

XBinder

XML Schema Compiler
Version 1.2
C++ Runtime
Reference Manual

The software described in this document is furnished under a license agreement and may be used only in accordance with the terms of this agreement.

Copyright Notice

Copyright ©1997-2007 Objective Systems, Inc. All rights reserved.

This document may be distributed in any form, electronic or otherwise, provided that it is distributed in its entirety and that the copyright and this notice are included.

Author's Contact Information

Comments, suggestions, and inquiries regarding XBinder may be submitted via electronic mail to info@obj-sys.com.

Contents

1	XBinder Main Page	1
2	XBinder Module Index	2
2.1	XBinder Modules	2
3	XBinder Hierarchical Index	3
3.1	XBinder Class Hierarchy	3
4	XBinder Class Index	5
4.1	XBinder Class List	5
5	XBinder File Index	7
5.1	XBinder File List	7
6	XBinder Module Documentation	9
6.1	Generic Input Stream Classes	9
6.2	Message Buffer Classes	10
6.3	Generic Output Stream Classes	11
6.4	TCP/IP or UDP Socket Classes	12
6.5	ASN.1 Stream Classes	13
7	XBinder Class Documentation	14
7.1	OSAnyAttrClass Class Reference	14
7.2	OSAnyElementClass Class Reference	18
7.3	OSBufferedInputStream Class Reference	21
7.4	OSDynOctStrClass Class Reference	23
7.5	OSRTBaseType Class Reference	27
7.6	OSRTContext Class Reference	28
7.7	OSRTCtxtHolder Class Reference	34
7.8	OSRTCtxtHolderIF Class Reference	38
7.9	OSRTCtxtPtr Class Reference	41

7.10 OSRTDListBaseClass Class Reference	44
7.11 OSRTDListClass Class Reference	46
7.12 OSRTDListNodeBaseClass Class Reference	49
7.13 OSRTDListNodeClass Class Reference	50
7.14 OSRTFastString Class Reference	53
7.15 OSRTFileInputStream Class Reference	56
7.16 OSRTFileOutputStream Class Reference	58
7.17 OSRTInputStream Class Reference	61
7.18 OSRTInputStreamIF Class Reference	68
7.19 OSRTInputStreamPtr Class Reference	72
7.20 OSRTException Class Reference	73
7.21 OSRTMemBuf Class Reference	75
7.22 OSRTMemoryInputStream Class Reference	76
7.23 OSRTMemoryOutputStream Class Reference	78
7.24 OSRTMessageBuffer Class Reference	80
7.25 OSRTMessageBufferIF Class Reference	85
7.26 OSRTObjListClass Class Reference	89
7.27 OSRTObjListNodeClass Class Reference	92
7.28 OSRTOutputStream Class Reference	95
7.29 OSRTOutputStreamIF Class Reference	100
7.30 OSRTOutputStreamPtr Class Reference	102
7.31 OSRTSocket Class Reference	103
7.32 OSRTSocketInputStream Class Reference	111
7.33 OSRTSocketOutputStream Class Reference	114
7.34 OSRTStream Class Reference	117
7.35 OSRTStreamIF Class Reference	122
7.36 OSRTString Class Reference	124
7.37 OSRTStringIF Class Reference	127
7.38 OSRTUTF8String Class Reference	130
7.39 OSSAXException Class Reference	133
7.40 OSSStreamException Class Reference	136
7.41 OSXMLAnyHandler Class Reference	137
7.42 OSXMLBase Class Reference	139
7.43 OSXMLBasePtr Class Reference	140
7.44 OSXMLContentHandler Class Reference	141
7.45 OSXMLDefaultHandler Class Reference	143
7.46 OSXMLDefaultHandler::ErrorInfo Struct Reference	146

7.47	OSXMLDefaultHandlerIF Class Reference	147
7.48	OSXMLDefaultHandlerPtr Class Reference	148
7.49	OSXMLErrorHandler Class Reference	149
7.50	OSXMLErrorInfo Class Reference	151
7.51	OSXMLNamespaceClass Class Reference	152
7.52	OSXMLParserCtxt Class Reference	154
7.53	OSXMLParserCtxtIF Class Reference	155
7.54	OSXMLReaderClass Class Reference	156
7.55	OSXMLSimpleTypeHandler Class Reference	158
7.56	OSXMLSoapHandler Class Reference	160
7.57	OSXMLStringClass Class Reference	162
7.58	OSXMLStrListHandler Class Reference	169
7.59	OSXSDDateClass Class Reference	170
7.60	OSXSDDateTimeClass Class Reference	173
7.61	OSXSDGlobalElement Class Reference	179
7.62	OSXSDDTimeClass Class Reference	185
8	XBinder File Documentation	188
8.1	OSRTBaseType.h File Reference	188
8.2	OSRTContext.h File Reference	189
8.3	OSRTCtxtHolder.h File Reference	190
8.4	OSRTCtxtHolderIF.h File Reference	191
8.5	OSRTFastString.h File Reference	192
8.6	OSRTFileInputStream.h File Reference	193
8.7	OSRTFileOutputStream.h File Reference	194
8.8	OSRTInputStream.h File Reference	195
8.9	OSRTInputStreamIF.h File Reference	196
8.10	OSRTMemBuf.h File Reference	197
8.11	OSRTMemoryInputStream.h File Reference	198
8.12	OSRTMemoryOutputStream.h File Reference	199
8.13	OSRTMsgBuf.h File Reference	200
8.14	OSRTMsgBufIF.h File Reference	201
8.15	OSRTOutputStream.h File Reference	202
8.16	OSRTOutputStreamIF.h File Reference	203
8.17	OSRTSocket.h File Reference	204
8.18	OSRTSocketInputStream.h File Reference	205
8.19	OSRTSocketOutputStream.h File Reference	206

8.20	OSRTStream.h File Reference	207
8.21	OSRTStreamIF.h File Reference	208
8.22	OSRTString.h File Reference	209
8.23	OSRTStringIF.h File Reference	210
8.24	OSRTUTF8String.h File Reference	211
8.25	rtSaxCppAny.h File Reference	212
8.26	rtSaxCppSimpleType.h File Reference	213
8.27	rtSaxCppSoap.h File Reference	214
8.28	rtSaxCppStrList.h File Reference	215
8.29	rtxCppAnyAttr.h File Reference	216
8.30	rtxCppAnyElement.h File Reference	217
8.31	rtxCppBufferedInputStream.h File Reference	218
8.32	rtxCppDateTime.h File Reference	219
8.33	rtxCppDList.h File Reference	220
8.34	rtxCppDynOctStr.h File Reference	221
8.35	rtxCppException.h File Reference	222
8.36	rtxCppTypes.h File Reference	223
8.37	rtxCppXmlString.h File Reference	224
8.38	rtXmlCppEncFuncs.h File Reference	225
8.39	rtXmlCppMsgBuf.h File Reference	227
8.40	rtXmlCppNamespace.h File Reference	228
8.41	rtXmlCppXSDElement.h File Reference	229

Chapter 1

XBinder Main Page

C++ Common / XML Runtime Library Classes

The **OSRT C++ run-time classes** are wrapper classes that provide an object-oriented interface to the common C run-time library functions. The categories of classes provided are as follows:

- Context management classes manage the context structure (OSCTXT) used to keep track of the working variables required to encode or decode XML messages.
- Message buffer classes are used to manage message buffers for encoding or decoding XML messages.
- XSD type base classes are used as the base for compiler-generated C++ data structures.
- Stream classes are used to read and write messages to and from files, sockets, and memory buffers.

Chapter 2

XBinder Module Index

2.1 XBinder Modules

Here is a list of all modules:

Generic Input Stream Classes	9
Message Buffer Classes	10
Generic Output Stream Classes	11
TCP/IP or UDP Socket Classes	12
ASN.1 Stream Classes	13

Chapter 3

XBinder Hierarchical Index

3.1 XBinder Class Hierarchy

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

OSRTBaseType	27
OSAnyAttrClass	14
OSAnyElementClass	18
OSDynOctStrClass	23
OSRTDListBaseClass	44
OSRTDListClass	46
OSRTObjListClass	89
OSRTUTF8String	130
OSXMLStringClass	162
OSXSDDateTimeClass	173
OSXSDDateClass	170
OSXSDDTimeClass	185
OSRTContext	28
OSRTCtxtHolderIF	38
OSRTCtxtHolder	34
OSRTMessageBuffer	80
OSRTStream	117
OSRTInputStream	61
OSBufferedInputStream	21
OSRTFileInputStream	56
OSRTMemoryInputStream	76
OSRTSocketInputStream	111
OSRTOutputStream	95
OSRTFileOutputStream	58
OSRTMemoryOutputStream	78
OSRTSocketOutputStream	114
OSRTMessageBufferIF	85
OSRTMessageBuffer	80
OSRTStreamIF	122
OSRTInputStreamIF	68
OSRTInputStream	61
OSRTOutputStreamIF	100

OSRTOutputStream	95
OSRTStream	117
OSRTCtxtPtr	41
OSRTDListNodeBaseClass	49
OSRTDListNodeClass	50
OSRTObjListNodeClass	92
OSRTInputStreamPtr	72
OSRTException	73
OSSAXException	133
OSSStreamException	136
OSRTMemBuf	75
OSRTOutputStreamPtr	102
OSRTSocket	103
OSRTStringIF	127
OSRTFastString	53
OSRTString	124
OSXMLBase	139
OSXMLReaderClass	156
OSXMLBasePtr	140
OSXMLContentHandler	141
OSXMLDefaultHandlerIF	147
OSXMLDefaultHandler	143
OSXMLAnyHandler	137
OSXMLSimpleTypeHandler	158
OSXMLSoapHandler	160
OSXMLDefaultHandler::ErrorInfo	146
OSXMLDefaultHandlerPtr	148
OSXMLErrorHandler	149
OSXMLErrorInfo	151
OSXMLDefaultHandlerIF	147
OSXMLNamespaceClass	152
OSXMLParserCtxtIF	155
OSXMLParserCtxt	154
OSXMLStrListHandler	169
OSXSDGlobalElement	179

Chapter 4

XBinder Class Index

4.1 XBinder Class List

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

OSAnyAttrClass (Any attribute)	14
OSAnyElementClass (Any element)	18
OSBufferedInputStream (The buffered input stream class)	21
OSDynOctStrClass (Dynamic binary string)	23
OSRTBaseType (C++ structured type base class)	27
OSRTContext (Reference counted context class)	28
OSRTCtxtHolder (Abstract message buffer or stream interface class)	34
OSRTCtxtHolderIF (Abstract message buffer or stream interface class)	38
OSRTCtxtPtr (Context reference counted pointer class)	41
OSRTDListBaseClass (This class is a base class for C++ representations of a doubly-linked list classes)	44
OSRTDListClass (This class represents a doubly-linked list structure)	46
OSRTDListNodeBaseClass (This class is a base class for C++ representations of a node for the doubly-linked list structure)	49
OSRTDListNodeClass (This class represents a doubly-linked list node structure)	50
OSRTFastString (C++ fast string class definition)	53
OSRTFileInputStream (Generic file input stream)	56
OSRTFileOutputStream (Generic file output stream)	58
OSRTInputStream (This is the base class for input streams)	61
OSRTInputStreamIF	68
OSRTInputStreamPtr	72
OSRTLException (The base exception class for the C++ run-time)	73
OSRTMemBuf (Memory Buffer class)	75
OSRTMemoryInputStream (Generic memory input stream)	76
OSRTMemoryOutputStream (Generic memory output stream)	78
OSRTMessageBuffer (Abstract message buffer base class)	80
OSRTMessageBufferIF (Abstract message buffer or stream interface class)	85
OSRTObjListClass (This class represents a doubly-linked list structure for objects)	89
OSRTObjListNodeClass (This class represents a doubly-linked list node structure for OSRTBaseType instances)	92
OSRTOutputStream (The base class definition for operations with output streams)	95
OSRTOutputStreamIF	100
OSRTOutputStreamPtr	102
OSRTSocket (Wrapper class for TCP/IP or UDP sockets)	103

OSRTSocketInputStream (Generic socket input stream)	111
OSRTSocketOutputStream (Generic socket output stream)	114
OSRTStream (The default base class for using I/O streams)	117
OSRTStreamIF	122
OSRTString (C++ string class definition)	124
OSRTStringIF (C++ string class interface)	127
OSRTUTF8String (UTF-8 string)	130
OSSAXException	133
OSStreamException (Exception class for streams)	136
OSXMLAnyHandler	137
OSXMLBase	139
OSXMLBasePtr	140
OSXMLContentHandler (Receive notification of general document events)	141
OSXMLDefaultHandler (This class is derived from the SAX class DefaultHandler base class)	143
OSXMLDefaultHandler::ErrorInfo	146
OSXMLDefaultHandlerIF (This class is derived from the SAX class DefaultHandler base class)	147
OSXMLDefaultHandlerPtr	148
OSXMLErrorHandler	149
OSXMLErrorInfo	151
OSXMLNamespaceClass (This class is used to hold an XML namespace prefix to URI mapping)	152
OSXMLParserCtxt	154
OSXMLParserCtxtIF	155
OSXMLReaderClass	156
OSXMLSimpleTypeHandler	158
OSXMLSoapHandler	160
OSXMLStringClass (XML string)	162
OSXMLStrListHandler (OSXMLStrListHandler)	169
OSXSDDateClass	170
OSXSDDateTimeClass	173
OSXSDDGlobalElement (XSD global element base class)	179
OSXSDDTimeClass	185

Chapter 5

XBinder File Index

5.1 XBinder File List

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

OSRTBaseType.h (C++ run-time base class for structured type definitions)	188
OSRTContext.h (C++ run-time context class definition)	189
OSRTCtxtHolder.h (C++ run-time message buffer interface class definition)	190
OSRTCtxtHolderIF.h (C++ run-time message buffer interface class definition)	191
OSRTFastString.h (C++ fast string class definition)	192
OSRTFileInputStream.h (C++ base class definitions for operations with input file streams)	193
OSRTFileOutputStream.h (C++ base class definitions for operations with output file streams)	194
OSRTInputStream.h (C++ base class definitions for operations with input streams)	195
OSRTInputStreamIF.h (C++ interface class definitions for operations with input streams)	196
OSRTMemBuf.h	197
OSRTMemoryInputStream.h (C++ base class definitions for operations with input memory streams)	198
OSRTMemoryOutputStream.h (C++ base class definitions for operations with output memory streams)	199
OSRTMsgBuf.h (C++ run-time message buffer class definition)	200
OSRTMsgBufIF.h (C++ run-time message buffer interface class definition)	201
OSRTOutputStream.h (C++ base class definitions for operations with output streams)	202
OSRTOutputStreamIF.h (C++ interface class definitions for operations with output streams)	203
OSRTSocket.h (TCP/IP or UDP socket class definitions)	204
OSRTSocketInputStream.h (C++ base class definitions for operations with input socket streams)	205
OSRTSocketOutputStream.h (C++ base class definitions for operations with output socket streams)	206
OSRTStream.h (C++ base class definitions for operations with I/O streams)	207
OSRTStreamIF.h (C++ interface class definitions for operations with I/O streams)	208
OSRTString.h (C++ string class definition)	209
OSRTStringIF.h (C++ string class interface)	210
OSRTUTF8String.h (C++ UTF-8 string class definition)	211
rtSaxCppAny.h	212
rtSaxCppParser.h	??
rtSaxCppParserIF.h	??
rtSaxCppSimpleType.h	213
rtSaxCppSoap.h	214
rtSaxCppStrList.h	215
rtxCppAnyAttr.h (C++ any element class definition)	216
rtxCppAnyElement.h (C++ any element class definition)	217
rtxCppBufferedInputStream.h	218

rtxCppDateTime.h (C++ XML schema date/time definition)	219
rtxCppDList.h	220
rtxCppDynOctStr.h (C++ dynamic binary string class definition)	221
rtxCppException.h (C++ run-time deprecated definition)	222
rtxCppTypes.h (C++ common type and class definitions)	223
rtxCppXmlString.h (C++ XML string class definition)	224
rtXmlCppEncFuncs.h (XML low-level C++ encode functions)	225
rtXmlCppMsgBuf.h (This file is deprecated)	227
rtXmlCppNamespace.h (XML namespace handling structures and function definitions)	228
rtXmlCppXSDElement.h (C++ run-time XML schema global element class definition)	229

Chapter 6

XBinder Module Documentation

6.1 Generic Input Stream Classes

The C++ interface class definitions for operations with input streams.

Classes

- class [OSRTInputStream](#)
This is the base class for input streams.
- class [OSRTInputStreamIF](#)
- class [OSRTInputStreamPtr](#)

6.1.1 Detailed Description

The C++ interface class definitions for operations with input streams.

Classes that implement this interface are used to input data from the various stream types, not to decode ASN.1 messages.

6.2 Message Buffer Classes

These classes are used to manage message buffers.

Classes

- class [OSRTMessageBuffer](#)
Abstract message buffer base class.
- class [OSRTMessageBufferIF](#)
Abstract message buffer or stream interface class.

6.2.1 Detailed Description

These classes are used to manage message buffers.

During encoding, messages are constructed within these buffers. During decoding, the messages to be decoded are held in these buffers.

6.3 Generic Output Stream Classes

The interface class definition for operations with output streams.

Classes

- class [OSRTOutputStream](#)
The base class definition for operations with output streams.
- class [OSRTOutputStreamIF](#)
- class [OSRTOutputStreamPtr](#)

6.3.1 Detailed Description

The interface class definition for operations with output streams.

Classes that implement this interface are used for writing data to the various stream types, not to encode ASN.1 messages.

6.4 TCP/IP or UDP Socket Classes

These classes provide utility methods for doing socket I/O.

Classes

- class [OSRSocket](#)

Wrapper class for TCP/IP or UDP sockets.

6.4.1 Detailed Description

These classes provide utility methods for doing socket I/O.

6.5 ASN.1 Stream Classes

Classes that read or write ASN.1 messages to files, sockets, memory buffers, et c., are derived from this class.

Classes

- class [OSRTStream](#)
The default base class for using I/O streams.
- class [OSRTStreamIF](#)

6.5.1 Detailed Description

Classes that read or write ASN.1 messages to files, sockets, memory buffers, et c., are derived from this class.

Chapter 7

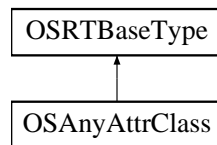
XBinder Class Documentation

7.1 OSAnyAttrClass Class Reference

Any attribute.

```
#include <rtxCppAnyAttr.h>
```

Inheritance diagram for OSAnyAttrClass::



Public Member Functions

- [OSAnyAttrClass \(\)](#)
The default constructor creates an empty attribute.
- [OSAnyAttrClass \(const OSUTF8CHAR *pname, const OSUTF8CHAR *pvalue\)](#)
This constructor initializes the attribute to contain the given data values.
- [OSAnyAttrClass \(const char *pname, const char *pvalue\)](#)
This constructor initializes the attribute to contain the given data values.
- [OSAnyAttrClass \(OSUTF8CHAR *pname, OSUTF8CHAR *pvalue\)](#)
This constructor initializes the attribute to contain the given data values.
- [OSAnyAttrClass \(OSAnyAttr &os\)](#)
This copy constructor initializes the attribute to contain the given data values from the C data structure.
- [OSAnyAttrClass \(const OSAnyAttrClass &os\)](#)
This copy constructor initializes the attribute to contain the given data values from the C++ data object.
- [virtual ~OSAnyAttrClass \(\)](#)

The destructor frees string memory.

- **OSRTBaseType * clone** () const
Clone method.
- void **copyValue** (const OSUTF8CHAR *pname, const OSUTF8CHAR *pvalue)
This method copies the given attribute value to the internal string storage variable.
- void **setValue** (const OSUTF8CHAR *pname, const OSUTF8CHAR *pvalue)
This method sets the attribute value to the given name/value.
- void **setValue** (const OSUTF8CHAR *pname, const OSUTF8CHAR *pvalue, size_t namebytes, size_t valuebytes=0)
This method sets the attribute value to the given name/value.
- **OSAnyAttrClass & operator=** (const **OSAnyAttrClass** &original)
Assignment operator.

7.1.1 Detailed Description

Any attribute.

This is the base class for generated C++ data type classes for any attribute declarations (xsd:anyAttr).

Definition at line 40 of file rtxCppAnyAttr.h.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 OSAnyAttrClass::OSAnyAttrClass (const OSUTF8CHAR * pname, const OSUTF8CHAR * pvalue)

This constructor initializes the attribute to contain the given data values.

Parameters:

pname - attribute name

pvalue - attribute contents

7.1.2.2 OSAnyAttrClass::OSAnyAttrClass (const char * pname, const char * pvalue)

This constructor initializes the attribute to contain the given data values.

This version allows the name/value arguments to be passed as standard C character string literal values.

Parameters:

pname - attribute name

pvalue - attribute contents

7.1.2.3 OSAnyAttrClass::OSAnyAttrClass (OSUTF8CHAR * *pname*, OSUTF8CHAR * *pvalue*)

This constructor initializes the attribute to contain the given data values.

Parameters:

pname - Attribute name.

pvalue - Attribute value.

7.1.2.4 OSAnyAttrClass::OSAnyAttrClass (OSAnyAttr & *os*)

This copy constructor initializes the attribute to contain the given data values from the C data structure.

It performs a deep copy.

Parameters:

os - C binary string structure.

7.1.2.5 OSAnyAttrClass::OSAnyAttrClass (const OSAnyAttrClass & *os*)

This copy constructor initializes the attribute to contain the given data values from the C++ data object.

It performs a deep copy.

Parameters:

os - C++ binary string object reference.

7.1.3 Member Function Documentation

7.1.3.1 OSRTBaseType* OSAnyAttrClass::clone () const [inline, virtual]

Clone method.

Creates a copied instance and returns pointer to OSRTBaseType.

Reimplemented from OSRTBaseType.

Definition at line 107 of file rtxCppAnyAttr.h.

7.1.3.2 void OSAnyAttrClass::copyValue (const OSUTF8CHAR * *pname*, const OSUTF8CHAR * *pvalue*)

This method copies the given attribute value to the internal string storage variable.

A deep-copy of the given value is done; the class will delete this memory when the object is deleted.

Parameters:

pname - Attribute name.

pvalue - Attribute value.

7.1.3.3 void OSAnyAttrClass::setValue (const OSUTF8CHAR * *pname*, const OSUTF8CHAR * *pvalue*)

This method sets the attribute value to the given name/value.

A deep-copy of the given value is not done; the pointer is stored directly in the class member variable.

Parameters:

pname - Attribute name.

pvalue - Attribute value.

7.1.3.4 void OSAnyAttrClass::setValue (const OSUTF8CHAR * *pname*, const OSUTF8CHAR * *pvalue*, size_t *namebytes*, size_t *valuebytes* = 0)

This method sets the attribute value to the given name/value.

A deep-copy of the given value is not done; the pointer is stored directly in the class member variable.

Parameters:

pname - Attribute name.

pvalue - Attribute value.

namebytes - Attribute name length.

valuebytes - Attribute value length.

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

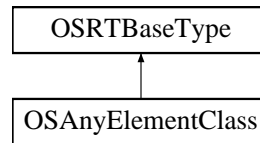
- [rtxCppAnyAttr.h](#)

7.2 OSAnyElementClass Class Reference

Any element.

```
#include <rtxCppAnyElement.h>
```

Inheritance diagram for OSAnyElementClass::



Public Member Functions

- [OSAnyElementClass \(\)](#)
The default constructor creates an empty element.
- [OSAnyElementClass \(const OSUTF8CHAR *pname, const OSUTF8CHAR *pvalue\)](#)
This constructor initializes the element to contain the given data values.
- [OSAnyElementClass \(const char *pname, const char *pvalue\)](#)
This constructor initializes the element to contain the given data values.
- [OSAnyElementClass \(OSAnyElement &os\)](#)
This copy constructor initializes the element to contain the given data values from the C data structure.
- [OSAnyElementClass \(const OSAnyElementClass &os\)](#)
This copy constructor initializes the element to contain the given data values from the C++ data object.
- [virtual ~OSAnyElementClass \(\)](#)
The destructor frees string memory.
- [void copyValue \(const OSUTF8CHAR *pname, const OSUTF8CHAR *pvalue\)](#)
This method copies the given element value to the internal string storage variable.
- [void print \(const char *pname\)](#)
This method prints the given element value to standard output.
- [void setValue \(const OSUTF8CHAR *pname, const OSUTF8CHAR *pvalue\)](#)
This method copies the given element value to the internal string storage variable.
- [OSRTBaseType * clone \(\) const](#)

7.2.1 Detailed Description

Any element.

This is the base class for generated C++ data type classes for any element declarations (xsd:any).

Definition at line 41 of file rtxCppAnyElement.h.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 `OSAnyElementClass::OSAnyElementClass (const OSUTF8CHAR * pname, const OSUTF8CHAR * pvalue)`

This constructor initializes the element to contain the given data values.

Parameters:

pname - element name

pvalue - element contents

7.2.2.2 `OSAnyElementClass::OSAnyElementClass (const char * pname, const char * pvalue)`

This constructor initializes the element to contain the given data values.

This version allows the name/value arguments to be passed as standard C character string literal values.

Parameters:

pname - element name

pvalue - element contents

7.2.2.3 `OSAnyElementClass::OSAnyElementClass (OSAnyElement & os)`

This copy constructor initializes the element to contain the given data values from the C data structure.

A deep copy is performed.

Parameters:

os - C binary string structure.

7.2.2.4 `OSAnyElementClass::OSAnyElementClass (const OSAnyElementClass & os)`

This copy constructor initializes the element to contain the given data values from the C++ data object.

A deep copy is performed.

Parameters:

os - C++ binary string object reference.

7.2.3 Member Function Documentation

7.2.3.1 `void OSAnyElementClass::copyValue (const OSUTF8CHAR * pname, const OSUTF8CHAR * pvalue)`

This method copies the given element value to the internal string storage variable.

A deep-copy of the given value is done; the class will delete this memory when the object is deleted.

Parameters:

pname - Element name.

pvalue - Element value.

7.2.3.2 void OSAnyElementClass::print (const char * *pname*) [inline]

This method prints the given element value to standard output.

Parameters:

pname - Name of generated string variable.

Definition at line 109 of file rtxCppAnyElement.h.

7.2.3.3 void OSAnyElementClass::setValue (const OSUTF8CHAR * *pname*, const OSUTF8CHAR * *pvalue*)

This method copies the given element value to the internal string storage variable.

A deep-copy of the given value is done; the class will delete this memory when the object is deleted.

Parameters:

pname - Element name.

pvalue - Element value.

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

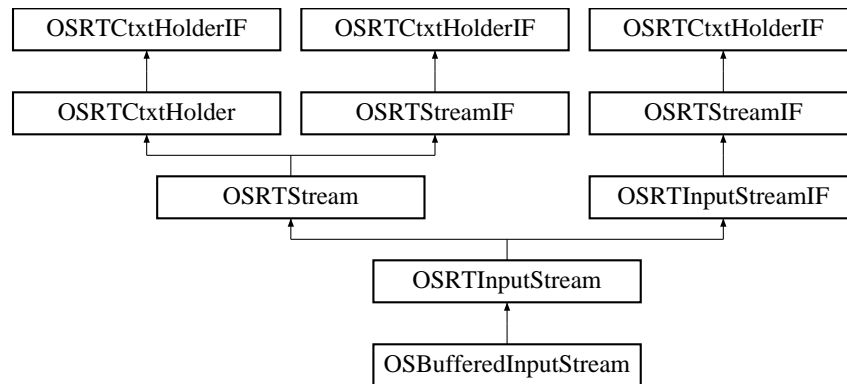
- [rtxCppAnyElement.h](#)

7.3 OSBufferedInputStream Class Reference

The buffered input stream class.

```
#include <rtxCppBufferedInputStream.h>
```

Inheritance diagram for OSBufferedInputStream::



Public Member Functions

- [OSBufferedInputStream \(OSRTInputStream &in\)](#)

The default constructor.

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

Virtual destructor.

7.3.1 Detailed Description

The buffered input stream class.

Definition at line 35 of file rtxCppBufferedInputStream.h.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 OSBufferedInputStream::OSBufferedInputStream (OSRTInputStream & in)

The default constructor.

It initializes a buffered stream. A buffered stream maintains data in memory before reading or writing to the device. This generally provides better performance than an unbuffered stream.

Exceptions:

[OSStreamException](#) Stream create or initialize failed.

7.3.2.2 virtual OSBufferedInputStream::~~OSBufferedInputStream () [virtual]

Virtual destructor.

Closes the stream if it was opened.

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

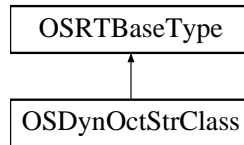
- [rtxCppBufferedInputStream.h](#)

7.4 OSDynOctStrClass Class Reference

Dynamic binary string.

```
#include <rtxCppDynOctStr.h>
```

Inheritance diagram for OSDynOctStrClass::



Public Member Functions

- [OSDynOctStrClass \(\)](#)
The default constructor creates an empty binary string.
- [OSDynOctStrClass \(OSUINT32 numocts_, const OSOCTET *data_\)](#)
This constructor initializes the binary string to contain the given data values.
- [OSDynOctStrClass \(OSDynOctStr &os\)](#)
The copy constructor initializes the binary string to contain the given data values from the C data structure.
- [OSDynOctStrClass \(const OSDynOctStrClass &os\)](#)
This copy constructor initializes the binary string to contain the given data values from the C++ data object.
- [virtual ~OSDynOctStrClass \(\)](#)
The destructor frees string memory.
- [OSRTBaseType * clone \(\) const](#)
Clone method.
- [void copyValue \(OSUINT32 numocts_, const OSOCTET *data_\)](#)
This method copies the given binary string value to the internal string storage variable.
- [const OSOCTET * getValue \(\) const](#)
This method returns a pointer to the binary data field.
- [size_t getLength \(\) const](#)
This method returns the length in octets of the binary data field.
- [size_t length \(\) const](#)
This method returns the length in octets of the binary data field.
- [void setValue \(OSUINT32 numocts_, const OSOCTET *data_\)](#)
This method copies the given binary string value to the internal string storage variable.
- [int setValue \(const char *hexstr, size_t nchars=0\)](#)

This method converts hex characters into binary form and sets the value.

- int `setValueFromBase64` (const char *base64str, size_t nchars=0)

This method converts base64-encoded characters into binary form and sets the value.

- `OSDynOctStrClass` & `operator=` (const `OSDynOctStrClass` &original)

Assignment operator.

Protected Attributes

- OSUINT32 `numocts`
- OSOCTET * `data`

7.4.1 Detailed Description

Dynamic binary string.

This is the base class for generated C++ data type classes for XSD binary types (hexBinary and base64Binary).

Definition at line 38 of file `rtxCppDynOctStr.h`.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 `OSDynOctStrClass::OSDynOctStrClass` (OSUINT32 `numocts_`, const OSOCTET * `data_`)

This constructor initializes the binary string to contain the given data values.

Parameters:

`numocts_` - Number of bytes in the binary string.

`data_` - The binary string data values.

7.4.2.2 `OSDynOctStrClass::OSDynOctStrClass` (`OSDynOctStr` & `os`)

The copy constructor initializes the binary string to contain the given data values from the C data structure.

Parameters:

`os` - C binary string structure.

7.4.2.3 `OSDynOctStrClass::OSDynOctStrClass` (const `OSDynOctStrClass` & `os`)

This copy constructor initializes the binary string to contain the given data values from the C++ data object.

Parameters:

`os` - C++ binary string object reference.

7.4.3 Member Function Documentation

7.4.3.1 `OSRTBaseType* OSDynOctStrClass::clone () const` [inline, virtual]

Clone method.

Creates a copied instance and returns pointer to `OSRTBaseType`.

Reimplemented from `OSRTBaseType`.

Definition at line 83 of file `rtxCppDynOctStr.h`.

7.4.3.2 `void OSDynOctStrClass::copyValue (OSUINT32 numocts_, const OSOCTET * data_)`

This method copies the given binary string value to the internal string storage variable.

A deep-copy of the given value is done; the class will delete this memory when the object is deleted.

Parameters:

numocts_ - Number of bytes in the binary string.

data_ - The binary string data values.

7.4.3.3 `void OSDynOctStrClass::setValue (OSUINT32 numocts_, const OSOCTET * data_)`

This method copies the given binary string value to the internal string storage variable.

A deep-copy of the given value is done; the class will delete this memory when the object is deleted.

Parameters:

numocts_ - Number of bytes in the binary string.

data_ - The binary string data values.

7.4.3.4 `int OSDynOctStrClass::setValue (const char * hexstr, size_t nchars = 0)`

This method converts hex characters into binary form and sets the value.

Parameters:

hexstr - Hex char string value.

nchars - Number of characters in string. If zero, characters are read up to null-terminator.

Returns:

- Status of operation: zero if success or a negative status code on error.

7.4.3.5 `int OSDynOctStrClass::setValueFromBase64 (const char * base64str, size_t nchars = 0)`

This method converts base64-encoded characters into binary form and sets the value.

Parameters:

base64str - Base64 char string value.

nchars - Number of characters in string. If zero, characters are read up to null-terminator.

Returns:

- Status of operation: zero if success or a negative status code on error.

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

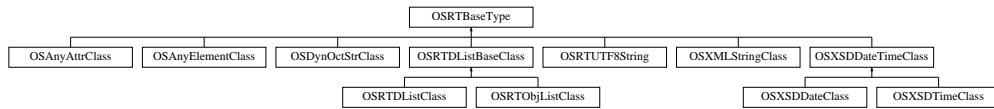
- [rtxCppDynOctStr.h](#)

7.5 OSRTBaseType Class Reference

C++ structured type base class.

```
#include <OSRTBaseType.h>
```

Inheritance diagram for OSRTBaseType::



Public Member Functions

- [OSRTBaseType \(\)](#)
- [virtual ~OSRTBaseType \(\)](#)
- [virtual OSRTBaseType * clone \(\) const](#)

7.5.1 Detailed Description

C++ structured type base class.

This is the base class for all generated structured types. It defines special new and delete operators to allow memory for the structure to be allocated using the built-in memory management algorithm.

Definition at line 39 of file OSRTBaseType.h.

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

- [OSRTBaseType.h](#)

7.6 OSRTContext Class Reference

Reference counted context class.

```
#include <OSRTContext.h>
```

Public Member Functions

- [OSRTContext](#) ()
The default constructor initializes the mCtx member variable and sets the reference count variable (mCount) to zero.
- virtual [~OSRTContext](#) ()
The destructor frees all memory held by the context.
- OSCTXT * [getPtr](#) ()
The getPtr method returns a pointer to the mCtx member variable.
- const OSCTXT * [getPtr](#) () const
- OSUINT32 [getRefCount](#) ()
The getRefCount method returns the current reference count.
- int [getStatus](#) () const
The getStatus method returns the runtime status code value.
- OSBOOL [isInitialized](#) ()
Returns TRUE, if initialized correctly, FALSE otherwise.
- void [_ref](#) ()
The _ref method increases the reference count by one.
- void [_unref](#) ()
The _unref method decreases the reference count by one.
- char * [getErrorInfo](#) ()
Returns error text in a dynamic memory buffer.
- char * [getErrorInfo](#) (size_t *pBufSize)
Returns error text in a dynamic memory buffer.
- char * [getErrorInfo](#) (char *pBuf, size_t &bufSize)
Returns error text in a memory buffer.
- void * [memAlloc](#) (size_t numocts)
The memAlloc method allocates memory using the C runtime memory management functions.
- void [memFreeAll](#) ()
The memFreeAll method will free all memory currently tracked within the context.
- void [memFreePtr](#) (void *ptr)
The memFreePtr method frees the memory at a specific location.

- void * [memRealloc](#) (void *ptr, size_t numocts)
The memRealloc method reallocates memory using the C runtime memory management functions.
- void [memReset](#) ()
The memReset method resets dynamic memory using the C runtime memory management functions.
- void [printErrorInfo](#) ()
The printErrorInfo method prints information on errors contained within the context.
- void [resetErrorInfo](#) ()
The resetErrorInfo method resets information on errors contained within the context.
- OSBOOL [setDiag](#) (OSBOOL value=TRUE)
The setDiag method will turn diagnostic tracing on or off.
- virtual int [setRunTimeKey](#) (const OSOCTET *key, size_t keylen)
This method sets run-time key to the context.
- int [setStatus](#) (int stat)
This method sets error status in the context.

Protected Attributes

- OSCTXT [mCtxt](#)
The mCtxt member variable is a standard C runtime context variable used in most C runtime function calls.
- OSUINT32 [mCount](#)
The mCount member variable holds the reference count of this context.
- OSBOOL [mbInitialized](#)
TRUE, if initialized correctly, FALSE otherwise.
- int [mStatus](#)
The mStatus variable holds the return status from C run-time function calls.

7.6.1 Detailed Description

Reference counted context class.

This keeps track of all encode/decode function variables between function invocations. It is reference counted to allow a message buffer and type class to share access to it.

Definition at line 65 of file OSRTContext.h.

7.6.2 Member Function Documentation

7.6.2.1 OSCTXT* OSRTContext::getPtr () [inline]

The getPtr method returns a pointer to the mCtxt member variable.

A user can use this function to get the the context pointer variable for use in a C runtime function call.

Definition at line 110 of file OSRTContext.h.

7.6.2.2 int OSRTContext::getStatus () const [inline]

The getStatus method returns the runtime status code value.

Returns:

Runtime status code:

- 0 (0) = success,
- negative return value is error.

Definition at line 125 of file OSRTContext.h.

7.6.2.3 OSBOOL OSRTContext::isInitialized () [inline]

Returns TRUE, if initialized correctly, FALSE otherwise.

Returns:

TRUE, if initialized correctly, FALSE otherwise.

Definition at line 133 of file OSRTContext.h.

7.6.2.4 char* OSRTContext::getErrorInfo ()

Returns error text in a dynamic memory buffer.

Buffer will be allocated using 'operator new []'. The calling routine is responsible for freeing the memory by using 'operator delete []'.

Returns:

A pointer to a newly allocated buffer with error text.

7.6.2.5 char* OSRTContext::getErrorInfo (size_t * pBufSize)

Returns error text in a dynamic memory buffer.

Buffer will be allocated using 'operator new []'. The calling routine is responsible for freeing the memory by using 'operator delete []'.

Parameters:

pBufSize A pointer to buffer size. It will receive the size of allocated dynamic buffer.

Returns:

A pointer to a newly allocated buffer with error text.

7.6.2.6 char* OSRTContext::getErrorInfo (char * pBuf, size_t & bufSize)

Returns error text in a memory buffer.

If buffer pointer is specified in parameters (not NULL) then error text will be copied in the passed buffer. Otherwise, this method allocates memory using the 'operator new []' function. The calling routine is responsible to free the memory by using 'operator delete []'.

Parameters:

pBuf A pointer to a destination buffer to obtain the error text. If NULL, dynamic buffer will be allocated.

bufSize A reference to buffer size. If pBuf is NULL it will receive the size of allocated dynamic buffer.

Returns:

A pointer to a buffer with error text. If pBuf is not NULL, the return pointer will be equal to it. Otherwise, returns newly allocated buffer with error text. NULL, if error occurred.

7.6.2.7 void* OSRTContext::memAlloc (size_t numocts) [inline]

The memAlloc method allocates memory using the C runtime memory management functions.

The memory is tracked in the underlying context structure. When both this [OSXSDGlobalElement](#) derived control class object and the message buffer object are destroyed, this memory will be freed.

Parameters:

numocts - Number of bytes of memory to allocate

Definition at line 192 of file OSRTContext.h.

7.6.2.8 void OSRTContext::memFreeAll () [inline]

The memFreeAll method will free all memory currently tracked within the context.

This includes all memory allocated with the memAlloc method as well as any memory allocated using the C `rtx-MemAlloc` function with the context returned by the `getCtxtPtr` method.

Definition at line 202 of file OSRTContext.h.

7.6.2.9 void OSRTContext::memFreePtr (void * ptr) [inline]

The memFreePtr method frees the memory at a specific location.

This memory must have been allocated using the memAlloc method described earlier.

Parameters:

ptr - Pointer to a block of memory allocated with `memAlloc`

Definition at line 214 of file OSRTContext.h.

7.6.2.10 `void* OSRTContext::memRealloc (void * ptr, size_t numocts)` [inline]

The memRealloc method reallocates memory using the C runtime memory management functions.

Parameters:

- ptr* - Original pointer containing dynamic memory to be resized.
- numocts* - Number of bytes of memory to allocate

Returns:

Reallocated memory pointer

Definition at line 227 of file OSRTContext.h.

7.6.2.11 `OSBOOL OSRTContext::setDiag (OSBOOL value = TRUE)` [inline]

The setDiag method will turn diagnostic tracing on or off.

Parameters:

- value* - Boolean value (default = TRUE = on)

Returns:

- Previous state of the diagnostics enabled boolean

Definition at line 261 of file OSRTContext.h.

7.6.2.12 `virtual int OSRTContext::setRunTimeKey (const OSOCTET * key, size_t keylen)` [virtual]

This method sets run-time key to the context.

This method does nothing for unlimited redistribution libraries.

Parameters:

- key* - array of octets with the key
- keylen* - number of octets in key array.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

7.6.2.13 `int OSRTContext::setStatus (int stat)` [inline]

This method sets error status in the context.

Parameters:

- stat* Status value.

Returns:

Error status value being set.

Definition at line 283 of file OSRTContext.h.

7.6.3 Member Data Documentation

7.6.3.1 `int OSRTContext::mStatus` [protected]

The mStatus variable holds the return status from C run-time function calls.

The getStatus method will either return this status or the last status on the context error list.

Definition at line 91 of file OSRTContext.h.

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

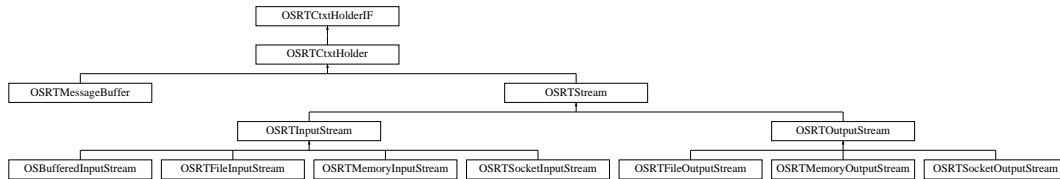
- [OSRTContext.h](#)

7.7 OSRTCtxtHolder Class Reference

Abstract message buffer or stream interface class.

```
#include <OSRTCtxtHolder.h>
```

Inheritance diagram for OSRTCtxtHolder::



Public Member Functions

- virtual [OSRTCtxtPtr](#) [getContext](#) ()
The `getContext` method returns the underlying context smart-pointer object.
- virtual `OSCTXT *` [getCtxtPtr](#) ()
The `getCtxtPtr` method returns the underlying C runtime context.
- virtual `char *` [getErrorInfo](#) ()
Returns error text in a dynamic memory buffer.
- virtual `char *` [getErrorInfo](#) (`char *pBuf`, `size_t &bufSize`)
Returns error text in a memory buffer.
- virtual `int` [getStatus](#) () `const`
This method returns the completion status of previous operation.
- virtual `void` [printErrorInfo](#) ()
The `printErrorInfo` method prints information on errors contained within the context.
- virtual `void` [resetErrorInfo](#) ()
The `resetErrorInfo` method resets information on errors contained within the context.

Protected Member Functions

- [OSRTCtxtHolder](#) (`OSRTContext *pContext=0`)
The protected constructor creates a new context and sets the buffer class type.
- virtual `~OSRTCtxtHolder` ()

Protected Attributes

- [OSRTCtxtPtr](#) `mpContext`
The `mpContext` member variable holds a reference-counted C runtime variable.

7.7.1 Detailed Description

Abstract message buffer or stream interface class.

This is the base class for both the in-memory message buffer classes and the run-time stream classes.

Definition at line 38 of file OSRTCtxtHolder.h.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 OSRTCtxtHolder::OSRTCtxtHolder (OSRTContext * pContext = 0) [protected]

The protected constructor creates a new context and sets the buffer class type.

Parameters:

pContext Pointer to a context to use. If NULL, new context will be allocated.

7.7.3 Member Function Documentation

7.7.3.1 virtual OSRTCtxtPtr OSRTCtxtHolder::getContext () [virtual]

The getContext method returns the underlying context smart-pointer object.

Returns:

Context smart pointer object.

Implements OSRTCtxtHolderIF.

Reimplemented in OSRTInputStream, OSRTMessageBuffer, OSRTOutputStream, and OSRTStream.

Referenced by OSRTStream::getContext(), and OSRTMessageBuffer::getContext().

7.7.3.2 virtual OSCTXT* OSRTCtxtHolder::getCtxtPtr () [virtual]

The getCtxtPtr method returns the underlying C runtime context.

This context can be used in calls to C runtime functions.

Returns:

The pointer to C runtime context.

Implements OSRTCtxtHolderIF.

Reimplemented in OSRTInputStream, OSRTMessageBuffer, OSRTOutputStream, and OSRTStream.

Referenced by OSRTMessageBuffer::getByteIndex(), OSRTStream::getCtxtPtr(), OSRTMessageBuffer::getCtxtPtr(), and OSRTMessageBuffer::getMsgPtr().

7.7.3.3 virtual char* OSRTCtxtHolder::getErrorInfo () [virtual]

Returns error text in a dynamic memory buffer.

Buffer will be allocated by 'operator new []'. The calling routine is responsible to free the memory by using 'operator delete []'.

Returns:

A pointer to a newly allocated buffer with error text.

Implements [OSRTCtxtHolderIF](#).

Reimplemented in [OSRTInputStream](#), [OSRTMessageBuffer](#), [OSRTOutputStream](#), and [OSRTStream](#).

Referenced by [OSRTStream::getErrorInfo\(\)](#), and [OSRTMessageBuffer::getErrorInfo\(\)](#).

7.7.3.4 virtual char* OSRTCtxtHolder::getErrorInfo (char * pBuf, size_t & bufSize) [virtual]

Returns error text in a memory buffer.

If buffer pointer is specified in parameters (not NULL) then error text will be copied in the passed buffer. Otherwise, this method allocates memory using the 'operator new []' function. The calling routine is responsible to free the memory by using 'operator delete []'.

Parameters:

pBuf A pointer to a destination buffer to obtain the error text. If NULL, dynamic buffer will be allocated.

bufSize A reference to buffer size. If pBuf is NULL it will receive the size of allocated dynamic buffer.

Returns:

A pointer to a buffer with error text. If pBuf is not NULL, the return pointer will be equal to it. Otherwise, returns newly allocated buffer with error text. NULL, if error occurred.

Implements [OSRTCtxtHolderIF](#).

Reimplemented in [OSRTInputStream](#), [OSRTMessageBuffer](#), [OSRTOutputStream](#), and [OSRTStream](#).

7.7.3.5 virtual int OSRTCtxtHolder::getStatus () const [virtual]

This method returns the completion status of previous operation.

It can be used to check completion status of constructors or methods, which do not return completion status. If error occurs, use [printErrorInfo](#) method to print out the error's description and stack trace. Method [resetError](#) can be used to reset error to continue operations after recovering from the error.

Returns:

Runtime status code:

- 0 (0) = success,
- negative return value is error.

Implements [OSRTCtxtHolderIF](#).

Reimplemented in [OSRTInputStream](#), [OSRTMessageBuffer](#), [OSRTOutputStream](#), and [OSRTStream](#).

Referenced by [OSRTMessageBuffer::getStatus\(\)](#).

7.7.4 Member Data Documentation**7.7.4.1 OSRTCtxtPtr OSRTCtxtHolder::mpContext [protected]**

The mpContext member variable holds a reference-counted C runtime variable.

This context is used in calls to all C run-time functions.

Definition at line 44 of file OSRTCtxtHolder.h.

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

- [OSRTCtxtHolder.h](#)

7.8.2 Constructor & Destructor Documentation

7.8.2.1 `virtual OSRTCtxtHolderIF::~~OSRTCtxtHolderIF ()` [inline, protected, virtual]

The virtual destructor does nothing.

It is overridden by derived versions of this class.

Definition at line 44 of file OSRTCtxtHolderIF.h.

7.8.3 Member Function Documentation

7.8.3.1 `virtual OSRTCtxtPtr OSRTCtxtHolderIF::getContext ()` [pure virtual]

The `getContext` method returns the underlying context smart-pointer object.

Returns:

Context smart pointer object.

Implemented in [OSRTCtxtHolder](#), [OSRTInputStream](#), [OSRTMessageBuffer](#), [OSRTOutputStream](#), and [OSRTStream](#).

Referenced by `OSXSDGlobalElement::OSXSDGlobalElement()`.

7.8.3.2 `virtual OSCTXT* OSRTCtxtHolderIF::getCtxtPtr ()` [pure virtual]

The `getCtxtPtr` method returns the underlying C runtime context.

This context can be used in calls to C runtime functions.

Returns:

The pointer to C runtime context.

Implemented in [OSRTCtxtHolder](#), [OSRTInputStream](#), [OSRTMessageBuffer](#), [OSRTOutputStream](#), and [OSRTStream](#).

7.8.3.3 `virtual char* OSRTCtxtHolderIF::getErrorInfo ()` [pure virtual]

Returns error text in a dynamic memory buffer.

Buffer will be allocated by 'operator new []'. The calling routine is responsible to free the memory by using 'operator delete []'.

Returns:

A pointer to a newly allocated buffer with error text.

Implemented in [OSRTCtxtHolder](#), [OSRTInputStream](#), [OSRTMessageBuffer](#), [OSRTOutputStream](#), and [OSRTStream](#).

7.8.3.4 `virtual char* OSRTCtxtHolderIF::getErrorInfo (char * pBuf, size_t & bufSize)` [pure virtual]

Returns error text in a memory buffer.

If buffer pointer is specified in parameters (not NULL) then error text will be copied in the passed buffer. Otherwise, this method allocates memory using the 'operator new []' function. The calling routine is responsible to free the memory by using 'operator delete []'.

Parameters:

pBuf A pointer to a destination buffer to obtain the error text. If NULL, dynamic buffer will be allocated.

bufSize A reference to buffer size. If pBuf is NULL it will receive the size of allocated dynamic buffer.

Returns:

A pointer to a buffer with error text. If pBuf is not NULL, the return pointer will be equal to it. Otherwise, returns newly allocated buffer with error text. NULL, if error occurred.

Implemented in [OSRTCtxtHolder](#), [OSRTInputStream](#), [OSRTMessageBuffer](#), [OSRTOutputStream](#), and [OSRTStream](#).

7.8.3.5 virtual int OSRTCtxtHolderIF::getStatus () const [pure virtual]

This method returns the completion status of previous operation.

It can be used to check completion status of constructors or methods, which do not return completion status. If error occurs, use `printErrorInfo` method to print out the error's description and stack trace. Method `resetError` can be used to reset error to continue operations after recovering from the error.

Returns:

Runtime status code:

- 0 (0) = success,
- negative return value is error.

Implemented in [OSRTCtxtHolder](#), [OSRTInputStream](#), [OSRTMessageBuffer](#), [OSRTOutputStream](#), and [OSRTStream](#).

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

- [OSRTCtxtHolderIF.h](#)

7.9 OSRTCtxtPtr Class Reference

Context reference counted pointer class.

```
#include <OSRTContext.h>
```

Public Member Functions

- **OSRTCtxtPtr** (**OSRTContext** *rf=0)
This constructor set the internal context pointer to the given value and, if it is non-zero, increases the reference count by one.
- **OSRTCtxtPtr** (const **OSRTCtxtPtr** &o)
The copy constructor copies the pointer from the source pointer object and, if it is non-zero, increases the reference count by one.
- virtual **~OSRTCtxtPtr** ()
The destructor decrements the reference counter to the internal context pointer object.
- **OSRTCtxtPtr** & **operator=** (const **OSRTCtxtPtr** &rf)
*This assignment operator assigns this **OSRTCtxtPtr** to another.*
- **OSRTCtxtPtr** & **operator=** (**OSRTContext** *rf)
*This assignment operator assigns does a direct assignment of an **OSRTContext** object to this **OSRTCtxtPtr** object.*
- **operator OSRTContext *** ()
The 'OSRTContext' operator returns the context object pointer.*
- **operator const OSRTContext *** () const
- **OSRTContext *** **operator ->** ()
The '->' operator returns the context object pointer.
- const **OSRTContext *** **operator ->** () const
- **OSBOOL** **operator==** (const **OSRTContext** *o) const
*The '==' operator compares two **OSRTContext** pointer values.*
- **OSBOOL** **isNull** () const
The isNull method returns TRUE if the underlying context pointer is NULL.
- **OSCTXT *** **getCtxtPtr** ()
This method returns the standard context pointer used in C function calls.

Protected Attributes

- **OSRTContext *** **mPointer**
The mPointer member variable is a pointer to a reference-counted ASN.1 context wrapper class object.

7.9.1 Detailed Description

Context reference counted pointer class.

This class allows a context object to automatically be released when its reference count goes to zero. It is very similar to the standard C++ library `auto_ptr` smart pointer class but only works with an `OSRTContext` object.

Definition at line 296 of file `OSRTContext.h`.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 `OSRTCtxPtr::OSRTCtxPtr (OSRTContext * rf = 0)` [inline]

This constructor set the internal context pointer to the given value and, if it is non-zero, increases the reference count by one.

Parameters:

rf - Pointer to `OSRTContext` object

Definition at line 311 of file `OSRTContext.h`.

7.9.2.2 `OSRTCtxPtr::OSRTCtxPtr (const OSRTCtxPtr & o)` [inline]

The copy constructor copies the pointer from the source pointer object and, if it is non-zero, increases the reference count by one.

Parameters:

o - Reference to `OSRTCtxPtr` object to be copied

Definition at line 321 of file `OSRTContext.h`.

7.9.2.3 `virtual OSRTCtxPtr::~OSRTCtxPtr ()` [inline, virtual]

The destructor decrements the reference counter to the internal context pointer object.

The context object will delete itself if its reference count goes to zero.

Definition at line 330 of file `OSRTContext.h`.

7.9.3 Member Function Documentation

7.9.3.1 `OSRTCtxPtr& OSRTCtxPtr::operator= (const OSRTCtxPtr & rf)` [inline]

This assignment operator assigns this `OSRTCtxPtr` to another.

The reference count of the context object managed by this object is first decremented. Then the new pointer is assigned and that object's reference count is incremented.

Parameters:

rf - Pointer to `OSRTCtxPtr` smart-pointer object

Definition at line 340 of file OSRTContext.h.

References OSRTContext::_ref(), and mPointer.

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

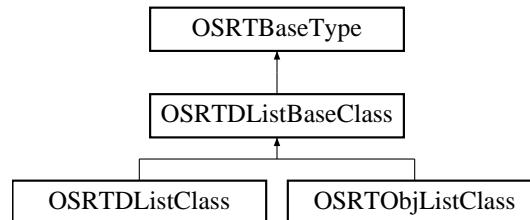
- [OSRTContext.h](#)

7.10 OSRTDListBaseClass Class Reference

This class is a base class for C++ representations of a doubly-linked list classes.

```
#include <rtxCppDList.h>
```

Inheritance diagram for OSRTDListBaseClass::



Public Member Functions

- [operator OSRTDList *](#) ()
- [operator const OSRTDList *](#) () const
- [OSRTDListBaseClass](#) ()
The default constructor initializes the list contents to empty.
- virtual [~OSRTDListBaseClass](#) ()
The destructor will delete all the nodes in the list.
- OSUINT32 [getCount](#) () const
This method returns count of items in the list.
- OSRTDList * [getList](#) ()
This method returns a pointer to OSRTDList structure for the list instance.
- const OSRTDList * [getList](#) () const
This method returns a const pointer to OSRTDList structure for the list instance.
- void [remove](#) (int index)
The remove method removes the data item at the given index from the list.

Protected Member Functions

- [OSRTDListBaseClass](#) (const [OSRTDListBaseClass](#) &o)

7.10.1 Detailed Description

This class is a base class for C++ representations of a doubly-linked list classes.

It is derived from the [OSRTBaseType](#) class as well as the C OSRTDList structure. This class provides a basic functionality for C++ doubly-linked list.

Definition at line 180 of file rtxCppDList.h.

7.10.2 Member Function Documentation

7.10.2.1 OSUINT32 OSRTDListBaseClass::getCount () const [inline]

This method returns count of items in the list.

Returns:

- Count of items in the list

Definition at line 203 of file rtxCppDList.h.

7.10.2.2 OSRTDList* OSRTDListBaseClass::getList () [inline]

This method returns a pointer to OSRTDList structure for the list instance.

Returns:

- a pointer to OSRTDList structure for the list instance.

Definition at line 212 of file rtxCppDList.h.

7.10.2.3 const OSRTDList* OSRTDListBaseClass::getList () const [inline]

This method returns a const pointer to OSRTDList structure for the list instance.

Returns:

- a const pointer to OSRTDList structure for the list instance.

Definition at line 221 of file rtxCppDList.h.

7.10.2.4 void OSRTDListBaseClass::remove (int *index*)

The remove method removes the data item at the given index from the list.

The index is zero-based.

Parameters:

- index* - Zero-based index of item to be removed.

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

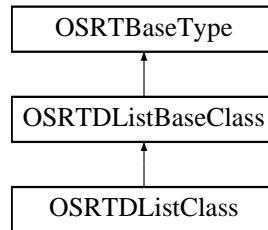
- [rtxCppDList.h](#)

7.11 OSRTDListClass Class Reference

This class represents a doubly-linked list structure.

```
#include <rtxCppDList.h>
```

Inheritance diagram for OSRTDListClass::



Public Member Functions

- [OSRTDListClass \(\)](#)
The default constructor initializes the list contents to empty.
- [OSRTDListClass \(const OSRTDListClass &o\)](#)
The copy constructor makes a copy of the list object.
- void [append](#) (void *pdata)
The append method adds an item to the end of the list.
- void [appendCopy](#) (void *pdata, size_t nbytes)
The appendCopy method adds a copy of an item to the end of the list.
- [OSRTDListNodeClass * getHead \(\)](#)
This method returns a pointer to a head node of the list.
- const [OSRTDListNodeClass * getHead \(\)](#) const
This method returns a pointer to a head node of the list.
- const void * [getItem](#) (int idx) const
The getItem method retrieves the data item from the list at the given index.
- [OSRTDListNodeClass * getTail \(\)](#)
This method returns a pointer to a tail node of the list.
- const [OSRTDListNodeClass * getTail \(\)](#) const
This method returns a pointer to a tail node of the list.
- void [insert](#) (int index, void *pdata)
The insert method inserts a data item into the list at the given indexed location.

7.11.1 Detailed Description

This class represents a doubly-linked list structure.

It extends the C++ `OSRTDListBaseClass` type. It provides methods for adding, retrieving, and removing items from linked lists. This list class is used to hold primitive types which are NOT derived from `OSRTBaseType`. See description of `OSRTObjListClass` for list of objects class.

Definition at line 241 of file `rtxCppDList.h`.

7.11.2 Member Function Documentation

7.11.2.1 `void OSRTDListClass::append (void * pdata)`

The `append` method adds an item to the end of the list.

Parameters:

pdata - Pointer to data item to be appended to list. Note the pointer itself is appended - a copy is not made.

7.11.2.2 `void OSRTDListClass::appendCopy (void * pdata, size_t nbytes)`

The `appendCopy` method adds a copy of an item to the end of the list.

Parameters:

pdata - Pointer to data item to be appended to list. Note that `clone()` is called on the data item, and the returned copy is stored in the list.

nbytes - Size of the data pointed to in bytes.

7.11.2.3 `OSRTDListNodeClass* OSRTDListClass::getHead () [inline]`

This method returns a pointer to a head node of the list.

Returns:

- Pointer to head node.

Definition at line 276 of file `rtxCppDList.h`.

7.11.2.4 `const OSRTDListNodeClass* OSRTDListClass::getHead () const [inline]`

This method returns a pointer to a head node of the list.

Returns:

- Pointer to head node.

Definition at line 285 of file `rtxCppDList.h`.

7.11.2.5 `const void* OSRTDListClass::getItem (int idx) const` [inline]

The `getItem` method retrieves the data item from the list at the given index.

The index is zero-based.

Parameters:

idx - Zero-based index of the node to retrieve.

Returns:

- Pointer to node structure containing the indexed data item.

Definition at line 296 of file `rtxCppDList.h`.

References `OSRTDListNodeClass::getData()`.

7.11.2.6 `OSRTDListNodeClass* OSRTDListClass::getTail ()` [inline]

This method returns a pointer to a tail node of the list.

Returns:

- Pointer to tail node.

Definition at line 307 of file `rtxCppDList.h`.

7.11.2.7 `const OSRTDListNodeClass* OSRTDListClass::getTail () const` [inline]

This method returns a pointer to a tail node of the list.

Returns:

- Pointer to tail node.

Definition at line 316 of file `rtxCppDList.h`.

7.11.2.8 `void OSRTDListClass::insert (int index, void * pdata)`

The `insert` method inserts a data item into the list at the given indexed location.

The index is zero-based.

Parameters:

index - Zero-based index of insertion point.

pdata - Pointer to data item to be inserted into list. Note the pointer itself is inserted - a copy is not made.

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

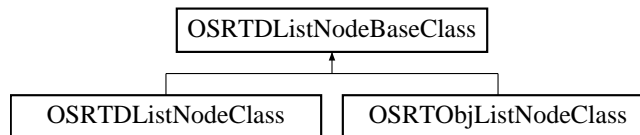
- [rtxCppDList.h](#)

7.12 OSRDLListNodeBaseClass Class Reference

This class is a base class for C++ representations of a node for the doubly-linked list structure.

```
#include <rtxCppDList.h>
```

Inheritance diagram for OSRDLListNodeBaseClass::



Public Member Functions

- virtual [~OSRDLListNodeBaseClass](#) ()

Protected Member Functions

- [operator OSRDLListNode *](#) ()
- [operator const OSRDLListNode *](#) () const

Friends

- class [OSRDLListBaseClass](#)
- class [OSRDLListClass](#)
- class [OSRTObjListClass](#)

7.12.1 Detailed Description

This class is a base class for C++ representations of a node for the doubly-linked list structure.

It extends the C OSRDLListNode type.

Definition at line 38 of file rtxCppDList.h.

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

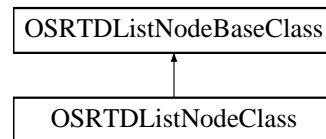
- [rtxCppDList.h](#)

7.13 OSRTDListNodeClass Class Reference

This class represents a doubly-linked list node structure.

```
#include <rtxCppDList.h>
```

Inheritance diagram for OSRTDListNodeClass::



Public Member Functions

- [OSRTDListNodeClass](#) (void *pdata)
 - void * [getData](#) ()
This method returns a pointer to a data associated with the node.
- const void * [getData](#) () const
This method returns a pointer to a data associated with the node.
- [OSRTDListNodeClass](#) * [getNext](#) ()
This method returns a pointer to a next node in the list.
- const [OSRTDListNodeClass](#) * [getNext](#) () const
This method returns a pointer to a next node in the list.
- [OSRTDListNodeClass](#) * [getPrev](#) ()
This method returns a pointer to a previous node in the list.
- const [OSRTDListNodeClass](#) * [getPrev](#) () const
This method returns a pointer to a previous node in the list.

7.13.1 Detailed Description

This class represents a doubly-linked list node structure.

It extends the C++ [OSRTDListNodeBaseClass](#) type.

Definition at line 56 of file [rtxCppDList.h](#).

7.13.2 Member Function Documentation

7.13.2.1 void* OSRTDListNodeClass::getData () [inline]

This method returns a pointer to a data associated with the node.

Returns:

Node data pointer.

Definition at line 67 of file rtxCppDList.h.

Referenced by OSRTDListClass::getItem().

7.13.2.2 const void* OSRTDListNodeClass::getData () const [inline]

This method returns a pointer to a data associated with the node.

Returns:

Node data pointer.

Definition at line 74 of file rtxCppDList.h.

7.13.2.3 OSRTDListNodeClass* OSRTDListNodeClass::getNext () [inline]

This method returns a pointer to a next node in the list.

Returns:

Pointer to the next node.

Definition at line 81 of file rtxCppDList.h.

7.13.2.4 const OSRTDListNodeClass* OSRTDListNodeClass::getNext () const [inline]

This method returns a pointer to a next node in the list.

Returns:

Pointer to the next node.

Definition at line 90 of file rtxCppDList.h.

7.13.2.5 OSRTDListNodeClass* OSRTDListNodeClass::getPrev () [inline]

This method returns a pointer to a previous node in the list.

Returns:

Pointer to the previous node.

Definition at line 99 of file rtxCppDList.h.

7.13.2.6 const OSRTDListNodeClass* OSRTDListNodeClass::getPrev () const [inline]

This method returns a pointer to a previous node in the list.

Returns:

Pointer to the previous node.

Definition at line 108 of file rtxCppDList.h.

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

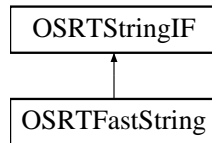
- [rtxCppDList.h](#)

7.14 OSRTFastString Class Reference

C++ fast string class definition.

```
#include <OSRTFastString.h>
```

Inheritance diagram for OSRTFastString::



Public Member Functions

- [OSRTFastString](#) ()
The default constructor sets the internal string member variable pointer to null.
- [OSRTFastString](#) (const char *strval)
This constructor initializes the string to contain the given standard ASCII string value.
- [OSRTFastString](#) (const OSUTF8CHAR *strval)
This constructor initializes the string to contain the given UTF-8 string value.
- [OSRTFastString](#) (const [OSRTFastString](#) &str)
Copy constructor.
- virtual [~OSRTFastString](#) ()
The destructor does nothing.
- virtual [OSRTStringIF](#) * [clone](#) ()
This method creates a copy of the given string object.
- virtual const char * [getValue](#) () const
This method returns the pointer to UTF-8 null terminated string as a standard ASCII string.
- virtual const OSUTF8CHAR * [getUTF8Value](#) () const
This method returns the pointer to UTF-8 null terminated string as a UTF-8 string.
- virtual void [print](#) (const char *name)
This method prints the string value to standard output.
- virtual void [setValue](#) (const char *str)
This method sets the string value to the given string.
- virtual void [setValue](#) (const OSUTF8CHAR *str)
This method sets the string value to the given UTF-8 string value.
- [OSRTFastString](#) & [operator=](#) (const [OSRTFastString](#) &original)
Assignment operator.

Protected Attributes

- `const OSUTF8CHAR * mValue`

7.14.1 Detailed Description

C++ fast string class definition.

This can be used to hold standard ASCII or UTF-8 strings. This string class implementations directly assigns any assigned pointers to internal member variables. It does no memory management.

Definition at line 43 of file OSRTFastString.h.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 OSRTFastString::OSRTFastString (const char * *strval*)

This constructor initializes the string to contain the given standard ASCII string value.

Parameters:

strval - Null-terminated C string value

7.14.2.2 OSRTFastString::OSRTFastString (const OSUTF8CHAR * *strval*)

This constructor initializes the string to contain the given UTF-8 string value.

Parameters:

strval - Null-terminated C string value

7.14.2.3 OSRTFastString::OSRTFastString (const OSRTFastString & *str*)

Copy constructor.

String data is not copied; the pointer is simply assigned to the target class member variable.

Parameters:

str - C++ string object to be copied.

7.14.3 Member Function Documentation

7.14.3.1 virtual void OSRTFastString::print (const char * *name*) [inline, virtual]

This method prints the string value to standard output.

Parameters:

name - Name of generated string variable.

Implements [OSRTStringIF](#).

Definition at line 109 of file OSRTFastString.h.

7.14.3.2 virtual void OSRTFastString::setValue (const char * *str*) [virtual]

This method sets the string value to the given string.

Parameters:

str - C null-terminated string.

Implements [OSRTStringIF](#).

7.14.3.3 virtual void OSRTFastString::setValue (const OSUTF8CHAR * *str*) [virtual]

This method sets the string value to the given UTF-8 string value.

Parameters:

str - C null-terminated UTF-8 string.

Implements [OSRTStringIF](#).

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

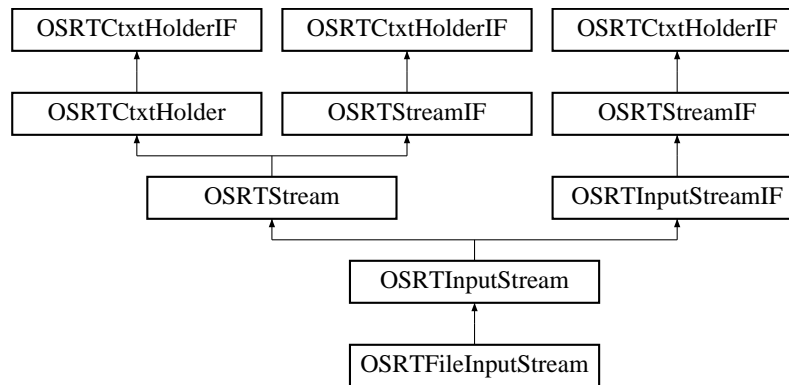
- [OSRTFastString.h](#)

7.15 OSRTFileInputStream Class Reference

Generic file input stream.

```
#include <OSRTFileInputStream.h>
```

Inheritance diagram for OSRTFileInputStream::



Public Member Functions

- [OSRTFileInputStream](#) (const char *pFilename)
Creates and initializes a file input stream using the name of file.
- [OSRTFileInputStream](#) (OSRTContext *pContext, const char *pFilename)
Creates and initializes a file input stream using the name of file.
- [OSRTFileInputStream](#) (FILE *file)
Initializes the file input stream using the opened FILE structure descriptor.
- [OSRTFileInputStream](#) (OSRTContext *pContext, FILE *file)
Initializes the file input stream using the opened FILE structure descriptor.

7.15.1 Detailed Description

Generic file input stream.

This class opens an existing file for input in binary mode and reads data from it.

Definition at line 37 of file OSRTFileInputStream.h.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 OSRTFileInputStream::OSRTFileInputStream (const char * pFilename)

Creates and initializes a file input stream using the name of file.

Parameters:

pFilename Name of file.

See also:

rtxStreamFileOpen

7.15.2.2 OSRTFileInputStream::OSRTFileInputStream ([OSRTContext](#) * *pContext*, const char * *pFilename*)

Creates and initializes a file input stream using the name of file.

Parameters:

pContext Pointer to a context to use.

pFilename Name of file.

See also:

rtxStreamFileOpen

7.15.2.3 OSRTFileInputStream::OSRTFileInputStream (FILE * *file*)

Initializes the file input stream using the opened FILE structure descriptor.

Parameters:

file Pointer to FILE structure.

See also:

rtxStreamFileAttach

7.15.2.4 OSRTFileInputStream::OSRTFileInputStream ([OSRTContext](#) * *pContext*, FILE * *file*)

Initializes the file input stream using the opened FILE structure descriptor.

Parameters:

pContext Pointer to a context to use.

file Pointer to FILE structure.

See also:

rtxStreamFileAttach

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

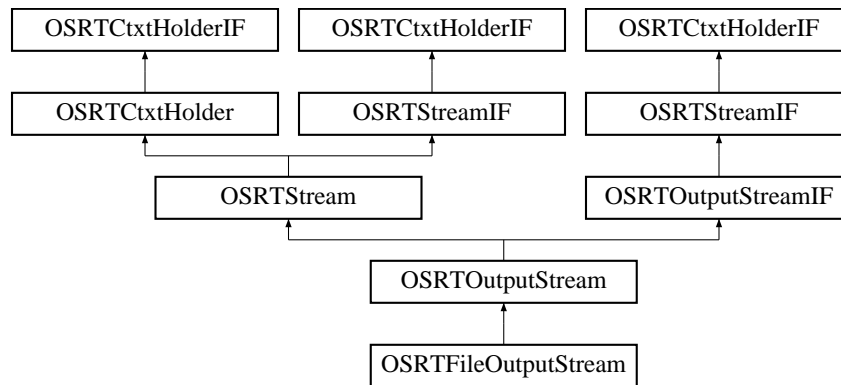
- [OSRTFileInputStream.h](#)

7.16 OSRTFileOutputStream Class Reference

Generic file output stream.

```
#include <OSRTFileOutputStream.h>
```

Inheritance diagram for OSRTFileOutputStream::



Public Member Functions

- [OSRTFileOutputStream](#) (const char *pFilename)
Creates and initializes a file output stream using the name of file.
- [OSRTFileOutputStream](#) (OSRTContext *pContext, const char *pFilename)
Creates and initializes a file output stream using the name of file.
- [OSRTFileOutputStream](#) (FILE *file)
Initializes the file output stream using the opened FILE structure descriptor.
- [OSRTFileOutputStream](#) (OSRTContext *pContext, FILE *file)
Initializes the file output stream using the opened FILE structure descriptor.

7.16.1 Detailed Description

Generic file output stream.

This class opens an existing file for output in binary mode and reads data from it.

Definition at line 37 of file OSRTFileOutputStream.h.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 OSRTFileOutputStream::OSRTFileOutputStream (const char * pFilename)

Creates and initializes a file output stream using the name of file.

Parameters:

pFilename Name of file.

Exceptions:

OSStreamException Stream create or initialize failed.

See also:

rtxStreamFileOpen

7.16.2.2 OSRTFileOutputStream::OSRTFileOutputStream (OSRTContext * pContext, const char * pFilename)

Creates and initializes a file output stream using the name of file.

Parameters:

pContext Pointer to a context to use.

pFilename Name of file.

Exceptions:

OSStreamException Stream create or initialize failed.

See also:

rtxStreamFileOpen

7.16.2.3 OSRTFileOutputStream::OSRTFileOutputStream (FILE * file)

Initializes the file output stream using the opened FILE structure descriptor.

Parameters:

file Pointer to FILE structure.

Exceptions:

OSStreamException Stream create or initialize failed.

See also:

rtxStreamFileAttach

7.16.2.4 OSRTFileOutputStream::OSRTFileOutputStream (OSRTContext * pContext, FILE * file)

Initializes the file output stream using the opened FILE structure descriptor.

Parameters:

pContext Pointer to a context to use.

file Pointer to FILE structure.

Exceptions:

OSStreamException Stream create or initialize failed.

See also:

`rtxStreamFileAttach`

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

- [OSRTFileOutputStream.h](#)

- virtual OSBOOL `isOpened ()`
Checks, is the stream opened or not.
- virtual OSBOOL `markSupported ()`
Tests if this input stream supports the mark and reset methods.
- virtual int `mark (size_t readAheadLimit)`
This method marks the current position in this input stream.
- void `printErrorInfo ()`
The printErrorInfo method prints information on errors contained within the context.
- void `resetErrorInfo ()`
The resetErrorInfo method resets information on errors contained within the context.
- virtual long `read (OSOCKET *pDestBuf, size_t maxToRead)`
Read data from the stream.
- virtual long `readBlocking (OSOCKET *pDestBuf, size_t toReadBytes)`
Read data from the stream.
- virtual int `reset ()`
Repositions this stream to the position at the time the mark method was last called on this input stream.
- virtual int `skip (size_t n)`
Skips over and discards the specified amount of data octets from this input stream.

7.17.1 Detailed Description

This is the base class for input streams.

These streams are buffered (I/O is stored in memory prior to being written) to provide higher performance.

Definition at line 41 of file OSRTInputStream.h.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 OSRTInputStream::OSRTInputStream ()

The default constructor.

It initializes a buffered stream. A buffered stream maintains data in memory before reading or writing to the device. This generally provides better performance than an unbuffered stream.

Exceptions:

OSRTStreamException Stream create or initialize failed.

7.17.2.2 virtual OSRTInputStream::~~OSRTInputStream () [virtual]

Virtual destructor.

Closes the stream if it was opened.

7.17.3 Member Function Documentation

7.17.3.1 virtual int OSRTInputStream::close () [virtual]

Closes the input or output stream and releases any system resources associated with the stream.

For output streams this function also flushes all internal buffers to the stream.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

`rtxStreamClose`, `rtxStreamBufClose`

Reimplemented from [OSRTStream](#).

7.17.3.2 virtual size_t OSRTInputStream::currentPos () [virtual]

This method returns the current position in the stream (in octets).

Returns:

The number of octets already read from the stream.

Implements [OSRTInputStreamIF](#).

7.17.3.3 virtual int OSRTInputStream::flush () [virtual]

Flushes the buffered data to the stream.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

`rtxStreamFlush`, `rtxStreamBufFlush`

Reimplemented from [OSRTStream](#).

7.17.3.4 **virtual OSRTCtxPtr OSRTInputStream::getContext ()** [inline, virtual]

This method returns a pointer to the underlying [OSRTContext](#) object.

Returns:

A reference-counted pointer to an [OSRTContext](#) object. The [OSRTContext](#) object will not be released until all referenced-counted pointer variables go out of scope. This allows safe sharing of the context between different run-time classes.

Reimplemented from [OSRTStream](#).

Definition at line 98 of file `OSRTInputStream.h`.

References `OSRTStream::getContext()`.

7.17.3.5 **virtual OSCTXT* OSRTInputStream::getCtxtPtr ()** [inline, virtual]

This method returns a pointer to the underlying `OSCTXT` object.

This is the structure used in calls to low-level C encode/decode functions.

Returns:

Pointer to a context (`OSCTXT`) structure.

Reimplemented from [OSRTStream](#).

Definition at line 108 of file `OSRTInputStream.h`.

References `OSRTStream::getCtxtPtr()`.

7.17.3.6 **virtual char* OSRTInputStream::getErrorInfo ()** [inline, virtual]

Returns error text in a dynamic memory buffer.

Buffer will be allocated by 'operator new []'. The calling routine is responsible to free the memory by using 'operator delete []'.

Returns:

A pointer to a newly allocated buffer with error text.

Reimplemented from [OSRTStream](#).

Definition at line 119 of file `OSRTInputStream.h`.

References `OSRTStream::getErrorInfo()`.

7.17.3.7 **virtual char* OSRTInputStream::getErrorInfo (char * pBuf, size_t & bufSize)** [inline, virtual]

Returns error text in a memory buffer.

If buffer pointer is specified in parameters (not NULL) then error text will be copied in the passed buffer. Otherwise, this method allocates memory using the 'operator new []' function. The calling routine is responsible to free the memory by using 'operator delete []'.

Parameters:

pBuf A pointer to a destination buffer to obtain the error text. If NULL, dynamic buffer will be allocated.

bufSize A reference to buffer size. If pBuf is NULL it will receive the size of allocated dynamic buffer.

Returns:

A pointer to a buffer with error text. If pBuf is not NULL, the return pointer will be equal to it. Otherwise, returns newly allocated buffer with error text. NULL, if error occurred.

Reimplemented from [OSRTStream](#).

Definition at line 139 of file OSRTInputStream.h.

References OSRTStream::getErrorInfo().

7.17.3.8 virtual int OSRTInputStream::getStatus () const [inline, virtual]

This method returns the completion status of previous operation.

It can be used to check completion status of constructors or methods, which do not return completion status.

Returns:

Runtime status code:

- 0 = success,
- negative return value is error.

Reimplemented from [OSRTStream](#).

Definition at line 152 of file OSRTInputStream.h.

References OSRTStream::getStatus().

7.17.3.9 virtual OSBOOL OSRTInputStream::isOpened () [virtual]

Checks, is the stream opened or not.

Returns:

s TRUE, if the stream is opened, FALSE otherwise.

See also:

rtxStreamIsOpened

Reimplemented from [OSRTStream](#).

7.17.3.10 virtual OSBOOL OSRTInputStream::markSupported () [virtual]

Tests if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance. By default, it returns FALSE.

Returns:

TRUE if this stream instance supports the mark and reset methods; FALSE otherwise.

See also:

rtxStreamIsMarkSupported

Implements [OSRTInputStreamIF](#).

7.17.3.11 virtual int OSRTInputStream::mark (size_t readAheadLimit) [virtual]

This method marks the current position in this input stream.

A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. The readAheadLimit argument tells this input stream to allow that many bytes to be read before the mark position gets invalidated.

Parameters:

readAheadLimit the maximum limit of bytes that can be read before the mark position becomes invalid.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

rtxStreamMark, rtxStreamReset

Implements [OSRTInputStreamIF](#).

7.17.3.12 virtual long OSRTInputStream::read (OSOCKET * pDestBuf, size_t maxToRead) [virtual]

Read data from the stream.

This method reads up to maxToRead bytes from the stream. It may return a value less than this if the maximum number of bytes is not available.

Parameters:

pDestBuf Pointer to a buffer to receive a data.

maxToRead Size of the buffer.

See also:

rtxStreamRead

Implements [OSRTInputStreamIF](#).

7.17.3.13 virtual long OSRTInputStream::readBlocking (OSOCKET * *pDestBuf*, size_t *toReadBytes*)
[virtual]

Read data from the stream.

This method reads up to `maxToRead` bytes from the stream. It may return a value less than this if the maximum number of bytes is not available.

Parameters:

- pDestBuf* Pointer to a buffer to receive a data.
- toReadBytes* Number of bytes to be read.

See also:

`rtxStreamRead`

Implements [OSRTInputStreamIF](#).

7.17.3.14 virtual int OSRTInputStream::reset () [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

Returns:

- Completion status of operation:
- 0 = success,
 - negative return value is error.

See also:

`rtxStreamMark`, `rtxStreamReset`

Implements [OSRTInputStreamIF](#).

7.17.3.15 virtual int OSRTInputStream::skip (size_t *n*) [virtual]

Skips over and discards the specified amount of data octets from this input stream.

Parameters:

- n* The number of octets to be skipped.

Returns:

- Completion status of operation:
- 0 = success,
 - negative return value is error.

See also:

`rtxStreamSkip`

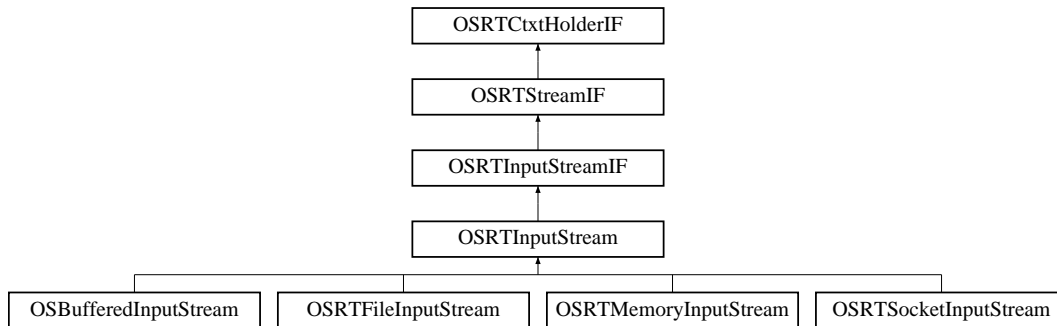
Implements [OSRTInputStreamIF](#).

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

- [OSRTInputStream.h](#)

7.18 OSRTInputStreamIF Class Reference

Inheritance diagram for OSRTInputStreamIF::



Public Member Functions

- virtual `~OSRTInputStreamIF ()`
Virtual destructor.
- virtual `size_t currentPos ()=0`
This method returns the current position in the stream (in octets).
- virtual `OSBOOL markSupported ()=0`
Tests if this input stream supports the mark and reset methods.
- virtual `int mark (size_t readAheadLimit)=0`
This method marks the current position in this input stream.
- virtual `long read (OSOCKET *pDestBuf, size_t maxToRead)=0`
Read data from the stream.
- virtual `long readBlocking (OSOCKET *pDestBuf, size_t toReadBytes)=0`
Read data from the stream.
- virtual `int reset ()=0`
Repositions this stream to the position at the time the mark method was last called on this input stream.
- virtual `int skip (size_t n)=0`
Skips over and discards the specified amount of data octets from this input stream.

7.18.1 Detailed Description

Definition at line 43 of file OSRTInputStreamIF.h.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 virtual OSRTInputStreamIF::~~OSRTInputStreamIF () [virtual]

Virtual destructor.

Closes the stream if it was opened.

7.18.3 Member Function Documentation

7.18.3.1 virtual size_t OSRTInputStreamIF::currentPos () [pure virtual]

This method returns the current position in the stream (in octets).

Returns:

The number of octets already read from the stream.

Implemented in [OSRTInputStream](#).

7.18.3.2 virtual OSBOOL OSRTInputStreamIF::markSupported () [pure virtual]

Tests if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance. By default, it returns FALSE.

Returns:

TRUE if this stream instance supports the mark and reset methods; FALSE otherwise.

See also:

`rtxStreamIsMarkSupported`

Implemented in [OSRTInputStream](#).

7.18.3.3 virtual int OSRTInputStreamIF::mark (size_t readAheadLimit) [pure virtual]

This method marks the current position in this input stream.

A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. The `readAheadLimit` argument tells this input stream to allow that many bytes to be read before the mark position gets invalidated.

Parameters:

readAheadLimit the maximum limit of bytes that can be read before the mark position becomes invalid.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

rtxStreamMark, rtxStreamReset

Implemented in [OSRTInputStream](#).

7.18.3.4 virtual long OSRTInputStreamIF::read (OSOCKET * *pDestBuf*, size_t *maxToRead*) [pure virtual]

Read data from the stream.

This method reads up to `maxToRead` bytes from the stream. It may return a value less than this if the maximum number of bytes is not available.

Parameters:

pDestBuf Pointer to a buffer to receive a data.

maxToRead Size of the buffer.

Returns:

The total number of octets read into the buffer, or negative value with error code if any error is occurred.

See also:

rtxStreamRead

Implemented in [OSRTInputStream](#).

7.18.3.5 virtual long OSRTInputStreamIF::readBlocking (OSOCKET * *pDestBuf*, size_t *toReadBytes*) [pure virtual]

Read data from the stream.

This method reads up to `maxToRead` bytes from the stream. It may return a value less than this if the maximum number of bytes is not available.

Parameters:

pDestBuf Pointer to a buffer to receive a data.

toReadBytes Number of bytes to be read.

Returns:

The total number of octets read into the buffer, or negative value with error code if any error is occurred.

See also:

rtxStreamRead

Implemented in [OSRTInputStream](#).

7.18.3.6 `virtual int OSRTInputStreamIF::reset ()` [pure virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

`rtxStreamMark`, `rtxStreamReset`

Implemented in [OSRTInputStream](#).

7.18.3.7 `virtual int OSRTInputStreamIF::skip (size_t n)` [pure virtual]

Skips over and discards the specified amount of data octets from this input stream.

Parameters:

n The number of octets to be skipped.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

`rtxStreamSkip`

Implemented in [OSRTInputStream](#).

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

- [OSRTInputStreamIF.h](#)

7.19 OSRTInputStreamPtr Class Reference

Public Member Functions

- [OSRTInputStreamPtr \(OSRTInputStreamIF *ptr\)](#)
- [~OSRTInputStreamPtr \(\)](#)
- [operator OSRTInputStreamIF * \(\)](#)
- [OSRTInputStreamIF * operator → \(\)](#)

7.19.1 Detailed Description

Definition at line 138 of file OSRTInputStreamIF.h.

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

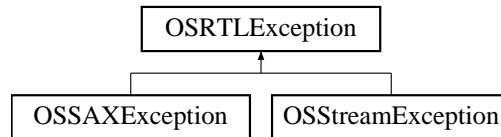
- [OSRTInputStreamIF.h](#)

7.20 OSRTLEException Class Reference

The base exception class for the C++ run-time.

```
#include <rtxCppException.h>
```

Inheritance diagram for OSRTLEException::



Public Member Functions

- [OSRTLEException](#) (int stat)
This constructor sets the status member variable value.
- [OSRTLEException](#) (OSRTContext *pContext, int stat)
This constructor sets the status member variable value.
- [OSRTLEException](#) (const [OSRTLEException](#) &o)
This is a copy constructor.
- [~OSRTLEException](#) ()
The virtual destructor does nothing.
- int [getStatus](#) () const
The getStatus method returns the runtime status code value.
- void [printErrorInfo](#) ()
Prints error information, if context is set.

Protected Member Functions

- [OSRTLEException](#) ()

Protected Attributes

- [OSRTCtxPtr mpContext](#)
- int [mStatus](#)
The mStatus member variable holds the status value which caused the exception to be thrown.

7.20.1 Detailed Description

The base exception class for the C++ run-time.

Definition at line 88 of file rtxCppException.h.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 OSRTLEException::OSRTLEException (int *stat*) [inline]

This constructor sets the status member variable value.

Parameters:

stat - The status value that caused the exception to be thrown.

Definition at line 107 of file rtxCppException.h.

7.20.2.2 OSRTLEException::OSRTLEException (OSRTContext * *pContext*, int *stat*) [inline]

This constructor sets the status member variable value.

Parameters:

pContext - The pointer to context to retrieve error information.

stat - The status value that caused the exception to be thrown.

Definition at line 116 of file rtxCppException.h.

7.20.2.3 OSRTLEException::OSRTLEException (const OSRTLEException & *o*) [inline]

This is a copy constructor.

Parameters:

o - Exception object to be copied.

Definition at line 124 of file rtxCppException.h.

7.20.2.4 OSRTLEException::~OSRTLEException () [inline]

The virtual destructor does nothing.

It is overridden by derived versions of this class.

Definition at line 131 of file rtxCppException.h.

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

- [rtxCppException.h](#)

7.21 OSRTMemBuf Class Reference

Memory Buffer class.

```
#include <OSRTMemBuf.h>
```

Public Member Functions

- const OSOCTET * [getData](#) ()
- size_t [getDataLen](#) ()

7.21.1 Detailed Description

Memory Buffer class.

This is the base class for generated C++ data type classes for XSD string types (string, token, NMTOKEN, etc.).

Definition at line 44 of file OSRTMemBuf.h.

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

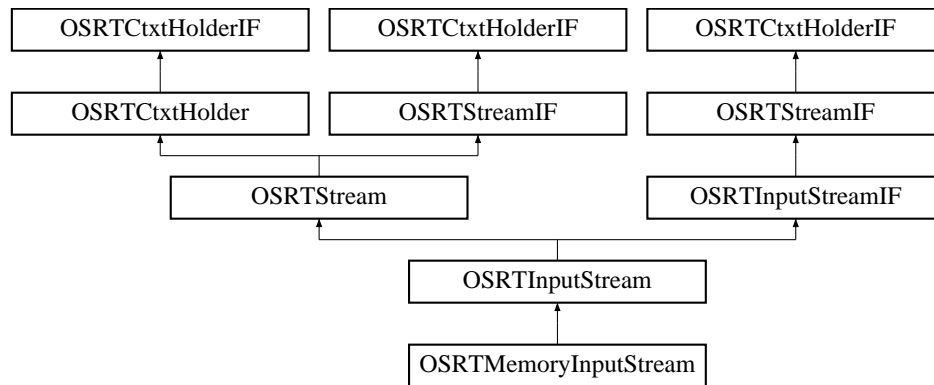
- [OSRTMemBuf.h](#)

7.22 OSRTMemoryInputStream Class Reference

Generic memory input stream.

```
#include <OSRTMemoryInputStream.h>
```

Inheritance diagram for OSRTMemoryInputStream::



Public Member Functions

- [OSRTMemoryInputStream](#) (const OSOCTET *pMemBuf, size_t bufSize)
Initializes the memory input stream using the specified memory buffer.
- [OSRTMemoryInputStream](#) (OSRTContext *pContext, const OSOCTET *pMemBuf, size_t bufSize)
Initializes the memory input stream using the specified memory buffer.

7.22.1 Detailed Description

Generic memory input stream.

This class opens an existing file for input in binary mode and reads data from it.

Definition at line 37 of file OSRTMemoryInputStream.h.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 OSRTMemoryInputStream::OSRTMemoryInputStream (const OSOCTET * pMemBuf, size_t bufSize)

Initializes the memory input stream using the specified memory buffer.

Parameters:

- pMemBuf* The pointer to the buffer.
- bufSize* The size of the buffer.

See also:

rtxStreamMemoryAttach

7.22.2.2 OSRTMemoryInputStream::OSRTMemoryInputStream ([OSRTContext](#) * *pContext*, const OSOCKET * *pMemBuf*, [size_t](#) *bufSize*)

Initializes the memory input stream using the specified memory buffer.

Parameters:

pContext Pointer to a context to use.

pMemBuf The pointer to the buffer.

bufSize The size of the buffer.

See also:

[rtxStreamMemoryAttach](#)

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

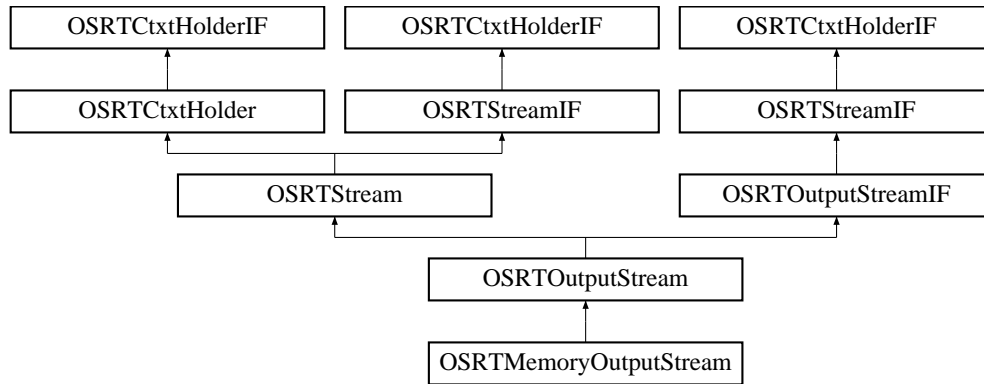
- [OSRTMemoryInputStream.h](#)

7.23 OSRTMemoryOutputStream Class Reference

Generic memory output stream.

```
#include <OSRTMemoryOutputStream.h>
```

Inheritance diagram for OSRTMemoryOutputStream::



Public Member Functions

- [OSRTMemoryOutputStream](#) (OSOCKET *pMemBuf, size_t bufSize)
Initializes the memory output stream using the specified memory buffer.
- [OSRTMemoryOutputStream](#) (OSRTContext *pContext, OSOCKET *pMemBuf, size_t bufSize)
Initializes the memory output stream using the specified memory buffer.

7.23.1 Detailed Description

Generic memory output stream.

This class opens an existing file for output in binary mode and reads data from it.

Definition at line 37 of file OSRTMemoryOutputStream.h.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 OSRTMemoryOutputStream::OSRTMemoryOutputStream (OSOCKET * pMemBuf, size_t bufSize)

Initializes the memory output stream using the specified memory buffer.

Parameters:

- pMemBuf* The pointer to the buffer.
bufSize The size of the buffer.

Exceptions:

- OSCStreamException* stream can't be created or initialized.

See also:

rtxStreamMemoryAttach

7.23.2.2 OSRTMemoryOutputStream::OSRTMemoryOutputStream (OSRTContext * pContext, OSOCTET * pMemBuf, size_t bufSize)

Initializes the memory output stream using the specified memory buffer.

Parameters:

pContext Pointer to a context to use.

pMemBuf The pointer to the buffer.

bufSize The size of the buffer.

Exceptions:

OSCStreamException stream can't be created or initialized.

See also:

rtxStreamMemoryAttach

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

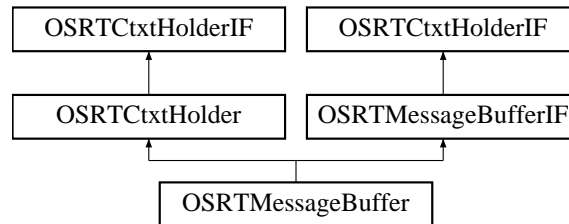
- [OSRTMemoryOutputStream.h](#)

7.24 OSRTMessageBuffer Class Reference

Abstract message buffer base class.

```
#include <OSRTMsgBuf.h>
```

Inheritance diagram for OSRTMessageBuffer::



Public Member Functions

- virtual `~OSRTMessageBuffer ()`
The virtual destructor does nothing.
- virtual `void * getAppInfo ()`
Returns a pointer to application-specific information block.
- virtual `size_t getByteIndex ()`
The `getByteIndex` method is used to fetch the current byte offset within the current working buffer.
- virtual `OSRTCtxtPtr getContext ()`
The `getContext` method returns the underlying context smart-pointer object.
- virtual `OSCTXT * getCtxtPtr ()`
The `getCtxtPtr` method returns the underlying C runtime context.
- virtual `char * getErrorInfo ()`
Returns error text in a dynamic memory buffer.
- virtual `char * getErrorInfo (char *pBuf, size_t &bufSize)`
Returns error text in a memory buffer.
- virtual `OSOCKET * getMsgCopy ()`
The `getMsgCopy` method will return a copy of the encoded message managed by the object.
- virtual `const OSOCKET * getMsgPtr ()`
The `getMsgPtr` method will return a const pointer to the encoded message managed by the object.
- `int getStatus () const`
This method returns the completion status of previous operation.
- virtual `int init ()`
Initializes message buffer.

- virtual int `initBuffer` (OSOCKET *pMsgBuf, size_t msgBufLen)
This version of the overloaded `initBuffer` method initializes the message buffer to point at the given null-terminated character string.
- virtual void `printErrorInfo` ()
The `printErrorInfo` method prints information on errors contained within the context.
- virtual void `resetErrorInfo` ()
The `resetErrorInfo` method resets information on errors contained within the context.
- virtual void `setAppInfo` (void *)
Sets the application-specific information block.
- virtual void `setDiag` (OSBOOL value=TRUE)
The `setDiag` method will turn diagnostic tracing on or off.

Protected Member Functions

- `OSRTMessageBuffer` (Type bufferType, OSRTContext *pContext=0)
The protected constructor creates a new context and sets the buffer class type.

Protected Attributes

- Type `mBufferType`
The `mBufferType` member variable holds information on the derived message buffer class type (for example, `XMLEncode`).

7.24.1 Detailed Description

Abstract message buffer base class.

This class is used to manage an encode or decode message buffer. For encoding, this is the buffer into which the message is being built. For decoding, it describes a message that was read into memory to be decoded. Further classes are derived from this to handle encoding and decoding of messages for different encoding rules types.

Definition at line 46 of file `OSRTMsgBuf.h`.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 `OSRTMessageBuffer::OSRTMessageBuffer` (Type bufferType, OSRTContext * pContext = 0) [protected]

The protected constructor creates a new context and sets the buffer class type.

Parameters:

- bufferType* Type of message buffer that is being created (for example, `XMLEncode`).
- pContext* Pointer to a context to use. If `NULL`, new context will be allocated.

7.24.2.2 **virtual OSRTMessageBuffer::~~OSRTMessageBuffer ()** [inline, virtual]

The virtual destructor does nothing.

It is overridden by derived versions of this class.

Definition at line 73 of file OSRTMsgBuf.h.

7.24.3 Member Function Documentation

7.24.3.1 **virtual size_t OSRTMessageBuffer::getByteIndex ()** [inline, virtual]

The getByteIndex method is used to fetch the current byte offset within the current working buffer.

For encoding, this is the next location that will be written to. For decoding, this is the next byte the parser will read.

Implements [OSRTMessageBufferIF](#).

Definition at line 86 of file OSRTMsgBuf.h.

References OSRTCtxtHolder::getCtxtPtr().

7.24.3.2 **virtual OSCTXT* OSRTMessageBuffer::getCtxtPtr ()** [inline, virtual]

The getCtxtPtr method returns the underlying C runtime context.

This context can be used in calls to C runtime functions.

Reimplemented from [OSRTCtxtHolder](#).

Definition at line 102 of file OSRTMsgBuf.h.

References OSRTCtxtHolder::getCtxtPtr().

7.24.3.3 **virtual char* OSRTMessageBuffer::getErrorInfo ()** [inline, virtual]

Returns error text in a dynamic memory buffer.

The buffer is allocated using 'operator new []'. The calling routine is responsible to free the memory by using 'operator delete []'.

Returns:

A pointer to a newly allocated buffer with error text.

Reimplemented from [OSRTCtxtHolder](#).

Definition at line 113 of file OSRTMsgBuf.h.

References OSRTCtxtHolder::getErrorInfo().

7.24.3.4 **virtual char* OSRTMessageBuffer::getErrorInfo (char * *pBuf*, size_t & *bufSize*)** [inline, virtual]

Returns error text in a memory buffer.

If buffer pointer is specified in parameters (not NULL) then error text will be copied in the passed buffer. Otherwise, this method allocates memory using the 'operator new []' function. The calling routine is responsible to free the memory by using 'operator delete []'.

Parameters:

pBuf A pointer to a destination buffer to obtain the error text. If NULL, dynamic buffer will be allocated.

bufSize A reference to buffer size. If pBuf is NULL it will receive the size of allocated dynamic buffer.

Returns:

A pointer to a buffer with error text. If pBuf is not NULL, the return pointer will be equal to it. Otherwise, returns newly allocated buffer with error text. NULL, if error occurred.

Reimplemented from [OSRTCtxtHolder](#).

Definition at line 133 of file OSRTMsgBuf.h.

References OSRTCtxtHolder::getErrorInfo().

7.24.3.5 int OSRTMessageBuffer::getStatus () const [inline, virtual]

This method returns the completion status of previous operation.

It can be used to check completion status of constructors or methods, which do not return completion status.

Returns:

Runtime status code:

- 0 = success,
- negative return value is error.

Reimplemented from [OSRTCtxtHolder](#).

Definition at line 162 of file OSRTMsgBuf.h.

References OSRTCtxtHolder::getStatus().

7.24.3.6 virtual int OSRTMessageBuffer::init () [inline, virtual]

Initializes message buffer.

Returns:

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

Implements [OSRTMessageBufferIF](#).

Definition at line 173 of file OSRTMsgBuf.h.

7.24.3.7 virtual int OSRTMessageBuffer::initBuffer (OSOCKET * pMsgBuf, size_t msgBufLen) [virtual]

This version of the overloaded initBuffer method initializes the message buffer to point at the given null-terminated character string.

Parameters:

pMsgBuf Pointer to message buffer.

msgBufLen Length of message buffer in bytes.

Returns:

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

Implements [OSRTMessageBufferIF](#).

7.24.3.8 virtual void OSRTMessageBuffer::setDiag (OSBOOL *value* = TRUE) [virtual]

The setDiag method will turn diagnostic tracing on or off.

Parameters:

value - Boolean value (default = TRUE = on)

Implements [OSRTMessageBufferIF](#).

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

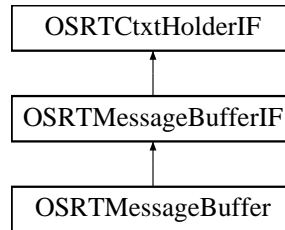
- [OSRTMsgBuf.h](#)

7.25 OSRTMessageBufferIF Class Reference

Abstract message buffer or stream interface class.

```
#include <OSRTMsgBufIF.h>
```

Inheritance diagram for OSRTMessageBufferIF::



Public Types

- enum [Type](#)

Public Member Functions

- virtual void * [getAppInfo](#) ()=0
Returns a pointer to application-specific information block.
- virtual size_t [getByteIndex](#) ()=0
The [getByteIndex](#) method is used to fetch the current byte offset within the current working buffer.
- virtual OSOCTET * [getMsgCopy](#) ()=0
The [getMsgCopy](#) method will return a copy of the encoded ASN.1 message managed by the object.
- virtual const OSOCTET * [getMsgPtr](#) ()=0
The [getMsgPtr](#) method will return a const pointer to the encoded ASN.1 message managed by the object.
- virtual int [init](#) ()=0
Initializes message buffer.
- virtual int [initBuffer](#) (OSOCTET *pMsgBuf, size_t msgBufLen)=0
This version of the overloaded [initBuffer](#) method initializes the message buffer to point at the given null-terminated character string.
- virtual OSBOOL [isA](#) (int bufferType)=0
This method checks the type of the message buffer.
- virtual void [setAppInfo](#) (void *pAppInfo)=0
Sets the application-specific information block.
- virtual void [setNamespace](#) (const OSUTF8CHAR *, const OSUTF8CHAR *, OSRTDList *)=0
Sets the namespace information.

- virtual void [setDiag](#) (OSBOOL value=TRUE)=0
The setDiag method will turn diagnostic tracing on or off.

Protected Member Functions

- virtual [~OSRTMessageBufferIF](#) ()
The virtual destructor does nothing.

7.25.1 Detailed Description

Abstract message buffer or stream interface class.

This is the base class for both the in-memory message buffer classes and the run-time stream classes.

Definition at line 47 of file OSRTMsgBufIF.h.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 virtual [OSRTMessageBufferIF::~OSRTMessageBufferIF](#) () [inline, protected, virtual]

The virtual destructor does nothing.

It is overridden by derived versions of this class.

Definition at line 58 of file OSRTMsgBufIF.h.

7.25.3 Member Function Documentation

7.25.3.1 virtual [size_t OSRTMessageBufferIF::getByteIndex](#) () [pure virtual]

The [getByteIndex](#) method is used to fetch the current byte offset within the current working buffer.

For encoding, this is the next location that will be written to. For decoding, this is the next byte the parser will read.

Implemented in [OSRTMessageBuffer](#).

7.25.3.2 virtual [OSOCKET* OSRTMessageBufferIF::getMsgCopy](#) () [pure virtual]

The [getMsgCopy](#) method will return a copy of the encoded ASN.1 message managed by the object.

The memory for the copy is allocated by new [] operator, user is responsible to free it by delete [] operator.

Returns:

The pointer to copied encoded ASN.1 message. NULL, if error occurred.

Implemented in [OSRTMessageBuffer](#).

7.25.3.3 `virtual const OSOCTET* OSRTMessageBufferIF::getMsgPtr ()` [pure virtual]

The `getMsgPtr` method will return a const pointer to the encoded ASN.1 message managed by the object.

Returns:

The pointer to the encoded ASN.1 message.

Implemented in [OSRTMessageBuffer](#).

7.25.3.4 `virtual int OSRTMessageBufferIF::init ()` [pure virtual]

Initializes message buffer.

Returns:

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

Implemented in [OSRTMessageBuffer](#).

7.25.3.5 `virtual int OSRTMessageBufferIF::initBuffer (OSOCTET * pMsgBuf, size_t msgBufLen)` [pure virtual]

This version of the overloaded `initBuffer` method initializes the message buffer to point at the given null-terminated character string.

Parameters:

pMsgBuf Pointer to message buffer.

msgBufLen Length of message buffer in bytes. string.

Returns:

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

Implemented in [OSRTMessageBuffer](#).

7.25.3.6 `virtual OSBOOL OSRTMessageBufferIF::isA (int bufferType)` [pure virtual]

This method checks the type of the message buffer.

Parameters:

bufferType Enumerated identifier specifying a derived class. Possible values are: `BEREncode`, `BERDecode`, `PEREncode`, `PERDecode`, `XEREncode`, `XERDecode`, `XMLEncode`, `XMLDecode`, `Stream`.

Returns:

Boolean result of the match operation. True if this is the class corresponding to the identifier argument.

7.25.3.7 virtual void OSRTMessageBufferIF::setDiag (OSBOOL *value* = TRUE) [pure virtual]

The setDiag method will turn diagnostic tracing on or off.

Parameters:

value - Boolean value (default = TRUE = on)

Implemented in [OSRTMessageBuffer](#).

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

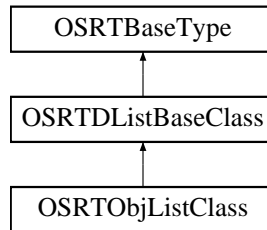
- [OSRTMsgBufIF.h](#)

7.26 OSRTObjListClass Class Reference

This class represents a doubly-linked list structure for objects.

```
#include <rtxCppDList.h>
```

Inheritance diagram for OSRTObjListClass::



Public Member Functions

- **OSRTObjListClass** ()
The default constructor initializes the list contents to empty.
- void **append** (OSRTBaseType *pdata)
The append method adds an item to the end of the list.
- void **appendCopy** (const OSRTBaseType *pdata)
The appendCopy method adds a copy of an item to the end of the list.
- OSRTBaseType * **clone** () const
- OSRTObjListNodeClass * **getHead** ()
This method returns a pointer to a head node of the list.
- const OSRTObjListNodeClass * **getHead** () const
This method returns a pointer to a head node of the list.
- const OSRTBaseType * **getItem** (int idx) const
The getItem method retrieves the data item from the list at the given index.
- OSRTObjListNodeClass * **getTail** ()
This method returns a pointer to a tail node of the list.
- const OSRTObjListNodeClass * **getTail** () const
This method returns a pointer to a tail node of the list.
- void **insert** (int index, OSRTBaseType *pdata)
The insert method inserts a data item into the list at the given indexed location.
- OSRTObjListClass & **operator=** (const OSRTObjListClass &)
Assignment operator.

7.26.1 Detailed Description

This class represents a doubly-linked list structure for objects.

It extends the C++ `OSRTDListBaseClass` type. It is similar to the `OSRTDListClass` described above except that the base type for items in the list is `OSRTBaseType`. This allows items in the list to be properly destructed when memory ownership for the items is transferred to the list object.

Definition at line 340 of file `rtxCppDList.h`.

7.26.2 Member Function Documentation

7.26.2.1 `void OSRTObjListClass::append (OSRTBaseType * pdata)`

The `append` method adds an item to the end of the list.

Parameters:

pdata - Pointer to data item to be appended to list. Note the pointer itself is appended - a copy is not made.

7.26.2.2 `void OSRTObjListClass::appendCopy (const OSRTBaseType * pdata)`

The `appendCopy` method adds a copy of an item to the end of the list.

Parameters:

pdata - Pointer to data item to be appended to list. Note that `clone()` is called on the data item, and the returned copy is stored in the list.

7.26.2.3 `OSRTObjListNodeClass* OSRTObjListClass::getHead () [inline]`

This method returns a pointer to a head node of the list.

Returns:

- Pointer to head node.

Definition at line 376 of file `rtxCppDList.h`.

7.26.2.4 `const OSRTObjListNodeClass* OSRTObjListClass::getHead () const [inline]`

This method returns a pointer to a head node of the list.

Returns:

- Pointer to head node.

Definition at line 385 of file `rtxCppDList.h`.

7.26.2.5 `const OSRTBaseType* OSRTObjListClass::getItem (int idx) const` [inline]

The `getItem` method retrieves the data item from the list at the given index.

The index is zero-based.

Parameters:

idx - Zero-based index of the node to retrieve.

Returns:

- Pointer to node structure containing the indexed data item.

Definition at line 396 of file `rtxCppDList.h`.

References `OSRTObjListNodeClass::getData()`.

7.26.2.6 `OSRTObjListNodeClass* OSRTObjListClass::getTail ()` [inline]

This method returns a pointer to a tail node of the list.

Returns:

- Pointer to tail node.

Definition at line 407 of file `rtxCppDList.h`.

7.26.2.7 `const OSRTObjListNodeClass* OSRTObjListClass::getTail () const` [inline]

This method returns a pointer to a tail node of the list.

Returns:

- Pointer to tail node.

Definition at line 416 of file `rtxCppDList.h`.

7.26.2.8 `void OSRTObjListClass::insert (int index, OSRTBaseType * pdata)`

The `insert` method inserts a data item into the list at the given indexed location.

The index is zero-based.

Parameters:

index - Zero-based index of insertion point.

pdata - Pointer to data item to be inserted into list. Note the pointer itself is inserted - a copy is not made.

7.26.2.9 `OSRTObjListClass& OSRTObjListClass::operator= (const OSRTObjListClass &)`

Assignment operator.

Sets the list's value to the value of the given list. Note that a copy of each object in the given list is made.

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

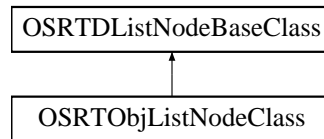
- [rtxCppDList.h](#)

7.27 OSRTObjListNodeClass Class Reference

This class represents a doubly-linked list node structure for `OSRTBaseType` instances.

```
#include <rtxCppDList.h>
```

Inheritance diagram for `OSRTObjListNodeClass`:



Public Member Functions

- `OSRTBaseType * getData ()`
This method returns a pointer to a data associated with the node.
- `const OSRTBaseType * getData () const`
This method returns a pointer to a data associated with the node.
- `OSRTObjListNodeClass * getNext ()`
This method returns a pointer to a next node in the list.
- `const OSRTObjListNodeClass * getNext () const`
This method returns a pointer to a next node in the list.
- `OSRTObjListNodeClass * getPrev ()`
This method returns a pointer to a previous node in the list.
- `const OSRTObjListNodeClass * getPrev () const`
This method returns a pointer to a previous node in the list.

7.27.1 Detailed Description

This class represents a doubly-linked list node structure for `OSRTBaseType` instances.

It extends the C++ `OSRTDListNodeBaseClass` type.

Definition at line 117 of file `rtxCppDList.h`.

7.27.2 Member Function Documentation

7.27.2.1 `OSRTBaseType* OSRTObjListNodeClass::getData () [inline]`

This method returns a pointer to a data associated with the node.

Returns:

Node data pointer.

Definition at line 128 of file rtxCppDList.h.

Referenced by OSRTObjListClass::getItem().

7.27.2.2 `const OSRTBaseType* OSRTObjListNodeClass::getData () const` [inline]

This method returns a pointer to a data associated with the node.

Returns:

Node data pointer.

Definition at line 135 of file rtxCppDList.h.

7.27.2.3 `OSRTObjListNodeClass* OSRTObjListNodeClass::getNext ()` [inline]

This method returns a pointer to a next node in the list.

Returns:

Pointer to the next node.

Definition at line 142 of file rtxCppDList.h.

7.27.2.4 `const OSRTObjListNodeClass* OSRTObjListNodeClass::getNext () const` [inline]

This method returns a pointer to a next node in the list.

Returns:

Pointer to the next node.

Definition at line 151 of file rtxCppDList.h.

7.27.2.5 `OSRTObjListNodeClass* OSRTObjListNodeClass::getPrev ()` [inline]

This method returns a pointer to a previous node in the list.

Returns:

Pointer to the previous node.

Definition at line 160 of file rtxCppDList.h.

7.27.2.6 `const OSRTObjListNodeClass* OSRTObjListNodeClass::getPrev () const` [inline]

This method returns a pointer to a previous node in the list.

Returns:

Pointer to the previous node.

Definition at line 169 of file rtxCppDList.h.

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

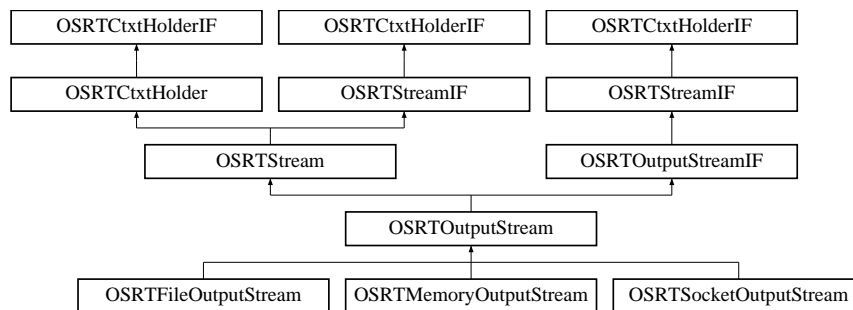
- [rtxCppDList.h](#)

7.28 OSRTOutputStream Class Reference

The base class definition for operations with output streams.

```
#include <OSRTOutputStream.h>
```

Inheritance diagram for OSRTOutputStream::



Public Member Functions

- [OSRTOutputStream \(\)](#)
The default constructor.
- virtual [~OSRTOutputStream \(\)](#)
Virtual destructor.
- virtual int [close \(\)](#)
Closes the output or output stream and releases any system resources associated with the stream.
- virtual int [flush \(\)](#)
Flushes the buffered data to the stream.
- virtual [OSRTCtxtPtr getContext \(\)](#)
This method returns a pointer to the underlying OSRTContext object.
- virtual [OSCTXT * getCtxtPtr \(\)](#)
This method returns a pointer to the underlying OSCTXT object.
- virtual [char * getErrorInfo \(\)](#)
Returns error text in a dynamic memory buffer.
- virtual [char * getErrorInfo \(char *pBuf, size_t &bufSize\)](#)
Returns error text in a memory buffer.
- virtual int [getStatus \(\) const](#)
This method returns the completion status of previous operation.
- virtual [OSBOOL isOpened \(\)](#)
Checks, is the stream opened or not.

- void [printErrorInfo](#) ()
The printErrorInfo method prints information on errors contained within the context.
- void [resetErrorInfo](#) ()
The resetErrorInfo method resets information on errors contained within the context.
- virtual long [write](#) (const OSOCKETET *pdata, size_t size)
Write data to the stream.

7.28.1 Detailed Description

The base class definition for operations with output streams.

As with the input stream, this implementation is backed by memory buffers to improve I/O performance.

Definition at line 43 of file OSRTOutputStream.h.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 OSRTOutputStream::OSRTOutputStream ()

The default constructor.

It initializes a buffered stream. A buffered stream maintains data in memory before reading or writing to the device. This generally provides better performance than an unbuffered stream.

7.28.2.2 virtual OSRTOutputStream::~~OSRTOutputStream () [virtual]

Virtual destructor.

Closes the stream if it was opened.

7.28.3 Member Function Documentation

7.28.3.1 virtual int OSRTOutputStream::close () [virtual]

Closes the output or output stream and releases any system resources associated with the stream.

For output streams this function also flushes all internal buffers to the stream.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

[rtxStreamClose](#), [rtxStreamBufClose](#)

Reimplemented from [OSRTStream](#).

7.28.3.2 `virtual int OSRTOutputStream::flush ()` [virtual]

Flushes the buffered data to the stream.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

`rtxStreamFlush`, `rtxStreamBufFlush`

Reimplemented from [OSRTStream](#).

7.28.3.3 `virtual OSRTCtxPtr OSRTOutputStream::getContext ()` [inline, virtual]

This method returns a pointer to the underlying [OSRTContext](#) object.

Returns:

A reference-counted pointer to an [OSRTContext](#) object. The [OSRTContext](#) object will not be released until all referenced-counted pointer variables go out of scope. This allows safe sharing of the context between different run-time classes.

Reimplemented from [OSRTStream](#).

Definition at line 91 of file `OSRTOutputStream.h`.

References `OSRTStream::getContext()`.

7.28.3.4 `virtual OSCTXT* OSRTOutputStream::getCtxtPtr ()` [inline, virtual]

This method returns a pointer to the underlying `OSCTXT` object.

This is the structure used in calls to low-level C encode/decode functions.

Returns:

Pointer to a context (`OSCTXT`) structure.

Reimplemented from [OSRTStream](#).

Definition at line 101 of file `OSRTOutputStream.h`.

References `OSRTStream::getCtxtPtr()`.

7.28.3.5 `virtual char* OSRTOutputStream::getErrorInfo ()` [inline, virtual]

Returns error text in a dynamic memory buffer.

Buffer will be allocated by `'operator new []'`. The calling routine is responsible to free the memory by using `'operator delete []'`.

Returns:

A pointer to a newly allocated buffer with error text.

Reimplemented from [OSRTStream](#).

Definition at line 112 of file OSRTOutputStream.h.

References OSRTStream::getErrorInfo().

7.28.3.6 `virtual char* OSRTOutputStream::getErrorInfo (char * pBuf, size_t & bufSize)` [inline, virtual]

Returns error text in a memory buffer.

If buffer pointer is specified in parameters (not NULL) then error text will be copied in the passed buffer. Otherwise, this method allocates memory using the 'operator new []' function. The calling routine is responsible to free the memory by using 'operator delete []'.

Parameters:

pBuf A pointer to a destination buffer to obtain the error text. If NULL, dynamic buffer will be allocated.

bufSize A reference to buffer size. If pBuf is NULL it will receive the size of allocated dynamic buffer.

Returns:

A pointer to a buffer with error text. If pBuf is not NULL, the return pointer will be equal to it. Otherwise, returns newly allocated buffer with error text. NULL, if error occurred.

Reimplemented from [OSRTStream](#).

Definition at line 132 of file OSRTOutputStream.h.

References OSRTStream::getErrorInfo().

7.28.3.7 `virtual int OSRTOutputStream::getStatus () const` [inline, virtual]

This method returns the completion status of previous operation.

It can be used to check completion status of constructors or methods, which do not return completion status.

Returns:

Runtime status code:

- 0 = success,
- negative return value is error.

Reimplemented from [OSRTStream](#).

Definition at line 145 of file OSRTOutputStream.h.

References OSRTStream::getStatus().

7.28.3.8 `virtual OSBOOL OSRTOutputStream::isOpened ()` [virtual]

Checks, is the stream opened or not.

Returns:

s TRUE, if the stream is opened, FALSE otherwise.

See also:

[rtxStreamIsOpened](#)

Reimplemented from [OSRTStream](#).

7.28.3.9 virtual long OSRTOutputStream::write (const OSOCTET * *pdata*, size_t *size*) [virtual]

Write data to the stream.

This method writes the given number of octets from the given array to the output stream.

Parameters:

pdata The pointer to the data to be written.

size The number of octets to write.

Returns:

The total number of octets written into the stream, or negative value with error code if any error is occurred.

See also:

[rtxStreamWrite](#)

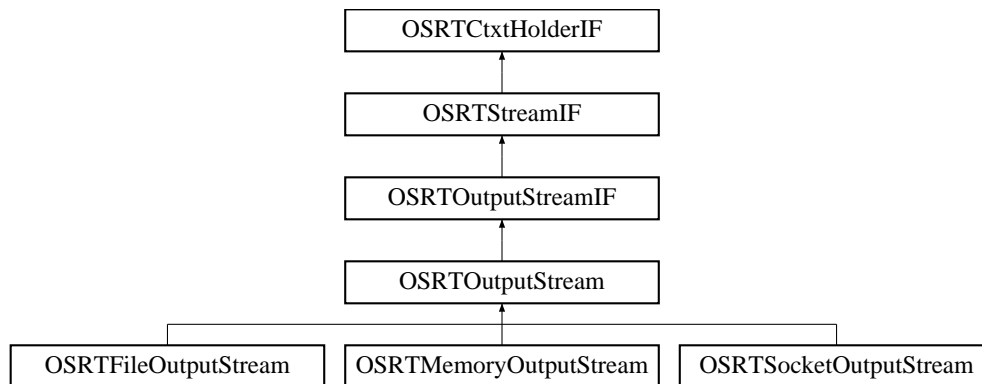
Implements [OSRTOutputStreamIF](#).

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

- [OSRTOutputStream.h](#)

7.29 OSRTOutputStreamIF Class Reference

Inheritance diagram for OSRTOutputStreamIF::



Public Member Functions

- virtual `~OSRTOutputStreamIF ()`
Virtual destructor.
- virtual long `write (const OSOCTET *pdata, size_t size)=0`
Write data to the stream.

7.29.1 Detailed Description

Definition at line 43 of file OSRTOutputStreamIF.h.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 virtual OSRTOutputStreamIF::~~OSRTOutputStreamIF () [virtual]

Virtual destructor.

Closes the stream if it was opened.

7.29.3 Member Function Documentation

7.29.3.1 virtual long OSRTOutputStreamIF::write (const OSOCTET * *pdata*, size_t *size*) [pure virtual]

Write data to the stream.

This method writes the given number of octets from the given array to the output stream.

Parameters:

pdata Pointer to the data to be written.

size The number of octets to write.

Returns:

The total number of octets written into the stream, or negative value with error code if any error is occurred.

See also:

`rtxStreamWrite`

Implemented in [OSRTOutputStream](#).

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

- [OSRTOutputStreamIF.h](#)

7.30 OSRTOutputStreamPtr Class Reference

Public Member Functions

- [OSRTOutputStreamPtr \(OSRTOutputStreamIF *ptr\)](#)
- [~OSRTOutputStreamPtr \(\)](#)
- [operator OSRTOutputStreamIF * \(\)](#)
- [OSRTOutputStreamIF * operator → \(\)](#)

7.30.1 Detailed Description

Definition at line 65 of file OSRTOutputStreamIF.h.

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

- [OSRTOutputStreamIF.h](#)

7.31 OSRTSocket Class Reference

Wrapper class for TCP/IP or UDP sockets.

```
#include <OSRTSocket.h>
```

Public Member Functions

- [OSRTSocket \(\)](#)
This is the default constructor.
- [OSRTSocket \(OSRTSOCKET socket, OSBOOL ownership=FALSE\)](#)
This constructor initializes an instance by using an existing socket.
- [OSRTSocket \(const OSRTSocket &socket\)](#)
The copy constructor.
- [~OSRTSocket \(\)](#)
The destructor.
- [OSRTSocket * accept \(OSIPADDR *destIP=0, int *port=0\)](#)
This method permits an incoming connection attempt on a socket.
- [int bind \(OSIPADDR addr, int port\)](#)
This method associates a local address with a socket.
- [int bind \(const char *pAddrStr, int port\)](#)
This method associates a local address with a socket.
- [int bind \(int port\)](#)
This method associates only a local port with a socket.
- [int blockingRead \(OSOCKET *pbuf, size_t readBytes\)](#)
This method receives data from the connected socket.
- [int close \(\)](#)
This method closes this socket.
- [int connect \(const char *host, int port\)](#)
This method establishes a connection to this socket.
- [OSBOOL getOwnership \(\)](#)
Returns the ownership of underlying O/S socket.
- [OSRTSOCKET getSocket \(\) const](#)
This method returns the handle of the socket.
- [int getStatus \(\)](#)
Returns a completion status of last operation.

- int [listen](#) (int maxConnections)
This method places a socket into a state where it is listening for an incoming connection.
- int [recv](#) (OSOCKET *pbuf, OSUINT32 bufsize)
This method receives data from a connected socket.
- void [setOwnership](#) (OSBOOL ownership)
Transfers an ownership of the underlying O/S socket to or from the socket object.
- int [send](#) (const OSOCKET *pdata, OSUINT32 size)
This method sends data on a connected socket.

Static Public Member Functions

- static const char * [addrToString](#) (OSIPADDR ipAddr, char *pAddrStr, int bufsize)
This method converts an IP address to its string representation.
- static OSIPADDR [stringToAddr](#) (const char *pAddrStr)
This method converts a string containing an Internet Protocol dotted address into a proper OSIPADDR address.

Protected Member Functions

- OSBOOL [isInitialized](#) ()

Protected Attributes

- OSRTSOCKET [mSocket](#)
handle of the socket
- int [mInitStatus](#)
- int [mStatus](#)
- OSBOOL [mOwner](#)
indicates this class owns the socket

7.31.1 Detailed Description

Wrapper class for TCP/IP or UDP sockets.

Definition at line 48 of file OSRTSocket.h.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 OSRTSocket::OSRTSocket ()

This is the default constructor.

It initializes all internal members with default values and creates a new socket structure. Use [getStatus\(\)](#) method to determine has error occurred during the initialization or not.

7.31.2.2 OSRTSocket::OSRTSocket (OSR_SOCKET *socket*, OS_BOOL *ownership* = FALSE)

This constructor initializes an instance by using an existing socket.

Parameters:

socket An existing socket handle.

ownership Boolean flag that specifies who is the owner of the socket. If it is TRUE then the socket will be destroyed in the destructor. Otherwise, the user is responsible to close and destroy the socket.

7.31.2.3 OSRTSocket::OSRTSocket (const OSRTSocket & *socket*)

The copy constructor.

The copied instance will have the same socket handle as the original one, but will not be the owner of the handle.

7.31.2.4 OSRTSocket::~OSRTSocket ()

The destructor.

This closes socket if the instance is the owner of the socket.

7.31.3 Member Function Documentation

7.31.3.1 OSRTSocket* OSRTSocket::accept (OS_IPADDR * *destIP* = 0, int * *port* = 0)

This method permits an incoming connection attempt on a socket.

It extracts the first connection on the queue of pending connections on the socket. It then creates a new socket and returns an instance of the new socket. The newly created socket will handle the actual connection and has the same properties as the original socket.

Parameters:

destIP Optional pointer to a buffer that receives the IP address of the connecting entity. It may be NULL.

port Optional pointer to a buffer that receives the port of the connecting entity. It may be NULL.

Returns:

An instance of the new socket class. NULL, if error occur. Use getStatus method to obtain error code.

See also:

rtSocketAccept

7.31.3.2 static const char* OSRTSocket::addrToString (OS_IPADDR *ipAddr*, char * *pAddrStr*, int *bufsize*) [static]

This method converts an IP address to its string representation.

Parameters:

ipAddr The IP address to be converted.

pAddrStr Pointer to the buffer to receive a string with the IP address.

bufsize Size of the buffer.

Returns:

Pointer to a string with IP-address. NULL, if error occur.

7.31.3.3 int OSRTSocket::bind (OSIPADDR *addr*, int *port*)

This method associates a local address with a socket.

It is used on an unconnected socket before subsequent calls to the connect or listen methods.

Parameters:

addr The local IP address to assign to the socket.

port The local port number to assign to the socket.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

rtSocketBind

7.31.3.4 int OSRTSocket::bind (const char * *pAddrStr*, int *port*)

This method associates a local address with a socket.

It is used on an unconnected socket before subsequent calls to the connect or listen methods.

Parameters:

pAddrStr Null-terminated character string representing a number expressed in the Internet standard "." (dotted) notation.

port The local port number to assign to the socket.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

rtSocketBind

7.31.3.5 `int OSRTSocket::bind (int port) [inline]`

This method associates only a local port with a socket.

It is used on an unconnected socket before subsequent calls to the [OSRTSocket::connect](#) or [OSRTSocket::listen](#) methods.

Parameters:

port The local port number to assign to the socket.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

[rtSocketBind](#)
[bind \(\)](#)

Definition at line 168 of file OSRTSocket.h.

7.31.3.6 `int OSRTSocket::blockingRead (OSOCKET * pbuf, size_t readBytes)`

This method receives data from the connected socket.

In this case, the connection is blocked until either the requested number of bytes is received or the socket is closed or an error occurs.

Parameters:

pbuf Pointer to the buffer for the incoming data.

readBytes Number of bytes to receive.

Returns:

If no error occurs, returns the number of bytes received. Otherwise, the negative value is error code.

7.31.3.7 `int OSRTSocket::close ()`

This method closes this socket.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

[rtSocketClose](#)

7.31.3.8 `int OSRTSocket::connect (const char * host, int port)`

This method establishes a connection to this socket.

It is used to create a connection to the specified destination. When the socket call completes successfully, the socket is ready to send and receive data.

Parameters:

host Null-terminated character string representing a number expressed in the Internet standard "." (dotted) notation.

port The destination port to connect.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

rtSocketConnect

7.31.3.9 `OSBOOL OSRTSocket::getOwnership () [inline]`

Returns the ownership of underlying O/S socket.

Returns:

TRUE, if the socket object has the ownership of underlying O/S socket.

Definition at line 218 of file OSRTSocket.h.

7.31.3.10 `OSRTSOCKET OSRTSocket::getSocket () const [inline]`

This method returns the handle of the socket.

Returns:

The handle of the socket.

Definition at line 225 of file OSRTSocket.h.

7.31.3.11 `int OSRTSocket::getStatus () [inline]`

Returns a completion status of last operation.

Returns:

Completion status of last operation:

- 0 = success,
- negative return value is error.

Definition at line 234 of file OSRTSocket.h.

7.31.3.12 `int OSRTSocket::listen (int maxConnections)`

This method places a socket into a state where it is listening for an incoming connection.

Parameters:

maxConnections Maximum length of the queue of pending connections.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

`rtSocketListen`

7.31.3.13 `int OSRTSocket::recv (OSOCKET * pbuf, OSUINT32 bufsize)`

This method receives data from a connected socket.

It is used to read incoming data on sockets. The socket must be connected before calling this function.

Parameters:

pbuf Pointer to the buffer for the incoming data.

bufsize Length of the buffer.

Returns:

If no error occurs, returns the number of bytes received. Negative error code if error occurred.

See also:

`rtSocketRecv`

7.31.3.14 `void OSRTSocket::setOwnership (OSBOOL ownership)` `[inline]`

Transfers an ownership of the underlying O/S socket to or from the socket object.

If the socket object has the ownership of the underlying O/S socket it will close the O/S socket when the socket object is being closed or destroyed.

Parameters:

ownership TRUE, if socket object should have ownership of the underlying O/S socket; FALSE, otherwise.

Definition at line 273 of file OSRTSocket.h.

7.31.3.15 `int OSRTSocket::send (const OSOCKET * pdata, OSUINT32 size)`

This method sends data on a connected socket.

It is used to write outgoing data on a connected socket.

Parameters:

pdata Buffer containing the data to be transmitted.

size Length of the data in *pdata*.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

`rtSocketSend`

7.31.3.16 `static OSIPADDR OSRTSocket::stringToAddr (const char * pAddrStr) [static]`

This method converts a string containing an Internet Protocol dotted address into a proper OSIPADDR address.

Parameters:

pAddrStr Null-terminated character string representing a number expressed in the Internet standard "." (dotted) notation.

Returns:

If no error occurs, returns OSIPADDR. OSIPADDR_INVALID, if error occurred.

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

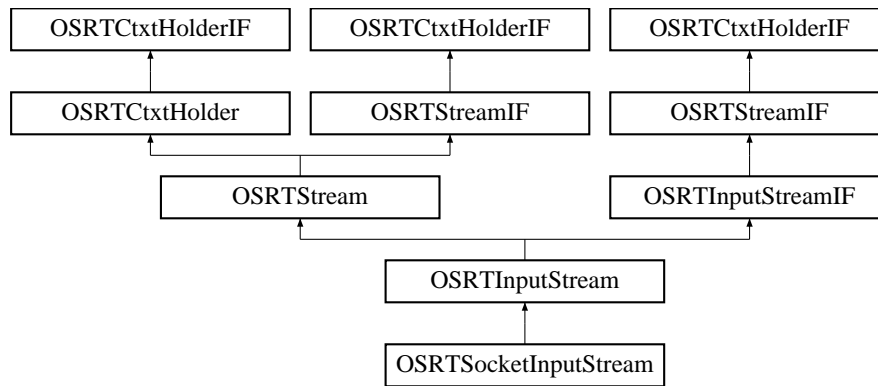
- [OSRTSocket.h](#)

7.32 OSRTSocketInputStream Class Reference

Generic socket input stream.

```
#include <OSRTSocketInputStream.h>
```

Inheritance diagram for OSRTSocketInputStream::



Public Member Functions

- [OSRTSocketInputStream \(OSRTSocket &socket\)](#)
Creates and initializes a socket input stream using the [OSRTSocket](#) instance of socket.
- [OSRTSocketInputStream \(OSRTContext *pContext, OSRTSocket &socket\)](#)
Creates and initializes a socket input stream using the [OSRTSocket](#) instance of socket.
- [OSRTSocketInputStream \(OSRTSOCKET socket, OSBOOL ownership=FALSE\)](#)
Creates and initializes the socket input stream using the socket handle.
- [OSRTSocketInputStream \(OSRTContext *pContext, OSRTSOCKET socket, OSBOOL ownership=FALSE\)](#)
Creates and initializes the socket input stream using the socket handle.

Protected Attributes

- [OSRTSocket mSocket](#)
a socket

7.32.1 Detailed Description

Generic socket input stream.

This class opens an existing socket for input in binary mode and reads data from it.

Definition at line 38 of file OSRTSocketInputStream.h.

7.32.2 Constructor & Destructor Documentation

7.32.2.1 OSRTSocketInputStream::OSRTSocketInputStream (OSRTSocket & socket)

Creates and initializes a socket input stream using the OSRTSocket instance of socket.

Parameters:

socket Reference to OSRTSocket instance.

See also:

rtxStreamSocketOpen

7.32.2.2 OSRTSocketInputStream::OSRTSocketInputStream (OSRTContext * pContext, OSRTSocket & socket)

Creates and initializes a socket input stream using the OSRTSocket instance of socket.

Parameters:

pContext Pointer to a context to use.

socket Reference to OSRTSocket instance.

See also:

rtxStreamSocketOpen

7.32.2.3 OSRTSocketInputStream::OSRTSocketInputStream (OSRTSOCKET socket, OSBOOL ownership = FALSE)

Creates and initializes the socket input stream using the socket handle.

Parameters:

socket Handle of the socket.

ownership Indicates ownership of the socket. Set to TRUE to pass ownership to this object instance. The socket will be closed when this object instance is deleted or goes out of scope.

See also:

rtxStreamSocketAttach

7.32.2.4 OSRTSocketInputStream::OSRTSocketInputStream (OSRTContext * pContext, OSRTSOCKET socket, OSBOOL ownership = FALSE)

Creates and initializes the socket input stream using the socket handle.

Parameters:

pContext Pointer to a context to use.

socket Handle of the socket.

ownership Indicates ownership of the socket. Set to TRUE to pass ownership to this object instance. The socket will be closed when this object instance is deleted or goes out of scope.

See also:

rtxStreamSocketAttach

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

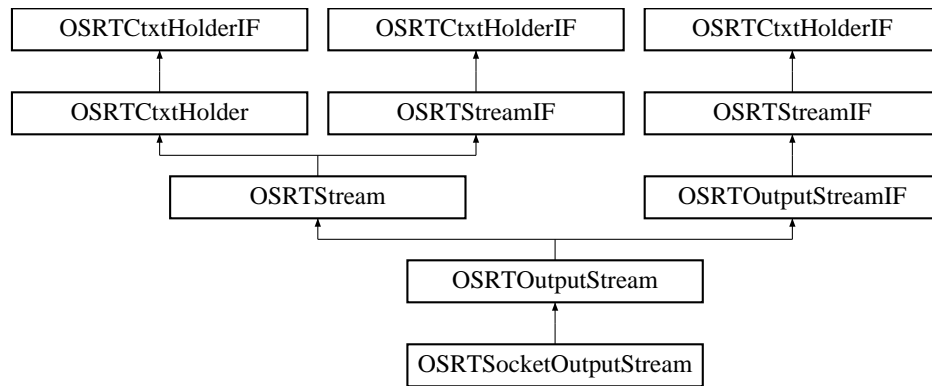
- [OSRTSocketInputStream.h](#)

7.33 OSRTSocketOutputStream Class Reference

Generic socket output stream.

```
#include <OSRTSocketOutputStream.h>
```

Inheritance diagram for OSRTSocketOutputStream::



Public Member Functions

- [OSRTSocketOutputStream](#) ([OSRTSocket](#) &socket)
Creates and initializes a socket output stream using the [OSRTSocket](#) instance of socket.
- [OSRTSocketOutputStream](#) ([OSRTContext](#) *pContext, [OSRTSocket](#) &socket)
Creates and initializes a socket output stream using the [OSRTSocket](#) instance of socket.
- [OSRTSocketOutputStream](#) ([OSRTSOCKET](#) socket, [OSBOOL](#) ownership=FALSE)
Initializes the socket output stream using the socket handle.
- [OSRTSocketOutputStream](#) ([OSRTContext](#) *pContext, [OSRTSOCKET](#) socket, [OSBOOL](#) ownership=FALSE)
Initializes the socket output stream using the socket handle.

Protected Attributes

- [OSRTSocket](#) mSocket
a socket

7.33.1 Detailed Description

Generic socket output stream.

This class opens an existing socket for output in binary mode and reads data from it.

Definition at line 38 of file OSRTSocketOutputStream.h.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 OSRTSocketOutputStream::OSRTSocketOutputStream (OSRTSocket & socket)

Creates and initializes a socket output stream using the OSRTSocket instance of socket.

Parameters:

socket Reference to OSRTSocket instance.

See also:

rtxStreamSocketOpen

7.33.2.2 OSRTSocketOutputStream::OSRTSocketOutputStream (OSRTContext * pContext, OSRTSocket & socket)

Creates and initializes a socket output stream using the OSRTSocket instance of socket.

Parameters:

pContext Pointer to a context to use.

socket Reference to OSRTSocket instance.

See also:

rtxStreamSocketOpen

7.33.2.3 OSRTSocketOutputStream::OSRTSocketOutputStream (OSRTSOCKET socket, OSBOOL ownership = FALSE)

Initializes the socket output stream using the socket handle.

Parameters:

socket Handle of the socket.

ownership Indicates ownership of the socket. Set to TRUE to pass ownership to this object instance. The socket will be closed when this object instance is deleted or goes out of scope.

See also:

rtxStreamSocketAttach

7.33.2.4 OSRTSocketOutputStream::OSRTSocketOutputStream (OSRTContext * pContext, OSRTSOCKET socket, OSBOOL ownership = FALSE)

Initializes the socket output stream using the socket handle.

Parameters:

pContext Pointer to a context to use.

socket Handle of the socket.

ownership Indicates ownership of the socket. Set to TRUE to pass ownership to this object instance. The socket will be closed when this object instance is deleted or goes out of scope.

See also:

rtxStreamSocketAttach

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

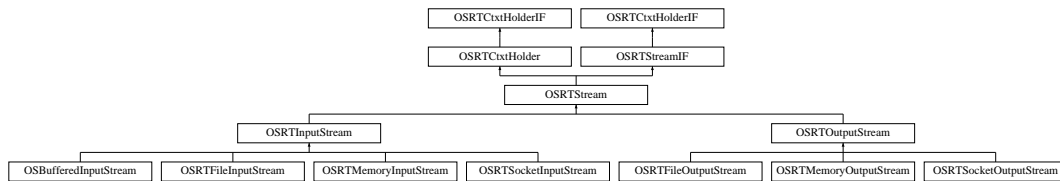
- [OSRTSocketOutputStream.h](#)

7.34 OSRTStream Class Reference

The default base class for using I/O streams.

```
#include <OSRTStream.h>
```

Inheritance diagram for OSRTStream::



Public Member Functions

- virtual `~OSRTStream ()`
Virtual destructor.
- virtual `int close ()`
Closes the input or output stream and releases any system resources associated with the stream.
- virtual `int flush ()`
Flushes the buffered data to the stream.
- virtual `OSRTCtxtPtr getContext ()`
This method returns a pointer to the underlying OSRTContext object.
- virtual `OSCTXT * getCtxtPtr ()`
This method returns a pointer to the underlying OSCTXT object.
- virtual `char * getErrorInfo ()`
Returns error text in a dynamic memory buffer.
- virtual `char * getErrorInfo (char *pBuf, size_t &bufSize)`
Returns error text in a memory buffer.
- `int getStatus () const`
This method returns the completion status of previous operation.
- `OSBOOL isInitialized ()`
- virtual `OSBOOL isOpened ()`
Checks, is the stream opened or not.
- `void printErrorInfo ()`
The printErrorInfo method prints information on errors contained within the context.
- `void resetErrorInfo ()`
The resetErrorInfo method resets information on errors contained within the context.

Protected Member Functions

- [OSRTStream \(\)](#)

The default constructor.

Protected Attributes

- OSBOOL [mbAttached](#)

Flag, TRUE for "attached" streams.

- int [mStatus](#)

Last stream operation status.

- int [mInitStatus](#)

Initialization status. 0 if initialized successfully.

7.34.1 Detailed Description

The default base class for using I/O streams.

This class may be subclassed, as in the case of [OSRTInputStream](#) and [OSRTOutputStream](#) or other custom implementations.

Definition at line 44 of file OSRTStream.h.

7.34.2 Constructor & Destructor Documentation

7.34.2.1 OSRTStream::OSRTStream () [protected]

The default constructor.

It initializes a buffered stream. A buffered stream maintains data in memory before reading or writing to the device. This generally provides better performance than an unbuffered stream.

7.34.2.2 virtual OSRTStream::~~OSRTStream () [virtual]

Virtual destructor.

Closes the stream if it was opened.

7.34.3 Member Function Documentation

7.34.3.1 virtual int OSRTStream::close () [virtual]

Closes the input or output stream and releases any system resources associated with the stream.

For output streams this function also flushes all internal buffers to the stream.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

`rtxStreamClose`

Implements [OSRTStreamIF](#).

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

7.34.3.2 virtual int OSRTStream::flush () [virtual]

Flushes the buffered data to the stream.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

`rtxStreamFlush`

Implements [OSRTStreamIF](#).

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

7.34.3.3 virtual OSRTCtxtPtr OSRTStream::getContext () [inline, virtual]

This method returns a pointer to the underlying [OSRTContext](#) object.

Returns:

A reference-counted pointer to an [OSRTContext](#) object. The [OSRTContext](#) object will not be released until all referenced-counted pointer variables go out of scope. This allows safe sharing of the context between different run-time classes.

Reimplemented from [OSRTCtxtHolder](#).

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

Definition at line 99 of file `OSRTStream.h`.

References `OSRTCtxtHolder::getContext()`.

Referenced by `OSRTOutputStream::getContext()`, and `OSRTInputStream::getContext()`.

7.34.3.4 virtual OSCTXT* OSRTStream::getCtxtPtr () [inline, virtual]

This method returns a pointer to the underlying `OSCTXT` object.

This is the structure used in calls to low-level C encode/decode functions.

Returns:

Pointer to a context (OSCTXT) structure.

Reimplemented from [OSRTCtxtHolder](#).

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

Definition at line 109 of file OSRTStream.h.

References [OSRTCtxtHolder::getCtxtPtr\(\)](#).

Referenced by [OSRTOutputStream::getCtxtPtr\(\)](#), and [OSRTInputStream::getCtxtPtr\(\)](#).

7.34.3.5 virtual char* OSRTStream::getErrorInfo () [inline, virtual]

Returns error text in a dynamic memory buffer.

Buffer will be allocated by 'operator new []'. The calling routine is responsible to free the memory by using 'operator delete []'.

Returns:

A pointer to a newly allocated buffer with error text.

Reimplemented from [OSRTCtxtHolder](#).

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

Definition at line 120 of file OSRTStream.h.

References [OSRTCtxtHolder::getErrorInfo\(\)](#).

Referenced by [OSRTOutputStream::getErrorInfo\(\)](#), and [OSRTInputStream::getErrorInfo\(\)](#).

7.34.3.6 virtual char* OSRTStream::getErrorInfo (char * pBuf, size_t & bufSize) [inline, virtual]

Returns error text in a memory buffer.

If buffer pointer is specified in parameters (not NULL) then error text will be copied in the passed buffer. Otherwise, this method allocates memory using the 'operator new []' function. The calling routine is responsible to free the memory by using 'operator delete []'.

Parameters:

pBuf A pointer to a destination buffer to obtain the error text. If NULL, dynamic buffer will be allocated.

bufSize A reference to buffer size. If pBuf is NULL it will receive the size of allocated dynamic buffer.

Returns:

A pointer to a buffer with error text. If pBuf is not NULL, the return pointer will be equal to it. Otherwise, returns newly allocated buffer with error text. NULL, if error occurred.

Reimplemented from [OSRTCtxtHolder](#).

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

Definition at line 140 of file OSRTStream.h.

References [OSRTCtxtHolder::getErrorInfo\(\)](#).

7.34.3.7 `int OSRTStream::getStatus () const` [inline, virtual]

This method returns the completion status of previous operation.

It can be used to check completion status of constructors or methods, which do not return completion status.

Returns:

Runtime status code:

- 0 = success,
- negative return value is error.

Reimplemented from [OSRTCtxtHolder](#).

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

Definition at line 153 of file `OSRTStream.h`.

Referenced by `OSRTOutputStream::getStatus()`, and `OSRTInputStream::getStatus()`.

7.34.3.8 `virtual OSBOOL OSRTStream::isOpened ()` [virtual]

Checks, is the stream opened or not.

Returns:

TRUE, if the stream is opened, FALSE otherwise.

See also:

`rtxStreamIsOpened`

Implements [OSRTStreamIF](#).

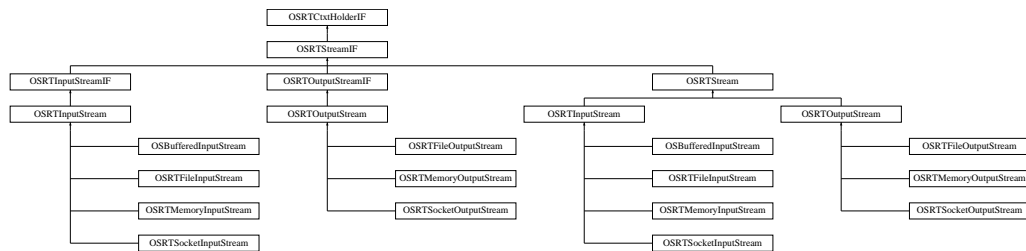
Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

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

- [OSRTStream.h](#)

7.35 OSRTStreamIF Class Reference

Inheritance diagram for OSRTStreamIF::



Public Member Functions

- virtual int `close ()=0`
Closes the input or output stream and releases any system resources associated with the stream.
- virtual int `flush ()=0`
Flushes the buffered data to the stream.
- virtual OSBOOL `isOpened ()=0`
Checks, is the stream opened or not.

7.35.1 Detailed Description

Definition at line 41 of file OSRTStreamIF.h.

7.35.2 Member Function Documentation

7.35.2.1 virtual int OSRTStreamIF::close () [pure virtual]

Closes the input or output stream and releases any system resources associated with the stream.

For output streams this function also flushes all internal buffers to the stream.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

`rtxStreamClose`

Implemented in `OSRTInputStream`, `OSRTOutputStream`, and `OSRTStream`.

7.35.2.2 `virtual int OSRTStreamIF::flush ()` [pure virtual]

Flushes the buffered data to the stream.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

See also:

`rtxStreamFlush`

Implemented in [OSRTInputStream](#), [OSRTOutputStream](#), and [OSRTStream](#).

7.35.2.3 `virtual OSBOOL OSRTStreamIF::isOpened ()` [pure virtual]

Checks, is the stream opened or not.

Returns:

TRUE, if the stream is opened, FALSE otherwise.

See also:

`rtxStreamIsOpened`

Implemented in [OSRTInputStream](#), [OSRTOutputStream](#), and [OSRTStream](#).

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

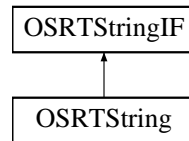
- [OSRTStreamIF.h](#)

7.36 OSRTString Class Reference

C++ string class definition.

```
#include <OSRTString.h>
```

Inheritance diagram for OSRTString::



Public Member Functions

- [OSRTString \(\)](#)
The default constructor creates an empty string.
- [OSRTString \(const char *strval\)](#)
This constructor initializes the string to contain the given standard ASCII string value.
- [OSRTString \(const OSUTF8CHAR *strval\)](#)
This constructor initializes the string to contain the given UTF-8 string value.
- [OSRTString \(const OSRTString &str\)](#)
Copy constructor.
- [virtual ~OSRTString \(\)](#)
The destructor frees string memory using the standard 'delete' operator.
- [virtual OSRTStringIF * clone \(\)](#)
This method creates a copy of the given string object.
- [virtual const char * getValue \(\) const](#)
This method returns the pointer to UTF-8 null terminated string as a standard ASCII string.
- [virtual const OSUTF8CHAR * getUTF8Value \(\) const](#)
This method returns the pointer to UTF-8 null terminated string as a UTF-8 string.
- [virtual void print \(const char *name\)](#)
This method prints the string value to standard output.
- [virtual void setValue \(const char *str\)](#)
This method sets the string value to the given string.
- [virtual void setValue \(const OSUTF8CHAR *str\)](#)
This method sets the string value to the given UTF-8 string value.
- [OSRTString & operator= \(const OSRTString &original\)](#)
Assignment operator.

Protected Attributes

- OSUTF8CHAR * [mValue](#)

7.36.1 Detailed Description

C++ string class definition.

This can be used to hold standard ASCII or UTF-8 strings. The standard C++ 'new' and 'delete' operators are used to allocate/free memory for the strings. All strings are deep-copied.

Definition at line 49 of file OSRTString.h.

7.36.2 Constructor & Destructor Documentation

7.36.2.1 OSRTString::OSRTString (const char * *strval*)

This constructor initializes the string to contain the given standard ASCII string value.

Parameters:

strval - Null-terminated C string value

7.36.2.2 OSRTString::OSRTString (const OSUTF8CHAR * *strval*)

This constructor initializes the string to contain the given UTF-8 string value.

Parameters:

strval - Null-terminated C string value

7.36.2.3 OSRTString::OSRTString (const [OSRTString](#) & *str*)

Copy constructor.

Parameters:

str - C++ string object to be copied.

7.36.3 Member Function Documentation

7.36.3.1 virtual void OSRTString::print (const char * *name*) [inline, virtual]

This method prints the string value to standard output.

Parameters:

name - Name of generated string variable.

Implements [OSRTStringIF](#).

Definition at line 114 of file OSRTString.h.

7.36.3.2 virtual void OSRTString::setValue (const char * *str*) [virtual]

This method sets the string value to the given string.

Parameters:

str - C null-terminated string.

Implements [OSRTStringIF](#).

7.36.3.3 virtual void OSRTString::setValue (const OSUTF8CHAR * *str*) [virtual]

This method sets the string value to the given UTF-8 string value.

Parameters:

str - C null-terminated UTF-8 string.

Implements [OSRTStringIF](#).

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

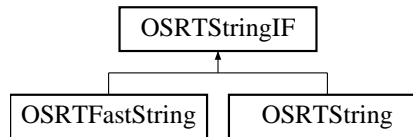
- [OSRTString.h](#)

7.37 OSRTStringIF Class Reference

C++ string class interface.

```
#include <OSRTStringIF.h>
```

Inheritance diagram for OSRTStringIF::



Public Member Functions

- virtual `~OSRTStringIF ()`
The destructor frees string memory using the standard 'delete' operator.
- virtual `OSRTStringIF * clone ()=0`
This method creates a copy of the given string object.
- virtual `const char * getValue () const=0`
This method returns the pointer to UTF-8 null terminated string as a standard ASCII string.
- virtual `const OSUTF8CHAR * getUTF8Value () const=0`
This method returns the pointer to UTF-8 null terminated string as a UTF-8 string.
- virtual `void print (const char *name)=0`
This method prints the string value to standard output.
- virtual `void setValue (const char *str)=0`
This method sets the string value to the given string.
- virtual `void setValue (const OSUTF8CHAR *utf8str)=0`
This method sets the string value to the given UTF-8 string value.

Protected Member Functions

- `OSRTStringIF ()`
The default constructor creates an empty string.
- `OSRTStringIF (const char *)`
This constructor initializes the string to contain the given standard ASCII string value.
- `OSRTStringIF (const OSUTF8CHAR *)`
This constructor initializes the string to contain the given UTF-8 string value.

7.37.1 Detailed Description

C++ string class interface.

This defines an interface to allow different types of string derived classes to be implemented. Currently, implementations include a standard string class ([OSRTString](#)) which deep-copies all values using new/delete, and a fast string class ([OSRTFastString](#)) that just copies pointers (i.e does no memory management).

Definition at line 49 of file OSRTStringIF.h.

7.37.2 Constructor & Destructor Documentation

7.37.2.1 OSRTStringIF::OSRTStringIF (const char *) [inline, protected]

This constructor initializes the string to contain the given standard ASCII string value.

Parameters:

strval - Null-terminated C string value

Definition at line 62 of file OSRTStringIF.h.

7.37.2.2 OSRTStringIF::OSRTStringIF (const OSUTF8CHAR *) [inline, protected]

This constructor initializes the string to contain the given UTF-8 string value.

Parameters:

strval - Null-terminated C string value

Definition at line 70 of file OSRTStringIF.h.

7.37.3 Member Function Documentation

7.37.3.1 virtual void OSRTStringIF::print (const char * name) [pure virtual]

This method prints the string value to standard output.

Parameters:

name - Name of generated string variable.

Implemented in [OSRTFastString](#), and [OSRTString](#).

7.37.3.2 virtual void OSRTStringIF::setValue (const char * str) [pure virtual]

This method sets the string value to the given string.

Parameters:

str - C null-terminated string.

Implemented in [OSRTFastString](#), and [OSRTString](#).

7.37.3.3 virtual void OSRTStringIF::setValue (const OSUTF8CHAR * *utf8str*) [pure virtual]

This method sets the string value to the given UTF-8 string value.

Parameters:

utf8str - C null-terminated UTF-8 string.

Implemented in [OSRTFastString](#), and [OSRTString](#).

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

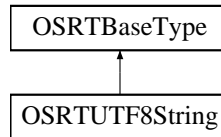
- [OSRTStringIF.h](#)

7.38 OSRTUTF8String Class Reference

UTF-8 string.

```
#include <OSRTUTF8String.h>
```

Inheritance diagram for OSRTUTF8String::



Public Member Functions

- [OSRTUTF8String \(\)](#)
The default constructor creates an empty string.
- [OSRTUTF8String \(const char *strval\)](#)
This constructor initializes the string to contain the given character string value.
- [OSRTUTF8String \(const OSUTF8CHAR *strval\)](#)
This constructor initializes the string to contain the given UTF-8 character string value.
- [OSRTUTF8String \(const OSRTUTF8String &str\)](#)
Copy constructor.
- [virtual ~OSRTUTF8String \(\)](#)
The destructor frees string memory if the memory ownership flag is set.
- [OSRTBaseType * clone \(\) const](#)
Clone method.
- [void copyValue \(const char *str\)](#)
This method copies the given string value to the internal string storage variable.
- [const char * c_str \(\) const](#)
This method returns the pointer to C null terminated string.
- [const char * getValue \(\) const](#)
This method returns the pointer to UTF-8 null terminated string.
- [void print \(const char *name\)](#)
This method prints the string value to standard output.
- [void setValue \(const char *str\)](#)
This method sets the string value to the given string.
- [OSRTUTF8String & operator= \(const OSRTUTF8String &original\)](#)
Assignment operator.

7.38.1 Detailed Description

UTF-8 string.

This is the base class for generated C++ data type classes for XSD string types (string, token, NMTOKEN, etc.).
Definition at line 39 of file OSRTUTF8String.h.

7.38.2 Constructor & Destructor Documentation

7.38.2.1 OSRTUTF8String::OSRTUTF8String (const char * *strval*)

This constructor initializes the string to contain the given character string value.

Parameters:

strval - String value

7.38.2.2 OSRTUTF8String::OSRTUTF8String (const OSUTF8CHAR * *strval*)

This constructor initializes the string to contain the given UTF-8 character string value.

Parameters:

strval - String value

7.38.2.3 OSRTUTF8String::OSRTUTF8String (const OSRTUTF8String & *str*)

Copy constructor.

Parameters:

str - C++ XML string class.

7.38.3 Member Function Documentation

7.38.3.1 OSRTBaseType* OSRTUTF8String::clone () const [inline, virtual]

Clone method.

Creates a copied instance and returns pointer to OSRTBaseType.

Reimplemented from OSRTBaseType.

Definition at line 81 of file OSRTUTF8String.h.

7.38.3.2 void OSRTUTF8String::copyValue (const char * *str*)

This method copies the given string value to the internal string storage variable.

A deep-copy of the given value is done; the class will delete this memory when the object is deleted.

Parameters:

str - C null-terminated string.

7.38.3.3 void OSRTUTF8String::print (const char * *name*) [inline]

This method prints the string value to standard output.

Parameters:

name - Name of generated string variable.

Definition at line 111 of file OSRTUTF8String.h.

7.38.3.4 void OSRTUTF8String::setValue (const char * *str*)

This method sets the string value to the given string.

A deep-copy of the given value is not done; the pointer is stored directly in the class member variable.

Parameters:

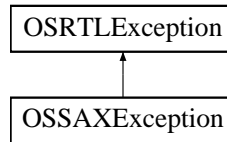
str - C null-terminated string.

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

- [OSRTUTF8String.h](#)

7.39 OSSAXException Class Reference

Inheritance diagram for OSSAXException::



Public Member Functions

- [OSSAXException \(\)](#)
Default constructor.
- [OSSAXException \(OSRTContext *pContext, const OSUTF8CHAR *const msg\)](#)
Constructor.
- [OSSAXException \(OSRTContext *pContext, const OSUTF8CHAR *const msg, int stat_, const OSUTF8CHAR *const file_, int line_\)](#)
Constructor.
- [OSSAXException \(OSRTContext *pContext, int stat_, const OSUTF8CHAR *file_, int line_\)](#)
Constructor.
- [OSSAXException \(const OSSAXException &toCopy\)](#)
Copy constructor.
- [~OSSAXException \(\)](#)
Destructor.
- const OSUTF8CHAR * [getMessage \(\)](#)
- const OSUTF8CHAR * [getSrcFileName \(\)](#)
- int [getSrcLineNum \(\)](#)
- void [printErrorInfo \(\)](#)
Prints error information, if context is set.

Protected Attributes

- const OSUTF8CHAR * [mpMsg](#)
- const OSUTF8CHAR * [mpFile](#)
- int [line](#)

7.39.1 Detailed Description

Definition at line 161 of file `rtxCppException.h`.

7.39.2 Constructor & Destructor Documentation

7.39.2.1 `OSSAXException::OSSAXException (OSRTContext * pContext, const OSUTF8CHAR *const msg)` [inline]

Constructor.

Parameters:

pContext Pointer to a context to use.

msg The error or warning message.

Definition at line 174 of file rtxCppException.h.

7.39.2.2 `OSSAXException::OSSAXException (OSRTContext * pContext, const OSUTF8CHAR *const msg, int stat_, const OSUTF8CHAR *const file_, int line_)` [inline]

Constructor.

Parameters:

pContext Pointer to a context to use.

msg The error or warning message.

stat_ The completion status to be reported.

file_ The name of source file from where exception is thrown.

line_ The line number of source file from where exception is thrown.

Definition at line 186 of file rtxCppException.h.

7.39.2.3 `OSSAXException::OSSAXException (OSRTContext * pContext, int stat_, const OSUTF8CHAR * file_, int line_)` [inline]

Constructor.

Parameters:

pContext Pointer to a context to use.

stat_ The completion status to be reported.

file_ The name of source file from where exception is thrown.

line_ The line number of source file from where exception is thrown.

Definition at line 198 of file rtxCppException.h.

7.39.2.4 `OSSAXException::OSSAXException (const OSSAXException & toCopy)` [inline]

Copy constructor.

Parameters:

toCopy The exception to be copy constructed

Definition at line 206 of file rtxCppException.h.

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

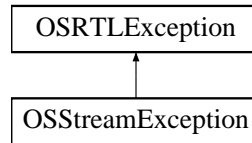
- [rtxCppException.h](#)

7.40 OSStreamException Class Reference

Exception class for streams.

```
#include <rtxCppException.h>
```

Inheritance diagram for OSStreamException::



Public Member Functions

- [OSStreamException](#) (int stat)
Constructor.
- [OSStreamException](#) (OSRTContext *pContext, int stat)
Constructor.
- [OSStreamException](#) (const [OSStreamException](#) &o)
Copy constructor.

7.40.1 Detailed Description

Exception class for streams.

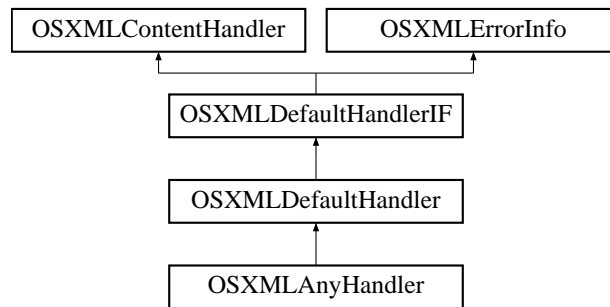
Definition at line 147 of file [rtxCppException.h](#).

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

- [rtxCppException.h](#)

7.41 OSXMLAnyHandler Class Reference

Inheritance diagram for OSXMLAnyHandler::



Public Member Functions

- virtual int [startElement](#) (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const *attrs)
Receive notification of the beginning of an element.
- virtual int [endElement](#) (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)
Receive notification of the end of an element.

7.41.1 Detailed Description

Definition at line 39 of file rtSaxCppAny.h.

7.41.2 Member Function Documentation

- 7.41.2.1** virtual int OSXMLAnyHandler::startElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const * attrs) [virtual]

Receive notification of the beginning of an element.

The Parser will invoke this method at the beginning of every element in the XML document; there will be a corresponding [endElement\(\)](#) event for every [startElement\(\)](#) event (even when the element is empty). All of the element's content will be reported, in order, before the corresponding [endElement\(\)](#) event.

Parameters:

- uri* The URI of the associated namespace for this element
- localname* The local part of the element name
- qname* The QName of this element
- attrs* The attributes name/value pairs attached to the element, if any.

See also:

[endElement](#)

Reimplemented from [OSXMLDefaultHandler](#).

7.41.2.2 `virtual int OSXMLAnyHandler::endElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)` [virtual]

Receive notification of the end of an element.

The SAX parser will invoke this method at the end of every element in the XML document; there will be a corresponding `startElement()` event for every `endElement()` event (even when the element is empty).

Parameters:

uri The URI of the associated namespace for this element

localname The local part of the element name

qname The QName of this element

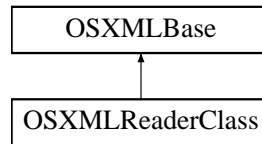
Reimplemented from [OSXMLDefaultHandler](#).

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

- [rtSaxCppAny.h](#)

7.42 OSXMLBase Class Reference

Inheritance diagram for OSXMLBase::



Protected Member Functions

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

7.42.1 Detailed Description

Definition at line 189 of file `rtSaxCppParserIF.h`.

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

- `rtSaxCppParserIF.h`

7.43 OSXMLBasePtr Class Reference

Public Member Functions

- [OSXMLBasePtr \(\)](#)
- [OSXMLBasePtr \(OSXMLBase *ptr\)](#)
- [~OSXMLBasePtr \(\)](#)
- [operator OSXMLBase * \(\) const](#)
- [OSXMLBase * operator= \(OSXMLBase *ptr\)](#)

7.43.1 Detailed Description

Definition at line 37 of file `rtSaxCppParser.h`.

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

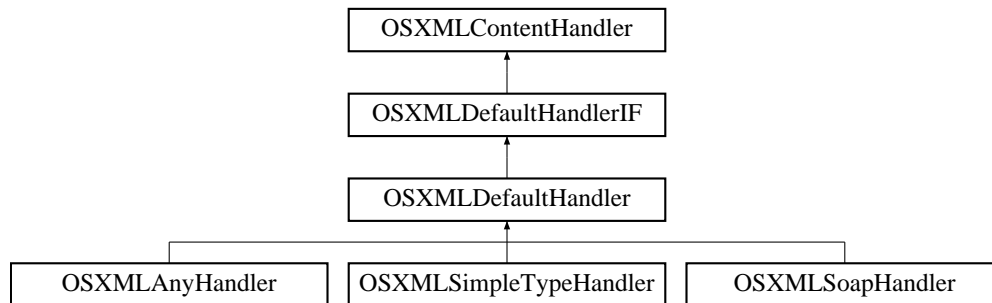
- `rtSaxCppParser.h`

7.44 OSXMLContentHandler Class Reference

Receive notification of general document events.

```
#include <rtSaxCppParserIF.h>
```

Inheritance diagram for OSXMLContentHandler::



Public Member Functions

- virtual `~OSXMLContentHandler()`

The virtual document handler interface

- virtual int `characters` (const OSUTF8CHAR *const chars, unsigned int length)=0
Receive notification of character data.
- virtual int `endElement` (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)=0
Receive notification of the end of an element.
- virtual int `startElement` (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const *attrs)=0
Receive notification of the beginning of an element.

7.44.1 Detailed Description

Receive notification of general document events.

Definition at line 115 of file rtSaxCppParserIF.h.

7.44.2 Member Function Documentation

7.44.2.1 virtual int OSXMLContentHandler::characters (const OSUTF8CHAR *const chars, unsigned int length) [pure virtual]

Receive notification of character data.

The Parser will call this method to report each chunk of character data. SAX parsers may return all contiguous character data in a single chunk, or they may split it into several chunks; however, all of the characters in any single event must come from the same external entity, so that the Locator provides useful information.

Parameters:

chars The characters from the XML document.

length The length of chars.

Implemented in [OSXMLDefaultHandler](#), [OSXMLSimpleTypeHandler](#), and [OSXMLSoapHandler](#).

7.44.2.2 `virtual int OSXMLContentHandler::endElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname) [pure virtual]`

Receive notification of the end of an element.

The SAX parser will invoke this method at the end of every element in the XML document; there will be a corresponding [startElement\(\)](#) event for every [endElement\(\)](#) event (even when the element is empty).

Parameters:

uri The URI of the associated namespace for this element

localname The local part of the element name

qname The QName of this element

Implemented in [OSXMLAnyHandler](#), [OSXMLDefaultHandler](#), [OSXMLSimpleTypeHandler](#), and [OSXMLSoapHandler](#).

7.44.2.3 `virtual int OSXMLContentHandler::startElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const * attrs) [pure virtual]`

Receive notification of the beginning of an element.

The Parser will invoke this method at the beginning of every element in the XML document; there will be a corresponding [endElement\(\)](#) event for every [startElement\(\)](#) event (even when the element is empty). All of the element's content will be reported, in order, before the corresponding [endElement\(\)](#) event.

Parameters:

uri The URI of the associated namespace for this element

localname The local part of the element name

qname The QName of this element

attrs The attributes name/value pairs attached to the element, if any.

See also:

[endElement](#)

Implemented in [OSXMLAnyHandler](#), [OSXMLDefaultHandler](#), [OSXMLSimpleTypeHandler](#), and [OSXMLSoapHandler](#).

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

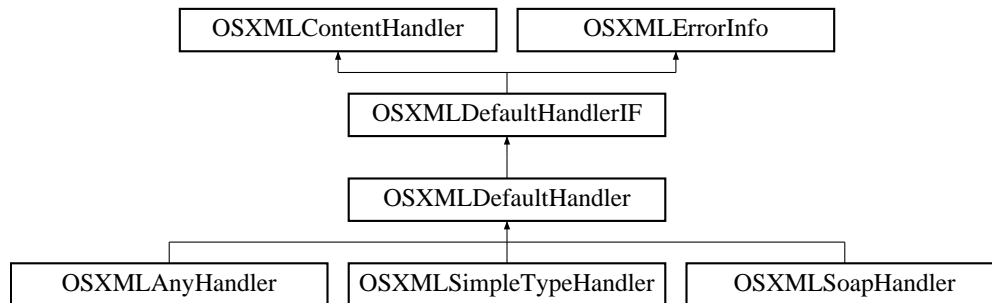
- [rtSaxCppParserIF.h](#)

7.45 OSXMLDefaultHandler Class Reference

This class is derived from the SAX class DefaultHandler base class.

```
#include <rtSaxCppParser.h>
```

Inheritance diagram for OSXMLDefaultHandler::



Public Member Functions

- [OSXMLDefaultHandler](#) ([OSRTCtPtr](#) *pContext, const [OSUTF8CHAR](#) *elemName=0, [OSINT32](#) level=0)
- virtual [~OSXMLDefaultHandler](#) ()
- virtual int [startElement](#) (const [OSUTF8CHAR](#) *const uri, const [OSUTF8CHAR](#) *const localname, const [OSUTF8CHAR](#) *const qname, const [OSUTF8CHAR](#) *const *attrs)
Receive notification of the beginning of an element.
- virtual int [characters](#) (const [OSUTF8CHAR](#) *const chars, unsigned int length)
Receive notification of character data.
- virtual int [endElement](#) (const [OSUTF8CHAR](#) *const uri, const [OSUTF8CHAR](#) *const localname, const [OSUTF8CHAR](#) *const qname)
Receive notification of the end of an element.
- [OSINT16](#) [getState](#) ()
This method returns the current state of the decoding process.
- virtual void [init](#) (int level=0)
- void [setElemName](#) (const [OSUTF8CHAR](#) *elemName)
- [OSBOOL](#) [isComplete](#) ()

Protected Attributes

- [OSRTCtPtr](#) mpContext
- const [OSUTF8CHAR](#) * mpElemName
- [OSINT32](#) mLevel
- [OSINT16](#) mStartLevel
- [OSINT16](#) mReqElemCnt
- [OSINT16](#) mCurrElemIdx
- [OSINT16](#) mState

Classes

- struct [ErrorInfo](#)

7.45.1 Detailed Description

This class is derived from the SAX class `DefaultHandler` base class.

It contains variables and methods specific to decoding XML messages. It is used to intercept standard SAX parser events, such as `startElement`, `characters`, `endElement`. This class is used as a base class for `XBinder` generated global element control classes (`<elem>_CC`).

Definition at line 60 of file `rtSaxCppParser.h`.

7.45.2 Member Function Documentation

7.45.2.1 `virtual int OSXMLDefaultHandler::startElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const * attrs)` [virtual]

Receive notification of the beginning of an element.

The Parser will invoke this method at the beginning of every element in the XML document; there will be a corresponding `endElement()` event for every `startElement()` event (even when the element is empty). All of the element's content will be reported, in order, before the corresponding `endElement()` event.

Parameters:

uri The URI of the associated namespace for this element

localname The local part of the element name

qname The QName of this element

attrs The attributes name/value pairs attached to the element, if any.

See also:

[endElement](#)

Implements [OSXMLContentHandler](#).

Reimplemented in [OSXMLAnyHandler](#), [OSXMLSimpleTypeHandler](#), and [OSXMLSoapHandler](#).

7.45.2.2 `virtual int OSXMLDefaultHandler::characters (const OSUTF8CHAR *const chars, unsigned int length)` [virtual]

Receive notification of character data.

The Parser will call this method to report each chunk of character data. SAX parsers may return all contiguous character data in a single chunk, or they may split it into several chunks; however, all of the characters in any single event must come from the same external entity, so that the Locator provides useful information.

Parameters:

chars The characters from the XML document.

length The length of chars.

Implements [OSXMLContentHandler](#).

Reimplemented in [OSXMLSimpleTypeHandler](#), and [OSXMLSoapHandler](#).

7.45.2.3 `virtual int OSXMLDefaultHandler::endElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)` [virtual]

Receive notification of the end of an element.

The SAX parser will invoke this method at the end of every element in the XML document; there will be a corresponding [startElement\(\)](#) event for every [endElement\(\)](#) event (even when the element is empty).

Parameters:

uri The URI of the associated namespace for this element

localname The local part of the element name

qname The QName of this element

Implements [OSXMLContentHandler](#).

Reimplemented in [OSXMLAnyHandler](#), [OSXMLSimpleTypeHandler](#), and [OSXMLSoapHandler](#).

7.45.2.4 `OSINT16 OSXMLDefaultHandler::getState ()` [inline]

This method returns the current state of the decoding process.

Returns:

The state of the decoding process as type `OSXMLState`. Can be `XMLINIT`, `XMLSTART`, `XMLDATA`, or `XMLEND`.

Definition at line 126 of file `rtSaxCppParser.h`.

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

- `rtSaxCppParser.h`

7.46 OSXMLDefaultHandler::ErrorInfo Struct Reference

Public Member Functions

- [ErrorInfo](#) ()

Public Attributes

- int [stat](#)
- const char * [file](#)
- int [line](#)

7.46.1 Detailed Description

Definition at line 71 of file `rtSaxCppParser.h`.

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

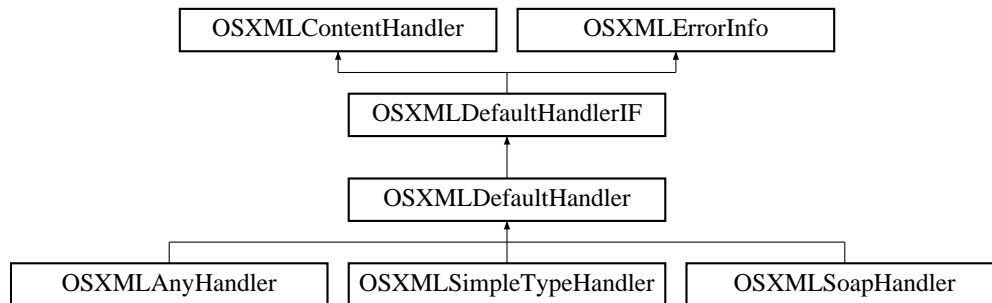
- `rtSaxCppParser.h`

7.47 OSXMLDefaultHandlerIF Class Reference

This class is derived from the SAX class DefaultHandler base class.

```
#include <rtSaxCppParserIF.h>
```

Inheritance diagram for OSXMLDefaultHandlerIF::



Public Member Functions

- virtual [~OSXMLDefaultHandlerIF](#) ()

7.47.1 Detailed Description

This class is derived from the SAX class DefaultHandler base class.

It contains variables and methods specific to decoding XML messages. It is used to intercept standard SAX parser events, such as startElement, characters, endElement. This class is used as a base class for XBinder generated global element control classes (<elem>_CC).

Definition at line 263 of file rtSaxCppParserIF.h.

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

- rtSaxCppParserIF.h

7.48 OSXMLDefaultHandlerPtr Class Reference

Public Member Functions

- [OSXMLDefaultHandlerPtr](#) ()
- [OSXMLDefaultHandlerPtr](#) ([OSXMLDefaultHandler](#) *ptr)
- [~OSXMLDefaultHandlerPtr](#) ()
- [operator OSXMLDefaultHandler *](#) ()
- [operator const OSXMLDefaultHandler *](#) () const
- [OSXMLDefaultHandler *](#) [operator →](#) () const
- [OSXMLDefaultHandler *](#) [operator=](#) ([OSXMLDefaultHandler](#) *ptr)
- [int operator==](#) (const [OSXMLDefaultHandler](#) *ptr) const
- [int operator!=](#) (const [OSXMLDefaultHandler](#) *ptr) const
- [int operator!](#) () const

Friends

- [int operator!=](#) (const void *ptr, const [OSXMLDefaultHandlerPtr](#) &ptr2)
- [int operator==](#) (const void *ptr, const [OSXMLDefaultHandlerPtr](#) &ptr2)

7.48.1 Detailed Description

Definition at line 149 of file `rtSaxCppParser.h`.

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

- `rtSaxCppParser.h`

7.49 OSXMLErrorHandler Class Reference

Public Member Functions

- virtual `~OSXMLErrorHandler ()`

The error handler interface

- virtual void `warning ()=0`
Receive notification of a warning.
- virtual void `error ()=0`
Receive notification of a recoverable error.
- virtual void `fatalError ()=0`
Receive notification of a non-recoverable error.
- virtual void `resetErrors ()=0`
Reset the Error handler object on its reuse.

7.49.1 Detailed Description

Definition at line 45 of file `rtSaxCppParserIF.h`.

7.49.2 Member Function Documentation

7.49.2.1 virtual void OSXMLErrorHandler::warning () [pure virtual]

Receive notification of a warning.

SAX parsers will use this method to report conditions that are not errors or fatal errors as defined by the XML 1.0 recommendation. The default behaviour is to take no action.

The SAX parser must continue to provide normal parsing events after invoking this method: it should still be possible for the application to process the document through to the end.

7.49.2.2 virtual void OSXMLErrorHandler::error () [pure virtual]

Receive notification of a recoverable error.

This corresponds to the definition of "error" in section 1.2 of the W3C XML 1.0 Recommendation. For example, a validating parser would use this callback to report the violation of a validity constraint. The default behaviour is to take no action.

The SAX parser must continue to provide normal parsing events after invoking this method: it should still be possible for the application to process the document through to the end. If the application cannot do so, then the parser should report a fatal error even if the XML 1.0 recommendation does not require it to do so.

7.49.2.3 virtual void OSXMLErrorHandler::fatalError () [pure virtual]

Receive notification of a non-recoverable error.

This corresponds to the definition of "fatal error" in section 1.2 of the W3C XML 1.0 Recommendation. For example, a parser would use this callback to report the violation of a well-formedness constraint.

The application must assume that the document is unusable after the parser has invoked this method, and should continue (if at all) only for the sake of collecting additional error messages: in fact, SAX parsers are free to stop reporting any other events once this method has been invoked.

7.49.2.4 virtual void OSXMLErrorHandler::resetErrors () [pure virtual]

Reset the Error handler object on its reuse.

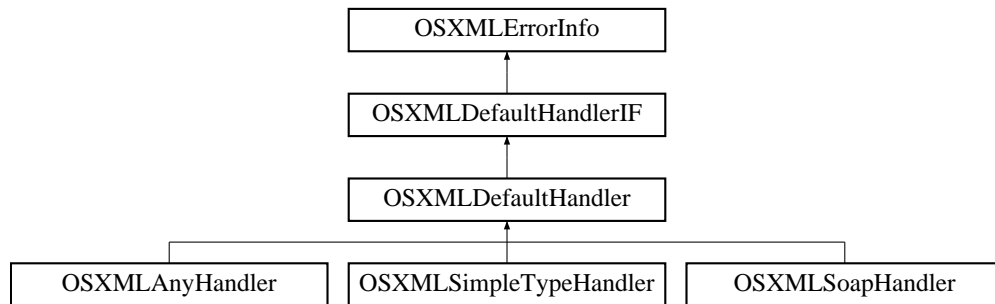
This method helps in resetting the Error handler object implementation defaults each time the Error handler is begun.

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

- rtSaxCppParserIF.h

7.50 OSXMLErrorInfo Class Reference

Inheritance diagram for OSXMLErrorInfo::



Public Member Functions

- virtual [~OSXMLErrorInfo](#) ()

7.50.1 Detailed Description

Definition at line 37 of file rtSaxCppParserIF.h.

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

- rtSaxCppParserIF.h

7.51 OSXMLNamespaceClass Class Reference

This class is used to hold an XML namespace prefix to URI mapping.

```
#include <rtXmlCppNamespace.h>
```

Public Member Functions

- [OSXMLNamespaceClass \(\)](#)
The default constructor sets the namespace prefix and URI values to empty values.
- [~OSXMLNamespaceClass \(\)](#)
The destructor deletes the prefix and uri string variables.
- [OSXMLNamespaceClass \(const OSUTF8CHAR *nsPrefix, const OSUTF8CHAR *nsURI\)](#)
The parameterized constructor sets the namespace prefix and URI values to the given values.
- [OSXMLNamespaceClass \(const OSUTF8CHAR *nsPrefix, size_t nsPrefixBytes, const OSUTF8CHAR *nsURI, size_t nsURIBytes\)](#)
The parameterized constructor sets the namespace prefix and URI values to the given values.
- [OSXMLNamespaceClass \(const OSXMLNamespaceClass &o\)](#)
The copy constructor make a deep-copy of the prefix and URI values.
- `const OSUTF8CHAR * getPrefix \(\) const`
This method is used to get the namespace prefix value.
- `const OSUTF8CHAR * getURI \(\) const`
This method is used to get the namespace URI value.
- `void setPrefix (const OSUTF8CHAR *nsPrefix)`
This method is used to set the namespace prefix value.
- `void setURI (const OSUTF8CHAR *nsURI)`
This method is used to set the namespace URI value.

7.51.1 Detailed Description

This class is used to hold an XML namespace prefix to URI mapping.

Definition at line 38 of file rtXmlCppNamespace.h.

7.51.2 Constructor & Destructor Documentation

7.51.2.1 OSXMLNamespaceClass::OSXMLNamespaceClass (const OSUTF8CHAR * nsPrefix, const OSUTF8CHAR * nsURI)

The parameterized constructor sets the namespace prefix and URI values to the given values.

A deep copy of the values is done.

Parameters:

nsPrefix Namespace prefix value.

nsURI Namespace URI value.

7.51.2.2 OSXMLNamespaceClass::OSXMLNamespaceClass (const OSUTF8CHAR * *nsPrefix*, size_t *nsPrefixBytes*, const OSUTF8CHAR * *nsURI*, size_t *nsURIBytes*)

The parameterized constructor sets the namespace prefix and URI values to the given values.

A deep copy of the values is done.

Parameters:

nsPrefix Namespace prefix value.

nsPrefixBytes Namespace prefix value size in bytes.

nsURI Namespace URI value.

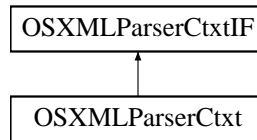
nsURIBytes Namespace URI value size in bytes.

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

- [rtXmlCppNamespace.h](#)

7.52 OSXMLParserCtxt Class Reference

Inheritance diagram for OSXMLParserCtxt::



7.52.1 Detailed Description

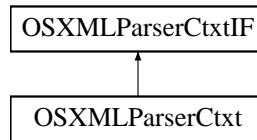
Definition at line 181 of file rtSaxCppParser.h.

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

- rtSaxCppParser.h

7.53 OSXMLParserCtxtIF Class Reference

Inheritance diagram for OSXMLParserCtxtIF::



Public Member Functions

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

7.53.1 Detailed Description

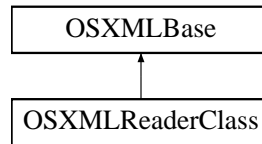
Definition at line 290 of file `rtSaxCppParserIF.h`.

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

- `rtSaxCppParserIF.h`

7.54 OSXMLReaderClass Class Reference

Inheritance diagram for OSXMLReaderClass::



Public Member Functions

- virtual int `parse (OSRTInputStreamIF &source)=0`
Parse an XML document.
- virtual int `parse (const char *const pBuffer, size_t bufSize)=0`
Parse an XML document from memory buffer.
- virtual int `parse (const char *const systemId)=0`
Parse an XML document from a system identifier (URI).

7.54.1 Detailed Description

Definition at line 201 of file rtSaxCppParserIF.h.

7.54.2 Member Function Documentation

7.54.2.1 virtual int OSXMLReaderClass::parse (OSRTInputStreamIF & source) [pure virtual]

Parse an XML document.

The application can use this method to instruct the SAX parser to begin parsing an XML document from any valid input source (a character stream, a byte stream, or a URI).

Applications may not invoke this method while a parse is in progress (they should create a new Parser instead for each additional XML document). Once a parse is complete, an application may reuse the same Parser object, possibly with a different input source.

Parameters:

source The input source for the top-level of the XML document.

7.54.2.2 virtual int OSXMLReaderClass::parse (const char *const pBuffer, size_t bufSize) [pure virtual]

Parse an XML document from memory buffer.

Parameters:

pBuffer Buffer containing the XML data to be parsed.

bufSize Buffer size, in octets.

7.54.2.3 `virtual int OSXMLReaderClass::parse (const char *const systemId)` [pure virtual]

Parse an XML document from a system identifier (URI).

This method is a shortcut for the common case of reading a document from a system identifier. It is the exact equivalent of the following:

```
parse(new URLInputSource(systemId));
```

If the system identifier is a URL, it must be fully resolved by the application before it is passed to the parser.

Parameters:

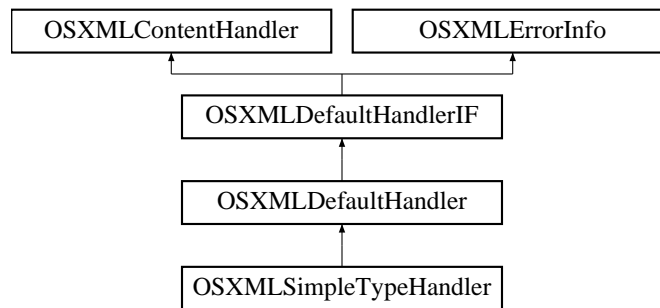
systemId The system identifier (URI).

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

- `rtSaxCppParserIF.h`

7.55 OSXMLSimpleTypeHandler Class Reference

Inheritance diagram for OSXMLSimpleTypeHandler::



Public Member Functions

- virtual int [startElement](#) (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const *attrs)
Receive notification of the beginning of an element.
- virtual int [characters](#) (const OSUTF8CHAR *const chars, unsigned int length)
Receive notification of character data.
- virtual int [endElement](#) (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)
Receive notification of the end of an element.

7.55.1 Detailed Description

Definition at line 39 of file rtSaxCppSimpleType.h.

7.55.2 Member Function Documentation

7.55.2.1 virtual int OSXMLSimpleTypeHandler::startElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const * attrs) [virtual]

Receive notification of the beginning of an element.

The Parser will invoke this method at the beginning of every element in the XML document; there will be a corresponding [endElement\(\)](#) event for every [startElement\(\)](#) event (even when the element is empty). All of the element's content will be reported, in order, before the corresponding [endElement\(\)](#) event.

Parameters:

- uri* The URI of the associated namespace for this element
- localname* The local part of the element name
- qname* The QName of this element

attrs The attributes name/value pairs attached to the element, if any.

See also:

[endElement](#)

Reimplemented from [OSXMLDefaultHandler](#).

7.55.2.2 virtual int OSXMLSimpleTypeHandler::characters (const OSUTF8CHAR *const *chars*, unsigned int *length*) [virtual]

Receive notification of character data.

The Parser will call this method to report each chunk of character data. SAX parsers may return all contiguous character data in a single chunk, or they may split it into several chunks; however, all of the characters in any single event must come from the same external entity, so that the Locator provides useful information.

Parameters:

chars The characters from the XML document.

length The length of chars.

Reimplemented from [OSXMLDefaultHandler](#).

7.55.2.3 virtual int OSXMLSimpleTypeHandler::endElement (const OSUTF8CHAR *const *uri*, const OSUTF8CHAR *const *localname*, const OSUTF8CHAR *const *qname*) [virtual]

Receive notification of the end of an element.

The SAX parser will invoke this method at the end of every element in the XML document; there will be a corresponding [startElement\(\)](#) event for every [endElement\(\)](#) event (even when the element is empty).

Parameters:

uri The URI of the associated namespace for this element

localname The local part of the element name

qname The QName of this element

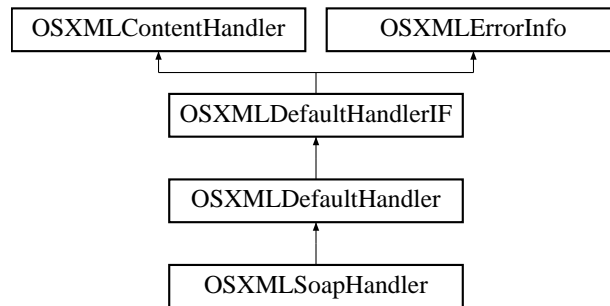
Reimplemented from [OSXMLDefaultHandler](#).

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

- [rtSaxCppSimpleType.h](#)

7.56 OSXMLSoapHandler Class Reference

Inheritance diagram for OSXMLSoapHandler::



Public Member Functions

- virtual int [startElement](#) (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const *attrs)
Receive notification of the beginning of an element.
- virtual int [characters](#) (const OSUTF8CHAR *const chars, unsigned int length)
Receive notification of character data.
- virtual int [endElement](#) (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)
Receive notification of the end of an element.

7.56.1 Detailed Description

Definition at line 40 of file rtSaxCppSoap.h.

7.56.2 Member Function Documentation

7.56.2.1 virtual int OSXMLSoapHandler::startElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const * attrs) [virtual]

Receive notification of the beginning of an element.

The Parser will invoke this method at the beginning of every element in the XML document; there will be a corresponding [endElement\(\)](#) event for every [startElement\(\)](#) event (even when the element is empty). All of the element's content will be reported, in order, before the corresponding [endElement\(\)](#) event.

Parameters:

- uri* The URI of the associated namespace for this element
- localname* The local part of the element name
- qname* The QName of this element

attrs The attributes name/value pairs attached to the element, if any.

See also:

[endElement](#)

Reimplemented from [OSXMLDefaultHandler](#).

7.56.2.2 virtual int OSXMLSoapHandler::characters (const OSUTF8CHAR *const *chars*, unsigned int *length*) [virtual]

Receive notification of character data.

The Parser will call this method to report each chunk of character data. SAX parsers may return all contiguous character data in a single chunk, or they may split it into several chunks; however, all of the characters in any single event must come from the same external entity, so that the Locator provides useful information.

Parameters:

chars The characters from the XML document.

length The length of chars.

Reimplemented from [OSXMLDefaultHandler](#).

7.56.2.3 virtual int OSXMLSoapHandler::endElement (const OSUTF8CHAR *const *uri*, const OSUTF8CHAR *const *localname*, const OSUTF8CHAR *const *qname*) [virtual]

Receive notification of the end of an element.

The SAX parser will invoke this method at the end of every element in the XML document; there will be a corresponding [startElement\(\)](#) event for every [endElement\(\)](#) event (even when the element is empty).

Parameters:

uri The URI of the associated namespace for this element

localname The local part of the element name

qname The QName of this element

Reimplemented from [OSXMLDefaultHandler](#).

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

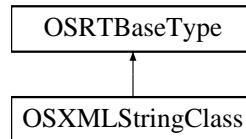
- [rtSaxCppSoap.h](#)

7.57 OSXMLStringClass Class Reference

XML string.

```
#include <rtxCppXmlString.h>
```

Inheritance diagram for OSXMLStringClass::



Public Member Functions

- [OSXMLStringClass \(\)](#)
The default constructor creates an empty string.
- [OSXMLStringClass \(const OSUTF8CHAR *strval, OSBOOL cdata_=FALSE\)](#)
This constructor initializes the string to contain the value.
- [OSXMLStringClass \(const OSUTF8CHAR *strval, size_t nbytes, OSBOOL cdata_=FALSE\)](#)
This constructor initializes the string to contain the value.
- [OSXMLStringClass \(const char *strval, OSBOOL cdata_=FALSE\)](#)
This constructor initializes the string to contain the value.
- [OSXMLStringClass \(const OSXMLSTRING &str\)](#)
Copy constructor.
- [OSXMLStringClass \(const OSXMLStringClass &str\)](#)
Copy constructor.
- [virtual ~OSXMLStringClass \(\)](#)
The destructor frees string memory if the memory ownership flag is set.
- [void appendValue \(const OSUTF8CHAR *utf8str, size_t nbytes=0\)](#)
This method copies the given string value to the end of internal string storage variable.
- [OSRTBaseType * clone \(\) const](#)
Clone method.
- [void copyValue \(const OSUTF8CHAR *utf8str, size_t nbytes=0\)](#)
This method copies the given string value to the internal string storage variable.
- [void copyValue \(const char *cstring, size_t nbytes=0\)](#)
This method copies the given string value to the internal string storage variable.
- [const char * c_str \(\) const](#)

This method returns the pointer to C null terminated string.

- virtual int `decodeXML` (OSCTXT *pctxt)

This method decodes XML content at the current stream/buffer position into this string object.

- virtual int `encodeXML` (OSRTMessageBufferIF &msgbuf, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This method encodes the data in this string object into XML content in the encode data stream.

- const OSUTF8CHAR * `getValue` () const

This method returns a pointer to the UTF-8 null terminated string.

- OSBOOL `isCDATA` () const

This method returns the value of the cdata member variable.

- void `setCDATA` (OSBOOL bvalue)

This method sets the value of the cdata member variable.

- void `print` (const char *name)

This method prints the string value to standard output.

- void `setValue` (const OSUTF8CHAR *utf8str, size_t nbytes=0)

This method sets the string value to the given string.

- void `setValue` (const char *cstring, size_t nbytes=0)

This method sets the string value to the given string.

- void `setValue` (OSRTMemBuf &membuf)

This method sets the string value to the value of the data in the given memory buffer object.

- OSXMLStringClass & `operator=` (const OSXMLStringClass &original)

Assignment operator.

- OSXMLStringClass & `operator=` (const char *original)

Assignment operator for C strings.

- OSXMLStringClass & `operator=` (const OSUTF8CHAR *original)

Assignment operator for C UTF-8 strings.

- `operator const char *` () const

String to C const char type conversion operator.*

- `operator const OSUTF8CHAR *` () const

String to C const OSUTF8CHAR type conversion operator.*

- int `length` ()

This method returns the number of characters.

- int `size` ()

This method returns the number of bytes.

Protected Member Functions

- void `newString` (const char *pString, size_t nbytes=0)
- void `xstrcat` (OSUTF8CHAR *dststr, const OSUTF8CHAR *srcstr)

Protected Attributes

- OSUTF8CHAR * `value`
- OSBOOL `CDATA`
- size_t `mCapacityIncrement`
- size_t `mCurrentBufferSize`
- size_t `mCurrentStringSize`

7.57.1 Detailed Description

XML string.

This is the base class for generated C++ data type classes for XSD string types (string, token, NMTOKEN, etc.).

Definition at line 46 of file `rtxCppXmlString.h`.

7.57.2 Constructor & Destructor Documentation

7.57.2.1 OSXMLStringClass::OSXMLStringClass (const OSUTF8CHAR * *strval*, OSBOOL *CDATA_* = FALSE)

This constructor initializes the string to contain the value.

A deep-copy of the given value is done.

Parameters:

strval - String value

CDATA_ - Should string be encoded as a CDATA section?

7.57.2.2 OSXMLStringClass::OSXMLStringClass (const OSUTF8CHAR * *strval*, size_t *nbytes*, OSBOOL *CDATA_* = FALSE)

This constructor initializes the string to contain the value.

It copies up to the given number of bytes from the source string. A deep-copy of the given value is done.

Parameters:

strval - String value

nbytes - Number of bytes to copy from source string

CDATA_ - Should string be encoded as a CDATA section?

7.57.2.3 OSXMLStringClass::OSXMLStringClass (const char * *strval*, OSBOOL *CDATA_* = FALSE)

This constructor initializes the string to contain the value.

A deep-copy of the given value is done.

Parameters:

strval - String value

CDATA_ - Should string be encoded as a CDATA section?

7.57.2.4 OSXMLStringClass::OSXMLStringClass (const OSXMLSTRING & *str*)

Copy constructor.

Parameters:

str - C XML string structure.

7.57.2.5 OSXMLStringClass::OSXMLStringClass (const OSXMLStringClass & *str*)

Copy constructor.

Parameters:

str - C++ XML string class.

7.57.3 Member Function Documentation

7.57.3.1 void OSXMLStringClass::appendValue (const OSUTF8CHAR * *utf8str*, size_t *nbytes* = 0)

This method copies the given string value to the end of internal string storage variable.

A deep-copy of the given value is done; the class will delete this memory when the object is deleted.

Parameters:

utf8str - C null-terminated string.

nbytes - length of *utf8str* in bytes.

7.57.3.2 OSRTBaseType* OSXMLStringClass::clone () const [inline, virtual]

Clone method.

Creates a copied instance and returns pointer to [OSRTBaseType](#).

Reimplemented from [OSRTBaseType](#).

Definition at line 144 of file `rtxCppXmlString.h`.

7.57.3.3 void OSXMLStringClass::copyValue (const OSUTF8CHAR * utf8str, size_t nbytes = 0)

This method copies the given string value to the internal string storage variable.

A deep-copy of the given value is done; the class will delete this memory when the object is deleted.

Parameters:

utf8str - C null-terminated string.

nbytes - length of utf8str in bytes.

7.57.3.4 void OSXMLStringClass::copyValue (const char * cstring, size_t nbytes = 0) [inline]

This method copies the given string value to the internal string storage variable.

A deep-copy of the given value is done; the class will delete this memory when the object is deleted.

Parameters:

cstring - C null-terminated string.

nbytes - length of cstring in bytes.

Definition at line 166 of file rtxCppXmlString.h.

7.57.3.5 virtual int OSXMLStringClass::decodeXML (OSCTXT * ptxt) [virtual]

This method decodes XML content at the current stream/buffer position into this string object.

This method is normally overridden by a decodeXML method in a generated class.

Parameters:

ptxt Pointer to context block structure.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

7.57.3.6 virtual int OSXMLStringClass::encodeXML (OSRTMessageBufferIF & msgbuf, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS) [virtual]

This method encodes the data in this string object into XML content in the encode data stream.

This method is normally overridden by an encodeXML method in a generated class.

Parameters:

msgbuf Message buffer or stream object reference.

elemName XML element name that should be added to encoded fragment.

pNS Pointer to namespace structure.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

7.57.3.7 OSBOOL OSXMLStringClass::isCDATA () const [inline]

This method returns the value of the cdata member variable.

This indicates if this string should be encoded as a CDATA section in an XML document.

Returns:

- True if string is to be encoded as CDATA section

Definition at line 219 of file rtxCppXmlString.h.

7.57.3.8 void OSXMLStringClass::setCDATA (OSBOOL *bvalue*) [inline]

This method sets the value of the cdata member variable.

This indicates if this string should be encoded as a CDATA section in an XML document.

Parameters:

bvalue - Boolean value.

Definition at line 228 of file rtxCppXmlString.h.

7.57.3.9 void OSXMLStringClass::print (const char * *name*) [inline]

This method prints the string value to standard output.

Parameters:

name - Name of generated string variable.

Definition at line 235 of file rtxCppXmlString.h.

7.57.3.10 void OSXMLStringClass::setValue (const OSUTF8CHAR * *utf8str*, size_t *nbytes* = 0)

This method sets the string value to the given string.

A deep-copy of the given value is done.

Parameters:

utf8str - UTF8 null-terminated string.

nbytes - Number of bytes to copy from the source string. If zero, bytes are copied up to the null-terminator.

7.57.3.11 void OSXMLStringClass::setValue (const char * *cstring*, size_t *nbytes* = 0) [inline]

This method sets the string value to the given string.

A deep-copy of the given value is done.

Parameters:

cstring - C null-terminated string.

nbytes - Number of bytes to copy from the source string. If zero, bytes are copied up to the null-terminator.

Definition at line 255 of file rtxCppXmlString.h.

7.57.3.12 void OSXMLStringClass::setValue (OSRTMemBuf & *membuf*) [inline]

This method sets the string value to the value of the data in the given memory buffer object.

A deep-copy of the value is done.

Parameters:

membuf - Reference to a memory buffer object.

Definition at line 266 of file rtxCppXmlString.h.

References OSRTMemBuf::getData(), and OSRTMemBuf::getDataLen().

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

- [rtxCppXmlString.h](#)

7.58 OSXMLStrListHandler Class Reference

[OSXMLStrListHandler](#).

```
#include <rtSaxCppStrList.h>
```

Static Public Member Functions

- static int [match](#) (OSCTXT *)

7.58.1 Detailed Description

[OSXMLStrListHandler](#).

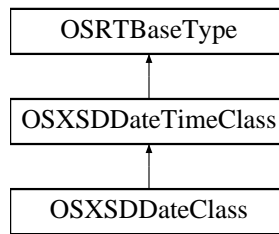
Definition at line 41 of file `rtSaxCppStrList.h`.

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

- [rtSaxCppStrList.h](#)

7.59 OSXSDDateClass Class Reference

Inheritance diagram for OSXSDDateClass::



Public Member Functions

- [OSXSDDateClass \(\)](#)
OSXSDDateClass(): This is a default constructor, sets the date and time fields to zero.
- [OSXSDDateClass \(const OSUTF8CHAR *dtString\)](#)
OSXSDDateClass(string): This is a parameterized constructor, parses string and sets the date and time fields.
- [OSXSDDateClass \(OSINT32 year, OSUINT8 mon, OSUINT8 day, OSUINT8 hour, OSUINT8 min, OSREAL sec, OSBOOL tz_flag, OSINT32 tzo\)](#)
OSXSDDateTimeClass(param1,param2,.
- [OSXSDDateClass \(const OSXSDDateTimeClass &dt\)](#)
OSXSDDateTimeClass(const OSXSDDateTimeClass& dt): This is a copy constructor, sets the date and time fields to that of equal to supplied OSXSDDateTimeClass type object.
- [OSXSDDateClass \(const OSXSDDateTime &dt\)](#)
OSXSDDateTimeClass(const OSXSDDateTime& dt): This is a copy constructor, sets the date and time fields to that of equal to supplied OSXSDDateTimeClass type object.
- [OSRTBaseType * clone \(\) const](#)
Clone method.
- virtual int [parseString](#) (const OSUTF8CHAR *dtString)
parseString: This method parses the date string and sets the date and time value
- void [print](#) (const char *name)
This method prints the datetime value to standard output.
- virtual const OSUTF8CHAR * [toString](#) (OSUTF8CHAR *buffer, size_t bufsize)
This method sets the date and time fields to the values of current date and time.

7.59.1 Detailed Description

Definition at line 211 of file rtxCppDateTime.h.

7.59.2 Constructor & Destructor Documentation

7.59.2.1 OSXSDDateClass::OSXSDDateClass (OSINT32 *year*, OSUINT8 *mon*, OSUINT8 *day*, OSUINT8 *hour*, OSUINT8 *min*, OSREAL *sec*, OSBOOL *tz_flag*, OSINT32 *tzo*) [inline]

[OSXSDDateTimeClass](#)(param1,param2,.

.): This is a parameterized constructor, sets the date and time elements with the supplied parameter values .

Parameters:

year OSINT32 sets year field
mon OSUINT8 sets month field
day OSUINT8 sets day field
hour OSUINT8 sets hour field
min OSUINT8 sets minute filed
sec OSREAL sets second field
tz_flag OSBOOL sets timezone flag
tzo OSINT32 sets timezone value

Definition at line 239 of file rtxCppDateTime.h.

7.59.2.2 OSXSDDateClass::OSXSDDateClass (const [OSXSDDateTimeClass](#) & *dt*) [inline]

[OSXSDDateTimeClass](#)(const [OSXSDDateTimeClass](#)& dt): This is a copy constructor, sets the date and time fields to that of equal to supplied [OSXSDDateTimeClass](#) type object.

Parameters:

dt [OSXSDDateTimeClass](#) type object.

Definition at line 250 of file rtxCppDateTime.h.

7.59.2.3 OSXSDDateClass::OSXSDDateClass (const [OSXSDDateTime](#) & *dt*) [inline]

[OSXSDDateTimeClass](#)(const [OSXSDDateTime](#)& dt): This is a copy constructor, sets the date and time fields to that of equal to supplied [OSXSDDateTimeClass](#) type object.

Parameters:

dt [OSXSDDateTimeClass](#) type object.

Definition at line 259 of file rtxCppDateTime.h.

7.59.3 Member Function Documentation

7.59.3.1 [OSRTBaseType](#)* OSXSDDateClass::clone () const [inline, virtual]

Clone method.

Creates a copied instance and returns pointer to [OSRTBaseType](#).

Reimplemented from [OSXSDDateTimeClass](#).

Definition at line 266 of file rtxCppDateTime.h.

7.59.3.2 `virtual int OSXSDDateClass::parseString (const OSUTF8CHAR * dtString) [virtual]`

`parseString`: This method parses the date string and sets the date and time value

Parameters:

dtString `const OSUTF8CHAR*` - Date and time string

Returns:

Completion status of operation:

- 0(RT_OK) = success,
- negative return value is error

Reimplemented from [OSXSDDateTimeClass](#).

7.59.3.3 `void OSXSDDateClass::print (const char * name)`

This method prints the datetime value to standard output.

Parameters:

name - Name of generated string variable.

Reimplemented from [OSXSDDateTimeClass](#).

7.59.3.4 `virtual const OSUTF8CHAR* OSXSDDateClass::toString (OSUTF8CHAR * buffer, size_t bufsize) [virtual]`

This method sets the date and time fields to the values of current date and time.

Parameters:

buffer `OSUTF8CHAR*` - pointer to Date and time string

bufsize `size_t` specifies buffer size

Returns:

`const OSUTF8CHAR*` returns the datetime string

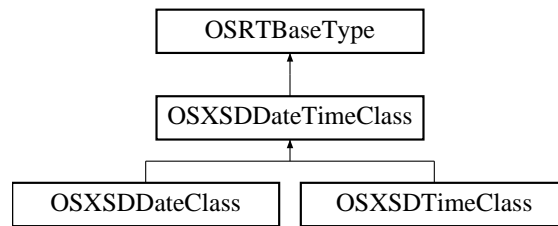
Reimplemented from [OSXSDDateTimeClass](#).

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

- [rtxCppDateTime.h](#)

7.60 OSXSDDateTimeClass Class Reference

Inheritance diagram for OSXSDDateTimeClass::



Public Member Functions

- [OSXSDDateTimeClass \(\)](#)
OSXSDDateTimeClass(): This is a default constructor, sets the date and time fields to zero.
- [OSXSDDateTimeClass \(const OSUTF8CHAR *dtString\)](#)
OSXSDDateTimeClass(string): This is a parameterized constructor, parses string and sets the date and time fields.
- [OSXSDDateTimeClass \(OSINT32 year_, OSUINT8 mon_, OSUINT8 day_, OSUINT8 hour_, OSUINT8 min_, OSREAL sec_, OSBOOL tz_flag_, OSINT32 tzo_\)](#)
OSXSDDateTimeClass(param1,param2,.
- [OSXSDDateTimeClass \(const OSXSDDateTimeClass &dt\)](#)
OSXSDDateTimeClass(const OSXSDDateTimeClass& dt): This is a copy constructor, sets the date and time fields to that of equal to supplied OSXSDDateTimeClass type object.
- [OSXSDDateTimeClass \(const OSXSDDateTime &dt\)](#)
OSXSDDateTimeClass(const OSXSDDateTime& dt): This is a copy constructor, sets the date and time fields to that of equal to supplied OSXSDDateTimeClass type object.
- [OSRTBaseType * clone \(\) const](#)
Clone method.
- [int getTime \(time_t &timeMs\)](#)
This method converts the datetime value to a calendar time encoded as a value of type time_t.
- [virtual int parseString \(const OSUTF8CHAR *dtString\)](#)
parseString: This method parses the datetime string and sets the date and time value
- [void print \(const char *name\)](#)
This method prints the datetime value to standard output.
- [int setCurrent \(\)](#)
setCurrent: This method sets the date and time fields to the values of current date and time.
- [int setCurrentTz \(\)](#)
setCurrentTz: This method sets the date, time and timezone fields to the values of current date and time.

- int [setDateTime](#) (struct tm *time)
This method converts a structure of type tm to the datetime value.
- int [setLocalTime](#) (time_t timeMs)
This method sets local date and time to the datetime value.
- int [setUtcTime](#) (time_t timeMs)
This method sets UTC date and time to the datetime value.
- void [setValue](#) (const OSUTF8CHAR *utf8str)
This method sets the string value to the given datetime instance.
- void [setValue](#) (const OSXSDDateTime &dt)
This method sets the datetime value to the given datetime instance.
- const OSUTF8CHAR * [toString](#) ()
toString: This method sets the date and time fields to the values of current date and time.
- virtual const OSUTF8CHAR * [toString](#) (OSUTF8CHAR *buffer, size_t bufsize)
This method sets the date and time fields to the values of current date and time.
- [OSXSDDateTimeClass](#) & [operator=](#) (const [OSXSDDateTimeClass](#) &orig)
Assignment operator.

7.60.1 Detailed Description

Definition at line 34 of file rtxCppDateTime.h.

7.60.2 Constructor & Destructor Documentation

7.60.2.1 OSXSDDateTimeClass::OSXSDDateTimeClass (OSINT32 year_, OSUINT8 mon_, OSUINT8 day_, OSUINT8 hour_, OSUINT8 min_, OSREAL sec_, OSBOOL tz_flag_, OSINT32 tzo_)

[OSXSDDateTimeClass](#)(param1,param2,.

.): This is a parameterized constructor, sets the date and time elements with the supplied parameter values .

Parameters:

year_ OSINT32 sets year field
mon_ OSUINT8 sets month field
day_ OSUINT8 sets day field
hour_ OSUINT8 sets hour field
min_ OSUINT8 sets minute filed
sec_ OSREAL sets second field
tz_flag_ OSBOOL sets timezone flag
tzo_ OSINT32 sets timezone value

7.60.2.2 OSXSDDateTimeClass::OSXSDDateTimeClass (const OSXSDDateTimeClass & dt)

[OSXSDDateTimeClass\(const OSXSDDateTimeClass& dt\)](#): This is a copy constructor, sets the date and time fields to that of equal to supplied [OSXSDDateTimeClass](#) type object.

Parameters:

dt [OSXSDDateTimeClass](#) type object.

7.60.2.3 OSXSDDateTimeClass::OSXSDDateTimeClass (const OSXSDDateTime & dt)

[OSXSDDateTimeClass\(const OSXSDDateTime& dt\)](#): This is a copy constructor, sets the date and time fields to that of equal to supplied [OSXSDDateTimeClass](#) type object.

Parameters:

dt [OSXSDDateTimeClass](#) type object.

7.60.3 Member Function Documentation

7.60.3.1 OSRTBaseType* OSXSDDateTimeClass::clone () const [inline, virtual]

Clone method.

Creates a copied instance and returns pointer to [OSRTBaseType](#).

Reimplemented from [OSRTBaseType](#).

Reimplemented in [OSXSDDateClass](#), and [OSXSDDateTimeClass](#).

Definition at line 85 of file [rtxCppDateTime.h](#).

7.60.3.2 int OSXSDDateTimeClass::getTime (time_t & timeMs)

This method converts the datetime value to a calendar time encoded as a value of type `time_t`.

Parameters:

timeMs A pointer to `time_t` value to be set.

Returns:

Completion status of operation:

- 0(RT_OK) = success,
- negative return value is error.

7.60.3.3 virtual int OSXSDDateTimeClass::parseString (const OSUTF8CHAR * dtString) [virtual]

`parseString`: This method parses the datetime string and sets the date and time value

Parameters:

dtString `const OSUTF8CHAR*` - Date and time string

Returns:

Completion status of operation:

- 0(RT_OK) = success,
- negative return value is error

Reimplemented in [OSXSDDateClass](#), and [OSXSDDateTimeClass](#).

7.60.3.4 void OSXSDDateTimeClass::print (const char * name)

This method prints the datetime value to standard output.

Parameters:

name - Name of generated string variable.

Reimplemented in [OSXSDDateClass](#), and [OSXSDDateTimeClass](#).

7.60.3.5 int OSXSDDateTimeClass::setCurrent ()

setCurrent: This method sets the date and time fields to the values of current date and time.

Returns:

Completion status of operation:

- 0(RT_OK) = success,
- negative return value is error

7.60.3.6 int OSXSDDateTimeClass::setCurrentTz ()

setCurrentTz: This method sets the date, time and timezone fields to the values of current date and time.

Returns:

Completion status of operation:

- 0(RT_OK) = success,
- negative return value is error

7.60.3.7 int OSXSDDateTimeClass::setDateTime (struct tm * time)

This method converts a structure of type tm to the datetime value.

Parameters:

time A pointer to tm structure to be converted.

Returns:

Completion status of operation:

- 0(RT_OK) = success,
- negative return value is error.

7.60.3.8 int OSXSDDateTimeClass::setLocalTime (time_t *timeMs*)

This method sets local date and time to the datetime value.

Parameters:

timeMs A calendar time encoded as a value of type time_t.

Returns:

Completion status of operation:

- 0(RT_OK) = success,
- negative return value is error.

7.60.3.9 int OSXSDDateTimeClass::setUtcTime (time_t *timeMs*)

This method sets UTC date and time to the datetime value.

Parameters:

timeMs A calendar time encoded as a value of type time_t. The time is represented as seconds elapsed since midnight (00:00:00), January 1, 1970, coordinated universal time (UTC).

Returns:

Completion status of operation:

- 0(RT_OK) = success,
- negative return value is error.

7.60.3.10 void OSXSDDateTimeClass::setValue (const OSUTF8CHAR * *utf8str*)

This method sets the string value to the given datetime instance.

Parameters:

utf8str - C null-terminated string.

7.60.3.11 void OSXSDDateTimeClass::setValue (const OSXSDDateTime & *dt*)

This method sets the datetime value to the given datetime instance.

Parameters:

dt - OSXSDDateTimeClass type object.

7.60.3.12 const OSUTF8CHAR* OSXSDDateTimeClass::toString ()

toString: This method sets the date and time fields to the values of current date and time.

Returns:

const OSUTF8CHAR* pointer returns the datetime string

7.60.3.13 `virtual const OSUTF8CHAR* OSXSDDateTimeClass::toString (OSUTF8CHAR * buffer, size_t bufsize)` [virtual]

This method sets the date and time fields to the values of current date and time.

Parameters:

buffer OSUTF8CHAR* - pointer to Date and time string

bufsize size_t specifies buffer size

Returns:

const OSUTF8CHAR* returns the datetime string

Reimplemented in [OSXSDDateClass](#), and [OSXSDDTimeClass](#).

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

- [rtxCppDateTime.h](#)

7.61 OSXSDGlobalElement Class Reference

XSD global element base class.

```
#include <rtXmlCppXSDElement.h>
```

Public Member Functions

- [OSXSDGlobalElement](#) ([OSRTMessageBufferIF](#) &msgBuf)
This constructor sets the internal message buffer pointer to point at the given message buffer or stream object.
- [OSXSDGlobalElement](#) (const [OSXSDGlobalElement](#) &o)
The copy constructor sets the internal message buffer pointer and context to point at the message buffer and context from the original OSCType object.
- virtual [~OSXSDGlobalElement](#) ()
The virtual destructor does nothing.
- int [decode](#) ()
The decode method decodes the message described by the encapsulated message buffer object.
- virtual int [decodeFrom](#) ([OSRTMessageBufferIF](#) &)
The decodeFrom method decodes a message from the given message buffer or stream argument.
- int [encode](#) ()
The encode method encodes a message using the encoding rules specified by the derived message buffer object.
- virtual int [encodeTo](#) ([OSRTMessageBufferIF](#) &)
The encodeTo method encodes a message into the given message buffer or stream argument.
- [OSCTXT](#) * [getCtxtPtr](#) ()
The getCtxtPtr method returns the underlying C runtime context.
- void * [memAlloc](#) (size_t numocts)
The memAlloc method allocates memory using the C runtime memory management functions.
- void [memFreePtr](#) (void *ptr)
The memFreePtr method frees the memory at a specific location.
- void [setDefaultNamespace](#) (const [OSUTF8CHAR](#) *uri)
The setDefaultNamespace method sets the default namespace for the element to the given value.
- void [setDiag](#) ([OSBOOL](#) value=TRUE)
The setDiag method turns diagnostic tracing on or off.
- void [setNamespace](#) (const [OSUTF8CHAR](#) *prefix, const [OSUTF8CHAR](#) *uri)
The setNamespace method adds or modifies the namespace with the given URI in the namespace list to contain the given prefix.
- void [setXSIType](#) (const [OSUTF8CHAR](#) *typeName)

The setXSIType method sets a type name to be used in the xsi:type attribute in the top-level module element declaration.

- int `validate` ()

The validate method validates the message described by the encapsulated message buffer object.

- virtual int `validateFrom` (OSRTMessageBufferIF &)

The validateFrom method validates a message from the given message buffer or stream argument.

Protected Member Functions

- OSXSDGlobalElement ()

The default constructor sets the message pointer member variable to NULL and creates a new context object.

- OSXSDGlobalElement (OSRTContext &ctxt)

This constructor sets the message pointer member variable to NULL and initializes the context object to point at the given context value.

- void `setMsgBuf` (OSRTMessageBufferIF &msgBuf)

The setMsgBuf method is used to set the internal message buffer pointer to point at the given message buffer or stream object.

Protected Attributes

- OSRTCtxtPtr `mpContext`

The mpContext member variable holds a reference-counted C runtime variable.

- OSRTMessageBufferIF * `mpMsgBuf`

The mpMsgBuf member variable is a pointer to a derived message buffer or stream class that will manage the message being encoded or decoded.

7.61.1 Detailed Description

XSD global element base class.

This is the main base class for all generated global element control classes. It holds a variable of a generated data type as well as the associated message buffer or stream class to which a message will be encoded or from which a message will be decoded.

Definition at line 44 of file rtXmlCppXSDElement.h.

7.61.2 Constructor & Destructor Documentation

7.61.2.1 OSXSDGlobalElement::OSXSDGlobalElement (OSRTContext & ctxt) [inline, protected]

This constructor sets the message pointer member variable to NULL and initializes the context object to point at the given context value.

Parameters:

ctxt - Reference to a context object.

Definition at line 72 of file rtXmlCppXSDElement.h.

7.61.2.2 OSXSDGlobalElement::OSXSDGlobalElement (OSRTMessageBufferIF & msgBuf) [inline]

This constructor sets the internal message buffer pointer to point at the given message buffer or stream object.

The context is set to point at the context contained within the message buffer object. Thus, the message buffer and control class object share the context. It will not be released until both objects are destroyed.

Parameters:

msgBuf - Reference to a message buffer or stream object.

Definition at line 92 of file rtXmlCppXSDElement.h.

References OSRTCtxtHolderIF::getContext().

7.61.2.3 OSXSDGlobalElement::OSXSDGlobalElement (const OSXSDGlobalElement & o) [inline]

The copy constructor sets the internal message buffer pointer and context to point at the message buffer and context from the original OSCType object.

Parameters:

o - Reference to a global element object.

Definition at line 103 of file rtXmlCppXSDElement.h.

7.61.2.4 virtual OSXSDGlobalElement::~~OSXSDGlobalElement () [inline, virtual]

The virtual destructor does nothing.

It is overridden by derived versions of this class.

Definition at line 110 of file rtXmlCppXSDElement.h.

7.61.3 Member Function Documentation

7.61.3.1 void OSXSDGlobalElement::setMsgBuf (OSRTMessageBufferIF & msgBuf) [protected]

The setMsgBuf method is used to set the internal message buffer pointer to point at the given message buffer or stream object.

Parameters:

msgBuf - Reference to a message buffer or stream object.

7.61.3.2 `virtual int OSXSDGlobalElement::decodeFrom (OSRTMessageBufferIF &) [inline, virtual]`

The `decodeFrom` method decodes a message from the given message buffer or stream argument.

Parameters:

msgBuf - Message buffer or stream containing message to decode.

Definition at line 125 of file `rtXmlCppXSDElement.h`.

7.61.3.3 `virtual int OSXSDGlobalElement::encodeTo (OSRTMessageBufferIF &) [inline, virtual]`

The `encodeTo` method encodes a message into the given message buffer or stream argument.

Parameters:

msgBuf - Message buffer or stream to which the message is to be encoded.

Definition at line 140 of file `rtXmlCppXSDElement.h`.

7.61.3.4 `OSCTXT* OSXSDGlobalElement::getCtxtPtr () [inline]`

The `getCtxtPtr` method returns the underlying C runtime context.

This context can be used in calls to C runtime functions.

Definition at line 146 of file `rtXmlCppXSDElement.h`.

7.61.3.5 `void* OSXSDGlobalElement::memAlloc (size_t numocts) [inline]`

The `memAlloc` method allocates memory using the C runtime memory management functions.

The memory is tracked in the underlying context structure. When both this `OSXSDGlobalElement` derived control class object and the message buffer object are destroyed, this memory will be freed.

Parameters:

numocts - Number of bytes of memory to allocate

Definition at line 159 of file `rtXmlCppXSDElement.h`.

7.61.3.6 `void OSXSDGlobalElement::memFreePtr (void * ptr) [inline]`

The `memFreePtr` method frees the memory at a specific location.

This memory must have been allocated using the `memAlloc` method described earlier.

Parameters:

ptr - Pointer to a block of memory allocated with `memAlloc`

Definition at line 187 of file `rtXmlCppXSDElement.h`.

7.61.3.7 void OSXSDGlobalElement::setDefaultNamespace (const OSUTF8CHAR * uri) [inline]

The setDefaultNamespace method sets the default namespace for the element to the given value.

Parameters:

uri - Default namespace URI

Definition at line 197 of file rtXmlCxxXSDElement.h.

7.61.3.8 void OSXSDGlobalElement::setDiag (OSBOOL value = TRUE) [inline]

The setDiag method turns diagnostic tracing on or off.

Parameters:

value - Boolean on/off value (default = on)

Definition at line 206 of file rtXmlCxxXSDElement.h.

7.61.3.9 void OSXSDGlobalElement::setNamespace (const OSUTF8CHAR * prefix, const OSUTF8CHAR * uri) [inline]

The setNamespace method adds or modifies the namespace with the given URI in the namespace list to contain the given prefix.

Parameters:

prefix - Namespace prefix

uri - Namespace URI

Definition at line 217 of file rtXmlCxxXSDElement.h.

7.61.3.10 void OSXSDGlobalElement::setXSIType (const OSUTF8CHAR * typeName) [inline]

The setXSIType method sets a type name to be used in the xsi:type attribute in the top-level module element declaration.

Parameters:

typeName - XSI type name

Definition at line 227 of file rtXmlCxxXSDElement.h.

7.61.3.11 virtual int OSXSDGlobalElement::validateFrom (OSRTMessageBufferIF &) [inline, virtual]

The validateFrom method validates a message from the given message buffer or stream argument.

Parameters:

msgBuf - Message buffer or stream containing message to validate.

Definition at line 244 of file rtXmlCxxXSDElement.h.

7.61.4 Member Data Documentation

7.61.4.1 [OSRTCtxtPtr OSXSDGlobalElement::mpContext](#) [protected]

The mpContext member variable holds a reference-counted C runtime variable.

This context is used in calls to all C run-time functions. The context pointed at by this smart-pointer object is shared with the message buffer object contained within this class.

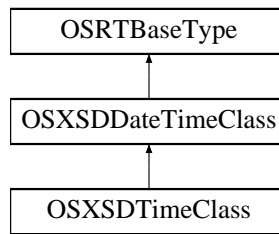
Definition at line 52 of file rtXmlCppXSDElement.h.

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

- [rtXmlCppXSDElement.h](#)

7.62 OSXSDDateTimeClass Class Reference

Inheritance diagram for OSXSDDateTimeClass::



Public Member Functions

- [OSXSDDateTimeClass \(\)](#)
OSXSDDateTimeClass(): This is a default constructor, sets the date and time fields to zero.
- [OSXSDDateTimeClass \(const OSUTF8CHAR *dtString\)](#)
OSXSDDateTimeClass(string): This is a parameterized constructor, parses string and sets the date and time fields.
- [OSXSDDateTimeClass \(OSINT32 year, OSUINT8 mon, OSUINT8 day, OSUINT8 hour, OSUINT8 min, OSREAL sec, OSBOOL tz_flag, OSINT32 tzo\)](#)
OSXSDDateTimeClass(param1,param2,.
- [OSXSDDateTimeClass \(const OSXSDDateTimeClass &dt\)](#)
OSXSDDateTimeClass(const OSXSDDateTimeClass& dt): This is a copy constructor, sets the date and time fields to that of equal to supplied OSXSDDateTimeClass type object.
- [OSXSDDateTimeClass \(const OSXSDDateTime &dt\)](#)
OSXSDDateTimeClass(const OSXSDDateTime& dt): This is a copy constructor, sets the date and time fields to that of equal to supplied OSXSDDateTimeClass type object.
- [OSRTBaseType * clone \(\) const](#)
Clone method.
- [virtual int parseString \(const OSUTF8CHAR *dtString\)](#)
parseString: This method parses the time string and sets the date and time value
- [void print \(const char *name\)](#)
This method prints the datetime value to standard output.
- [virtual const OSUTF8CHAR * toString \(OSUTF8CHAR *buffer, size_t bufsize\)](#)
This method sets the date and time fields to the values of current date and time.

7.62.1 Detailed Description

Definition at line 297 of file `rtxCppDateTime.h`.

7.62.2 Constructor & Destructor Documentation

7.62.2.1 OSXSDDateTimeClass::OSXSDDateTimeClass (OSINT32 *year*, OSUINT8 *mon*, OSUINT8 *day*, OSUINT8 *hour*, OSUINT8 *min*, OSREAL *sec*, OSBOOL *tz_flag*, OSINT32 *tzo*) [inline]

[OSXSDDateTimeClass](#)(param1,param2,.

.): This is a parameterized constructor, sets the date and time elements with the supplied parameter values .

Parameters:

year OSINT32 sets year field
mon OSUINT8 sets month field
day OSUINT8 sets day field
hour OSUINT8 sets hour field
min OSUINT8 sets minute filed
sec OSREAL sets second field
tz_flag OSBOOL sets timezone flag
tzo OSINT32 sets timezone value

Definition at line 325 of file rtxCppDateTime.h.

7.62.2.2 OSXSDDateTimeClass::OSXSDDateTimeClass (const [OSXSDDateTimeClass](#) & *dt*) [inline]

[OSXSDDateTimeClass](#)(const [OSXSDDateTimeClass](#)& *dt*): This is a copy constructor, sets the date and time fields to that of equal to supplied [OSXSDDateTimeClass](#) type object.

Parameters:

dt [OSXSDDateTimeClass](#) type object.

Definition at line 336 of file rtxCppDateTime.h.

7.62.2.3 OSXSDDateTimeClass::OSXSDDateTimeClass (const [OSXSDDateTime](#) & *dt*) [inline]

[OSXSDDateTimeClass](#)(const [OSXSDDateTime](#)& *dt*): This is a copy constructor, sets the date and time fields to that of equal to supplied [OSXSDDateTimeClass](#) type object.

Parameters:

dt [OSXSDDateTimeClass](#) type object.

Definition at line 345 of file rtxCppDateTime.h.

7.62.3 Member Function Documentation

7.62.3.1 [OSRTBaseType](#)* OSXSDDateTimeClass::clone () const [inline, virtual]

Clone method.

Creates a copied instance and returns pointer to [OSRTBaseType](#).

Reimplemented from [OSXSDDateTimeClass](#).

Definition at line 352 of file rtxCppDateTime.h.

7.62.3.2 virtual int OSXSDDateTimeClass::parseString (const OSUTF8CHAR * *dtString*) [virtual]

parseString: This method parses the time string and sets the date and time value

Parameters:

dtString const OSUTF8CHAR* - Date and time string

Returns:

Completion status of operation:

- 0(RT_OK) = success,
- negative return value is error

Reimplemented from [OSXSDDateTimeClass](#).

7.62.3.3 void OSXSDDateTimeClass::print (const char * *name*)

This method prints the datetime value to standard output.

Parameters:

name - Name of generated string variable.

Reimplemented from [OSXSDDateTimeClass](#).

7.62.3.4 virtual const OSUTF8CHAR* OSXSDDateTimeClass::toString (OSUTF8CHAR * *buffer*, size_t *bufsize*) [virtual]

This method sets the date and time fields to the values of current date and time.

Parameters:

buffer OSUTF8CHAR* - pointer to Date and time string

bufsize size_t specifies buffer size

Returns:

const OSUTF8CHAR* returns the datetime string

Reimplemented from [OSXSDDateTimeClass](#).

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

- [rtxCppDateTime.h](#)

Chapter 8

XBinder File Documentation

8.1 OSRTBaseType.h File Reference

C++ run-time base class for structured type definitions.

```
#include "rtxsrc/OSRTContext.h"
```

Classes

- class [OSRTBaseType](#)
C++ structured type base class.

8.1.1 Detailed Description

C++ run-time base class for structured type definitions.

Definition in file [OSRTBaseType.h](#).

8.2 OSRTContext.h File Reference

C++ run-time context class definition.

```
#include "rtxsrc/rtxContext.h"  
#include "rtxsrc/rtxDiag.h"  
#include "rtxsrc/rtxError.h"  
#include "rtxsrc/rtxMemory.h"
```

Classes

- class [OSRTContext](#)
Reference counted context class.
- class [OSRTCtxtPtr](#)
Context reference counted pointer class.

Functions

- void * [operator new](#) (size_t nbytes, OSCTXT *pctx)
Custom placement new function to allocate memory using context memory-management functions.
- void [operator delete](#) (void *pmem, OSCTXT *pctx)
Custom placement delete function to free memory using context memory-management functions.

8.2.1 Detailed Description

C++ run-time context class definition.

Definition in file [OSRTContext.h](#).

8.3 OSRTCtxtHolder.h File Reference

C++ run-time message buffer interface class definition.

```
#include "rtxsrc/OSRTCtxtHolderIF.h"
```

Classes

- class [OSRTCtxtHolder](#)

Abstract message buffer or stream interface class.

8.3.1 Detailed Description

C++ run-time message buffer interface class definition.

Definition in file [OSRTCtxtHolder.h](#).

8.4 OSRTCtxtHolderIF.h File Reference

C++ run-time message buffer interface class definition.

```
#include "rtxsrc/OSRTCtxtHolderIF.h"
```

Classes

- class [OSRTCtxtHolderIF](#)
Abstract message buffer or stream interface class.

8.4.1 Detailed Description

C++ run-time message buffer interface class definition.

Definition in file [OSRTCtxtHolderIF.h](#).

8.5 OSRTFastString.h File Reference

C++ fast string class definition.

```
#include "rtxsrc/rtxCommon.h"
```

```
#include "rtxsrc/rtxPrint.h"
```

Classes

- class [OSRTFastString](#)

C++ fast string class definition.

8.5.1 Detailed Description

C++ fast string class definition.

This can be used to hold standard ASCII or UTF-8 strings. This string class implementations directly assigns any assigned pointers to internal member variables. It does no memory management.

Definition in file [OSRTFastString.h](#).

8.6 OSRTFileInputStream.h File Reference

C++ base class definitions for operations with input file streams.

```
#include "rtxsrc/OSRTInputStream.h"
```

Classes

- class [OSRTFileInputStream](#)
Generic file input stream.

8.6.1 Detailed Description

C++ base class definitions for operations with input file streams.

Definition in file [OSRTFileInputStream.h](#).

8.7 OSRTFileOutputStream.h File Reference

C++ base class definitions for operations with output file streams.

```
#include "rtxsrc/OSRTOutputStream.h"
```

Classes

- class [OSRTFileOutputStream](#)
Generic file output stream.

8.7.1 Detailed Description

C++ base class definitions for operations with output file streams.

Definition in file [OSRTFileOutputStream.h](#).

8.8 OSRTInputStream.h File Reference

C++ base class definitions for operations with input streams.

```
#include "rtxsrc/OSRTInputStreamIF.h"
```

```
#include "rtxsrc/OSRTStream.h"
```

Classes

- class [OSRTInputStream](#)

This is the base class for input streams.

8.8.1 Detailed Description

C++ base class definitions for operations with input streams.

Definition in file [OSRTInputStream.h](#).

8.9 OSRTInputStreamIF.h File Reference

C++ interface class definitions for operations with input streams.

```
#include "rtxsrc/OSRTStreamIF.h"
```

Classes

- class [OSRTInputStreamIF](#)
- class [OSRTInputStreamPtr](#)

8.9.1 Detailed Description

C++ interface class definitions for operations with input streams.

Definition in file [OSRTInputStreamIF.h](#).

8.10 OSRTMemBuf.h File Reference

```
#include "rtxsrc/rtxMemBuf.h"
```

Classes

- class [OSRTMemBuf](#)
Memory Buffer class.

8.10.1 Detailed Description

Definition in file [OSRTMemBuf.h](#).

8.11 OSRTMemoryInputStream.h File Reference

C++ base class definitions for operations with input memory streams.

```
#include "rtxsrc/OSRTInputStream.h"
```

Classes

- class [OSRTMemoryInputStream](#)
Generic memory input stream.

8.11.1 Detailed Description

C++ base class definitions for operations with input memory streams.

Definition in file [OSRTMemoryInputStream.h](#).

8.12 OSRTMemoryOutputStream.h File Reference

C++ base class definitions for operations with output memory streams.

```
#include "rtxsrc/OSRTOutputStream.h"
```

Classes

- class [OSRTMemoryOutputStream](#)
Generic memory output stream.

8.12.1 Detailed Description

C++ base class definitions for operations with output memory streams.

Definition in file [OSRTMemoryOutputStream.h](#).

8.13 OSRTMsgBuf.h File Reference

C++ run-time message buffer class definition.

```
#include "rtxsrc/OSRTCtxtHolder.h"
```

```
#include "rtxsrc/OSRTMsgBufIF.h"
```

Classes

- class [OSRTMessageBuffer](#)

Abstract message buffer base class.

8.13.1 Detailed Description

C++ run-time message buffer class definition.

Definition in file [OSRTMsgBuf.h](#).

8.14 OSRTMsgBufIF.h File Reference

C++ run-time message buffer interface class definition.

```
#include "rtxsrc/OSRTContext.h"
```

```
#include "rtxsrc/OSRTCtxtHolderIF.h"
```

Classes

- class [OSRTMessageBufferIF](#)

Abstract message buffer or stream interface class.

8.14.1 Detailed Description

C++ run-time message buffer interface class definition.

Definition in file [OSRTMsgBufIF.h](#).

8.15 OSRTOutputStream.h File Reference

C++ base class definitions for operations with output streams.

```
#include "rtxsrc/OSRTOutputStreamIF.h"
```

```
#include "rtxsrc/OSRTStream.h"
```

Classes

- class [OSRTOutputStream](#)

The base class definition for operations with output streams.

8.15.1 Detailed Description

C++ base class definitions for operations with output streams.

Definition in file [OSRTOutputStream.h](#).

8.16 OSRTOutputStreamIF.h File Reference

C++ interface class definitions for operations with output streams.

```
#include "rtxsrc/OSRTStreamIF.h"
```

Classes

- class [OSRTOutputStreamIF](#)
- class [OSRTOutputStreamPtr](#)

8.16.1 Detailed Description

C++ interface class definitions for operations with output streams.

Definition in file [OSRTOutputStreamIF.h](#).

8.17 OSRTSocket.h File Reference

TCP/IP or UDP socket class definitions.

```
#include "rtxsrc/rtxSocket.h"
```

Classes

- class [OSRTSocket](#)

Wrapper class for TCP/IP or UDP sockets.

8.17.1 Detailed Description

TCP/IP or UDP socket class definitions.

Definition in file [OSRTSocket.h](#).

8.18 OSRTSocketInputStream.h File Reference

C++ base class definitions for operations with input socket streams.

```
#include "rtxsrc/OSRTSocket.h"
```

```
#include "rtxsrc/OSRTInputStream.h"
```

Classes

- class [OSRTSocketInputStream](#)
Generic socket input stream.

8.18.1 Detailed Description

C++ base class definitions for operations with input socket streams.

Definition in file [OSRTSocketInputStream.h](#).

8.19 OSRTSocketOutputStream.h File Reference

C++ base class definitions for operations with output socket streams.

```
#include "rtxsrc/OSRTSocket.h"
```

```
#include "rtxsrc/OSRTOutputStream.h"
```

Classes

- class [OSRTSocketOutputStream](#)
Generic socket output stream.

8.19.1 Detailed Description

C++ base class definitions for operations with output socket streams.

Definition in file [OSRTSocketOutputStream.h](#).

8.20 OSRTStream.h File Reference

C++ base class definitions for operations with I/O streams.

```
#include "rtxsrc/OSRCTxtHolder.h"
```

```
#include "rtxsrc/OSRTStreamIF.h"
```

Classes

- class [OSRTStream](#)

The default base class for using I/O streams.

8.20.1 Detailed Description

C++ base class definitions for operations with I/O streams.

Definition in file [OSRTStream.h](#).

8.21 OSRTStreamIF.h File Reference

C++ interface class definitions for operations with I/O streams.

```
#include "rtxsrc/rtxCommon.h"
```

```
#include "rtxsrc/OSRTCtxtHolderIF.h"
```

Classes

- class [OSRTStreamIF](#)

8.21.1 Detailed Description

C++ interface class definitions for operations with I/O streams.

Definition in file [OSRTStreamIF.h](#).

8.22 OSRTString.h File Reference

C++ string class definition.

```
#include "rtxsrc/rtxCommon.h"  
#include "rtxsrc/rtxPrint.h"  
#include "rtxsrc/OSRTStringIF.h"
```

Classes

- class [OSRTString](#)
C++ string class definition.

8.22.1 Detailed Description

C++ string class definition.

This can be used to hold standard ASCII or UTF-8 strings. The standard C++ 'new' and 'delete' operators are used to allocate/free memory for the strings. All strings are deep-copied.

Definition in file [OSRTString.h](#).

8.23 OSRTStringIF.h File Reference

C++ string class interface.

```
#include "rtxsrc/rtxCommon.h"
```

```
#include "rtxsrc/rtxPrint.h"
```

Classes

- class [OSRTStringIF](#)

C++ string class interface.

8.23.1 Detailed Description

C++ string class interface.

This defines an interface to allow different types of string derived classes to be implemented. Currently, implementations include a standard string class ([OSRTString](#)) which deep-copies all values using new/delete, and a fast string class ([OSRTFastString](#)) that just copies pointers (i.e does no memory management).

These classes can be used to hold standard ASCII or UTF-8 strings.

Definition in file [OSRTStringIF.h](#).

8.24 OSRTUTF8String.h File Reference

C++ UTF-8 string class definition.

```
#include "rtxsrc/OSRTBaseType.h"  
#include "rtxsrc/rtxPrint.h"  
#include "rtxsrc/rtxUTF8.h"
```

Classes

- class [OSRTUTF8String](#)
UTF-8 string.

8.24.1 Detailed Description

C++ UTF-8 string class definition.

Definition in file [OSRTUTF8String.h](#).

8.25 rtSaxCppAny.h File Reference

```
#include "rtxsrc/OSRTContext.h"  
#include "rtxmlsrc/osrtxml.h"  
#include "rtxmlsrc/rtSaxCppParser.h"  
#include "rtxmlsrc/rtXmlCppMsgBuf.h"  
#include "rtxsrc/rtxCppXmlString.h"  
#include "rtxmlsrc/OSXSDAnyTypeClass.h"
```

Classes

- class [OSXMLAnyHandler](#)

8.25.1 Detailed Description

Definition in file [rtSaxCppAny.h](#).

8.26 rtSaxCppSimpleType.h File Reference

```
#include "rtxmlsrc/osrtxml.h"  
#include "rtxmlsrc/rtSaxCppParser.h"
```

Classes

- class [OSXMLSimpleTypeHandler](#)

8.26.1 Detailed Description

Definition in file [rtSaxCppSimpleType.h](#).

8.27 rtSaxCppSoap.h File Reference

```
#include "rtxsrc/rtxCppDynOctStr.h"  
#include "rtxsrc/OSRTContext.h"  
#include "rtxsrc/OSRTMemBuf.h"  
#include "rtxsrc/rtxCppXmlString.h"  
#include "rtxmlsrc/osrtxml.h"  
#include "rtxmlsrc/rtSaxCppParser.h"  
#include "rtxmlsrc/rtXmlCppMsgBuf.h"
```

Classes

- class [OSXMLSoapHandler](#)

8.27.1 Detailed Description

Definition in file [rtSaxCppSoap.h](#).

8.28 rtSaxCppStrList.h File Reference

```
#include "rtxsrc/rtxToken.h"  
#include "rtxsrc/rtxDList.h"  
#include "rtxmlsrc/osrtxml.h"  
#include "rtxmlsrc/rtSaxCppParser.h"  
#include "rtxsrc/rtxCppDList.h"
```

Classes

- class [OSXMLStrListHandler](#)
OSXMLStrListHandler.

8.28.1 Detailed Description

Definition in file [rtSaxCppStrList.h](#).

8.29 rtxCppAnyAttr.h File Reference

C++ any element class definition.

```
#include "rtxsrc/rtxCommon.h"
```

```
#include "rtxsrc/OSRTBaseType.h"
```

Classes

- class [OSAnyAttrClass](#)
Any attribute.

Typedefs

- typedef OSUTF8NVP [OSAnyAttr](#)

8.29.1 Detailed Description

C++ any element class definition.

Definition in file [rtxCppAnyAttr.h](#).

8.30 rtxCppAnyElement.h File Reference

C++ any element class definition.

```
#include "rtxsrc/rtxCommon.h"  
#include "rtxsrc/OSRTBaseType.h"  
#include "rtxsrc/rtxPrint.h"
```

Classes

- class [OSAnyElementClass](#)

Any element.

Typedefs

- typedef OSUTF8NVP [OSAnyElement](#)

8.30.1 Detailed Description

C++ any element class definition.

Definition in file [rtxCppAnyElement.h](#).

8.31 rtxCppBufferedInputStream.h File Reference

```
#include "rtxsrc/OSRTInputStream.h"
```

Classes

- class [OSBufferedInputStream](#)
The buffered input stream class.

8.31.1 Detailed Description

Definition in file [rtxCppBufferedInputStream.h](#).

8.32 rtxCppDateTime.h File Reference

C++ XML schema date/time definition.

```
#include "rtxsrc/rtxCommon.h"
```

```
#include "rtxsrc/OSRTBaseType.h"
```

Classes

- class [OSXSDDateTimeClass](#)
- class [OSXSDDateClass](#)
- class [OSXSDDTimeClass](#)

8.32.1 Detailed Description

C++ XML schema date/time definition.

Definition in file [rtxCppDateTime.h](#).

8.33 rtxCppDList.h File Reference

```
#include "rtxsrc/rtxCommon.h"  
#include "rtxsrc/OSRTBaseType.h"  
#include "rtxsrc/rtxDList.h"
```

Classes

- class [OSRTDListNodeBaseClass](#)
This class is a base class for C++ representations of a node for the doubly-linked list structure.
- class [OSRTDListNodeClass](#)
This class represents a doubly-linked list node structure.
- class [OSRTObjListNodeClass](#)
This class represents a doubly-linked list node structure for OSRTBaseType instances.
- class [OSRTDListBaseClass](#)
This class is a base class for C++ representations of a doubly-linked list classes.
- class [OSRTDListClass](#)
This class represents a doubly-linked list structure.
- class [OSRTObjListClass](#)
This class represents a doubly-linked list structure for objects.

8.33.1 Detailed Description

Definition in file [rtxCppDList.h](#).

8.34 rtxCppDynOctStr.h File Reference

C++ dynamic binary string class definition.

```
#include "rtxsrc/rtxCommon.h"
```

```
#include "rtxsrc/OSRTBaseType.h"
```

Classes

- class [OSDynOctStrClass](#)
Dynamic binary string.

8.34.1 Detailed Description

C++ dynamic binary string class definition.

Definition in file [rtxCppDynOctStr.h](#).

8.35 rtxCppTypeException.h File Reference

C++ run-time deprecated definition.

```
#include "rtxsrc/rtxCommon.h"  
#include "rtxsrc/OSRTContext.h"
```

Classes

- class [OSRTLException](#)
The base exception class for the C++ run-time.
- class [OSSStreamException](#)
Exception class for streams.
- class [OSSAXException](#)

Defines

- #define [OSTRY](#) try
- #define [OSRTLTHROW1](#)(stat) throw [OSRTLException](#)(stat)
- #define [OSRTLTHROW2](#)(ctxt, stat) throw [OSRTLException](#)(ctxt,stat)
- #define [OSTHROW](#)(ex) throw (ex)
- #define [OSCATCH](#)(exType, ex, body) catch (exType ex) { body; }
- #define [OSSAXTHROW](#) throw [OSSAXException](#) ()
- #define [OSSAXTHROWSTR](#)(ctxt, str) throw [OSSAXException](#) (ctxt,str,RTERR_XMLPARSE,(const OSUTF8CHAR*)__FILE__, __LINE__)
- #define [OSSAXTHROWSTATUS](#)(ctxt, stat) throw [OSSAXException](#) (ctxt,LOG_RTERR(ctxt → get-Ptr(),stat),(const OSUTF8CHAR*)__FILE__, __LINE__)

8.35.1 Detailed Description

C++ run-time deprecated definition.

Definition in file [rtxCppTypeException.h](#).

8.36 rtxCppType.h File Reference

C++ common type and class definitions.

```
#include "rtxsrc/rtxCppAnyElement.h"  
#include "rtxsrc/rtxCppDList.h"  
#include "rtxsrc/rtxCppDynOctStr.h"  
#include "rtxsrc/rtxCppXmlString.h"
```

8.36.1 Detailed Description

C++ common type and class definitions.

Definition in file [rtxCppTypes.h](#).

8.37 `rtxCppXmlString.h` File Reference

C++ XML string class definition.

```
#include "rtxsrc/OSRTBaseType.h"
#include "rtxsrc/OSRTMemBuf.h"
#include "rtxsrc/rtxPrint.h"
#include "rtxsrc/rtxUTF8.h"
#include "rtxsrc/rtxXmlStr.h"
```

Classes

- class [OSXMLStringClass](#)
XML string.

Defines

- #define [DEFAULT_CAPACITY](#) 40

8.37.1 Detailed Description

C++ XML string class definition.

Definition in file [rtxCppXmlString.h](#).

8.38 rtXmlCppEncFuncs.h File Reference

XML low-level C++ encode functions.

```
#include "rtxmlsrc/osrtxml.h"
```

Functions

- int [rtXmlCppEncAnyAttr](#) (OSCTXT *pctxt, [OSRTObjListClass](#) *pAnyAttrList)
This function encodes a variable of the XSD any attribute type.
- int [rtXmlEncAny](#) (OSCTXT *pctxt, [OSXMLStringClass](#) *pxmlstr, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)
This function encodes a variable of the XSD any type.
- int [rtXmlEncString](#) (OSCTXT *pctxt, [OSXMLStringClass](#) *pxmlstr, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)
This function encodes a variable of the XSD string type.

8.38.1 Detailed Description

XML low-level C++ encode functions.

These are overloaded versions of C XML encode functions for use with C++.

Definition in file [rtXmlCppEncFuncs.h](#).

8.38.2 Function Documentation

8.38.2.1 int rtXmlCppEncAnyAttr (OSCTXT * pctxt, OSRTObjListClass * pAnyAttrList)

This function encodes a variable of the XSD any attribute type.

This is expressed as list of name/value pairs.

Parameters:

- pctxt* Pointer to context block structure.
- pAnyAttrList* List of name/value pair objects.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

8.38.2.2 int rtXmlEncAny (OSCTXT * pctxt, OSXMLStringClass * pxmlstr, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

This function encodes a variable of the XSD any type.

This is considered to be a fully-wrapped element of any type (for example: <myType>myData</myType>)

Parameters:

pctxt Pointer to context block structure.

pxmlstr Value to be encoded. This is a string containing the fully-encoded XML text to be copied to the output stream.

elemName XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.

nsPrefix XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

8.38.2.3 `int rtXmlEncString (OSCTXT * pctxt, OSXMLStringClass * pxmlstr, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)`

This function encodes a variable of the XSD string type.

Parameters:

pctxt Pointer to context block structure.

pxmlstr XML string value to be encoded.

elemName XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.

nsPrefix XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

Returns:

Completion status of operation:

- 0 = success,
- negative return value is error.

8.39 rtXmlCppMsgBuf.h File Reference

This file is deprecated.

```
#include "rtxmlsrc/OSXMLEncodeBuffer.h"  
#include "rtxmlsrc/OSXMLEncodeStream.h"  
#include "rtxmlsrc/OSXMLDecodeBuffer.h"
```

8.39.1 Detailed Description

This file is deprecated.

Users should use one or more of the individual headers files defined in the include statements below.

Definition in file [rtXmlCppMsgBuf.h](#).

8.40 rtXmlCppNamespace.h File Reference

XML namespace handling structures and function definitions.

```
#include "rtxmlsrc/osrtxml.h"  
#include "rtxsrc/OSRTBaseType.h"  
#include "rtxsrc/OSRTString.h"
```

Classes

- class [OSXMLNamespaceClass](#)

This class is used to hold an XML namespace prefix to URI mapping.

8.40.1 Detailed Description

XML namespace handling structures and function definitions.

Definition in file [rtXmlCppNamespace.h](#).

8.41 rtXmlCppXSDElement.h File Reference

C++ run-time XML schema global element class definition.

```
#include "rtxsrc/OSRTContext.h"
#include "rtxsrc/OSRTMsgBufIF.h"
#include "rtxsrc/rtxDiag.h"
#include "rtxsrc/rtxErrCodes.h"
#include "rtxmlsrc/osrtxml.h"
```

Classes

- class [OSXSDGlobalElement](#)
XSD global element base class.

8.41.1 Detailed Description

C++ run-time XML schema global element class definition.

Definition in file [rtXmlCppXSDElement.h](#).

Index

- ~OSBufferedInputStream
 - OSBufferedInputStream, 21
- ~OSRTCtxtHolderIF
 - OSRTCtxtHolderIF, 39
- ~OSRTCtxtPtr
 - OSRTCtxtPtr, 42
- ~OSRTInputStream
 - OSRTInputStream, 62
- ~OSRTInputStreamIF
 - OSRTInputStreamIF, 69
- ~OSRTException
 - OSRTException, 74
- ~OSRTMessageBuffer
 - OSRTMessageBuffer, 81
- ~OSRTMessageBufferIF
 - OSRTMessageBufferIF, 86
- ~OSRTOutputStream
 - OSRTOutputStream, 96
- ~OSRTOutputStreamIF
 - OSRTOutputStreamIF, 100
- ~OSRTSocket
 - OSRTSocket, 105
- ~OSRTStream
 - OSRTStream, 118
- ~OSXSDGlobalElement
 - OSXSDGlobalElement, 181
- accept
 - OSRTSocket, 105
- addrToString
 - OSRTSocket, 105
- append
 - OSRTDListClass, 47
 - OSRTObjListClass, 90
- appendCopy
 - OSRTDListClass, 47
 - OSRTObjListClass, 90
- appendValue
 - OSXMLStringClass, 165
- ASN.1 Stream Classes, 13
- bind
 - OSRTSocket, 106
- blockingRead
 - OSRTSocket, 107
- characters
 - OSXMLContentHandler, 141
 - OSXMLDefaultHandler, 144
 - OSXMLSimpleTypeHandler, 159
 - OSXMLSoapHandler, 161
- clone
 - OSAnyAttrClass, 16
 - OSDynOctStrClass, 25
 - OSRTUTF8String, 131
 - OSXMLStringClass, 165
 - OSXSDDateClass, 171
 - OSXSDDateTimeClass, 175
 - OSXSDDTimeClass, 186
- close
 - OSRTInputStream, 63
 - OSRTOutputStream, 96
 - OSRTSocket, 107
 - OSRTStream, 118
 - OSRTStreamIF, 122
- connect
 - OSRTSocket, 107
- copyValue
 - OSAnyAttrClass, 16
 - OSAnyElementClass, 19
 - OSDynOctStrClass, 25
 - OSRTUTF8String, 131
 - OSXMLStringClass, 165, 166
- currentPos
 - OSRTInputStream, 63
 - OSRTInputStreamIF, 69
- decodeFrom
 - OSXSDGlobalElement, 181
- decodeXML
 - OSXMLStringClass, 166
- encodeTo
 - OSXSDGlobalElement, 182
- encodeXML
 - OSXMLStringClass, 166
- endElement
 - OSXMLAnyHandler, 138
 - OSXMLContentHandler, 142
 - OSXMLDefaultHandler, 145
 - OSXMLSimpleTypeHandler, 159

- OSXMLSoapHandler, 161
- error
 - OSXMLErrorHandler, 149
- fatalError
 - OSXMLErrorHandler, 149
- flush
 - OSRTInputStream, 63
 - OSRTOutputStream, 96
 - OSRTStream, 119
 - OSRTStreamIF, 122
- Generic Input Stream Classes, 9
- Generic Output Stream Classes, 11
- getBytesIndex
 - OSRTMessageBuffer, 82
 - OSRTMessageBufferIF, 86
- getContext
 - OSRTCtxtHolder, 35
 - OSRTCtxtHolderIF, 39
 - OSRTInputStream, 63
 - OSRTOutputStream, 97
 - OSRTStream, 119
- getCount
 - OSRTDListBaseClass, 45
- getCtxtPtr
 - OSRTCtxtHolder, 35
 - OSRTCtxtHolderIF, 39
 - OSRTInputStream, 64
 - OSRTMessageBuffer, 82
 - OSRTOutputStream, 97
 - OSRTStream, 119
 - OSXSDDGlobalElement, 182
- getData
 - OSRTDListNodeClass, 50, 51
 - OSRTObjListNodeClass, 92, 93
- getErrorInfo
 - OSRTContext, 30, 31
 - OSRTCtxtHolder, 35, 36
 - OSRTCtxtHolderIF, 39
 - OSRTInputStream, 64
 - OSRTMessageBuffer, 82
 - OSRTOutputStream, 97, 98
 - OSRTStream, 120
- getHead
 - OSRTDListClass, 47
 - OSRTObjListClass, 90
- getItem
 - OSRTDListClass, 47
 - OSRTObjListClass, 90
- getList
 - OSRTDListBaseClass, 45
- getMsgCopy
 - OSRTMessageBufferIF, 86
- getMsgPtr
 - OSRTMessageBufferIF, 86
- getNext
 - OSRTDListNodeClass, 51
 - OSRTObjListNodeClass, 93
- getOwnership
 - OSRTSocket, 108
- getPrev
 - OSRTDListNodeClass, 51
 - OSRTObjListNodeClass, 93
- getPtr
 - OSRTContext, 30
- getSocket
 - OSRTSocket, 108
- getState
 - OSXMLDefaultHandler, 145
- getStatus
 - OSRTContext, 30
 - OSRTCtxtHolder, 36
 - OSRTCtxtHolderIF, 40
 - OSRTInputStream, 65
 - OSRTMessageBuffer, 83
 - OSRTOutputStream, 98
 - OSRTSocket, 108
 - OSRTStream, 120
- getTail
 - OSRTDListClass, 48
 - OSRTObjListClass, 91
- getTime
 - OSXSDDDateTimeClass, 175
- init
 - OSRTMessageBuffer, 83
 - OSRTMessageBufferIF, 87
- initBuffer
 - OSRTMessageBuffer, 83
 - OSRTMessageBufferIF, 87
- insert
 - OSRTDListClass, 48
 - OSRTObjListClass, 91
- isA
 - OSRTMessageBufferIF, 87
- isCDATA
 - OSXMLStringClass, 167
- isInitialized
 - OSRTContext, 30
- isOpened
 - OSRTInputStream, 65
 - OSRTOutputStream, 98
 - OSRTStream, 121
 - OSRTStreamIF, 123
- listen
 - OSRTSocket, 108

- mark
 - OSRTInputStream, 66
 - OSRTInputStreamIF, 69
- markSupported
 - OSRTInputStream, 65
 - OSRTInputStreamIF, 69
- memAlloc
 - OSRTContext, 31
 - OSXSDGlobalElement, 182
- memFreeAll
 - OSRTContext, 31
- memFreePtr
 - OSRTContext, 31
 - OSXSDGlobalElement, 182
- memRealloc
 - OSRTContext, 31
- Message Buffer Classes, 10
- mpContext
 - OSRTCtxtHolder, 36
 - OSXSDGlobalElement, 184
- mStatus
 - OSRTContext, 33
- operator=
 - OSRTCtxtPtr, 42
 - OSRTObjListClass, 91
- OSAnyAttrClass, 14
 - OSAnyAttrClass, 15, 16
- OSAnyAttrClass
 - clone, 16
 - copyValue, 16
 - OSAnyAttrClass, 15, 16
 - setValue, 16, 17
- OSAnyElementClass, 18
 - OSAnyElementClass, 19
- OSAnyElementClass
 - copyValue, 19
 - OSAnyElementClass, 19
 - print, 20
 - setValue, 20
- OSBufferedInputStream, 21
 - OSBufferedInputStream, 21
- OSBufferedInputStream
 - ~OSBufferedInputStream, 21
 - OSBufferedInputStream, 21
- OSDynOctStrClass, 23
 - OSDynOctStrClass, 24
- OSDynOctStrClass
 - clone, 25
 - copyValue, 25
 - OSDynOctStrClass, 24
 - setValue, 25
 - setValueFromBase64, 25
- OSRTBaseType, 27
 - OSRTBaseType.h, 188
- OSRTContext, 28
 - getErrorInfo, 30, 31
 - getPtr, 30
 - getStatus, 30
 - isInitialized, 30
 - memAlloc, 31
 - memFreeAll, 31
 - memFreePtr, 31
 - memRealloc, 31
 - mStatus, 33
 - setDiag, 32
 - setRunTimeKey, 32
 - setStatus, 32
- OSRTContext.h, 189
- OSRTCtxtHolder, 34
 - OSRTCtxtHolder, 35
- OSRTCtxtHolder
 - getContext, 35
 - getCtxtPtr, 35
 - getErrorInfo, 35, 36
 - getStatus, 36
 - mpContext, 36
 - OSRTCtxtHolder, 35
- OSRTCtxtHolder.h, 190
- OSRTCtxtHolderIF, 38
- OSRTCtxtHolderIF
 - ~OSRTCtxtHolderIF, 39
 - getContext, 39
 - getCtxtPtr, 39
 - getErrorInfo, 39
 - getStatus, 40
- OSRTCtxtHolderIF.h, 191
- OSRTCtxtPtr, 41
 - OSRTCtxtPtr, 42
- OSRTCtxtPtr
 - ~OSRTCtxtPtr, 42
 - operator=, 42
 - OSRTCtxtPtr, 42
- OSRTDListBaseClass, 44
- OSRTDListBaseClass
 - getCount, 45
 - getList, 45
 - remove, 45
- OSRTDListClass, 46
- OSRTDListClass
 - append, 47
 - appendCopy, 47
 - getHead, 47
 - getItem, 47
 - getTail, 48
 - insert, 48
- OSRTDListNodeBaseClass, 49
- OSRTDListNodeClass, 50

- OSRDLListNodeClass
 - getData, 50, 51
 - getNext, 51
 - getPrev, 51
- OSRTFastString, 53
 - OSRTFastString, 54
- OSRTFastString
 - OSRTFastString, 54
 - print, 54
 - setValue, 54, 55
- OSRTFastString.h, 192
- OSRTFileInputStream, 56
 - OSRTFileInputStream, 56, 57
- OSRTFileInputStream
 - OSRTFileInputStream, 56, 57
- OSRTFileInputStream.h, 193
- OSRTFileOutputStream, 58
 - OSRTFileOutputStream, 58, 59
- OSRTFileOutputStream
 - OSRTFileOutputStream, 58, 59
- OSRTFileOutputStream.h, 194
- OSRTInputStream, 61
 - OSRTInputStream, 62
- OSRTInputStream
 - ~OSRTInputStream, 62
 - close, 63
 - currentPos, 63
 - flush, 63
 - getContext, 63
 - getCtxtPtr, 64
 - getErrorInfo, 64
 - getStatus, 65
 - isOpened, 65
 - mark, 66
 - markSupported, 65
 - OSRTInputStream, 62
 - read, 66
 - readBlocking, 66
 - reset, 67
 - skip, 67
- OSRTInputStream.h, 195
- OSRTInputStreamIF, 68
- OSRTInputStreamIF
 - ~OSRTInputStreamIF, 69
 - currentPos, 69
 - mark, 69
 - markSupported, 69
 - read, 70
 - readBlocking, 70
 - reset, 70
 - skip, 71
- OSRTInputStreamIF.h, 196
- OSRTInputStreamPtr, 72
- OSRTException, 73
 - ~OSRTException, 74
 - OSRTException, 74
- OSRTMemBuf, 75
- OSRTMemBuf.h, 197
- OSRTMemoryInputStream, 76
 - OSRTMemoryInputStream, 76
- OSRTMemoryInputStream
 - OSRTMemoryInputStream, 76
- OSRTMemoryInputStream.h, 198
- OSRTMemoryOutputStream, 78
 - OSRTMemoryOutputStream, 78, 79
- OSRTMemoryOutputStream
 - OSRTMemoryOutputStream, 78, 79
- OSRTMemoryOutputStream.h, 199
- OSRTMessageBuffer, 80
 - OSRTMessageBuffer, 81
- OSRTMessageBuffer
 - ~OSRTMessageBuffer, 81
 - getByteIndex, 82
 - getCtxtPtr, 82
 - getErrorInfo, 82
 - getStatus, 83
 - init, 83
 - initBuffer, 83
 - OSRTMessageBuffer, 81
 - setDiag, 84
- OSRTMessageBufferIF, 85
- OSRTMessageBufferIF
 - ~OSRTMessageBufferIF, 86
 - getByteIndex, 86
 - getMsgCopy, 86
 - getMsgPtr, 86
 - init, 87
 - initBuffer, 87
 - isA, 87
 - setDiag, 87
- OSRTMsgBuf.h, 200
- OSRTMsgBufIF.h, 201
- OSRTObjListClass, 89
- OSRTObjListClass
 - append, 90
 - appendCopy, 90
 - getHead, 90
 - getItem, 90
 - getTail, 91
 - insert, 91
 - operator=, 91
- OSRTObjListNodeClass, 92
- OSRTObjListNodeClass
 - getData, 92, 93
 - getNext, 93
 - getPrev, 93
- OSRTOutputStream, 95
 - OSRTOutputStream, 96

- OSRTOutputStream
 - ~OSRTOutputStream, 96
 - close, 96
 - flush, 96
 - getContext, 97
 - getCtxtPtr, 97
 - getErrorInfo, 97, 98
 - getStatus, 98
 - isOpened, 98
 - OSRTOutputStream, 96
 - write, 99
- OSRTOutputStream.h, 202
- OSRTOutputStreamIF, 100
- OSRTOutputStreamIF
 - ~OSRTOutputStreamIF, 100
 - write, 100
- OSRTOutputStreamIF.h, 203
- OSRTOutputStreamPtr, 102
- OSRTSocket, 103
 - ~OSRTSocket, 105
 - accept, 105
 - addrToString, 105
 - bind, 106
 - blockingRead, 107
 - close, 107
 - connect, 107
 - getOwnership, 108
 - getSocket, 108
 - getStatus, 108
 - listen, 108
 - OSRTSocket, 104, 105
 - recv, 109
 - send, 109
 - setOwnership, 109
 - stringToAddr, 110
- OSRTSocket.h, 204
- OSRTSocketInputStream, 111
 - OSRTSocketInputStream, 112
- OSRTSocketInputStream
 - OSRTSocketInputStream, 112
- OSRTSocketInputStream.h, 205
- OSRTSocketOutputStream, 114
 - OSRTSocketOutputStream, 115
- OSRTSocketOutputStream
 - OSRTSocketOutputStream, 115
- OSRTSocketOutputStream.h, 206
- OSRTStream, 117
 - ~OSRTStream, 118
 - close, 118
 - flush, 119
 - getContext, 119
 - getCtxtPtr, 119
 - getErrorInfo, 120
 - getStatus, 120
 - isOpened, 121
 - OSRTStream, 118
- OSRTStream.h, 207
- OSRTStreamIF, 122
- OSRTStreamIF
 - close, 122
 - flush, 122
 - isOpened, 123
- OSRTStreamIF.h, 208
- OSRTString, 124
 - OSRTString, 125
 - print, 125
 - setValue, 125, 126
- OSRTString.h, 209
- OSRTStringIF, 127
 - OSRTStringIF, 128
- OSRTStringIF
 - OSRTStringIF, 128
 - print, 128
 - setValue, 128
- OSRTStringIF.h, 210
- OSRTUTF8String, 130
 - clone, 131
 - copyValue, 131
 - OSRTUTF8String, 131
 - print, 131
 - setValue, 132
- OSRTUTF8String.h, 211
- OSSAXException, 133
 - OSSAXException, 134
- OSSStreamException, 136
- OSXMLAnyHandler, 137
- OSXMLAnyHandler
 - endElement, 138
 - startElement, 137
- OSXMLBase, 139
- OSXMLBasePtr, 140
- OSXMLContentHandler, 141
- OSXMLContentHandler
 - characters, 141
 - endElement, 142
 - startElement, 142
- OSXMLDefaultHandler, 143
- OSXMLDefaultHandler
 - characters, 144
 - endElement, 145
 - getState, 145
 - startElement, 144
- OSXMLDefaultHandler::ErrorInfo, 146
- OSXMLDefaultHandlerIF, 147
- OSXMLDefaultHandlerPtr, 148
- OSXMLErrorHandler, 149
- OSXMLErrorHandler
 - error, 149

- fatalError, 149
- resetErrors, 150
- warning, 149
- OSXMLErrorInfo, 151
- OSXMLNamespaceClass, 152
 - OSXMLNamespaceClass, 152, 153
- OSXMLNamespaceClass
 - OSXMLNamespaceClass, 152, 153
- OSXMLParserCtxt, 154
- OSXMLParserCtxtIF, 155
- OSXMLReaderClass, 156
- OSXMLReaderClass
 - parse, 156, 157
- OSXMLSimpleTypeHandler, 158
- OSXMLSimpleTypeHandler
 - characters, 159
 - endElement, 159
 - startElement, 158
- OSXMLSoapHandler, 160
- OSXMLSoapHandler
 - characters, 161
 - endElement, 161
 - startElement, 160
- OSXMLStringClass, 162
 - OSXMLStringClass, 164, 165
- OSXMLStringClass
 - appendValue, 165
 - clone, 165
 - copyValue, 165, 166
 - decodeXML, 166
 - encodeXML, 166
 - isCDATA, 167
 - OSXMLStringClass, 164, 165
 - print, 167
 - setCDATA, 167
 - setValue, 167, 168
- OSXMLStrListHandler, 169
- OSXSDDateClass, 170
 - OSXSDDateClass, 171
- OSXSDDateClass
 - clone, 171
 - OSXSDDateClass, 171
 - parseString, 171
 - print, 172
 - toString, 172
- OSXSDDateTimeClass, 173
 - OSXSDDateTimeClass, 174, 175
- OSXSDDateTimeClass
 - clone, 175
 - getTime, 175
 - OSXSDDateTimeClass, 174, 175
 - parseString, 175
 - print, 176
 - setCurrent, 176
 - setCurrentTz, 176
 - setDateTime, 176
 - setLocalTime, 176
 - setUtcTime, 177
 - setValue, 177
 - toString, 177
- OSXSDDGlobalElement, 179
 - OSXSDDGlobalElement, 180, 181
- OSXSDDGlobalElement
 - ~OSXSDDGlobalElement, 181
 - decodeFrom, 181
 - encodeTo, 182
 - getCtxtPtr, 182
 - memAlloc, 182
 - memFreePtr, 182
 - mpContext, 184
 - OSXSDDGlobalElement, 180, 181
 - setDefaultNamespace, 182
 - setDiag, 183
 - setMsgBuf, 181
 - setNamespace, 183
 - setXSIType, 183
 - validateFrom, 183
- OSXSDDTimeClass, 185
 - OSXSDDTimeClass, 186
- OSXSDDTimeClass
 - clone, 186
 - OSXSDDTimeClass, 186
 - parseString, 186
 - print, 187
 - toString, 187
- parse
 - OSXMLReaderClass, 156, 157
- parseString
 - OSXSDDateClass, 171
 - OSXSDDateTimeClass, 175
 - OSXSDDTimeClass, 186
- print
 - OSAnyElementClass, 20
 - OSRTFastString, 54
 - OSRTString, 125
 - OSRTStringIF, 128
 - OSRTUTF8String, 131
 - OSXMLStringClass, 167
 - OSXSDDateClass, 172
 - OSXSDDateTimeClass, 176
 - OSXSDDTimeClass, 187
- read
 - OSRTInputStream, 66
 - OSRTInputStreamIF, 70
- readBlocking
 - OSRTInputStream, 66

- OSRTInputStreamIF, 70
- recv
 - OSRTSocket, 109
- remove
 - OSRTDListBaseClass, 45
- reset
 - OSRTInputStream, 67
 - OSRTInputStreamIF, 70
- resetErrors
 - OSXMLErrorHandler, 150
- rtSaxCppAny.h, 212
- rtSaxCppSimpleType.h, 213
- rtSaxCppSoap.h, 214
- rtSaxCppStrList.h, 215
- rtxCppAnyAttr.h, 216
- rtxCppAnyElement.h, 217
- rtxCppBufferedInputStream.h, 218
- rtxCppDateTime.h, 219
- rtxCppDList.h, 220
- rtxCppDynOctStr.h, 221
- rtxCppException.h, 222
- rtxCppTypes.h, 223
- rtxCppXmlString.h, 224
- rtXmlCppEncAnyAttr
 - rtXmlCppEncFuncs.h, 225
- rtXmlCppEncFuncs.h, 225
- rtXmlCppEncFuncs.h
 - rtXmlCppEncAnyAttr, 225
 - rtXmlEncAny, 225
 - rtXmlEncString, 226
- rtXmlCppMsgBuf.h, 227
- rtXmlCppNamespace.h, 228
- rtXmlCppXSDElement.h, 229
- rtXmlEncAny
 - rtXmlCppEncFuncs.h, 225
- rtXmlEncString
 - rtXmlCppEncFuncs.h, 226
- send
 - OSRTSocket, 109
- setCDATA
 - OSXMLStringClass, 167
- setCurrent
 - OSXSDDateTimeClass, 176
- setCurrentTz
 - OSXSDDateTimeClass, 176
- setDateTime
 - OSXSDDateTimeClass, 176
- setDefaultNamespace
 - OSXSDGlobalElement, 182
- setDiag
 - OSRTContext, 32
 - OSRTMessageBuffer, 84
 - OSRTMessageBufferIF, 87
 - OSXSDGlobalElement, 183
- setLocalTime
 - OSXSDDateTimeClass, 176
- setMsgBuf
 - OSXSDGlobalElement, 181
- setNamespace
 - OSXSDGlobalElement, 183
- setOwnership
 - OSRTSocket, 109
- setRunTimeKey
 - OSRTContext, 32
- setStatus
 - OSRTContext, 32
- setUtcTime
 - OSXSDDateTimeClass, 177
- setValue
 - OSAnyAttrClass, 16, 17
 - OSAnyElementClass, 20
 - OSDynOctStrClass, 25
 - OSRTFastString, 54, 55
 - OSRTString, 125, 126
 - OSRTStringIF, 128
 - OSRTUTF8String, 132
 - OSXMLStringClass, 167, 168
 - OSXSDDateTimeClass, 177
- setValueFromBase64
 - OSDynOctStrClass, 25
- setXSIType
 - OSXSDGlobalElement, 183
- skip
 - OSRTInputStream, 67
 - OSRTInputStreamIF, 71
- startElement
 - OSXMLAnyHandler, 137
 - OSXMLContentHandler, 142
 - OSXMLDefaultHandler, 144
 - OSXMLSimpleTypeHandler, 158
 - OSXMLSoapHandler, 160
- stringToAddr
 - OSRTSocket, 110
- TCP/IP or UDP Socket Classes, 12
- toString
 - OSXSDDateClass, 172
 - OSXSDDateTimeClass, 177
 - OSXSDDateTimeClass, 187
- validateFrom
 - OSXSDGlobalElement, 183
- warning
 - OSXMLErrorHandler, 149
- write
 - OSRTOutputStream, 99
 - OSRTOutputStreamIF, 100