  
parse, 85  
Rotation, 85–87  
Rotation\_, 88  
Rotation\_optional, 78  
Rotation\_traits, 78  
Rotation\_type, 78  
OpenGPS::Schemas::ISO5436\_-  
    2::AxisDescriptionType,  
        88  
    \_clone, 99  
AxisDescriptionType, 97, 98  
AxisType, 99–101  
AxisType\_, 107  
AxisType\_traits, 95  
AxisType\_type, 95  
DataType, 101–103  
DataType\_, 107  
DataType\_optional, 95  
DataType\_traits, 95  
DataType\_type, 95, 96  
Increment, 103–105  
Increment\_, 107  
Increment\_optional, 96  
Increment\_traits, 96  
Increment\_type, 96  
Offset, 105, 106  
Offset\_, 107  
Offset\_optional, 96, 97  
Offset\_traits, 97  
Offset\_type, 97  
parse, 107  
OpenGPS::Schemas::ISO5436\_-  
    2::AxisType, 107  
    \_clone, 113  
    \_xsd\_AxisType\_convert, 113,  
        114  
    \_xsd\_AxisType\_indexes\_, 115  
    \_xsd\_AxisType\_literals\_, 115  
AxisType, 110–112  
operator value, 114  
operator=, 114  
value, 109, 110  
OpenGPS::Schemas::ISO5436\_-  
    2::DataLinkType, 135  
    \_clone, 147  
DataLinkType, 145, 146  
MD5ChecksumPointData, 147–  
    149  
MD5ChecksumPointData\_, 156  
MD5ChecksumPointData\_traits,  
    143  
MD5ChecksumPointData\_type,  
    143  
MD5ChecksumValidPoints, 149–  
    151  
MD5ChecksumValidPoints\_, 156  
MD5ChecksumValidPoints\_-  
    optional, 143  
MD5ChecksumValidPoints\_-  
    traits, 143, 144

MD5ChecksumValidPoints\_type, 144  
parse, 151  
PointDataLink, 151–153  
PointDataLink\_, 156  
PointDataLink\_traits, 144  
PointDataLink\_type, 144  
ValidPointsLink, 153–155  
ValidPointsLink\_, 156  
ValidPointsLink\_optional, 144, 145  
ValidPointsLink\_traits, 145  
ValidPointsLink\_type, 145  
OpenGPS::Schemas::ISO5436\_-  
    2::DataListType, 156  
    \_clone, 161, 162  
    DataListType, 160, 161  
    Datum, 162, 163  
    Datum\_, 164  
    Datum\_const\_iterator, 159  
    Datum\_iterator, 159  
    Datum\_sequence, 159  
    Datum\_traits, 159, 160  
    Datum\_type, 160  
    parse, 163  
OpenGPS::Schemas::ISO5436\_-  
    2::DataType, 182  
    \_clone, 188  
    \_xsd\_DataType\_convert, 188  
    \_xsd\_DataType\_indexes\_, 189  
    \_xsd\_DataType\_literals\_, 189  
    DataType, 185–187  
    operator value, 188, 189  
    operator=, 189  
    value, 184  
OpenGPS::Schemas::ISO5436\_-  
    2::Datum, 190  
    \_clone, 194  
    Datum, 191–193  
OpenGPS::Schemas::ISO5436\_-  
    2::FeatureType, 220  
    \_clone, 224  
    FeatureType, 221–223  
OpenGPS::Schemas::ISO5436\_-  
    2::InstrumentType, 236  
    \_clone, 245  
    InstrumentType, 243, 244  
    Manufacturer, 245–247  
    Manufacturer\_, 252  
    Manufacturer\_traits, 242  
    Manufacturer\_type, 242  
Model, 247–249  
Model\_, 252  
Model\_traits, 242  
Model\_type, 242  
parse, 249  
Serial, 249, 250  
Serial\_, 253  
Serial\_traits, 242  
Serial\_type, 243  
Version, 251, 252  
Version\_, 253  
Version\_traits, 243  
Version\_type, 243  
OpenGPS::Schemas::ISO5436\_-  
    2::ISO5436\_2Type, 322  
    \_clone, 333  
    ISO5436\_2Type, 332, 333  
    parse, 334  
    Record1, 334, 335  
    Record1\_, 343  
    Record1\_traits, 329  
    Record1\_type, 329  
    Record2, 336–338  
    Record2\_, 343  
    Record2\_optional, 329, 330  
    Record2\_traits, 330  
    Record2\_type, 330  
    Record3, 338, 339  
    Record3\_, 344  
    Record3\_traits, 330  
    Record3\_type, 330  
    Record4, 339–341  
    Record4\_, 344  
    Record4\_traits, 331  
    Record4\_type, 331  
    VendorSpecificID, 341–343  
    VendorSpecificID\_, 344  
    VendorSpecificID\_optional, 331  
    VendorSpecificID\_traits, 331  
    VendorSpecificID\_type, 331, 332  
OpenGPS::Schemas::ISO5436\_-  
    2::MatrixDimensionType,  
        344  
        \_clone, 351  
        MatrixDimensionType, 350, 351  
        parse, 352  
        SizeX, 352, 353  
        SizeX\_, 356  
        SizeX\_traits, 348

SizeX\_type, 348  
SizeY, 353, 354  
SizeY\_, 356  
SizeY\_traits, 349  
SizeY\_type, 349  
SizeZ, 354, 355  
SizeZ\_, 356  
SizeZ\_traits, 349  
SizeZ\_type, 349  
OpenGPS::Schemas::ISO5436\_-  
    2::ProbingSystemType, 432  
        \_clone, 438  
        Identification, 439, 440  
        Identification\_, 442  
        Identification\_traits, 436  
        Identification\_type, 436  
        parse, 440, 441  
        ProbingSystemType, 437, 438  
        Type, 441, 442  
        Type\_, 442  
        Type\_traits, 436  
        Type\_type, 436  
OpenGPS::Schemas::ISO5436\_-  
    2::Record1Type, 443  
        \_clone, 450, 451  
        Axes, 451, 452  
        Axes\_, 456  
        Axes\_traits, 448  
        Axes\_type, 448  
        FeatureType, 453, 454  
        FeatureType\_, 456  
        FeatureType\_traits, 448  
        FeatureType\_type, 448  
        parse, 454  
        Record1Type, 449, 450  
        Revision, 455, 456  
        Revision\_, 456  
        Revision\_traits, 448  
        Revision\_type, 449  
OpenGPS::Schemas::ISO5436\_-  
    2::Record2Type, 457  
        \_clone, 470  
        CalibrationDate, 470-472  
        CalibrationDate\_, 482  
        CalibrationDate\_traits, 465  
        CalibrationDate\_type, 465, 466  
        Comment, 472-474  
        Comment\_, 482  
        Comment\_optional, 466  
        Comment\_traits, 466  
OpenGPS::Schemas::ISO5436\_-  
    2::Record3Type, 482  
        \_clone, 494  
        DataLink, 494-496  
        DataLink\_, 503  
        DataLink\_optional, 489  
        DataLink\_traits, 490  
        DataLink\_type, 490  
        DataList, 496-498  
        DataList\_, 503  
        DataList\_optional, 490  
        DataList\_traits, 490  
        DataList\_type, 490, 491  
        ListDimension, 499, 500  
        ListDimension\_, 503  
        ListDimension\_optional, 491  
        ListDimension\_traits, 491  
        ListDimension\_type, 491  
        MatrixDimension, 500-503  
        MatrixDimension\_, 503  
        MatrixDimension\_optional, 491,  
            492  
        MatrixDimension\_traits, 492  
        MatrixDimension\_type, 492  
        parse, 503  
        Record3Type, 492, 493  
OpenGPS::Schemas::ISO5436\_-  
    2::Record4Type, 504  
        \_clone, 508  
        ChecksumFile, 509, 510

ChecksumFile\_, 511  
ChecksumFile\_traits, 506  
ChecksumFile\_type, 506  
parse, 510  
Record4Type, 507, 508  
OpenGPS::Schemas::ISO5436\_-  
    2::RotationMatrixElementType,  
        511  
    \_clone, 515  
    RotationMatrixElementType,  
        512–514  
OpenGPS::Schemas::ISO5436\_-  
    2::RotationType, 516  
    \_clone, 532, 533  
    parse, 533  
    r11, 533–535  
    r11\_, 548  
    r11\_traits, 527  
    r11\_type, 527  
    r12, 535, 536  
    r12\_, 548  
    r12\_traits, 527  
    r12\_type, 527, 528  
    r13, 536–538  
    r13\_, 548  
    r13\_traits, 528  
    r13\_type, 528  
    r21, 538, 539  
    r21\_, 548  
    r21\_traits, 528  
    r21\_type, 528  
    r22, 540, 541  
    r22\_, 548  
    r22\_traits, 529  
    r22\_type, 529  
    r23, 541–543  
    r23\_, 548  
    r23\_traits, 529  
    r23\_type, 529  
    r31, 543, 544  
    r31\_, 548  
    r31\_traits, 529, 530  
    r31\_type, 530  
    r32, 544–546  
    r32\_, 548  
    r32\_traits, 530  
    r32\_type, 530  
    r33, 546, 547  
    r33\_, 548  
    r33\_traits, 530  
    r33\_type, 531  
    RotationType, 531, 532  
    OpenGPS::Schemas::ISO5436\_-  
        2::Type, 553  
    \_clone, 558  
    \_xsd\_Type\_convert, 559  
    \_xsd\_Type\_indexes\_, 560  
    \_xsd\_Type\_literals\_, 560  
    operator value, 559  
    operator=, 559  
    Type, 555–558  
    value, 555  
OpenGPS::String, 549  
    ~String, 551  
    BaseType, 550  
    ConvertFromMd5, 551  
    ConvertToMd5, 551  
    CopyTo, 552  
    ElementType, 550  
    FromChar, 552  
    String, 551  
    ToChar, 552  
OpenGPS::ValidBuffer, 560  
    ~ValidBuffer, 562  
    Allocate, 562  
    AllocateRaw, 562  
    IsAllocated, 563  
    IsValid, 563  
    m\_RawSize, 564  
    m\_ValidityBuffer, 564  
    Read, 563  
    Reset, 563  
    SetValid, 563  
    ValidBuffer, 562  
    Write, 564  
OpenGPS::VectorBuffer, 564  
    ~VectorBuffer, 567  
    GetPointVectorProxy, 567  
    GetValidityBuffer, 567, 568  
    GetValidityProvider, 568  
    GetX, 568  
    GetY, 568, 569  
    GetZ, 569  
    HasValidityBuffer, 569  
    m\_ValidBuffer, 570  
    m\_ValidityProvider, 570  
    m\_X, 570  
    m\_Y, 571  
    m\_Z, 571  
    SetValidityProvider, 569

SetX, 569  
SetY, 570  
SetZ, 570  
VectorBuffer, 567  
OpenGPS::VectorBufferBuilder, 571  
  ~VectorBufferBuilder, 572  
  BuildBuffer, 573  
  BuildValidityProvider, 573  
  BuildX, 573  
  BuildY, 573  
  BuildZ, 574  
  CreatePointBuffer, 574  
  GetBuffer, 574  
  m\_Buffer, 575  
  VectorBufferBuilder, 572  
OpenGPS::XmlPointVectorReaderContext, 575  
  ~XmlPointVectorReaderContext, 578  
  IsGood, 578  
  IsValid, 578  
  m\_Next, 581  
  m\_PointVectorList, 581  
  m\_Stream, 581  
  MoveNext, 579  
  Read, 579, 580  
  Reset, 580  
  Set, 581  
  Skip, 581  
  StringList, 578  
  XmlPointVectorReaderContext, 578  
OpenGPS::XmlPointVectorWriterContext, 582  
  ~XmlPointVectorWriterContext, 584  
  AppendSeparator, 585  
  Get, 585  
  IsGood, 585  
  m\_NeedsSeparator, 587  
  m\_PointVectorList, 587  
  m\_Stream, 587  
  MoveNext, 585  
  Reset, 585  
  Skip, 585  
  StringList, 584  
  Write, 586, 587  
  XmlPointVectorWriterContext, 584  
OpenGPS::ZipOutputStream, 588  
  ~ZipOutputStream, 589  
  BaseType, 589  
  write, 589  
  ZipOutputStream, 589  
OpenGPS::ZipStreamBuffer, 590  
  ~ZipStreamBuffer, 591  
  BaseType, 591  
  GetMd5, 591  
  m\_Handle, 592  
  m\_Md5Context, 592  
  xputn, 591  
  ZipStreamBuffer, 591  
operator value  
  OpenGPS::Schemas::ISO5436\_-  
    2::AxisType, 114  
  OpenGPS::Schemas::ISO5436\_-  
    2::DataType, 188, 189  
  OpenGPS::Schemas::ISO5436\_-  
    2::Type, 559  
operator!=  
  OpenGPS::Schemas::ISO5436\_2,  
    48, 49  
operator<<  
  OpenGPS::Schemas::ISO5436\_2,  
    49–54  
operator=  
  OpenGPS::DataPoint, 169  
  OpenGPS::ISO5436\_2, 277  
  OpenGPS::ISO5436\_-  
    2Container::PointIteratorImpl,  
      320  
  OpenGPS::PointVector, 384  
  OpenGPS::PointVectorProxy, 406  
  OpenGPS::PointVectorProxy::DataPointProxy,  
    412  
  OpenGPS::Schemas::ISO5436\_-  
    2::AxisType, 114  
  OpenGPS::Schemas::ISO5436\_-  
    2::DataType, 189  
  OpenGPS::Schemas::ISO5436\_-  
    2::Type, 559  
operator==  
  OpenGPS::Schemas::ISO5436\_2,  
    54, 55  
OutputBinaryFileStream  
  OpenGPS::OutputBinaryFileStream,  
    361  
P  
  md5.c, 711

parse  
  OpenGPS::Schemas::ISO5436\_-  
    2::AxesType, 85  
  OpenGPS::Schemas::ISO5436\_-  
    2::AxisDescriptionType,  
      107  
  OpenGPS::Schemas::ISO5436\_-  
    2::DataLinkType, 151  
  OpenGPS::Schemas::ISO5436\_-  
    2::DataListType, 163  
  OpenGPS::Schemas::ISO5436\_-  
    2::InstrumentType, 249  
  OpenGPS::Schemas::ISO5436\_-  
    2::ISO5436\_2Type, 334  
  OpenGPS::Schemas::ISO5436\_-  
    2::MatrixDimensionType,  
      352  
  OpenGPS::Schemas::ISO5436\_-  
    2::ProbingSystemType, 440,  
      441  
  OpenGPS::Schemas::ISO5436\_-  
    2::Record1Type, 454  
  OpenGPS::Schemas::ISO5436\_-  
    2::Record2Type, 480  
  OpenGPS::Schemas::ISO5436\_-  
    2::Record3Type, 503  
  OpenGPS::Schemas::ISO5436\_-  
    2::Record4Type, 510  
  OpenGPS::Schemas::ISO5436\_-  
    2::RotationType, 533

parsing  
  xml\_schema, 65

PathExists  
  OpenGPS::Environment, 212

performanceDouble  
  ISO5436\_2\_XML\_Demo.cxx,  
    681

performanceInt16  
  ISO5436\_2\_XML\_Demo.cxx,  
    682

point\_buffer.cxx, 731  
point\_buffer.hxx, 731  
point\_iterator.cxx, 732  
point\_iterator.h, 732  
  ogps\_FreePointIterator, 735  
  ogps\_GetCurrentPoint, 735  
  ogps\_GetListPosition, 735  
  ogps\_GetMatrixPosition, 736  
  ogps\_HasNextPoint, 736  
  ogps\_HasPrevPoint, 737

ogps\_MoveNextPoint, 737  
ogps\_MovePrevPoint, 737  
OGPS\_PointIteratorPtr, 734  
ogps\_ResetNextPointIterator,  
  738  
ogps\_ResetPrevPointIterator,  
  738  
ogps\_SetCurrentPoint, 738

point\_iterator.hxx, 739  
point\_iterator\_c.hxx, 740  
  ogps\_FreePointIterator, 741  
  ogps\_GetCurrentPoint, 742  
  ogps\_GetListPosition, 742  
  ogps\_GetMatrixPosition, 742  
  ogps\_HasNextPoint, 743  
  ogps\_HasPrevPoint, 743  
  ogps\_MoveNextPoint, 744  
  ogps\_MovePrevPoint, 744  
  ogps\_ResetNextPointIterator,  
    744  
  ogps\_ResetPrevPointIterator,  
    745  
  ogps\_SetCurrentPoint, 745

point\_iterator\_c.hxx, 745  
  OGPS\_PointIterator, 746  
  OGPS\_PointIteratorPtr, 746

point\_validity\_provider.cxx, 746  
point\_validity\_provider.hxx, 747  
point\_vector.cxx, 748  
point\_vector.h, 748  
  ogps\_CreatePointVector, 752  
  ogps\_FreePointVector, 752  
  ogps\_GetDoubleX, 753  
  ogps\_GetDoubleY, 753  
  ogps\_GetDoubleZ, 753  
  ogps\_GetFloatX, 754  
  ogps\_GetFloatY, 754  
  ogps\_GetFloatZ, 754  
  ogps.GetInt16X, 755  
  ogps.GetInt16Y, 755  
  ogps.GetInt16Z, 755  
  ogps.GetInt32X, 756  
  ogps.GetInt32Y, 756  
  ogps.GetInt32Z, 756  
  ogps\_GetPointTypeX, 757  
  ogps\_GetPointTypeY, 757  
  ogps\_GetPointTypeZ, 757  
  ogps\_GetX, 757  
  ogps\_GetXYZ, 757  
  ogps\_GetY, 758

ogps\_GetZ, 758  
ogps\_IsValidPoint, 758  
OGPS\_PointVectorPtr, 752  
ogps\_SetDoubleX, 758  
ogps\_SetDoubleY, 759  
ogps\_SetDoubleZ, 759  
ogps\_SetFloatX, 759  
ogps\_SetFloatY, 759  
ogps\_SetFloatZ, 759  
ogps\_SetInt16X, 760  
ogps\_SetInt16Y, 760  
ogps\_SetInt16Z, 760  
ogps\_SetInt32X, 760  
ogps\_SetInt32Y, 760  
ogps\_SetInt32Z, 761  
point\_vector.hxx, 761  
point\_vector\_base.hxx, 762  
point\_vector\_c.hxx, 763  
    ogps\_CreatePointVector, 767  
    ogps\_FreePointVector, 767  
    ogps\_GetDoubleX, 767  
    ogps\_GetDoubleY, 768  
    ogps\_GetDoubleZ, 768  
    ogps\_GetFloatX, 768  
    ogps\_GetFloatY, 769  
    ogps\_GetFloatZ, 769  
    ogps.GetInt16X, 769  
    ogps.GetInt16Y, 770  
    ogps.GetInt16Z, 770  
    ogps.GetInt32X, 770  
    ogps.GetInt32Y, 771  
    ogps.GetInt32Z, 771  
    ogps\_GetPointXYZ, 771  
    ogps\_GetPointTypeX, 771  
    ogps\_GetPointTypeY, 771  
    ogps\_GetPointTypeZ, 772  
    ogps\_GetX, 772  
    ogps\_GetXYZ, 772  
    ogps\_GetY, 772  
    ogps\_GetZ, 772  
    ogps\_IsValidPoint, 773  
    ogps\_SetDoubleX, 773  
    ogps\_SetDoubleY, 773  
    ogps\_SetDoubleZ, 773  
    ogps\_SetFloatX, 774  
    ogps\_SetFloatY, 774  
    ogps\_SetFloatZ, 774  
    ogps\_SetInt16X, 774  
    ogps\_SetInt16Y, 774  
    ogps\_SetInt16Z, 775  
    ogps\_SetInt32X, 775  
ogps\_SetInt32Y, 775  
ogps\_SetInt32Z, 775  
point\_vector\_c.hxx, 776  
    OGPS\_PointVector, 776  
    OGPS\_PointVectorPtr, 776  
point\_vector\_iostream.cxx, 777  
point\_vector\_iostream.hxx, 777  
point\_vector\_parser.cxx, 778  
point\_vector\_parser.hxx, 779  
point\_vector\_parser\_builder.cxx,  
    780  
point\_vector\_parser\_builder.hxx,  
    780  
point\_vector\_proxy.cxx, 781  
point\_vector\_proxy.hxx, 781  
point\_vector\_proxy\_context.cxx,  
    782  
point\_vector\_proxy\_context.hxx,  
    783  
point\_vector\_proxy\_context\_-  
    list.hxx, 784  
point\_vector\_proxy\_context\_-  
    list.hxx, 784  
point\_vector\_proxy\_context\_-  
    matrix.cxx, 785  
point\_vector\_proxy\_context\_-  
    matrix.hxx, 785  
point\_vector\_reader\_context.hxx,  
    786  
point\_vector\_writer\_context.hxx,  
    787  
PointBuffer  
    OpenGPS::PointBuffer, 364  
PointBufferAutoPtr  
    OpenGPS, 31  
PointDataLink  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataLinkType, 151–153  
PointDataLink\_-  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataLinkType, 156  
PointDataLink\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataLinkType, 144  
PointDataLink\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::DataLinkType, 144  
PointIterator  
    OpenGPS::PointIterator, 369  
PointIteratorAutoPtr

OpenGPS, 31  
PointIteratorImpl  
    OpenGPS::ISO5436\_-  
        2Container::PointIteratorImpl, PointVectorWriterContext  
            316, 317  
PointValidityProvider  
    OpenGPS::PointValidityProvider, 374  
PointVector  
    OpenGPS::PointVector, 379  
PointVectorAutoPtr  
    OpenGPS, 32  
PointVectorBase  
    OpenGPS::PointVectorBase, 388  
PointVectorInputStream  
    OpenGPS::PointVectorInputStream, 392  
PointVectorInvariantLocale  
    OpenGPS::PointVectorInvariantLocale, 393  
PointVectorOutputStream  
    OpenGPS::PointVectorOutputStream, 395  
PointVectorParser  
    OpenGPS::PointVectorParser, 397  
PointVectorParserBuilder  
    OpenGPS::PointVectorParserBuilder, 401  
PointVectorParserBuilderAutoPtr  
    OpenGPS, 32  
PointVectorProxy  
    OpenGPS::PointVectorProxy, 404, 405  
PointVectorProxyContext  
    OpenGPS::PointVectorProxyContext, 415  
PointVectorProxyContextAutoPtr  
    OpenGPS, 32  
PointVectorProxyContextList  
    OpenGPS::PointVectorProxyContextList, 417  
PointVectorProxyContextMatrix  
    OpenGPS::PointVectorProxyContextMatrix, 421  
PointVectorReaderContext  
    OpenGPS::PointVectorReaderContext, 424  
PointVectorReaderContextAutoPtr  
    iso5436\_2\_container.cxx, 675  
PointVectorWhitespaceFacet  
    OpenGPS::PointVectorWhitespaceFacet, 428  
    PointVectorWriterContextAutoPtr  
        iso5436\_2\_container.cxx, 675  
positive\_integer  
    xml\_schema, 65  
PrintCopyright  
    OpenGPS::Info, 233  
PrintLicense  
    OpenGPS::Info, 233  
PrintVersion  
    OpenGPS::Info, 233  
ProbingSystem  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record2Type, 480, 481  
    ProbingSystem\_  
        OpenGPS::Schemas::ISO5436\_-  
            2::Record2Type, 482  
    ProbingSystem\_traits  
        OpenGPS::Schemas::ISO5436\_-  
            2::Record2Type, 468  
    ProbingSystem\_type  
        OpenGPS::Schemas::ISO5436\_-  
            2::Record2Type, 468  
    ProbingSystemType  
        OpenGPS::Schemas::ISO5436\_-  
            2::ProbingSystemType, 437,  
            438  
properties  
    xml\_schema, 65  
PUT ULONG LE  
    md5.c, 711  
qname  
    xml\_schema, 65  
OpenGPS::Schemas::ISO5436\_-  
    2::RotationType, 533-535  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 548  
    OpenGPS::Schemas::ISO5436\_-  
        r11\_traits  
            OpenGPS::Schemas::ISO5436\_-  
                2::RotationType, 527  
        r11\_type

OpenGPS::Schemas::ISO5436\_-  
    2::RotationType, 527

r12  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 535, 536

r12\_-  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 548

r12\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 527

r12\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 527, 528

r13  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 536–538

r13\_-  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 548

r13\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 528

r13\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 528

r21  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 538, 539

r21\_-  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 548

r21\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 528

r21\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 528

r22  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 540, 541

r22\_-  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 548

r22\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 529

r22\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 529

r23  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 541–543

r23\_-  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 548

r23\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 529

r23\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 529

r31  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 543, 544

r31\_-  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 548

r31\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 529, 530

r31\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 530

r32  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 544–546

r32\_-  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 548

r32\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 530

r32\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 530

r33  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 546, 547

r33\_-  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 548

r33\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 530

r33\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 531

Read

OpenGPS::BinaryLSBPointVectorReaderContext  
    **117, 118**

OpenGPS::BinaryMSBPointVectorReaderContext  
    **123, 124**

OpenGPS::DataPointParser, **181**

OpenGPS::DoubleDataPointParser,  
    **196**

OpenGPS::FloatDataPointParser,  
    **226**

OpenGPS::Int16DataPointParser,  
    **254**

OpenGPS::Int32DataPointParser,  
    **260**

OpenGPS::MissingDataPointParser,  
    **358**

OpenGPS::PointVectorParser,  
    **398**

OpenGPS::PointVectorReaderContext,  
    **425, 426**

OpenGPS::ValidBuffer, **563**

OpenGPS::XmlPointVectorReaderContext  
    **579, 580**

ReadDocument  
    OpenGPS::ISO5436\_2Container,  
        **307**

ReadMd5FromFile  
    OpenGPS::ISO5436\_2Container,  
        **307**

readonlyExample  
    ISO5436\_2\_XML\_Demo.cxx,  
        **682**

readonlyExample2  
    ISO5436\_2\_XML\_Demo.cxx,  
        **682**

readonlyExample3  
    ISO5436\_2\_XML\_Demo.cxx,  
        **682**

readonlyExample4  
    ISO5436\_2\_XML\_Demo.cxx,  
        **682**

Read XmlDocument  
    OpenGPS::ISO5436\_2Container,  
        **307**

Record1  
    OpenGPS::Schemas::ISO5436\_-  
        **2::ISO5436\_2Type, 334, 335**

Record1\_  
    OpenGPS::Schemas::ISO5436\_-  
        **2::ISO5436\_2Type, 343**

Record1\_Context  
    OpenGPS::Schemas::ISO5436\_-  
        **2::ISO5436\_2Type, 329**

Record1\_type  
    Record1Type  
        OpenGPS::Schemas::ISO5436\_-  
            **2::Record1Type, 449, 450**

Record2  
    Record2\_2  
        OpenGPS::Schemas::ISO5436\_-  
            **2::ISO5436\_2Type, 336-338**

Record2\_  
    Record2\_optional  
        OpenGPS::Schemas::ISO5436\_-  
            **2::ISO5436\_2Type, 343**

Record2\_traits  
    Record2Type  
        OpenGPS::Schemas::ISO5436\_-  
            **2::Record2Type, 468, 469**

Record3  
    Record3\_2  
        OpenGPS::Schemas::ISO5436\_-  
            **2::ISO5436\_2Type, 338, 339**

Record3\_  
    Record3\_traits  
        OpenGPS::Schemas::ISO5436\_-  
            **2::ISO5436\_2Type, 344**

Record3\_type  
    Record3Type  
        OpenGPS::Schemas::ISO5436\_-  
            **2::Record3Type, 492, 493**

Record4  
    Record4\_2  
        OpenGPS::Schemas::ISO5436\_-  
            **2::ISO5436\_2Type, 339-341**

Record4\_  
    Record4\_traits  
        OpenGPS::Schemas::ISO5436\_-  
            **2::ISO5436\_2Type, 344**

Record4\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::ISO5436\_2Type, 331

Record4\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::ISO5436\_2Type, 331

Record4Type  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record4Type, 507, 508

RemoveDir  
    OpenGPS::Environment, 212

RemoveFile  
    OpenGPS::Environment, 212

RemoveTempDir  
    OpenGPS::ISO5436\_2Container,  
        307

RenameFile  
    OpenGPS::Environment, 212

Reset  
    OpenGPS::DataPoint, 169  
    OpenGPS::DataPointImpl, 176  
    OpenGPS::Environment, 213  
    OpenGPS::ExceptionHistory, 218  
    OpenGPS::ISO5436\_2Container,  
        307

OpenGPS::PointVectorProxy::DataPointProxy  
    412

OpenGPS::ValidBuffer, 563

OpenGPS::XmlPointVectorReaderContext  
    580

OpenGPS::XmlPointVectorWriterContext  
    585

ResetNext  
    OpenGPS::ISO5436\_-  
        2Container::PointIteratorImpl, S  
        320

    OpenGPS::PointIterator, 372

ResetPrev  
    OpenGPS::ISO5436\_-  
        2Container::PointIteratorImpl,  
        320

    OpenGPS::PointIterator, 372

ResetValidPointsLink  
    OpenGPS::ISO5436\_2Container,  
        307

ResetXmlPointList  
    OpenGPS::ISO5436\_2Container,  
        308

Revision  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record1Type, 455, 456

Revision\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record1Type, 456

Revision\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record1Type, 448

Revision\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::Record1Type, 449

Rotation  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxesType, 85-87

Rotation\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxesType, 88

Rotation\_optional  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxesType, 78

Rotation\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxesType, 78

Rotation\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::AxesType, 78

RotationMatrixElementType  
    OpenGPS::Schemas::ISO5436\_-  
        512-514

RotationType  
    OpenGPS::Schemas::ISO5436\_-  
        2::RotationType, 531, 532

md5.c, 711

SafeMultiplication  
    OpenGPS::ISO5436\_2Container,  
        308

SaveChecksumFile  
    OpenGPS::ISO5436\_2Container,  
        308

SavePointBuffer  
    OpenGPS::ISO5436\_2Container,  
        308

SaveValidPointsLink  
    OpenGPS::ISO5436\_2Container,  
        308

Save XmlDocument

OpenGPS::ISO5436\_2Container,  
309  
Serial  
  OpenGPS::Schemas::ISO5436\_-  
    2::InstrumentType, 249,  
    250  
  Serial\_-  
    OpenGPS::Schemas::ISO5436\_-  
      2::InstrumentType, 253  
  Serial\_traits  
    OpenGPS::Schemas::ISO5436\_-  
      2::InstrumentType, 242  
  Serial\_type  
    OpenGPS::Schemas::ISO5436\_-  
      2::InstrumentType, 243  
serialization  
  xml\_schema, 65  
Set  
  OpenGPS::DataPoint, 169, 170  
  OpenGPS::DataPointImpl, 176,  
    177  
  OpenGPS::DoublePointBuffer,  
    201  
  OpenGPS::FloatPointBuffer, 231  
  OpenGPS::Int16PointBuffer, 257  
  OpenGPS::Int32PointBuffer, 263  
  OpenGPS::PointBuffer, 366, 367  
  OpenGPS::PointVector, 384  
  OpenGPS::PointVectorBase, 390  
  OpenGPS::PointVectorProxy, 406  
  OpenGPS::PointVectorProxy::DataPoint  
    412, 413  
  OpenGPS::XmlPointVectorReaderConte  
    581  
SetCurrent  
  OpenGPS::ISO5436\_-  
    2Container::PointIteratorImpl,  
    320  
  OpenGPS::PointIterator, 372  
SetIndex  
  OpenGPS::PointVectorProxyContextList  
    418  
  OpenGPS::PointVectorProxyContextSMMatrix,  
    421  
SetLastException  
  OpenGPS::ExceptionHistory,  
    218, 219  
SetListPoint  
  OpenGPS::ISO5436\_2, 277  
OpenGPS::ISO5436\_2Container,  
309  
SetMatrixPoint  
  OpenGPS::ISO5436\_2, 277  
  OpenGPS::ISO5436\_2Container,  
    309  
SetValid  
  OpenGPS::DoubleInlineValidity,  
    198  
  OpenGPS::FloatInlineValidity,  
    228  
  OpenGPS::Int16ValidBuffer, 259  
  OpenGPS::Int32ValidBuffer, 265  
  OpenGPS::PointValidityProvider,  
    375  
  OpenGPS::ValidBuffer, 563  
SetValidityProvider  
  OpenGPS::VectorBuffer, 569  
SetX  
  OpenGPS::PointVector, 385  
  OpenGPS::PointVectorParser,  
    398  
  OpenGPS::VectorBuffer, 569  
SetY  
  OpenGPS::PointVector, 385, 386  
  OpenGPS::PointVectorParser,  
    398  
  OpenGPS::VectorBuffer, 570  
SetZ  
  OpenGPS::PointVector, 386, 387  
OpenGPS::PointVectorParser,  
398  
OpenGPS::VectorBuffer, 570  
severity  
  xml\_schema, 65  
short\_  
  xml\_schema, 65  
simple\_type  
  xml\_schema, 65  
simpleExample  
ISO5436\_2\_XML\_Demo.hxx,  
682  
SMMatrix,  
OpenGPS::Schemas::ISO5436\_-  
  2::MatrixDimensionType,  
    352, 353  
SizeX\_  
  OpenGPS::Schemas::ISO5436\_-  
    2::MatrixDimensionType,  
      356

SizeX\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::MatrixDimensionType,  
            348

SizeX\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::MatrixDimensionType,  
            348

SizeY  
    OpenGPS::Schemas::ISO5436\_-  
        2::MatrixDimensionType,  
            353, 354

SizeY\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::MatrixDimensionType,  
            356

SizeY\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::MatrixDimensionType,  
            349

SizeY\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::MatrixDimensionType,  
            349

SizeZ  
    OpenGPS::Schemas::ISO5436\_-  
        2::MatrixDimensionType,  
            354, 355

SizeZ\_  
    OpenGPS::Schemas::ISO5436\_-  
        2::MatrixDimensionType,  
            356

SizeZ\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::MatrixDimensionType,  
            349

SizeZ\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::MatrixDimensionType,  
            349

Skip  
    OpenGPS::BinaryPointVectorReaderContext,  
        131

    OpenGPS::BinaryPointVectorWriterContext,DATALINK\_PATH,  
        135

    OpenGPS::PointVectorReaderContext,  
        426

    OpenGPS::PointVectorWriterContext,  
        430

    OpenGPS::XmlPointVectorReaderContext,

            581

OpenGPS::XmlPointVectorWriterContext,  
    585

state  
    md5\_context, 357

std, 55

stdafx.hxx, 788

    \_ASSERT, 790

    OPENGPS\_BINFORMAT\_-  
        DOUBLE\_SIZE, 790

    OPENGPS\_BINFORMAT\_-  
        FLOAT\_SIZE, 790

    OPENGPS\_BINFORMAT\_-  
        INT16\_SIZE, 790

    OPENGPS\_BINFORMAT\_-  
        INT32\_SIZE, 790

    OPENGPS\_BUILD, 790

    OPENGPS\_DELETE, 790

    OPENGPS\_DELETE\_-  
        ARRAY, 790

    OPENGPS\_DESCRIPTION,  
        790

    OPENGPS\_ENV\_-  
        OPENGPS\_LOCATION,  
            790

    OPENGPS\_EXCEPTION\_-  
        MESG, 790

    OPENGPS\_FREE, 791

    OPENGPS\_ID, 791

    OPENGPS\_ISO5436\_-  
        LOCATION, 791

    OPENGPS\_MINVERSION,  
        791

    OPENGPS\_MSTR, 791

    OPENGPS\_NAME, 791

    OPENGPS\_REVISION, 791

    OPENGPS\_VERSION, 791

    OPENGPS\_-  
        VERSIONSTRING, 791

    OPENGPS\_-  
        VERSIONSTRING\_M,

            OPENGPS\_XSD\_ISO5436\_-  
                DATALINK\_PATH, 791

            OPENGPS\_XSD\_ISO5436\_-  
                LOCATION, 792

            OPENGPS\_XSD\_ISO5436\_-  
                MAIN\_CHECKSUM\_-  
                    PATH, 792

\_OPENGPS\_XSD\_ISO5436\_-  
  MAIN\_PATH, [792](#)  
\_OPENGPS\_XSD\_ISO5436\_-  
  NAMESPACE, [792](#)  
\_OPENGPS\_XSD\_ISO5436\_-  
  VALIDPOINTSLINK\_-  
  PATH, [792](#)  
  VERIFY, [792](#)

String  
  OpenGPS::String, [551](#)

string  
  xml\_schema, [65](#)

string.cxx, [792](#)

string.hxx, [793](#)

StringList  
  OpenGPS::ISO5436\_2Container, [289](#)  
  OpenGPS::XmlPointVectorReaderContext, [578](#)  
  OpenGPS::XmlPointVectorWriterContext, [584](#)

TestChecksums  
  OpenGPS::ISO5436\_2Container, [310](#)

time  
  xml\_schema, [65](#)

ToChar  
  OpenGPS::String, [552](#)

token  
  xml\_schema, [65](#)

total  
  md5\_context, [357](#)

tree\_node\_key  
  xml\_schema, [66](#)

TRUE  
  opengps.h, [729](#)

Type  
  OpenGPS::Schemas::ISO5436\_-  
    2::ProbingSystemType, [441](#), [442](#)  
  OpenGPS::Schemas::ISO5436\_-  
    2::Type, [555-558](#)

type  
  xml\_schema, [65](#)

Type\_-  
  OpenGPS::Schemas::ISO5436\_-  
    2::ProbingSystemType, [442](#)

Type\_traits

OpenGPS::Schemas::ISO5436\_-  
  2::ProbingSystemType, [436](#)

Type\_type  
  OpenGPS::Schemas::ISO5436\_-  
    2::ProbingSystemType, [436](#)

unexpected\_element  
  xml\_schema, [66](#)

unexpected\_enumerator  
  xml\_schema, [66](#)

unsigned\_byte  
  xml\_schema, [66](#)

unsigned\_int  
  xml\_schema, [66](#)

unsigned\_long  
  xml\_schema, [66](#)

unsigned\_short  
  xml\_schema, [66](#)

UnsignedByte  
  OpenGPS, [32](#)

UnsignedBytePtr  
  OpenGPS, [32](#)

uri  
  xml\_schema, [66](#)

valid\_buffer.cxx, [793](#)

valid\_buffer.hxx, [794](#)

ValidBuffer  
  OpenGPS::ValidBuffer, [562](#)

ValidPointsLink  
  OpenGPS::Schemas::ISO5436\_-  
    2::DataLinkType, [153-155](#)

ValidPointsLink\_-  
  OpenGPS::Schemas::ISO5436\_-  
    2::DataLinkType, [156](#)

ValidPointsLink\_optional  
  OpenGPS::Schemas::ISO5436\_-  
    2::DataLinkType, [144, 145](#)

ValidPointsLink\_traits  
  OpenGPS::Schemas::ISO5436\_-  
    2::DataLinkType, [145](#)

ValidPointsLink\_type  
  OpenGPS::Schemas::ISO5436\_-  
    2::DataLinkType, [145](#)

value  
  OpenGPS::Schemas::ISO5436\_-  
    2::AxisType, [109, 110](#)  
  OpenGPS::Schemas::ISO5436\_-  
    2::DataType, [184](#)

OpenGPS::Schemas::ISO5436\_-  
    2::Type, 555  
vector\_buffer.cxx, 795  
vector\_buffer.hxx, 796  
vector\_buffer\_builder.cxx, 796  
vector\_buffer\_builder.hxx, 797  
VectorBuffer  
    OpenGPS::VectorBuffer, 567  
VectorBufferAutoPtr  
    OpenGPS, 32  
VectorBufferBuilder  
    OpenGPS::VectorBufferBuilder,  
        572  
VectorBufferBuilderAutoPtr  
    OpenGPS, 32  
VendorSpecificID  
    OpenGPS::Schemas::ISO5436\_-  
        2::ISO5436\_2Type, 341–343  
VendorSpecificID\_-  
    OpenGPS::Schemas::ISO5436\_-  
        2::ISO5436\_2Type, 344  
VendorSpecificID\_optional  
    OpenGPS::Schemas::ISO5436\_-  
        2::ISO5436\_2Type, 331  
VendorSpecificID\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::ISO5436\_2Type, 331  
VendorSpecificID\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::ISO5436\_2Type, 331,  
            332  
VerifyChecksum  
    OpenGPS::ISO5436\_2Container,  
        310  
VerifyDataBinChecksum  
    OpenGPS::ISO5436\_2Container,  
        311  
VerifyMainChecksum  
    OpenGPS::ISO5436\_2Container,  
        311  
VerifyValidBinChecksum  
    OpenGPS::ISO5436\_2Container,  
        311  
Version  
    OpenGPS::Schemas::ISO5436\_-  
        2::InstrumentType, 251,  
            252  
Version\_-  
    OpenGPS::Schemas::ISO5436\_-  
        2::InstrumentType, 253  
Version\_traits  
    OpenGPS::Schemas::ISO5436\_-  
        2::InstrumentType, 243  
Version\_type  
    OpenGPS::Schemas::ISO5436\_-  
        2::InstrumentType, 243  
win32\_environment.cxx, 798  
win32\_environment.hxx, 798  
Write  
    OpenGPS::BinaryLSBPointVectorWriterContext,  
        120, 121  
    OpenGPS::BinaryMSBPointVectorWriterContext,  
        126, 127  
    OpenGPS::DataPointParser, 181  
    OpenGPS::DoubleDataPointParser,  
        196  
    OpenGPS::FloatDataPointParser,  
        226  
    OpenGPS::Int16DataPointParser,  
        254  
    OpenGPS::Int32DataPointParser,  
        261  
    OpenGPS::ISO5436\_2, 278  
    OpenGPS::ISO5436\_2Container,  
        311  
    OpenGPS::MissingDataPointParser,  
        359  
    OpenGPS::PointVectorParser,  
        399  
    OpenGPS::PointVectorWriterContext,  
        431, 432  
    OpenGPS::ValidBuffer, 564  
    OpenGPS::XmlPointVectorWriterContext,  
        586, 587  
write  
    OpenGPS::ZipOutputStream, 589  
WriteVendorSpecific  
    OpenGPS::ISO5436\_2Container,  
        311  
x  
    \_OGPS\_POINT\_VECTOR, 70  
xml\_point\_vector\_reader\_-  
    context.cxx, 798  
    \_CHECK\_ISGOOD\_AND\_-  
        THROW\_EXCEPTION,  
            799  
    \_CHECK\_STREAM\_AND\_-  
        THROW\_EXCEPTION,

799  
xml\_point\_vector\_reader -  
    context.hxx, 799  
xml\_point\_vector\_writer -  
    context.cxx, 800  
    \_CHECK\_ISGOOD\_AND -  
        THROW\_EXCEPTION,  
            801  
    \_CHECK\_STREAM\_AND -  
        THROW\_EXCEPTION,  
            801  
xml\_point\_vector\_writer -  
    context.hxx, 802  
xml\_schema, 60  
    base64\_binary, 62  
    boolean, 62  
    bounds, 62  
    buffer, 62  
    byte, 62  
    date, 62  
    date\_time, 62  
    day, 62  
    decimal, 62  
    diagnostics, 62  
    double\_, 62  
    duplicate\_id, 62  
    duration, 62  
    entities, 62  
    entity, 63  
    error, 63  
    error\_handler, 63  
    exception, 63  
    expected\_attribute, 63  
    expected\_element, 63  
    expected\_text\_content, 63  
    flags, 63  
    float\_, 63  
    hex\_binary, 63  
    id, 63  
    idref, 63  
    idrefs, 63  
    int\_, 63  
    integer, 63  
    language, 64  
    long\_, 64  
    month, 64  
    month\_day, 64  
    name, 64  
    namespace\_info, 64  
    namespace\_infomap, 64  
    ncname, 64  
    negative\_integer, 64  
    nmtoken, 64  
    nmtokens, 64  
    no\_namespace\_mapping, 64  
    no\_prefix\_mapping, 64  
    no\_type\_info, 64  
    non\_negative\_integer, 64  
    non\_positive\_integer, 65  
    normalized\_string, 65  
    not\_derived, 65  
    parsing, 65  
    positive\_integer, 65  
    properties, 65  
    qname, 65  
    serialization, 65  
    severity, 65  
    short\_, 65  
    simple\_type, 65  
    string, 65  
    time, 65  
    token, 65  
    tree\_node\_key, 66  
    type, 65  
    unexpected\_element, 66  
    unexpected\_enumerator, 66  
    unsigned\_byte, 66  
    unsigned\_int, 66  
    unsigned\_long, 66  
    unsigned\_short, 66  
    uri, 66  
    xsi\_already\_in\_use, 66  
    year, 66  
    year\_month, 66  
XmlPointVectorReaderContext  
    OpenGPS::XmlPointVectorReaderContext,  
        578  
XmlPointVectorWriterContext  
    OpenGPS::XmlPointVectorWriterContext,  
        584  
xsd\_Licence\_Header.c, 802  
xsi\_already\_in\_use  
    xml\_schema, 66  
xspputn  
    OpenGPS::ZipStreamBuffer, 591  
XYSSL\_MD5\_C  
    md5.c, 711  
y  
    \_OGPS\_POINT\_VECTOR, 70

year  
    xml\_schema, [66](#)  
year\_month  
    xml\_schema, [66](#)

z  
    \_OGPS\_POINT\_VECTOR, [70](#)  
zip\_stream\_buffer.cxx, [803](#)  
zip\_stream\_buffer.hxx, [803](#)  
ZipOutputStream  
    OpenGPS::ZipOutputStream, [589](#)  
ZipStreamBuffer  
    OpenGPS::ZipStreamBuffer, [591](#)