

XBinder

XML Schema Compiler
Version 2.5
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 ©1997–2017 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	Main Page	1
2	Module Index	2
2.1	Modules	2
3	Class Index	3
3.1	Class Hierarchy	3
4	Class Index	5
4.1	Class List	5
5	File Index	7
5.1	File List	7
6	Module Documentation	9
6.1	Generic Input Stream Classes	9
6.1.1	Detailed Description	9
6.2	Message Buffer Classes	10
6.2.1	Detailed Description	10
6.3	Generic Output Stream Classes	11
6.3.1	Detailed Description	11
6.4	TCP/IP or UDP Socket Classes	12
6.4.1	Detailed Description	12
6.5	ASN.1 Stream Classes	13
6.5.1	Detailed Description	13
7	Class Documentation	14
7.1	OSAnyAttrClass Class Reference	14
7.1.1	Detailed Description	15
7.1.2	Constructor & Destructor Documentation	15
7.1.2.1	OSAnyAttrClass	15

7.1.2.2	OSAnyAttrClass	15
7.1.2.3	OSAnyAttrClass	16
7.1.2.4	OSAnyAttrClass	16
7.1.2.5	OSAnyAttrClass	16
7.1.3	Member Function Documentation	16
7.1.3.1	clone	16
7.1.3.2	copyValue	16
7.1.3.3	setValue	17
7.1.3.4	setValue	17
7.2	OSAnyElementClass Class Reference	18
7.2.1	Detailed Description	18
7.2.2	Constructor & Destructor Documentation	19
7.2.2.1	OSAnyElementClass	19
7.2.2.2	OSAnyElementClass	19
7.2.2.3	OSAnyElementClass	19
7.2.2.4	OSAnyElementClass	19
7.2.3	Member Function Documentation	19
7.2.3.1	copyValue	19
7.2.3.2	print	20
7.2.3.3	setValue	20
7.3	OSBufferedInputStream Class Reference	21
7.3.1	Detailed Description	21
7.3.2	Constructor & Destructor Documentation	21
7.3.2.1	OSBufferedInputStream	21
7.3.2.2	~OSBufferedInputStream	21
7.4	OSDynOctStrClass Class Reference	23
7.4.1	Detailed Description	24
7.4.2	Constructor & Destructor Documentation	24
7.4.2.1	OSDynOctStrClass	24
7.4.2.2	OSDynOctStrClass	24
7.4.2.3	OSDynOctStrClass	24
7.4.3	Member Function Documentation	24
7.4.3.1	clone	24
7.4.3.2	copyValue	25
7.4.3.3	setValue	25
7.4.3.4	setValue	25
7.4.3.5	setValueFromBase64	25

7.5	OSRTBase64TextInputStream Class Reference	26
7.5.1	Detailed Description	26
7.5.2	Constructor & Destructor Documentation	26
7.5.2.1	OSRTBase64TextInputStream	26
7.5.2.2	~OSRTBase64TextInputStream	27
7.5.3	Member Function Documentation	27
7.5.3.1	isA	27
7.6	OSRTBaseType Class Reference	28
7.6.1	Detailed Description	28
7.7	OSRTCContext Class Reference	29
7.7.1	Detailed Description	30
7.7.2	Member Function Documentation	31
7.7.2.1	getErrorInfo	31
7.7.2.2	getErrorInfo	31
7.7.2.3	getErrorInfo	31
7.7.2.4	getPtr	31
7.7.2.5	getStatus	32
7.7.2.6	isInitialized	32
7.7.2.7	memAlloc	32
7.7.2.8	memAllocZ	32
7.7.2.9	memFreeAll	32
7.7.2.10	memFreePtr	33
7.7.2.11	memRealloc	33
7.7.2.12	setDiag	33
7.7.2.13	setRunTimeKey	33
7.7.2.14	setStatus	34
7.7.3	Member Data Documentation	34
7.7.3.1	mStatus	34
7.8	OSRTCtxtHolder Class Reference	35
7.8.1	Detailed Description	36
7.8.2	Constructor & Destructor Documentation	36
7.8.2.1	OSRTCtxtHolder	36
7.8.3	Member Function Documentation	36
7.8.3.1	getContext	36
7.8.3.2	getCtxtPtr	36
7.8.3.3	getErrorInfo	36
7.8.3.4	getErrorInfo	37

7.8.3.5	getStatus	37
7.8.4	Member Data Documentation	37
7.8.4.1	mpContext	37
7.9	OSRTCtxtHolderIF Class Reference	38
7.9.1	Detailed Description	38
7.9.2	Constructor & Destructor Documentation	39
7.9.2.1	~OSRTCtxtHolderIF	39
7.9.3	Member Function Documentation	39
7.9.3.1	getContext	39
7.9.3.2	getCtxtPtr	39
7.9.3.3	getErrorInfo	39
7.9.3.4	getErrorInfo	40
7.9.3.5	getStatus	40
7.10	OSRTCtxtPtr Class Reference	41
7.10.1	Detailed Description	42
7.10.2	Constructor & Destructor Documentation	42
7.10.2.1	OSRTCtxtPtr	42
7.10.2.2	OSRTCtxtPtr	42
7.10.2.3	~OSRTCtxtPtr	42
7.10.3	Member Function Documentation	42
7.10.3.1	operator=	42
7.11	OSRTDListBaseClass Class Reference	44
7.11.1	Detailed Description	44
7.11.2	Member Function Documentation	44
7.11.2.1	getCount	44
7.11.2.2	getList	45
7.11.2.3	getList	45
7.11.2.4	remove	45
7.12	OSRTDListClass Class Reference	46
7.12.1	Detailed Description	47
7.12.2	Member Function Documentation	47
7.12.2.1	append	47
7.12.2.2	appendCopy	47
7.12.2.3	getHead	47
7.12.2.4	getHead	47
7.12.2.5	getItem	48
7.12.2.6	getTail	48

7.12.2.7	getTail	48
7.12.2.8	insert	48
7.13	OSRTDListNodeBaseClass Class Reference	49
7.13.1	Detailed Description	49
7.14	OSRTDListNodeClass Class Reference	50
7.14.1	Detailed Description	50
7.14.2	Member Function Documentation	50
7.14.2.1	getData	50
7.14.2.2	getData	51
7.14.2.3	getNext	51
7.14.2.4	getNext	51
7.14.2.5	getPrev	51
7.14.2.6	getPrev	51
7.15	OSRTFastString Class Reference	52
7.15.1	Detailed Description	53
7.15.2	Constructor & Destructor Documentation	53
7.15.2.1	OSRTFastString	53
7.15.2.2	OSRTFastString	53
7.15.2.3	OSRTFastString	53
7.15.3	Member Function Documentation	53
7.15.3.1	print	53
7.15.3.2	setValue	54
7.15.3.3	setValue	54
7.16	OSRTFileInputStream Class Reference	55
7.16.1	Detailed Description	55
7.16.2	Constructor & Destructor Documentation	55
7.16.2.1	OSRTFileInputStream	55
7.16.2.2	OSRTFileInputStream	56
7.16.2.3	OSRTFileInputStream	56
7.16.2.4	OSRTFileInputStream	56
7.16.3	Member Function Documentation	56
7.16.3.1	isA	56
7.17	OSRTFileOutputStream Class Reference	58
7.17.1	Detailed Description	58
7.17.2	Constructor & Destructor Documentation	58
7.17.2.1	OSRTFileOutputStream	58
7.17.2.2	OSRTFileOutputStream	59

7.17.2.3	OSRTFileOutputStream	59
7.17.2.4	OSRTFileOutputStream	59
7.17.3	Member Function Documentation	60
7.17.3.1	isA	60
7.18	OSRTHexTextInputStream Class Reference	61
7.18.1	Detailed Description	61
7.18.2	Constructor & Destructor Documentation	61
7.18.2.1	OSRTHexTextInputStream	61
7.18.2.2	~OSRTHexTextInputStream	62
7.18.3	Member Function Documentation	62
7.18.3.1	isA	62
7.19	OSRTInputStream Class Reference	63
7.19.1	Detailed Description	64
7.19.2	Constructor & Destructor Documentation	64
7.19.2.1	OSRTInputStream	64
7.19.2.2	~OSRTInputStream	65
7.19.3	Member Function Documentation	65
7.19.3.1	close	65
7.19.3.2	currentPos	65
7.19.3.3	flush	65
7.19.3.4	getContext	66
7.19.3.5	getCtxtPtr	66
7.19.3.6	getErrorInfo	66
7.19.3.7	getErrorInfo	67
7.19.3.8	getPosition	67
7.19.3.9	getStatus	67
7.19.3.10	isA	67
7.19.3.11	isOpened	68
7.19.3.12	mark	68
7.19.3.13	markSupported	68
7.19.3.14	read	69
7.19.3.15	readBlocking	69
7.19.3.16	reset	69
7.19.3.17	setPosition	70
7.19.3.18	skip	70
7.20	OSRTLException Class Reference	71
7.20.1	Detailed Description	71

7.20.2	Constructor & Destructor Documentation	72
7.20.2.1	OSRTLException	72
7.20.2.2	OSRTLException	72
7.20.2.3	OSRTLException	72
7.20.2.4	~OSRTLException	72
7.21	OSRTMemBuf Class Reference	73
7.21.1	Detailed Description	73
7.22	OSRTMemoryInputStream Class Reference	74
7.22.1	Detailed Description	74
7.22.2	Constructor & Destructor Documentation	74
7.22.2.1	OSRTMemoryInputStream	74
7.22.2.2	OSRTMemoryInputStream	75
7.22.3	Member Function Documentation	75
7.22.3.1	isA	75
7.23	OSRTMemoryOutputStream Class Reference	76
7.23.1	Detailed Description	76
7.23.2	Constructor & Destructor Documentation	76
7.23.2.1	OSRTMemoryOutputStream	76
7.23.2.2	OSRTMemoryOutputStream	77
7.23.2.3	OSRTMemoryOutputStream	77
7.23.3	Member Function Documentation	77
7.23.3.1	getBuffer	77
7.23.3.2	isA	78
7.23.3.3	reset	78
7.24	OSRTMessageBuffer Class Reference	79
7.24.1	Detailed Description	80
7.24.2	Constructor & Destructor Documentation	80
7.24.2.1	OSRTMessageBuffer	80
7.24.2.2	~OSRTMessageBuffer	81
7.24.3	Member Function Documentation	81
7.24.3.1	getByteIndex	81
7.24.3.2	getCtxtPtr	81
7.24.3.3	getErrorInfo	81
7.24.3.4	getErrorInfo	81
7.24.3.5	getStatus	82
7.24.3.6	init	82
7.24.3.7	initBuffer	82

7.24.3.8	setDiag	83
7.25	OSRTMessageBufferIF Class Reference	84
7.25.1	Detailed Description	85
7.25.2	Constructor & Destructor Documentation	85
7.25.2.1	~OSRTMessageBufferIF	85
7.25.3	Member Function Documentation	85
7.25.3.1	getByteIndex	85
7.25.3.2	getMsgCopy	85
7.25.3.3	getMsgPtr	85
7.25.3.4	init	86
7.25.3.5	initBuffer	86
7.25.3.6	isA	86
7.25.3.7	setDiag	86
7.26	OSRTObjListClass Class Reference	88
7.26.1	Detailed Description	89
7.26.2	Member Function Documentation	89
7.26.2.1	append	89
7.26.2.2	appendCopy	89
7.26.2.3	getHead	89
7.26.2.4	getHead	89
7.26.2.5	getItem	90
7.26.2.6	getTail	90
7.26.2.7	getTail	90
7.26.2.8	insert	90
7.26.2.9	operator=	90
7.27	OSRTObjListNodeClass Class Reference	91
7.27.1	Detailed Description	91
7.27.2	Member Function Documentation	91
7.27.2.1	getData	91
7.27.2.2	getData	92
7.27.2.3	getNext	92
7.27.2.4	getNext	92
7.27.2.5	getPrev	92
7.27.2.6	getPrev	92
7.28	OSRTOutputStream Class Reference	93
7.28.1	Detailed Description	94
7.28.2	Constructor & Destructor Documentation	94

7.28.2.1	OSRTOutputStream	94
7.28.2.2	~OSRTOutputStream	94
7.28.3	Member Function Documentation	94
7.28.3.1	close	94
7.28.3.2	flush	95
7.28.3.3	getContext	95
7.28.3.4	getCtxtPtr	95
7.28.3.5	getErrorInfo	95
7.28.3.6	getErrorInfo	96
7.28.3.7	getStatus	96
7.28.3.8	isA	97
7.28.3.9	isOpened	97
7.28.3.10	write	97
7.28.3.11	write	97
7.29	OSRTSocket Class Reference	99
7.29.1	Detailed Description	100
7.29.2	Constructor & Destructor Documentation	100
7.29.2.1	OSRTSocket	100
7.29.2.2	OSRTSocket	101
7.29.2.3	OSRTSocket	101
7.29.2.4	~OSRTSocket	101
7.29.3	Member Function Documentation	101
7.29.3.1	accept	101
7.29.3.2	addrToString	101
7.29.3.3	bind	102
7.29.3.4	bind	102
7.29.3.5	bind	103
7.29.3.6	bindUrl	103
7.29.3.7	blockingRead	103
7.29.3.8	close	104
7.29.3.9	connect	104
7.29.3.10	connectUrl	104
7.29.3.11	getOwnership	105
7.29.3.12	getSocket	105
7.29.3.13	getStatus	105
7.29.3.14	listen	105
7.29.3.15	recv	106

7.29.3.16	send	106
7.29.3.17	setOwnership	106
7.29.3.18	stringToAddr	107
7.30	OSRTSocketInputStream Class Reference	108
7.30.1	Detailed Description	108
7.30.2	Constructor & Destructor Documentation	108
7.30.2.1	OSRTSocketInputStream	108
7.30.2.2	OSRTSocketInputStream	109
7.30.2.3	OSRTSocketInputStream	109
7.30.2.4	OSRTSocketInputStream	109
7.30.3	Member Function Documentation	110
7.30.3.1	isA	110
7.31	OSRTSocketOutputStream Class Reference	111
7.31.1	Detailed Description	111
7.31.2	Constructor & Destructor Documentation	111
7.31.2.1	OSRTSocketOutputStream	111
7.31.2.2	OSRTSocketOutputStream	112
7.31.2.3	OSRTSocketOutputStream	112
7.31.2.4	OSRTSocketOutputStream	112
7.31.3	Member Function Documentation	113
7.31.3.1	isA	113
7.32	OSRTStream Class Reference	114
7.32.1	Detailed Description	115
7.32.2	Constructor & Destructor Documentation	115
7.32.2.1	OSRTStream	115
7.32.2.2	~OSRTStream	115
7.32.3	Member Function Documentation	116
7.32.3.1	close	116
7.32.3.2	flush	116
7.32.3.3	getContext	116
7.32.3.4	getCtxtPtr	117
7.32.3.5	getErrorInfo	117
7.32.3.6	getErrorInfo	117
7.32.3.7	getStatus	118
7.32.3.8	isOpened	118
7.33	OSRTString Class Reference	119
7.33.1	Detailed Description	120

7.33.2	Constructor & Destructor Documentation	120
7.33.2.1	OSRTString	120
7.33.2.2	OSRTString	120
7.33.2.3	OSRTString	120
7.33.3	Member Function Documentation	120
7.33.3.1	print	120
7.33.3.2	setValue	120
7.33.3.3	setValue	121
7.34	OSRTStringIF Class Reference	122
7.34.1	Detailed Description	123
7.34.2	Constructor & Destructor Documentation	123
7.34.2.1	OSRTStringIF	123
7.34.2.2	OSRTStringIF	123
7.34.3	Member Function Documentation	123
7.34.3.1	print	123
7.34.3.2	setValue	123
7.34.3.3	setValue	124
7.35	OSRTUTF8String Class Reference	125
7.35.1	Detailed Description	126
7.35.2	Constructor & Destructor Documentation	126
7.35.2.1	OSRTUTF8String	126
7.35.2.2	OSRTUTF8String	126
7.35.2.3	OSRTUTF8String	126
7.35.3	Member Function Documentation	126
7.35.3.1	clone	126
7.35.3.2	copyValue	126
7.35.3.3	print	127
7.35.3.4	setValue	127
7.36	OSStreamException Class Reference	128
7.36.1	Detailed Description	128
7.37	OSXMLStringClass Class Reference	129
7.37.1	Detailed Description	131
7.37.2	Constructor & Destructor Documentation	131
7.37.2.1	OSXMLStringClass	131
7.37.2.2	OSXMLStringClass	131
7.37.2.3	OSXMLStringClass	131
7.37.2.4	OSXMLStringClass	132

7.37.2.5	OSXMLStringClass	132
7.37.3	Member Function Documentation	132
7.37.3.1	appendValue	132
7.37.3.2	clone	132
7.37.3.3	compare	132
7.37.3.4	copyValue	133
7.37.3.5	copyValue	133
7.37.3.6	decodeXML	133
7.37.3.7	encodeXML	133
7.37.3.8	isCDATA	134
7.37.3.9	print	134
7.37.3.10	setCDATA	134
7.37.3.11	setValue	134
7.37.3.12	setValue	135
7.37.3.13	setValue	135
7.38	OSXMLStringList Class Reference	136
7.38.1	Detailed Description	136
7.38.2	Constructor & Destructor Documentation	137
7.38.2.1	OSXMLStringList	137
7.38.3	Member Function Documentation	137
7.38.3.1	append	137
7.38.3.2	appendCopy	137
7.38.3.3	clone	137
7.38.3.4	operator=	137
8	File Documentation	138
8.1	OSRTBase64TextInputStream.h File Reference	138
8.1.1	Detailed Description	138
8.2	OSRTBaseType.h File Reference	139
8.2.1	Detailed Description	139
8.3	OSRTContext.h File Reference	140
8.3.1	Detailed Description	140
8.4	OSRTCtxtHolder.h File Reference	141
8.4.1	Detailed Description	141
8.5	OSRTCtxtHolderIF.h File Reference	142
8.5.1	Detailed Description	142
8.6	OSRTFastString.h File Reference	143

8.6.1	Detailed Description	143
8.7	OSRTFileInputStream.h File Reference	144
8.7.1	Detailed Description	144
8.8	OSRTFileOutputStream.h File Reference	145
8.8.1	Detailed Description	145
8.9	OSRTHexTextInputStream.h File Reference	146
8.9.1	Detailed Description	146
8.10	OSRTInputStream.h File Reference	147
8.10.1	Detailed Description	147
8.11	OSRTInputStreamIF.h File Reference	148
8.11.1	Detailed Description	148
8.12	OSRTMemBuf.h File Reference	149
8.12.1	Detailed Description	149
8.13	OSRTMemoryInputStream.h File Reference	150
8.13.1	Detailed Description	150
8.14	OSRTMemoryOutputStream.h File Reference	151
8.14.1	Detailed Description	151
8.15	OSRTMsgBuf.h File Reference	152
8.15.1	Detailed Description	152
8.16	OSRTMsgBufIF.h File Reference	153
8.16.1	Detailed Description	153
8.17	OSRTOutputStream.h File Reference	154
8.17.1	Detailed Description	154
8.18	OSRTOutputStreamIF.h File Reference	155
8.18.1	Detailed Description	155
8.19	OSRTSocket.h File Reference	156
8.19.1	Detailed Description	156
8.20	OSRTSocketInputStream.h File Reference	157
8.20.1	Detailed Description	157
8.21	OSRTSocketOutputStream.h File Reference	158
8.21.1	Detailed Description	158
8.22	OSRTStream.h File Reference	159
8.22.1	Detailed Description	159
8.23	OSRTStreamIF.h File Reference	160
8.23.1	Detailed Description	160
8.24	OSRTString.h File Reference	161
8.24.1	Detailed Description	161

8.25	OSRTStringIF.h File Reference	162
8.25.1	Detailed Description	162
8.26	OSRTUTF8String.h File Reference	163
8.26.1	Detailed Description	163
8.27	rtxCppAnyAttr.h File Reference	164
8.27.1	Detailed Description	164
8.28	rtxCppAnyElement.h File Reference	165
8.28.1	Detailed Description	165
8.29	rtxCppBitString.h File Reference	166
8.29.1	Detailed Description	166
8.29.2	Function Documentation	166
8.29.2.1	rtxCppCheckBitBounds	166
8.30	rtxCppBufferedInputStream.h File Reference	167
8.30.1	Detailed Description	167
8.31	rtxCppDateTime.h File Reference	168
8.31.1	Detailed Description	168
8.32	rtxCppDList.h File Reference	169
8.32.1	Detailed Description	169
8.33	rtxCppDynOctStr.h File Reference	170
8.33.1	Detailed Description	170
8.34	rtxCppException.h File Reference	171
8.34.1	Detailed Description	171
8.35	rtxCppTypes.h File Reference	172
8.35.1	Detailed Description	172
8.36	rtxCppXmlSTLString.h File Reference	173
8.36.1	Detailed Description	173
8.37	rtxCppXmlString.h File Reference	174
8.37.1	Detailed Description	174
8.38	rtxCppXmlStringList.h File Reference	175
8.38.1	Detailed Description	175

Chapter 1

Main Page

C++ Common 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

Module Index

2.1 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

Class Index

3.1 Class Hierarchy

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

OSRTBaseType	28
OSAnyAttrClass	14
OSAnyElementClass	18
OSDynOctStrClass	23
OSRTDListBaseClass	44
OSRTDListClass	46
OSRTObjListClass	88
OSRTUTF8String	125
OSXMLStringClass	129
OSXMLStringList	136
OSRTContext	29
OSRTCtxtHolderIF	38
OSRTCtxtHolder	35
OSRTCtxtPtr	41
OSRTDListNodeBaseClass	49
OSRTDListNodeClass	50
OSRTObjListNodeClass	91
OSRTException	71
OSStreamException	128
OSRTMemBuf	73
OSRTMessageBufferIF	84
OSRTMessageBuffer	79
OSRTSocket	99
OSRTStream	114
OSRTInputStream	63
OSBufferedInputStream	21
OSRTBase64TextInputStream	26
OSRTFileInputStream	55
OSRTHexTextInputStream	61
OSRTMemoryInputStream	74
OSRTSocketInputStream	108
OSRTOutputStream	93

OSRTFileOutputStream	58
OSRTMemoryOutputStream	76
OSRTSocketOutputStream	111
OSRTStringIF	122
OSRTFastString	52
OSRTString	119

Chapter 4

Class Index

4.1 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
OSRTBase64TextInputStream (Hexadecimal text input stream filter class)	26
OSRTBaseType (C++ structured type base class)	28
OSRTContext (Reference counted context class)	29
OSRTCtxtHolder (Abstract message buffer or stream interface class)	35
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)	52
OSRTFileInputStream (Generic file input stream)	55
OSRTFileOutputStream (Generic file output stream)	58
OSRTHexTextInputStream (Hexadecimal text input stream filter class)	61
OSRTInputStream (This is the base class for input streams)	63
OSRTLException (The base exception class for the C++ run-time)	71
OSRTMemBuf (Memory Buffer class)	73
OSRTMemoryInputStream (Generic memory input stream)	74
OSRTMemoryOutputStream (Generic memory output stream)	76
OSRTMessageBuffer (Abstract message buffer base class)	79
OSRTMessageBufferIF (Abstract message buffer or stream interface class)	84
OSRTObjListClass (This class represents a doubly-linked list structure for objects)	88
OSRTObjListNodeClass (This class represents a doubly-linked list node structure for OSRTBaseType instances)	91
OSRTOutputStream (The base class definition for operations with output streams)	93
OSRTSocket (Wrapper class for TCP/IP or UDP sockets)	99
OSRTSocketInputStream (Generic socket input stream)	108
OSRTSocketOutputStream (Generic socket output stream)	111

OSRTStream (The default base class for using I/O streams)	114
OSRTString (C++ string class definition)	119
OSRTStringIF (C++ string class interface)	122
OSRTUTF8String (UTF-8 string)	125
OSStreamException (Exception class for streams)	128
OSXMLStringClass (XML string)	129
OSXMLStringList (XML list string)	136

Chapter 5

File Index

5.1 File List

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

OSRTBase64TextInputStream.h (C++ hexadecimal text input stream filter class)	138
OSRTBaseType.h (C++ run-time base class for structured type definitions)	139
OSRTContext.h (C++ run-time context class definition)	140
OSRTCtxtHolder.h (C++ run-time message buffer interface class definition)	141
OSRTCtxtHolderIF.h (C++ run-time message buffer interface class definition)	142
OSRTFastString.h (C++ fast string class definition)	143
OSRTFileInputStream.h (C++ base class definitions for operations with input file streams)	144
OSRTFileOutputStream.h (C++ base class definitions for operations with output file streams)	145
OSRTHexTextInputStream.h (C++ hexadecimal text input stream filter class)	146
OSRTInputStream.h (C++ base class definitions for operations with input streams)	147
OSRTInputStreamIF.h (C++ interface class definitions for operations with input streams)	148
OSRTMemBuf.h	149
OSRTMemoryInputStream.h (C++ base class definitions for operations with input memory streams)	150
OSRTMemoryOutputStream.h (C++ base class definitions for operations with output memory streams)	151
OSRTMsgBuf.h (C++ run-time message buffer class definition)	152
OSRTMsgBufIF.h (C++ run-time message buffer interface class definition)	153
OSRTOutputStream.h (C++ base class definitions for operations with output streams)	154
OSRTOutputStreamIF.h (C++ interface class definitions for operations with output streams)	155
OSRTSocket.h (TCP/IP or UDP socket class definitions)	156
OSRTSocketInputStream.h (C++ base class definitions for operations with input socket streams)	157
OSRTSocketOutputStream.h (C++ base class definitions for operations with output socket streams)	158
OSRTStream.h (C++ base class definitions for operations with I/O streams)	159
OSRTStreamIF.h (C++ interface class definitions for operations with I/O streams)	160
OSRTString.h (C++ string class definition)	161
OSRTStringIF.h (C++ string class interface)	162
OSRTUTF8String.h (C++ UTF-8 string class definition)	163
rtxCppAnyAttr.h (C++ any element class definition)	164
rtxCppAnyElement.h (C++ any element class definition)	165
rtxCppBitString.h	(
• Contains utility function for sizing a bit string	
)	166
rtxCppBufferedInputStream.h	167
rtxCppDateTime.h (C++ XML schema date/time definition)	168

rtxCppDList.h	169
rtxCppDynOctStr.h (C++ dynamic binary string class definition)	170
rtxCppException.h (C++ run-time deprecated definition)	171
rtxCppTypes.h (C++ common type and class definitions)	172
rtxCppXmlSTLString.h (C++ XML STL string class definition)	173
rtxCppXmlString.h (C++ XML string class definition)	174
rtxCppXmlStringList.h (C++ XML string list class definition)	175

Chapter 6

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.

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.

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.

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

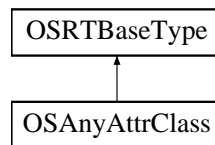
Class Documentation

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 rxCppAnyAttr.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*, 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.

7.1.3.4 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.

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

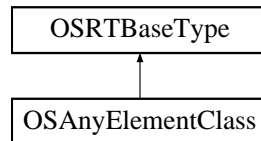
- [rtxCppAnyAttr.h](#)

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.

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:

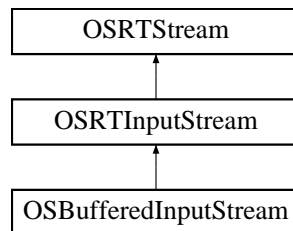
- [rtxCppAnyElement.h](#)

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

[OSSStreamException](#) 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:

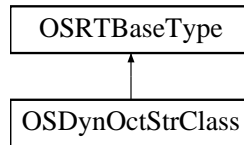
- [rtxCppBufferedInputStream.h](#)

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.

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 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.

7.4.3.4 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.

7.4.3.5 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:

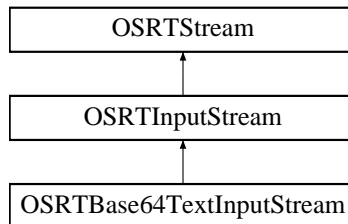
- [rtxCppDynOctStr.h](#)

7.5 OSRTBase64TextInputStream Class Reference

Hexadecimal text input stream filter class.

```
#include <OSRTBase64TextInputStream.h>
```

Inheritance diagram for OSRTBase64TextInputStream:



Public Member Functions

- EXTRMETHOD [OSRTBase64TextInputStream \(OSRTInputStream *pstream\)](#)
Initializes the input stream using the existing standard input stream.
- EXTRMETHOD [~OSRTBase64TextInputStream \(\)](#)
The destructor deletes the underlying stream object.
- virtual OSBOOL [isA \(StreamID id\) const](#)
This method is used to query a stream object in order to determine its actual type.
- void [setOwnUnderStream \(OSBOOL value=TRUE\)](#)
This method is used to transfer ownership of the underlying stream to the class.
- OSBOOL [isCertificate \(\)](#)
This method is used to determine if a certificate was parsed.

7.5.1 Detailed Description

Hexadecimal text input stream filter class. This class is created on top of an existing stream class to provide conversion of hexadecimal text input into binary form.

Definition at line 39 of file OSRTBase64TextInputStream.h.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 EXTRMETHOD OSRTBase64TextInputStream::OSRTBase64TextInputStream (OSRTInputStream * pstream)

Initializes the input stream using the existing standard input stream.

Only file and memory underlying stream types are supported.

Parameters

pstream The underlying input stream object. Note that this class will take control of the underlying stream object and delete it upon destruction.

See also

`rtxStreamHexTextAttach`

7.5.2.2 EXTRTMETHOD OSRTBase64TextInputStream::~OSRTBase64TextInputStream ()

The destructor deletes the underlying stream object.

That object should be used as nothing more to a surrogate to this object.

7.5.3 Member Function Documentation

7.5.3.1 virtual OSBOOL OSRTBase64TextInputStream::isA (StreamID *id*) const [inline, virtual]

This method is used to query a stream object in order to determine its actual type.

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented from [OSRTInputStream](#).

Definition at line 72 of file `OSRTBase64TextInputStream.h`.

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

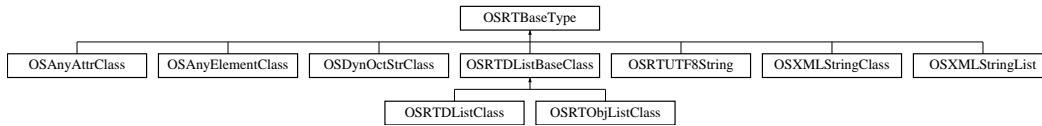
- [OSRTBase64TextInputStream.h](#)

7.6 OSRTBaseType Class Reference

C++ structured type base class.

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

Inheritance diagram for OSRTBaseType:



7.6.1 Detailed Description

C++ structured type base class. This is the base class for all generated structured types.

Definition at line 37 of file OSRTBaseType.h.

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

- [OSRTBaseType.h](#)

7.7 OSRTContext Class Reference

Reference counted context class.

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

Public Member Functions

- EXTRTMETHOD [OSRTContext](#) ()
The default constructor initializes the mCtx member variable and sets the reference count variable (mCount) to zero.
- virtual EXTRTMETHOD [~OSRTContext](#) ()
The destructor frees all memory held by the context.
- OSCTXT * [getPtr](#) ()
The getPtr method returns a pointer to the mCtx member variable.
- EXTRTMETHOD 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.
- EXTRTMETHOD void [_ref](#) ()
The _ref method increases the reference count by one.
- EXTRTMETHOD void [_unref](#) ()
The _unref method decreases the reference count by one.
- EXTRTMETHOD char * [getErrorInfo](#) ()
Returns error text in a dynamic memory buffer.
- EXTRTMETHOD char * [getErrorInfo](#) (size_t *pBufSize)
Returns error text in a dynamic memory buffer.
- EXTRTMETHOD 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 * [memAllocZ](#) (size_t numocts)
The memAllocZ method allocates and zeroes 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 EXTRTMETHOD 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.

7.7.1 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 OSRTCContext.h.

7.7.2 Member Function Documentation

7.7.2.1 EXTRTMETHOD 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.

7.7.2.2 EXTRTMETHOD 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, or (size_t)-1 if an error occurred.

Returns

A pointer to a newly allocated buffer with error text, or NULL if an error occurred.

7.7.2.3 EXTRTMETHOD 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, or NULL if an error occurred.

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

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

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

Definition at line 110 of file `OSRTContext.h`.

7.7.2.5 `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`.

7.7.2.6 `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`.

7.7.2.7 `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 196 of file `OSRTContext.h`.

7.7.2.8 `void* OSRTContext::memAllocZ (size_t numocts) [inline]`

The `memAllocZ` method allocates and zeroes 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 209 of file `OSRTContext.h`.

7.7.2.9 `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 `rtxMemAlloc` function with the context returned by the `getCtxtPtr` method.

Definition at line 219 of file `OSRTContext.h`.

7.7.2.10 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 231 of file OSRTContext.h.

7.7.2.11 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 244 of file OSRTContext.h.

7.7.2.12 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 278 of file OSRTContext.h.

7.7.2.13 virtual EXTRMETHOD 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.

7.7.2.14 `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 300 of file OSRTContext.h.

7.7.3 Member Data Documentation

7.7.3.1 `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:

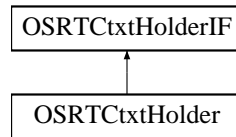
- [OSRTContext.h](#)

7.8 OSRTCtxtHolder Class Reference

Abstract message buffer or stream interface class.

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

Inheritance diagram for OSRTCtxtHolder:



Public Member Functions

- EXTRTMETHOD [OSRTCtxtHolder](#) ([OSRTCContext](#) *pContext=0)
The default constructor creates a new context and sets the buffer class type.
- virtual EXTRTMETHOD [OSRTCtxtPtr](#) [getContext](#) ()
The [getContext](#) method returns the underlying context smart-pointer object.
- virtual EXTRTMETHOD OSCTXT * [getCtxtPtr](#) ()
The [getCtxtPtr](#) method returns the underlying C runtime context.
- virtual EXTRTMETHOD char * [getErrorInfo](#) ()
Returns error text in a dynamic memory buffer.
- virtual EXTRTMETHOD char * [getErrorInfo](#) (char *pBuf, size_t &bufSize)
Returns error text in a memory buffer.
- virtual EXTRTMETHOD int [getStatus](#) () const
This method returns the completion status of previous operation.
- virtual EXTRTMETHOD void [printErrorInfo](#) ()
The [printErrorInfo](#) method prints information on errors contained within the context.
- virtual EXTRTMETHOD void [resetErrorInfo](#) ()
The [resetErrorInfo](#) method resets information on errors contained within the context.

Protected Attributes

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

7.8.1 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.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 EXTRTMETHOD OSRTCtxtHolder::OSRTCtxtHolder (OSRTCContext * *pContext* = 0)

The default 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.

7.8.3 Member Function Documentation

7.8.3.1 virtual EXTRTMETHOD OSRTCtxtPtr OSRTCtxtHolder::getContext () [virtual]

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

Returns

Context smart pointer object.

Implements [OSRTCtxtHolderIF](#).

7.8.3.2 virtual EXTRTMETHOD 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](#).

7.8.3.3 virtual EXTRTMETHOD 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](#).

7.8.3.4 virtual EXTRTMETHOD 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](#).

7.8.3.5 virtual EXTRTMETHOD 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](#).

7.8.4 Member Data Documentation

7.8.4.1 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:

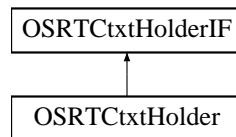
- [OSRTCtxtHolder.h](#)

7.9 OSRTCtxtHolderIF Class Reference

Abstract message buffer or stream interface class.

```
#include <OSRTCtxtHolderIF.h>
```

Inheritance diagram for OSRTCtxtHolderIF:



Public Member Functions

- virtual `OSRTCtxtPtr getContext ()=0`
The `getContext` method returns the underlying context smart-pointer object.
- virtual `OSCTXT * getCtxtPtr ()=0`
The `getCtxtPtr` method returns the underlying C runtime context.
- virtual `char * getErrorInfo ()=0`
Returns error text in a dynamic memory buffer.
- virtual `char * getErrorInfo (char *pBuf, size_t &bufSize)=0`
Returns error text in a memory buffer.
- virtual `int getStatus () const =0`
This method returns the completion status of previous operation.
- virtual `void printErrorInfo ()=0`
The `printErrorInfo` method prints information on errors contained within the context.
- virtual `void resetErrorInfo ()=0`
The `resetErrorInfo` method resets information on errors contained within the context.

Protected Member Functions

- virtual `~OSRTCtxtHolderIF ()`
The virtual destructor does nothing.

7.9.1 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 OSRTCtxtHolderIF.h.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 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.

7.9.3 Member Function Documentation

7.9.3.1 virtual OSRTCtxtPtr OSRTCtxtHolderIF::getContext () [pure virtual]

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

Returns

Context smart pointer object.

Implemented in [OSRTCtxtHolder](#).

7.9.3.2 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](#).

7.9.3.3 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](#).

7.9.3.4 `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](#).

7.9.3.5 `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](#).

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

- [OSRTCtxtHolderIF.h](#)

7.10 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.*
- **OSRTContext *** **operator->** ()
The '->' operator returns the context object pointer.
- **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.

7.10.1 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 313 of file `OSRTContext.h`.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 `OSRTCtxtPtr::OSRTCtxtPtr (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 328 of file `OSRTContext.h`.

7.10.2.2 `OSRTCtxtPtr::OSRTCtxtPtr (const OSRTCtxtPtr & 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 `OSRTCtxtPtr` object to be copied

Definition at line 338 of file `OSRTContext.h`.

7.10.2.3 `virtual OSRTCtxtPtr::~~OSRTCtxtPtr () [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 347 of file `OSRTContext.h`.

7.10.3 Member Function Documentation

7.10.3.1 `OSRTCtxtPtr& OSRTCtxtPtr::operator= (const OSRTCtxtPtr & rf) [inline]`

This assignment operator assigns this `OSRTCtxtPtr` 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 `OSRTCtxtPtr` smart-pointer object

Definition at line 357 of file OSRTContext.h.

References OSRTContext::_ref(), and mPointer.

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

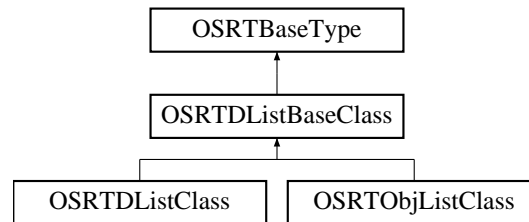
- [OSRTContext.h](#)

7.11 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

- [OSRTDListBaseClass \(\)](#)
The default constructor initializes the list contents to empty.
- virtual [~OSRTDListBaseClass \(\)](#)
The destructor will delete all the nodes in the list.
- `OSSIZE` [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.

7.11.1 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 179 of file `rtxCppDList.h`.

7.11.2 Member Function Documentation

7.11.2.1 `OSSIZE OSRTDListBaseClass::getCount ()` const `[inline]`

This method returns count of items in the list.

Returns

- Count of items in the list

Definition at line 202 of file rtxCppDList.h.

7.11.2.2 `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 220 of file rtxCppDList.h.

7.11.2.3 `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 211 of file rtxCppDList.h.

7.11.2.4 `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:

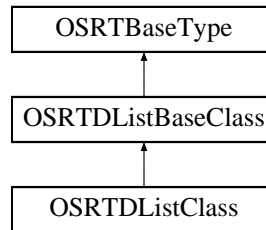
- [rtxCppDList.h](#)

7.12 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.

7.12.1 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 240 of file `rtxCppDList.h`.

7.12.2 Member Function Documentation

7.12.2.1 `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.

7.12.2.2 `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.

7.12.2.3 `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 284 of file `rtxCppDList.h`.

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

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

Returns

- Pointer to head node.

Definition at line 275 of file `rtxCppDList.h`.

7.12.2.5 `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 295 of file `rtxCppDList.h`.

References `OSRTDListNodeClass::getData()`.

7.12.2.6 `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 315 of file `rtxCppDList.h`.

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

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

Returns

- Pointer to tail node.

Definition at line 306 of file `rtxCppDList.h`.

7.12.2.8 `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:

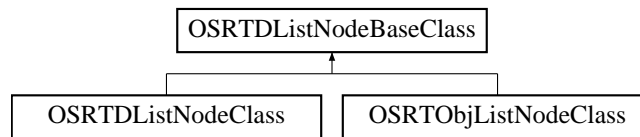
- [rtxCppDList.h](#)

7.13 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:



Friends

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

7.13.1 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 37 of file rtxCppDList.h.

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

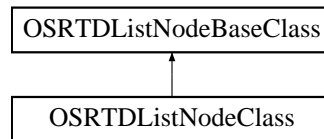
- [rtxCppDList.h](#)

7.14 OSRTDListNodeClass Class Reference

This class represents a doubly-linked list node structure.

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

Inheritance diagram for OSRTDListNodeClass:



Public Member Functions

- `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.

7.14.1 Detailed Description

This class represents a doubly-linked list node structure. It extends the C++ `OSRTDListNodeBaseClass` type. Definition at line 55 of file `rtxCppDList.h`.

7.14.2 Member Function Documentation

7.14.2.1 `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 73 of file `rtxCppDList.h`.

7.14.2.2 `void* OSRTDListNodeClass::getData () [inline]`

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

Returns

Node data pointer.

Definition at line 66 of file `rtxCppDList.h`.

Referenced by `OSRTDListClass::getItem()`.

7.14.2.3 `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 89 of file `rtxCppDList.h`.

7.14.2.4 `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 80 of file `rtxCppDList.h`.

7.14.2.5 `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 107 of file `rtxCppDList.h`.

7.14.2.6 `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 98 of file `rtxCppDList.h`.

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

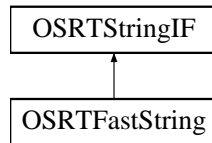
- [rtxCppDList.h](#)

7.15 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.

7.15.1 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.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 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

7.15.2.2 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

7.15.2.3 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.

7.15.3 Member Function Documentation

7.15.3.1 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.

7.15.3.2 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](#).

7.15.3.3 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](#).

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

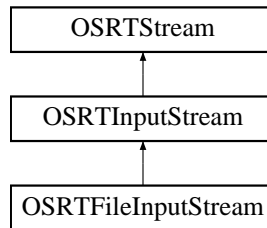
- [OSRTFastString.h](#)

7.16 OSRTFileInputStream Class Reference

Generic file input stream.

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

Inheritance diagram for OSRTFileInputStream:



Public Member Functions

- EXTRTMETHOD [OSRTFileInputStream](#) (const char *pFilename)
Creates and initializes a file input stream using the name of file.
- EXTRTMETHOD [OSRTFileInputStream](#) (OSRTContext *pContext, const char *pFilename)
Creates and initializes a file input stream using the name of file.
- EXTRTMETHOD [OSRTFileInputStream](#) (FILE *file)
Initializes the file input stream using the opened FILE structure descriptor.
- EXTRTMETHOD [OSRTFileInputStream](#) (OSRTContext *pContext, FILE *file)
Initializes the file input stream using the opened FILE structure descriptor.
- virtual OSBOOL [isA](#) (StreamID id) const
This method is used to query a stream object in order to determine its actual type.

7.16.1 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.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 EXTRTMETHOD 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

7.16.2.2 EXTRTMETHOD 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

7.16.2.3 EXTRTMETHOD OSRTFileInputStream::OSRTFileInputStream (FILE * *file*)

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

Parameters

file Pointer to FILE structure.

See also

rtxStreamFileAttach

7.16.2.4 EXTRTMETHOD 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

7.16.3 Member Function Documentation

7.16.3.1 virtual OSBOOL OSRTFileInputStream::isA (StreamID *id*) const [inline, virtual]

This method is used to query a stream object in order to determine its actual type.

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented from [OSRTInputStream](#).

Definition at line 82 of file OSRTFileInputStream.h.

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

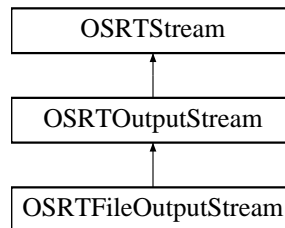
- [OSRTFileInputStream.h](#)

7.17 OSRTFileOutputStream Class Reference

Generic file output stream.

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

Inheritance diagram for OSRTFileOutputStream:



Public Member Functions

- EXTRTMETHOD [OSRTFileOutputStream](#) (const char *pFilename)
Creates and initializes a file output stream using the name of file.
- EXTRTMETHOD [OSRTFileOutputStream](#) (OSRTContext *pContext, const char *pFilename)
Creates and initializes a file output stream using the name of file.
- EXTRTMETHOD [OSRTFileOutputStream](#) (FILE *file)
Initializes the file output stream using the opened FILE structure descriptor.
- EXTRTMETHOD [OSRTFileOutputStream](#) (OSRTContext *pContext, FILE *file)
Initializes the file output stream using the opened FILE structure descriptor.
- virtual OSBOOL [isA](#) (StreamID id) const
This method is used to query a stream object in order to determine its actual type.

7.17.1 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.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 EXTRTMETHOD 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

7.17.2.2 EXTRTMETHOD 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

7.17.2.3 EXTRTMETHOD 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

7.17.2.4 EXTRTMETHOD 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`

7.17.3 Member Function Documentation

7.17.3.1 `virtual OSBOOL OSRTFileOutputStream::isA (StreamID id) const` `[inline, virtual]`

This method is used to query a stream object in order to determine its actual type.

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented from [OSRTOutputStream](#).

Definition at line 86 of file `OSRTFileOutputStream.h`.

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

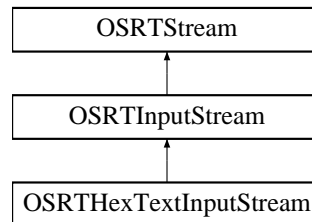
- [OSRTFileOutputStream.h](#)

7.18 OSRTHexTextInputStream Class Reference

Hexadecimal text input stream filter class.

```
#include <OSRTHexTextInputStream.h>
```

Inheritance diagram for OSRTHexTextInputStream:



Public Member Functions

- EXTRTMETHOD [OSRTHexTextInputStream \(OSRTInputStream *pstream\)](#)
Initializes the input stream using the existing standard input stream.
- EXTRTMETHOD [~OSRTHexTextInputStream \(\)](#)
The destructor deletes the underlying stream object.
- virtual OSBOOL [isA \(StreamID id\) const](#)
This method is used to query a stream object in order to determine its actual type.
- void [setOwnUnderStream \(OSBOOL value=TRUE\)](#)
This method transfers ownership of the underlying stream to the class.

7.18.1 Detailed Description

Hexadecimal text input stream filter class. This class is created on top of an existing stream class to provide conversion of hexadecimal text input into binary form.

Definition at line 38 of file OSRTHexTextInputStream.h.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 EXTRTMETHOD OSRTHexTextInputStream::OSRTHexTextInputStream (OSRTInputStream *pstream)

Initializes the input stream using the existing standard input stream.

Only file and memory underlying stream types are supported.

Parameters

- pstream* The underlying input stream object. Note that this class will take control of the underlying stream object and delete it upon destruction.

See also

`rtxStreamHexTextAttach`

7.18.2.2 EXTRMETHOD OSRTHexTextInputStream::~~OSRTHexTextInputStream ()

The destructor deletes the underlying stream object.

That object should be used as nothing more to a surrogate to this object.

7.18.3 Member Function Documentation

7.18.3.1 virtual OSBOOL OSRTHexTextInputStream::isA (StreamID *id*) const [inline, virtual]

This method is used to query a stream object in order to determine its actual type.

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented from [OSRTInputStream](#).

Definition at line 70 of file `OSRTHexTextInputStream.h`.

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

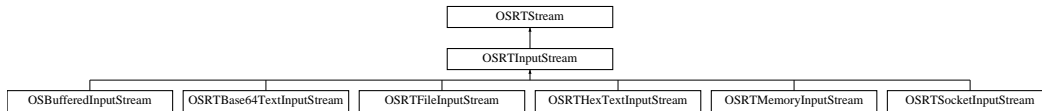
- [OSRTHexTextInputStream.h](#)

7.19 OSRTInputStream Class Reference

This is the base class for input streams.

```
#include <OSRTInputStream.h>
```

Inheritance diagram for OSRTInputStream:



Public Member Functions

- EXTRTMETHOD [OSRTInputStream \(\)](#)
The default constructor.
- virtual EXTRTMETHOD [~OSRTInputStream \(\)](#)
Virtual destructor.
- virtual EXTRTMETHOD int [close \(\)](#)
Closes the input or output stream and releases any system resources associated with the stream.
- virtual EXTRTMETHOD size_t [currentPos \(\)](#)
This method returns the current position in the stream (in octets).
- virtual EXTRTMETHOD int [flush \(\)](#)
Flushes the buffered data to the stream.
- virtual OSBOOL [isA \(StreamID id\) const](#)
This method is used to query a stream object in order to determine its actual type.
- virtual [OSRTCtxtPtr getContext \(\)](#)
This method returns a pointer to the underlying [OSRTCtxt](#) 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 [getPosition \(size_t *ppos\)](#)
Returns the current stream position.
- virtual int [getStatus \(\) const](#)
This method returns the completion status of previous operation.

- virtual EXTRTMETHOD OSBOOL `isOpened ()`
Checks, is the stream opened or not.
- virtual EXTRTMETHOD OSBOOL `markSupported ()`
Tests if this input stream supports the mark and reset methods.
- virtual EXTRTMETHOD 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 EXTRTMETHOD long `read (OSOCKETET *pDestBuf, size_t maxToRead)`
Read data from the stream.
- virtual EXTRTMETHOD long `readBlocking (OSOCKETET *pDestBuf, size_t toReadBytes)`
Read data from the stream.
- virtual EXTRTMETHOD int `reset ()`
Repositions this stream to the position at the time the mark method was last called on this input stream.
- virtual int `setPosition (size_t pos)`
Sets the current stream position to the given offset.
- virtual EXTRTMETHOD int `skip (size_t n)`
Skips over and discards the specified amount of data octets from this input stream.

7.19.1 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.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 EXTRTMETHOD 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.

7.19.2.2 virtual EXTRTMETHOD OSRTInputStream::~OSRTInputStream () [virtual]

Virtual destructor.

Closes the stream if it was opened.

7.19.3 Member Function Documentation

7.19.3.1 virtual EXTRTMETHOD 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`

Reimplemented from [OSRTStream](#).

7.19.3.2 virtual EXTRTMETHOD 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.

7.19.3.3 virtual EXTRTMETHOD 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`

Reimplemented from [OSRTStream](#).

7.19.3.4 virtual OSRTCtxtPtr 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 110 of file OSRTInputStream.h.

References OSRTStream::getContext().

7.19.3.5 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 120 of file OSRTInputStream.h.

References OSRTStream::getCtxtPtr().

Referenced by OSRTBase64TextInputStream::isCertificate().

7.19.3.6 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 151 of file OSRTInputStream.h.

References OSRTStream::getErrorInfo().

7.19.3.7 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 131 of file OSRTInputStream.h.

References OSRTStream::getErrorInfo().

7.19.3.8 virtual int OSRTInputStream::getPosition (size_t * ppos) [virtual]

Returns the current stream position.

This may be used with the `setPosition` method to reset back to an arbitrary point in the input stream.

Parameters

ppos Pointer to a variable to receive position.

Returns

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

7.19.3.9 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 175 of file OSRTInputStream.h.

References OSRTStream::getStatus().

7.19.3.10 virtual OSBOOL OSRTInputStream::isA (StreamID id) const [inline, virtual]

This method is used to query a stream object in order to determine its actual type.

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented in [OSRTBase64TextInputStream](#), [OSRTFileInputStream](#), [OSRTHexTextInputStream](#), [OSRTMemoryInputStream](#), and [OSRTSocketInputStream](#).

Definition at line 97 of file [OSRTInputStream.h](#).

7.19.3.11 virtual EXTRTMETHOD OSBOOL OSRTInputStream::isOpen() [virtual]

Checks, is the stream opened or not.

Returns

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

See also

[rtxStreamIsOpened](#)

Reimplemented from [OSRTStream](#).

7.19.3.12 virtual EXTRTMETHOD 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](#)

7.19.3.13 virtual EXTRTMETHOD 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

[rtxStreamMarkSupported](#)

7.19.3.14 virtual EXTRTMETHOD 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`

7.19.3.15 virtual EXTRTMETHOD 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`

7.19.3.16 virtual EXTRTMETHOD 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`

7.19.3.17 **virtual int OSRTInputStream::setPosition (size_t pos) [virtual]**

Sets the current stream position to the given offset.

Parameters

pos Position stream is to be reset to. This is normally obtained via a call to `getPosition`, although in most cases it is a zero-based offset.

Returns

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

7.19.3.18 **virtual EXTRMETHOD 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`

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

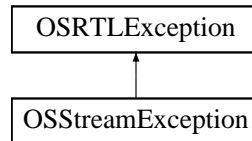
- [OSRTInputStream.h](#)

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 Attributes

- 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>
```

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:

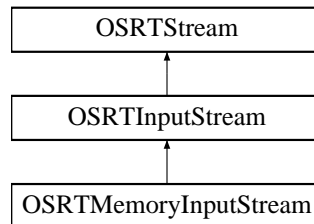
- [OSRTMemBuf.h](#)

7.22 OSRTMemoryInputStream Class Reference

Generic memory input stream.

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

Inheritance diagram for OSRTMemoryInputStream:



Public Member Functions

- EXTRTMETHOD [OSRTMemoryInputStream](#) (const OSOCTET *pMemBuf, size_t bufSize)
Initializes the memory input stream using the specified memory buffer.
- EXTRTMETHOD [OSRTMemoryInputStream](#) (OSRTContext *pContext, const OSOCTET *pMemBuf, size_t bufSize)
Initializes the memory input stream using the specified memory buffer.
- virtual OSBOOL [isA](#) (StreamID id) const
This method is used to query a stream object in order to determine its actual type.

7.22.1 Detailed Description

Generic memory input stream. This class provides methods for streaming data from an input memory buffer.

Definition at line 37 of file OSRTMemoryInputStream.h.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 EXTRTMETHOD 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 EXTRTMETHOD OSRTMemoryInputStream::OSRTMemoryInputStream (OSRTContext * *pContext*, const OSOCTET * *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`

7.22.3 Member Function Documentation

7.22.3.1 virtual OSBOOL OSRTMemoryInputStream::isA (StreamID *id*) const [`inline`, `virtual`]

This method is used to query a stream object in order to determine its actual type.

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented from [OSRTInputStream](#).

Definition at line 66 of file `OSRTMemoryInputStream.h`.

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

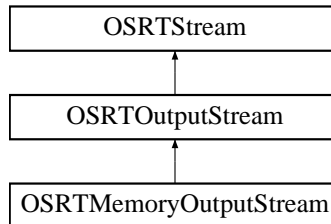
- [OSRTMemoryInputStream.h](#)

7.23 OSRTMemoryOutputStream Class Reference

Generic memory output stream.

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

Inheritance diagram for OSRTMemoryOutputStream:



Public Member Functions

- EXTRTMETHOD [OSRTMemoryOutputStream \(\)](#)
The default constructor initializes the memory output stream to use a dynamic memory output buffer.
- EXTRTMETHOD [OSRTMemoryOutputStream \(OSOCKET *pMemBuf, size_t bufSize\)](#)
Initializes the memory output stream using the specified memory buffer.
- EXTRTMETHOD [OSRTMemoryOutputStream \(OSRTContext *pContext, OSOCKET *pMemBuf, size_t bufSize\)](#)
Initializes the memory output stream using the specified memory buffer.
- EXTRTMETHOD OSOCKET * [getBuffer](#) (size_t *pSize=0)
This method returns the address of the memory buffer to which data was written.
- virtual OSBOOL [isA](#) (StreamID id) const
This method is used to query a stream object in order to determine its actual type.
- int [reset](#) ()
This method resets the output memory stream internal buffer to allow it to be overwritten with new data.

7.23.1 Detailed Description

Generic memory output stream. This class provides methods for streaming data to an output memory buffer.

Definition at line 37 of file OSRTMemoryOutputStream.h.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 EXTRTMETHOD OSRTMemoryOutputStream::OSRTMemoryOutputStream ()

The default constructor initializes the memory output stream to use a dynamic memory output buffer.

The status of the construction can be obtained by calling the `getStatus` method.

See also

rtxStreamMemoryCreate

7.23.2.2 EXTRTMETHOD OSRTMemoryOutputStream::OSRTMemoryOutputStream (OSOCKET * *pMemBuf*, size_t *bufSize*)

Initializes the memory output stream using the specified memory buffer.

The status of the construction can be obtained by calling the `getStatus` method.

Parameters

pMemBuf The pointer to the buffer.

bufSize The size of the buffer.

See also

rtxStreamMemoryAttach

7.23.2.3 EXTRTMETHOD OSRTMemoryOutputStream::OSRTMemoryOutputStream (OSRTContext * *pContext*, OSOCKET * *pMemBuf*, size_t *bufSize*)

Initializes the memory output stream using the specified memory buffer.

The status of the construction can be obtained by calling the `getStatus` method.

Parameters

pContext Pointer to a context to use.

pMemBuf The pointer to the buffer.

bufSize The size of the buffer.

See also

rtxStreamMemoryAttach

7.23.3 Member Function Documentation

7.23.3.1 EXTRTMETHOD OSOCKET* OSRTMemoryOutputStream::getBuffer (size_t * *pSize* = 0)

This method returns the address of the memory buffer to which data was written.

If the buffer memory is dynamic, it may be freed using the `rtxMemFreePtr` function or it will be freed when the stream object is destroyed.

Parameters

pSize Pointer to a size variable to receive the number of bytes written to the stream. This is an optional parameter, if a null pointer is passed, size is not returned.

Returns

Pointer to memory buffer.

7.23.3.2 virtual OSBOOL OSRTMemoryOutputStream::isA (StreamID *id*) const [inline, virtual]

This method is used to query a stream object in order to determine its actual type.

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented from [OSRTOutputStream](#).

Definition at line 92 of file OSRTMemoryOutputStream.h.

7.23.3.3 int OSRTMemoryOutputStream::reset ()

This method resets the output memory stream internal buffer to allow it to be overwritten with new data.

Memory for the buffer is not freed.

Returns

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

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

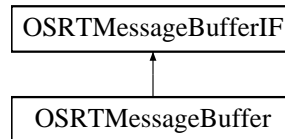
- [OSRTMemoryOutputStream.h](#)

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 `OSOCTET * getMsgCopy ()`
The `getMsgCopy` method will return a copy of the encoded message managed by the object.
- virtual `const OSOCTET * 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 `EXTRTMETHOD int initBuffer (OSOCTET *pMsgBuf, size_t msgBufLen)`

This version of the overloaded `initWithBuffer` 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 EXTRTMETHOD void `setDiag` (OSBOOL value=TRUE)
The `setDiag` method will turn diagnostic tracing on or off.

Protected Member Functions

- EXTRTMETHOD `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 EXTRTMETHOD `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.

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.

Implements [OSRTMessageBufferIF](#).

Definition at line 102 of file OSRTMsgBuf.h.

7.24.3.3 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.

Definition at line 133 of file OSRTMsgBuf.h.

7.24.3.4 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.

Definition at line 113 of file OSRTMsgBuf.h.

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

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.

Definition at line 162 of file OSRTMsgBuf.h.

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 EXTRTMETHOD 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 EXTRTMETHOD 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:

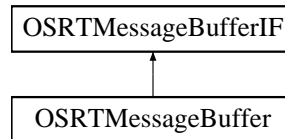
- [OSRTMsgBuf.h](#)

7.25 OSRTMessageBufferIF Class Reference

Abstract message buffer or stream interface class.

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

Inheritance diagram for OSRTMessageBufferIF:



Public 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](#) (Type 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 OSOCKET* 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 (OSOCKET * *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 (Type *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:

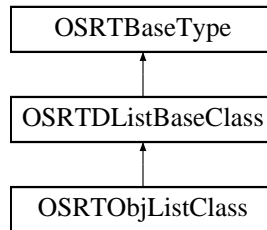
- [OSRTMsgBufIF.h](#)

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.
- **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 339 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 `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 384 of file `rtxCppDList.h`.

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

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

Returns

- Pointer to head node.

Definition at line 375 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 395 of file `rtxCppDList.h`.

References `OSRTObjListNodeClass::getData()`.

7.26.2.6 `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 415 of file `rtxCppDList.h`.

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

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

Returns

- Pointer to tail node.

Definition at line 406 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:

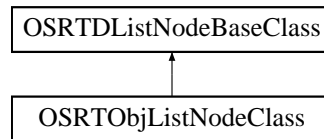
- [rtxCppDList.h](#)

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 116 of file [rtxCppDList.h](#).

7.27.2 Member Function Documentation

7.27.2.1 `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 134 of file [rtxCppDList.h](#).

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

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

Returns

Node data pointer.

Definition at line 127 of file rtxCppDList.h.

Referenced by OSRTObjListClass::getItem().

7.27.2.3 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 150 of file rtxCppDList.h.

7.27.2.4 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 141 of file rtxCppDList.h.

7.27.2.5 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 168 of file rtxCppDList.h.

7.27.2.6 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 159 of file rtxCppDList.h.

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

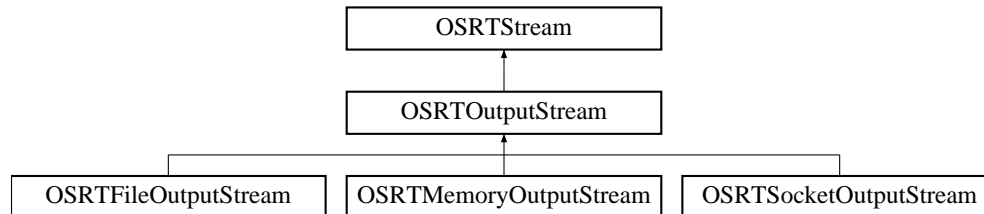
- [rtxCppDList.h](#)

7.28 OSRTOutputStream Class Reference

The base class definition for operations with output streams.

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

Inheritance diagram for OSRTOutputStream:



Public Member Functions

- EXTRTMETHOD [OSRTOutputStream \(\)](#)
The default constructor.
- virtual EXTRTMETHOD [~OSRTOutputStream \(\)](#)
Virtual destructor.
- virtual EXTRTMETHOD int [close \(\)](#)
Closes the output or output stream and releases any system resources associated with the stream.
- virtual EXTRTMETHOD int [flush \(\)](#)
Flushes the buffered data to the stream.
- virtual [OSRTCtxtPtr getContext \(\)](#)
This method returns a pointer to the underlying [OSRTCtxt](#) 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 [isA \(StreamID id\) const](#)
This method is used to query a stream object in order to determine its actual type.
- virtual EXTRTMETHOD OSBOOL [isOpen \(\)](#)
Checks if the stream open 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 EXTRTMETHOD long `write (const OSOCKET *pdata, size_t size)`
Write data to the stream.
- virtual EXTRTMETHOD long `write (const char *pdata)`
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 EXTRTMETHOD 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 EXTRTMETHOD OSRTOutputStream::~OSRTOutputStream () [virtual]

Virtual destructor.

Closes the stream if it was opened.

7.28.3 Member Function Documentation

7.28.3.1 virtual EXTRTMETHOD 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`

Reimplemented from [OSRTStream](#).

7.28.3.2 virtual EXTRTMETHOD 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`

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 94 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 104 of file `OSRTOutputStream.h`.

References `OSRTStream::getCtxtPtr()`.

7.28.3.5 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 135 of file OSRTOutputStream.h.

References OSRTStream::getErrorInfo().

7.28.3.6 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 115 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 148 of file OSRTOutputStream.h.

References OSRTStream::getStatus().

7.28.3.8 virtual OSBOOL OSRTOutputStream::isA (StreamID *id*) const [inline, virtual]

This method is used to query a stream object in order to determine its actual type.

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented in [OSRTFileOutputStream](#), [OSRTMemoryOutputStream](#), and [OSRTSocketOutputStream](#).

Definition at line 159 of file OSRTOutputStream.h.

7.28.3.9 virtual EXTRTMETHOD OSBOOL OSRTOutputStream::isOpen () [virtual]

Checks if the stream open or not.

Returns

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

See also

[rtxStreamIsOpened](#)

Reimplemented from [OSRTStream](#).

7.28.3.10 virtual EXTRTMETHOD long OSRTOutputStream::write (const char * *pdata*) [virtual]

Write data to the stream.

This method writes data from a null-terminated character string to the output stream.

Parameters

pdata The pointer to the data to be written.

Returns

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

See also

[rtxStreamWrite](#)

7.28.3.11 virtual EXTRTMETHOD 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

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

- [OSRTOutputStream.h](#)

7.29 OSRTSocket Class Reference

Wrapper class for TCP/IP or UDP sockets.

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

Public Member Functions

- EXTRTMETHOD [OSRTSocket](#) ()
This is the default constructor.
- EXTRTMETHOD [OSRTSocket](#) (OSRTOCKET socket, OSBOOL ownership=FALSE)
This constructor initializes an instance by using an existing socket.
- EXTRTMETHOD [OSRTSocket](#) (const [OSRTSocket](#) &socket)
The copy constructor.
- EXTRTMETHOD [~OSRTSocket](#) ()
The destructor.
- EXTRTMETHOD [OSRTSocket](#) * [accept](#) (OSIPADDR *destIP=0, int *port=0)
This method permits an incoming connection attempt on a socket.
- EXTRTMETHOD int [bind](#) (OSIPADDR addr, int port)
This method associates a local address with a socket.
- EXTRTMETHOD int [bindUrl](#) (const char *url)
This method associates a local address with a socket.
- EXTRTMETHOD 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.
- EXTRTMETHOD int [blockingRead](#) (OSOCKET *pbuf, size_t readBytes)
This method receives data from the connected socket.
- EXTRTMETHOD int [close](#) ()
This method closes this socket.
- EXTRTMETHOD int [connect](#) (const char *host, int port)
This method establishes a connection to this socket.
- EXTRTMETHOD int [connectUrl](#) (const char *url)
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.
- EXTRTMETHOD int [listen](#) (int maxConnections)
This method places a socket into a state where it is listening for an incoming connection.
- EXTRTMETHOD int [recv](#) (OSOCKET *pbuf, size_t 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.
- EXTRTMETHOD int [send](#) (const OSOCKET *pdata, size_t size)
This method sends data on a connected socket.

Static Public Member Functions

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

Protected Attributes

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

7.29.1 Detailed Description

Wrapper class for TCP/IP or UDP sockets.

Definition at line 50 of file OSRTSocket.h.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 EXTRTMETHOD 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.

7.29.2.2 EXTRTMETHOD OSRTSocket::OSRTSocket (OSR SOCKET *socket*, OSBOOL *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.

7.29.2.3 EXTRTMETHOD 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.

7.29.2.4 EXTRTMETHOD OSRTSocket::~OSRTSocket ()

The destructor.

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

7.29.3 Member Function Documentation

7.29.3.1 EXTRTMETHOD OSRTSocket* OSRTSocket::accept (OSIPADDR * *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 [OSRTSocket::getStatus](#) method to obtain error code.

See also

rtxSocketAccept

7.29.3.2 static EXTRTMETHOD const char* OSRTSocket::addrToString (OSIPADDR *ipAddr*, char * *pAddrStr*, size_t *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.

7.29.3.3 `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

`rtxSocketBind`
[bind \(\)](#)

Definition at line 183 of file OSRTSocket.h.

7.29.3.4 `EXTRTMETHOD 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 [OSRTSocket::connect](#) or [OSRTSocket::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

`rtxSocketBind`

7.29.3.5 EXTRTMETHOD 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 [OSRTSocket::connect](#) or [OSRTSocket::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

[rtxSocketBind](#)

7.29.3.6 EXTRTMETHOD int OSRTSocket::bindUrl (const char * *url*)

This method associates a local address with a socket.

It is used on an unconnected socket before subsequent calls to the [OSRTSocket::connect](#) or [OSRTSocket::listen](#) methods. This version of the method allows a URL to be used instead of address and port number.

Parameters

url Universal resource locator (URL) string.

Returns

Completion status of operation:

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

See also

[rtxSocketBind](#)

7.29.3.7 EXTRTMETHOD 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.

7.29.3.8 EXTRTMETHOD int OSRTSocket::close ()

This method closes this socket.

Returns

Completion status of operation:

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

See also

rtxSocketClose

7.29.3.9 EXTRTMETHOD 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

rtxSocketConnect

7.29.3.10 EXTRTMETHOD int OSRTSocket::connectUrl (const char * *url*)

This method establishes a connection to this socket.

It is used to create a connection to the specified destination. In this version, destination is specified using a URL.

Parameters

url Universal resource locator (URL) string.

Returns

Completion status of operation:

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

See also

rtxSocketConnect

7.29.3.11 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 246 of file OSRTSocket.h.

7.29.3.12 OSRTSOCKET OSRTSocket::getSocket () const [inline]

This method returns the handle of the socket.

Returns

The handle of the socket.

Definition at line 253 of file OSRTSocket.h.

7.29.3.13 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 262 of file OSRTSocket.h.

7.29.3.14 EXTRTMETHOD 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

rtxSocketListen

7.29.3.15 EXTRMETHOD int OSRTSocket::recv (OSOCKET * *pbuf*, size_t *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

rtxSocketRecv

7.29.3.16 EXTRMETHOD int OSRTSocket::send (const OSOCKET * *pdata*, size_t *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

rtxSocketSend

7.29.3.17 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 301 of file OSRTSocket.h.

**7.29.3.18 static EXTRTMETHOD 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:

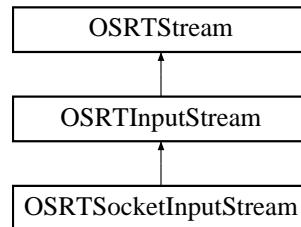
- [OSRTSocket.h](#)

7.30 OSRTSocketInputStream Class Reference

Generic socket input stream.

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

Inheritance diagram for OSRTSocketInputStream:



Public Member Functions

- EXTRTMETHOD [OSRTSocketInputStream](#) ([OSRTSocket](#) &socket)
Creates and initializes a socket input stream using the [OSRTSocket](#) instance of socket.
- EXTRTMETHOD [OSRTSocketInputStream](#) ([OSRTContext](#) *pContext, [OSRTSocket](#) &socket)
Creates and initializes a socket input stream using the [OSRTSocket](#) instance of socket.
- EXTRTMETHOD [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.
- virtual [OSBOOL](#) [isA](#) ([StreamID](#) id) const
This method is used to query a stream object in order to determine its actual type.

Protected Attributes

- [OSRTSocket](#) [mSocket](#)
a socket

7.30.1 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 40 of file OSRTSocketInputStream.h.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 EXTRTMETHOD 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

[rtxStreamSocketAttach](#)

7.30.2.2 EXTRTMETHOD 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

[rtxStreamSocketAttach](#)

7.30.2.3 EXTRTMETHOD 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](#)

7.30.2.4 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](#)

7.30.3 Member Function Documentation

7.30.3.1 virtual OSBOOL OSRTSocketInputStream::isA (StreamID *id*) const [inline, virtual]

This method is used to query a stream object in order to determine its actual type.

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented from [OSRTInputStream](#).

Definition at line 105 of file OSRTSocketInputStream.h.

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

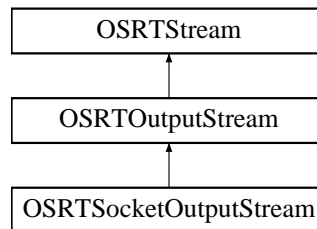
- [OSRTSocketInputStream.h](#)

7.31 OSRTSocketOutputStream Class Reference

Generic socket output stream.

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

Inheritance diagram for OSRTSocketOutputStream:



Public Member Functions

- EXTRTMETHOD [OSRTSocketOutputStream](#) ([OSRTSocket](#) &socket)
Creates and initializes a socket output stream using the [OSRTSocket](#) instance of socket.
- EXTRTMETHOD [OSRTSocketOutputStream](#) ([OSRTContext](#) *pContext, [OSRTSocket](#) &socket)
Creates and initializes a socket output stream using the [OSRTSocket](#) instance of socket.
- EXTRTMETHOD [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.
- virtual [OSBOOL](#) [isA](#) ([StreamID](#) id) const
This method is used to query a stream object in order to determine its actual type.

Protected Attributes

- [OSRTSocket](#) [mSocket](#)
a socket

7.31.1 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 40 of file [OSRTSocketOutputStream.h](#).

7.31.2 Constructor & Destructor Documentation

7.31.2.1 EXTRTMETHOD [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

[rtxStreamSocketAttach](#)

7.31.2.2 EXTRTMETHOD 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

[rtxStreamSocketAttach](#)

7.31.2.3 EXTRTMETHOD 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](#)

7.31.2.4 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](#)

7.31.3 Member Function Documentation

7.31.3.1 virtual OSBOOL OSRTSocketOutputStream::isA (StreamID *id*) const [inline, virtual]

This method is used to query a stream object in order to determine its actual type.

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented from [OSRTOutputStream](#).

Definition at line 103 of file OSRTSocketOutputStream.h.

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

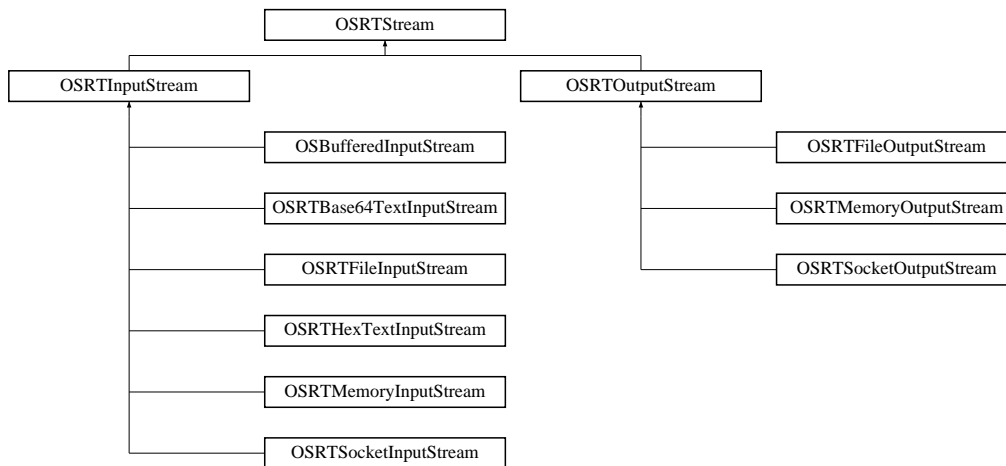
- [OSRTSocketOutputStream.h](#)

7.32 OSRStream Class Reference

The default base class for using I/O streams.

```
#include <OSRStream.h>
```

Inheritance diagram for OSRStream:



Public Member Functions

- virtual EXTRTMETHOD [~OSRStream](#) ()
Virtual destructor.
- virtual EXTRTMETHOD int [close](#) ()
Closes the input or output stream and releases any system resources associated with the stream.
- virtual EXTRTMETHOD int [flush](#) ()
Flushes the buffered data to the stream.
- virtual [OSRTCtxtPtr](#) [getContext](#) ()
This method returns a pointer to the underlying [OSRTCtxt](#) 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.
- virtual EXTRTMETHOD OSBOOL [isOpen](#) ()

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

- EXTRTMETHOD [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.

7.32.1 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.

7.32.2 Constructor & Destructor Documentation

7.32.2.1 EXTRTMETHOD 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.

7.32.2.2 virtual EXTRTMETHOD OSRTStream::~OSRTStream () [virtual]

Virtual destructor.

Closes the stream if it was opened.

7.32.3 Member Function Documentation

7.32.3.1 virtual EXTRTMETHOD 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](#)

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

7.32.3.2 virtual EXTRTMETHOD 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](#)

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

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

Definition at line 101 of file [OSRTStream.h](#).

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

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

Definition at line 111 of file OSRTStream.h.

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

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

Definition at line 142 of file OSRTStream.h.

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

Definition at line 122 of file OSRTStream.h.

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

7.32.3.7 `int OSRTStream::getStatus () const [inline]`

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

Definition at line 155 of file `OSRTStream.h`.

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

7.32.3.8 `virtual EXRTMETHOD OSBOOL OSRTStream::isOpen () [virtual]`

Checks, is the stream opened or not.

Returns

TRUE, if the stream is opened, FALSE otherwise.

See also

`rtxStreamIsOpened`

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

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

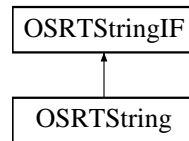
- [OSRTStream.h](#)

7.33 OSRTString Class Reference

C++ string class definition.

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

Inheritance diagram for OSRTString:



Public Member Functions

- EXTRTMETHOD [OSRTString](#) ()
The default constructor creates an empty string.
- EXTRTMETHOD [OSRTString](#) (const char *strval)
This constructor initializes the string to contain the given standard ASCII string value.
- EXTRTMETHOD [OSRTString](#) (const OSUTF8CHAR *strval)
This constructor initializes the string to contain the given UTF-8 string value.
- EXTRTMETHOD [OSRTString](#) (const [OSRTString](#) &str)
Copy constructor.
- virtual EXTRTMETHOD [~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 EXTRTMETHOD void [setValue](#) (const char *str)
This method sets the string value to the given string.
- virtual EXTRTMETHOD void [setValue](#) (const OSUTF8CHAR *str)
This method sets the string value to the given UTF-8 string value.
- EXTRTMETHOD [OSRTString](#) & [operator=](#) (const [OSRTString](#) &original)
Assignment operator.

7.33.1 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.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 EXTRTMETHOD 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

7.33.2.2 EXTRTMETHOD 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

7.33.2.3 EXTRTMETHOD OSRTString::OSRTString (const OSRTString & *str*)

Copy constructor.

Parameters

str - C++ string object to be copied.

7.33.3 Member Function Documentation

7.33.3.1 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.

7.33.3.2 virtual EXTRTMETHOD 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](#).

7.33.3.3 virtual EXTRTMETHOD 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](#).

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

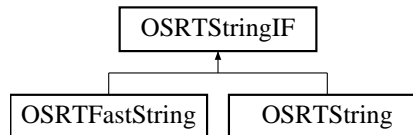
- [OSRTString.h](#)

7.34 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.

7.34.1 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.

7.34.2 Constructor & Destructor Documentation

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

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

Parameters

- Null-terminated C string value

Definition at line 62 of file OSRTStringIF.h.

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

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

Parameters

- Null-terminated C string value

Definition at line 70 of file OSRTStringIF.h.

7.34.3 Member Function Documentation

7.34.3.1 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](#).

7.34.3.2 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](#).

7.34.3.3 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](#).

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

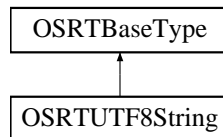
- [OSRTStringIF.h](#)

7.35 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.

7.35.1 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.

7.35.2 Constructor & Destructor Documentation

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

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

Parameters

strval - String value

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

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

Parameters

strval - String value

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

Copy constructor.

Parameters

str - C++ XML string class.

7.35.3 Member Function Documentation

7.35.3.1 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.

7.35.3.2 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.

7.35.3.3 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.

7.35.3.4 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:

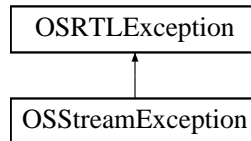
- [OSRTUTF8String.h](#)

7.36 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.

7.36.1 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:

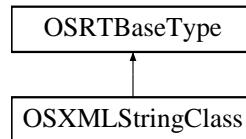
- [rtxCppException.h](#)

7.37 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.
- [int compare \(const OSUTF8CHAR *value2\) const](#)
This method does a standard string comparison operation (strcmp) with the UTF-8 null terminated string.
- [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, OSXMLNames-pace *pNS)`

This method encodes the data in this string object into XML content in the encode data stream.

- `OSBOOL equals (const OSUTF8CHAR *value2) const`

This method compares this string with the UTF-8 null terminated string for equality.

- `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.*

- `size_t length () const`

This method returns the number of characters.

- `size_t size ()`

This method returns the number of bytes.

7.37.1 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`.

7.37.2 Constructor & Destructor Documentation

7.37.2.1 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?

7.37.2.2 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?

7.37.2.3 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?

7.37.2.4 OSXMLStringClass::OSXMLStringClass (const OSXMLSTRING & *str*)

Copy constructor.

Parameters

str - C XML string structure.

7.37.2.5 OSXMLStringClass::OSXMLStringClass (const OSXMLStringClass & *str*)

Copy constructor.

Parameters

str - C++ XML string class.

7.37.3 Member Function Documentation

7.37.3.1 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.

7.37.3.2 OSRTBaseType* OSXMLStringClass::clone () const [inline, virtual]

Clone method.

Creates a copied instance and returns pointer to [OSRTBaseType](#).

Reimplemented from [OSRTBaseType](#).

Definition at line 142 of file `rtxCppXmlString.h`.

7.37.3.3 int OSXMLStringClass::compare (const OSUTF8CHAR * *value2*) const [inline]

This method does a standard string comparison operation (`strcmp`) with the UTF-8 null terminated string.

Returns

Zero (0) if the compared characters sequences are equal, -1 if the internal string value is less than the argument value, and +1 if the internal string value is greater than the argument value.

Definition at line 152 of file `rtxCppXmlString.h`.

7.37.3.4 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 176 of file rtxCppXmlString.h.

7.37.3.5 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.

7.37.3.6 virtual int OSXMLStringClass::decodeXML (OSCTXT * *pctxt*) [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

pctxt Pointer to context block structure.

Returns

Completion status of operation:

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

7.37.3.7 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.

7.37.3.8 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 237 of file rtxCppXmlString.h.

7.37.3.9 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 253 of file rtxCppXmlString.h.

7.37.3.10 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 246 of file rtxCppXmlString.h.

7.37.3.11 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 284 of file rtxCppXmlString.h.

7.37.3.12 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 273 of file rxCppXmlString.h.

7.37.3.13 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.

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

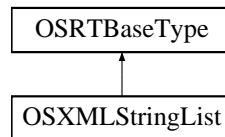
- [rxCppXmlString.h](#)

7.38 OSXMLStringList Class Reference

XML list string.

```
#include <rtxCppXmlStringList.h>
```

Inheritance diagram for OSXMLStringList:



Public Member Functions

- [OSXMLStringList \(\)](#)
The default constructor creates an empty list.
- [OSXMLStringList \(const OSXMLStringList &orig\)](#)
The copy constructor creates a deep-copy of the original list.
- [OSXMLStringList & operator= \(const OSXMLStringList &orig\)](#)
The assignment operator frees the existing list and then makes a deep-copy of the original list.
- void [append \(OSXMLStringClass *pdata\)](#)
The append method adds the given object to the end of the list.
- void [appendCopy \(OSXMLStringClass *pdata\)](#)
The appendCopy method adds a copy of the given object to the end of the list.
- virtual [OSRTBaseType * clone \(\) const](#)
The clone method makes a cloned copy of this object.
- [OSXMLStringClass * getItem \(int idx\)](#)
The getItem method returns a pointer to the indexed item in the list or NULL if the index is out-of-range.

Public Attributes

- [OSRTObjListClass mElemList](#)
List of OSXMLStringClass objects.

7.38.1 Detailed Description

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

Definition at line 39 of file rtxCppXmlStringList.h.

7.38.2 Constructor & Destructor Documentation

7.38.2.1 OSXMLStringList::OSXMLStringList (const OSXMLStringList & *orig*)

The copy constructor creates a deep-copy of the original list.

Parameters

orig Object to be copied.

7.38.3 Member Function Documentation

7.38.3.1 void OSXMLStringList::append (OSXMLStringClass * *pdata*)

The append method adds the given object to the end of the list.

The pointer is assigned directly (i.e. a deep-copy is not made).

Parameters

pdata Pointer to object to be appended.

7.38.3.2 void OSXMLStringList::appendCopy (OSXMLStringClass * *pdata*)

The appendCopy method adds a copy of the given object to the end of the list.

In this case, a deep-copy of the given object is made before appending it to the list.

Parameters

pdata Pointer to object to be appended.

7.38.3.3 virtual OSRTBaseType* OSXMLStringList::clone () const [virtual]

The clone method makes a cloned copy of this object.

It may be used to create a copy of any object derived from this base class.

Reimplemented from [OSRTBaseType](#).

7.38.3.4 OSXMLStringList& OSXMLStringList::operator= (const OSXMLStringList & *orig*)

The assignment operator frees the existing list and then makes a deep-copy of the original list.

Parameters

orig Object to be assigned.

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

- [rtxCppXmlStringList.h](#)

Chapter 8

File Documentation

8.1 OSRTBase64TextInputStream.h File Reference

C++ hexadecimal text input stream filter class.

```
#include "rtxsrc/OSRTInputStream.h"  
#include "rtxsrc/rtxStreamBase64Text.h"
```

Classes

- class [OSRTBase64TextInputStream](#)
Hexadecimal text input stream filter class.

8.1.1 Detailed Description

C++ hexadecimal text input stream filter class.

Definition in file [OSRTBase64TextInputStream.h](#).

8.2 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.

8.2.1 Detailed Description

C++ run-time base class for structured type definitions.

Definition in file [OSRTBaseType.h](#).

8.3 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

- EXTERNRT void * [operator new](#) (size_t nbytes, OSCTXT *pctx)
Custom placement new function to allocate memory using context memory-management functions.
- EXTERNRT void [operator delete](#) (void *pmem, OSCTXT *pctx)
Custom placement delete function to free memory using context memory-management functions.

8.3.1 Detailed Description

C++ run-time context class definition.

Definition in file [OSRTContext.h](#).

8.4 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.

8.4.1 Detailed Description

C++ run-time message buffer interface class definition.

Definition in file [OSRTCtxtHolder.h](#).

8.5 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.

8.5.1 Detailed Description

C++ run-time message buffer interface class definition.

Definition in file [OSRTCtxtHolderIF.h](#).

8.6 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.

8.6.1 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](#).

8.7 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.

8.7.1 Detailed Description

C++ base class definitions for operations with input file streams.

Definition in file [OSRTFileInputStream.h](#).

8.8 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.

8.8.1 Detailed Description

C++ base class definitions for operations with output file streams.

Definition in file [OSRTFileOutputStream.h](#).

8.9 OSRTHexTextInputStream.h File Reference

C++ hexadecimal text input stream filter class.

```
#include "rtxsrc/OSRTInputStream.h"
```

Classes

- class [OSRTHexTextInputStream](#)
Hexadecimal text input stream filter class.

8.9.1 Detailed Description

C++ hexadecimal text input stream filter class.

Definition in file [OSRTHexTextInputStream.h](#).

8.10 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.

8.10.1 Detailed Description

C++ base class definitions for operations with input streams.

Definition in file [OSRTInputStream.h](#).

8.11 OSRTInputStreamIF.h File Reference

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

```
#include "rtxsrc/OSRTStreamIF.h"
```

8.11.1 Detailed Description

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

Definition in file [OSRTInputStreamIF.h](#).

8.12 OSRTMemBuf.h File Reference

```
#include "rtxsrc/rtxMemBuf.h"
```

Classes

- class [OSRTMemBuf](#)
Memory Buffer class.

8.12.1 Detailed Description

Definition in file [OSRTMemBuf.h](#).

8.13 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.

8.13.1 Detailed Description

C++ base class definitions for operations with input memory streams.

Definition in file [OSRTMemoryInputStream.h](#).

8.14 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.

8.14.1 Detailed Description

C++ base class definitions for operations with output memory streams.

Definition in file [OSRTMemoryOutputStream.h](#).

8.15 OSRTMsgBuf.h File Reference

C++ run-time message buffer class definition.

```
#include "rtxsrc/OSRCTxtHolder.h"
```

```
#include "rtxsrc/OSRTMsgBufIF.h"
```

Classes

- class [OSRTMessageBuffer](#)

Abstract message buffer base class.

8.15.1 Detailed Description

C++ run-time message buffer class definition.

Definition in file [OSRTMsgBuf.h](#).

8.16 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.

8.16.1 Detailed Description

C++ run-time message buffer interface class definition.

Definition in file [OSRTMsgBufIF.h](#).

8.17 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.

8.17.1 Detailed Description

C++ base class definitions for operations with output streams.

Definition in file [OSRTOutputStream.h](#).

8.18 OSRTOutputStreamIF.h File Reference

C++ interface class definitions for operations with output streams.

```
#include "rtxsrc/OSRTStreamIF.h"
```

8.18.1 Detailed Description

C++ interface class definitions for operations with output streams.

Definition in file [OSRTOutputStreamIF.h](#).

8.19 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.

8.19.1 Detailed Description

TCP/IP or UDP socket class definitions.

Definition in file [OSRTSocket.h](#).

8.20 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.

8.20.1 Detailed Description

C++ base class definitions for operations with input socket streams.

Definition in file [OSRTSocketInputStream.h](#).

8.21 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.

8.21.1 Detailed Description

C++ base class definitions for operations with output socket streams.

Definition in file [OSRTSocketOutputStream.h](#).

8.22 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.

8.22.1 Detailed Description

C++ base class definitions for operations with I/O streams.

Definition in file [OSRTStream.h](#).

8.23 OSRTStreamIF.h File Reference

C++ interface class definitions for operations with I/O streams.

```
#include "rtxsrc/OSRTCtxtHolderIF.h"
```

8.23.1 Detailed Description

C++ interface class definitions for operations with I/O streams.

Definition in file [OSRTStreamIF.h](#).

8.24 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.

8.24.1 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](#).

8.25 OSRTStringIF.h File Reference

C++ string class interface.

```
#include "rtxsrc/rtxCommon.h"
```

```
#include "rtxsrc/rtxPrint.h"
```

Classes

- class [OSRTStringIF](#)
C++ string class interface.

8.25.1 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](#).

8.26 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.

8.26.1 Detailed Description

C++ UTF-8 string class definition.

Definition in file [OSRTUTF8String.h](#).

8.27 rtxCppAnyAttr.h File Reference

C++ any element class definition.

```
#include "rtxsrc/rtxCommon.h"
```

```
#include "rtxsrc/OSRTBaseType.h"
```

Classes

- class [OSAnyAttrClass](#)
Any attribute.

8.27.1 Detailed Description

C++ any element class definition.

Definition in file [rtxCppAnyAttr.h](#).

8.28 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.

8.28.1 Detailed Description

C++ any element class definition.

Definition in file [rtxCppAnyElement.h](#).

8.29 rtxCppBitString.h File Reference

- Contains utility function for sizing a bit string.

```
#include "rtxsrc/rtxContext.h"
```

Functions

- EXTERNRT int [rtxCppCheckBitBounds](#) (OSOCKET *&pBits, OSUINT32 &numocts, size_t minRequiredBits, size_t preferredLimitBits)

Check whether the given bit string is large enough, and expand it if necessary.

8.29.1 Detailed Description

- Contains utility function for sizing a bit string.

Definition in file [rtxCppBitString.h](#).

8.29.2 Function Documentation

8.29.2.1 EXTERNRT int rtxCppCheckBitBounds (OSOCKET *& pBits, OSUINT32 & numocts, size_t minRequiredBits, size_t preferredLimitBits)

Check whether the given bit string is large enough, and expand it if necessary.

Parameters

pBits pBits is a pointer to the bit string, or NULL if one has not been created. If the string is expanded, pBits receives a pointer to the new bit string.

numocts is the current size of the bit string in octets. If the bit string is expanded, numocts receives the new size.

minRequiredBits The minimum number of bits needed in the bit string. On return, pBits will point to a bit string with at least this many bits.

preferredLimitBits The number of bits over which we prefer not to go. If nonzero, no more bytes will be allocated than necessary for this many bits, unless explicitly required by minRequiredBits.

Returns

If successful, 0. Otherwise, an error code.

8.30 rtxCppBufferedInputStream.h File Reference

```
#include "rtxsrc/OSRTInputStream.h"
```

Classes

- class [OSBufferedInputStream](#)
The buffered input stream class.

8.30.1 Detailed Description

Definition in file [rtxCppBufferedInputStream.h](#).

8.31 rtxCppType.h File Reference

C++ XML schema date/time definition.

```
#include "rtxsrc/rtxCommon.h"  
#include "rtxsrc/OSRTBaseType.h"
```

8.31.1 Detailed Description

C++ XML schema date/time definition.

Definition in file [rtxCppType.h](#).

8.32 rtxCppDList.h File Reference

```
#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.

8.32.1 Detailed Description

Definition in file [rtxCppDList.h](#).

8.33 rtxCppDynOctStr.h File Reference

C++ dynamic binary string class definition.

```
#include "rtxsrc/rtxCommon.h"
```

```
#include "rtxsrc/OSRTBaseType.h"
```

Classes

- class [OSDynOctStrClass](#)
Dynamic binary string.

8.33.1 Detailed Description

C++ dynamic binary string class definition.

Definition in file [rtxCppDynOctStr.h](#).

8.34 rtxCppException.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 [OSStreamException](#)
Exception class for streams.

8.34.1 Detailed Description

C++ run-time deprecated definition.

Definition in file [rtxCppException.h](#).

8.35 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"
```

8.35.1 Detailed Description

C++ common type and class definitions.

Definition in file [rtxCppTypes.h](#).

8.36 rtxCppXmlSTLString.h File Reference

C++ XML STL string class definition.

8.36.1 Detailed Description

C++ XML STL string class definition.

Definition in file [rtxCppXmlSTLString.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.

8.37.1 Detailed Description

C++ XML string class definition.

Definition in file [rtxCppXmlString.h](#).

8.38 rtxCppXmlStringList.h File Reference

C++ XML string list class definition.

```
#include "rtxsrc/rtxCppDList.h"  
#include "rtxsrc/rtxCppXmlString.h"
```

Classes

- class [OSXMLStringList](#)
XML list string.

8.38.1 Detailed Description

C++ XML string list class definition.

Definition in file [rtxCppXmlStringList.h](#).

Index

- ~OSBufferedInputStream
 - OSBufferedInputStream, 21
- ~OSRTBase64TextInputStream
 - OSRTBase64TextInputStream, 27
- ~OSRTCtxtHolderIF
 - OSRTCtxtHolderIF, 39
- ~OSRTCtxtPtr
 - OSRTCtxtPtr, 42
- ~OSRTHexTextInputStream
 - OSRTHexTextInputStream, 62
- ~OSRTInputStream
 - OSRTInputStream, 64
- ~OSRTLException
 - OSRTLException, 72
- ~OSRTMessageBuffer
 - OSRTMessageBuffer, 80
- ~OSRTMessageBufferIF
 - OSRTMessageBufferIF, 85
- ~OSRTOutputStream
 - OSRTOutputStream, 94
- ~OSRTSocket
 - OSRTSocket, 101
- ~OSRTStream
 - OSRTStream, 115
- accept
 - OSRTSocket, 101
- addrToString
 - OSRTSocket, 101
- append
 - OSRTDListClass, 47
 - OSRTObjListClass, 89
 - OSXMLStringClass, 137
- appendCopy
 - OSRTDListClass, 47
 - OSRTObjListClass, 89
 - OSXMLStringClass, 137
- appendValue
 - OSXMLStringClass, 132
- ASN.1 Stream Classes, 13
- bind
 - OSRTSocket, 102
- bindUrl
 - OSRTSocket, 103
- blockingRead
 - OSRTSocket, 103
- clone
 - OSAnyAttrClass, 16
 - OSDynOctStrClass, 24
 - OSRTUTF8String, 126
 - OSXMLStringClass, 132
 - OSXMLStringList, 137
- close
 - OSRTInputStream, 65
 - OSRTOutputStream, 94
 - OSRTSocket, 103
 - OSRTStream, 116
- compare
 - OSXMLStringClass, 132
- connect
 - OSRTSocket, 104
- connectUrl
 - OSRTSocket, 104
- copyValue
 - OSAnyAttrClass, 16
 - OSAnyElementClass, 19
 - OSDynOctStrClass, 25
 - OSRTUTF8String, 126
 - OSXMLStringClass, 132, 133
- currentPos
 - OSRTInputStream, 65
- decodeXML
 - OSXMLStringClass, 133
- encodeXML
 - OSXMLStringClass, 133
- flush
 - OSRTInputStream, 65
 - OSRTOutputStream, 95
 - OSRTStream, 116
- Generic Input Stream Classes, 9
- Generic Output Stream Classes, 11
- getBuffer
 - OSRTMemoryOutputStream, 77
- getByteIndex
 - OSRTMessageBuffer, 81

- OSRTMessageBufferIF, 85
- getContext
 - OSRTCtxtHolder, 36
 - OSRTCtxtHolderIF, 39
 - OSRTInputStream, 65
 - OSRTOutputStream, 95
 - OSRTStream, 116
- getCount
 - OSRTDListBaseClass, 44
- getCtxtPtr
 - OSRTCtxtHolder, 36
 - OSRTCtxtHolderIF, 39
 - OSRTInputStream, 66
 - OSRTMessageBuffer, 81
 - OSRTOutputStream, 95
 - OSRTStream, 116
- getData
 - OSRTDListNodeClass, 50
 - OSRTObjListNodeClass, 91
- getErrorInfo
 - OSRTContext, 31
 - OSRTCtxtHolder, 36, 37
 - OSRTCtxtHolderIF, 39
 - OSRTInputStream, 66
 - OSRTMessageBuffer, 81
 - OSRTOutputStream, 95, 96
 - OSRTStream, 117
- getHead
 - OSRTDListClass, 47
 - OSRTObjListClass, 89
- getItem
 - OSRTDListClass, 47
 - OSRTObjListClass, 89
- getList
 - OSRTDListBaseClass, 45
- getMsgCopy
 - OSRTMessageBufferIF, 85
- getMsgPtr
 - OSRTMessageBufferIF, 85
- getNext
 - OSRTDListNodeClass, 51
 - OSRTObjListNodeClass, 92
- getOwnership
 - OSRTSocket, 104
- getPosition
 - OSRTInputStream, 67
- getPrev
 - OSRTDListNodeClass, 51
 - OSRTObjListNodeClass, 92
- getPtr
 - OSRTContext, 31
- getSocket
 - OSRTSocket, 105
- getStatus
 - OSRTContext, 31
 - OSRTCtxtHolder, 37
 - OSRTCtxtHolderIF, 40
 - OSRTInputStream, 67
 - OSRTMessageBuffer, 82
 - OSRTOutputStream, 96
 - OSRTSocket, 105
 - OSRTStream, 117
- getTail
 - OSRTDListClass, 48
 - OSRTObjListClass, 90
- init
 - OSRTMessageBuffer, 82
 - OSRTMessageBufferIF, 85
- initBuffer
 - OSRTMessageBuffer, 82
 - OSRTMessageBufferIF, 86
- insert
 - OSRTDListClass, 48
 - OSRTObjListClass, 90
- isA
 - OSRTBase64TextInputStream, 27
 - OSRTFileInputStream, 56
 - OSRTFileOutputStream, 60
 - OSRTHexTextInputStream, 62
 - OSRTInputStream, 67
 - OSRTMemoryInputStream, 75
 - OSRTMemoryOutputStream, 77
 - OSRTMessageBufferIF, 86
 - OSRTOutputStream, 96
 - OSRTSocketInputStream, 110
 - OSRTSocketOutputStream, 113
- isCDATA
 - OSXMLStringClass, 134
- isInitialized
 - OSRTContext, 32
- isOpened
 - OSRTInputStream, 68
 - OSRTOutputStream, 97
 - OSRTStream, 118
- listen
 - OSRTSocket, 105
- mark
 - OSRTInputStream, 68
- markSupported
 - OSRTInputStream, 68
- memAlloc
 - OSRTContext, 32
- memAllocZ
 - OSRTContext, 32
- memFreeAll
 - OSRTContext, 32

- memFreePtr
 - OSRTContext, 32
- memRealloc
 - OSRTContext, 33
- Message Buffer Classes, 10
- mpContext
 - OSRTCtxtHolder, 37
- mStatus
 - OSRTContext, 34
- operator=
 - OSRTCtxtPtr, 42
 - OSRTObjListClass, 90
 - OSXMLStringList, 137
- OSAnyAttrClass, 14
 - clone, 16
 - copyValue, 16
 - OSAnyAttrClass, 15, 16
 - setValue, 16, 17
- OSAnyElementClass, 18
 - copyValue, 19
 - OSAnyElementClass, 19
 - print, 20
 - setValue, 20
- OSBufferedInputStream, 21
 - ~OSBufferedInputStream, 21
 - OSBufferedInputStream, 21
- OSDynOctStrClass, 23
 - clone, 24
 - copyValue, 25
 - OSDynOctStrClass, 24
 - setValue, 25
 - setValueFromBase64, 25
- OSRTBase64TextInputStream, 26
 - ~OSRTBase64TextInputStream, 27
 - isA, 27
 - OSRTBase64TextInputStream, 26
- OSRTBase64TextInputStream.h, 138
- OSRTBaseType, 28
- OSRTBaseType.h, 139
- OSRTContext, 29
 - getErrorInfo, 31
 - getPtr, 31
 - getStatus, 31
 - isInitialized, 32
 - memAlloc, 32
 - memAllocZ, 32
 - memFreeAll, 32
 - memFreePtr, 32
 - memRealloc, 33
 - mStatus, 34
 - setDiag, 33
 - setRunTimeKey, 33
 - setStatus, 34
- OSRTContext.h, 140
- OSRTCtxtHolder, 35
 - getContext, 36
 - getCtxtPtr, 36
 - getErrorInfo, 36, 37
 - getStatus, 37
 - mpContext, 37
 - OSRTCtxtHolder, 36
- OSRTCtxtHolder.h, 141
- OSRTCtxtHolderIF, 38
 - ~OSRTCtxtHolderIF, 39
 - getContext, 39
 - getCtxtPtr, 39
 - getErrorInfo, 39
 - getStatus, 40
- OSRTCtxtHolderIF.h, 142
- OSRTCtxtPtr, 41
 - ~OSRTCtxtPtr, 42
 - operator=, 42
 - OSRTCtxtPtr, 42
- OSRTDListBaseClass, 44
 - getCount, 44
 - getList, 45
 - remove, 45
- OSRTDListClass, 46
 - append, 47
 - appendCopy, 47
 - getHead, 47
 - getItem, 47
 - getTail, 48
 - insert, 48
- OSRTDListNodeBaseClass, 49
- OSRTDListNodeClass, 50
 - getData, 50
 - getNext, 51
 - getPrev, 51
- OSRTFastString, 52
 - OSRTFastString, 53
 - print, 53
 - setValue, 53, 54
- OSRTFastString.h, 143
- OSRTFileInputStream, 55
 - isA, 56
 - OSRTFileInputStream, 55, 56
- OSRTFileInputStream.h, 144
- OSRTFileOutputStream, 58
 - isA, 60
 - OSRTFileOutputStream, 58, 59
- OSRTFileOutputStream.h, 145
- OSRTHexTextInputStream, 61
 - ~OSRTHexTextInputStream, 62
 - isA, 62
 - OSRTHexTextInputStream, 61
- OSRTHexTextInputStream.h, 146

- OSRTInputStream, 63
 - ~OSRTInputStream, 64
 - close, 65
 - currentPos, 65
 - flush, 65
 - getContext, 65
 - getCtxtPtr, 66
 - getErrorInfo, 66
 - getPosition, 67
 - getStatus, 67
 - isA, 67
 - isOpened, 68
 - mark, 68
 - markSupported, 68
 - OSRTInputStream, 64
 - read, 68
 - readBlocking, 69
 - reset, 69
 - setPosition, 69
 - skip, 70
- OSRTInputStream.h, 147
- OSRTInputStreamIF.h, 148
- OSRTLEException, 71
 - ~OSRTLEException, 72
 - OSRTLEException, 72
- OSRTMemBuf, 73
- OSRTMemBuf.h, 149
- OSRTMemoryInputStream, 74
 - isA, 75
 - OSRTMemoryInputStream, 74
- OSRTMemoryInputStream.h, 150
- OSRTMemoryOutputStream, 76
 - getBuffer, 77
 - isA, 77
 - OSRTMemoryOutputStream, 76, 77
 - reset, 78
- OSRTMemoryOutputStream.h, 151
- OSRTMessageBuffer, 79
 - ~OSRTMessageBuffer, 80
 - getByteIndex, 81
 - getCtxtPtr, 81
 - getErrorInfo, 81
 - getStatus, 82
 - init, 82
 - initBuffer, 82
 - OSRTMessageBuffer, 80
 - setDiag, 82
- OSRTMessageBufferIF, 84
 - ~OSRTMessageBufferIF, 85
 - getByteIndex, 85
 - getMsgCopy, 85
 - getMsgPtr, 85
 - init, 85
 - initBuffer, 86
- isA, 86
 - setDiag, 86
- OSRTMsgBuf.h, 152
- OSRTMsgBufIF.h, 153
- OSRTObjListClass, 88
 - append, 89
 - appendCopy, 89
 - getHead, 89
 - getItem, 89
 - getTail, 90
 - insert, 90
 - operator=, 90
- OSRTObjListNodeClass, 91
 - getData, 91
 - getNext, 92
 - getPrev, 92
- OSRTOutputStream, 93
 - ~OSRTOutputStream, 94
 - close, 94
 - flush, 95
 - getContext, 95
 - getCtxtPtr, 95
 - getErrorInfo, 95, 96
 - getStatus, 96
 - isA, 96
 - isOpened, 97
 - OSRTOutputStream, 94
 - write, 97
- OSRTOutputStream.h, 154
- OSRTOutputStreamIF.h, 155
- OSRTSocket, 99
 - ~OSRTSocket, 101
 - accept, 101
 - addrToString, 101
 - bind, 102
 - bindUrl, 103
 - blockingRead, 103
 - close, 103
 - connect, 104
 - connectUrl, 104
 - getOwnership, 104
 - getSocket, 105
 - getStatus, 105
 - listen, 105
 - OSRTSocket, 100, 101
 - recv, 105
 - send, 106
 - setOwnership, 106
 - stringToAddr, 106
- OSRTSocket.h, 156
- OSRTSocketInputStream, 108
 - isA, 110
 - OSRTSocketInputStream, 108, 109
- OSRTSocketInputStream.h, 157

- OSRTSocketOutputStream, 111
 - isA, 113
 - OSRTSocketOutputStream, 111, 112
- OSRTSocketOutputStream.h, 158
- OSRTStream, 114
 - ~OSRTStream, 115
 - close, 116
 - flush, 116
 - getContext, 116
 - getCtxtPtr, 116
 - getErrorInfo, 117
 - getStatus, 117
 - isOpened, 118
 - OSRTStream, 115
- OSRTStream.h, 159
- OSRTStreamIF.h, 160
- OSRTString, 119
 - OSRTString, 120
 - print, 120
 - setValue, 120, 121
- OSRTString.h, 161
- OSRTStringIF, 122
 - OSRTStringIF, 123
 - print, 123
 - setValue, 123
- OSRTStringIF.h, 162
- OSRTUTF8String, 125
 - clone, 126
 - copyValue, 126
 - OSRTUTF8String, 126
 - print, 126
 - setValue, 127
- OSRTUTF8String.h, 163
- OStreamException, 128
- OSXMLStringClass, 129
 - appendValue, 132
 - clone, 132
 - compare, 132
 - copyValue, 132, 133
 - decodeXML, 133
 - encodeXML, 133
 - isCDATA, 134
 - OSXMLStringClass, 131, 132
 - print, 134
 - setCDATA, 134
 - setValue, 134, 135
- OSXMLStringList, 136
 - append, 137
 - appendCopy, 137
 - clone, 137
 - operator=, 137
 - OSXMLStringList, 137
- print
 - OSAnyElementClass, 20
 - OSRTFastString, 53
 - OSRTString, 120
 - OSRTStringIF, 123
 - OSRTUTF8String, 126
 - OSXMLStringClass, 134
- read
 - OSRTInputStream, 68
- readBlocking
 - OSRTInputStream, 69
- recv
 - OSRTSocket, 105
- remove
 - OSRTDListBaseClass, 45
- reset
 - OSRTInputStream, 69
 - OSRTMemoryOutputStream, 78
- rtxCppAnyAttr.h, 164
- rtxCppAnyElement.h, 165
- rtxCppBitString.h, 166
 - rtxCppCheckBitBounds, 166
- rtxCppBufferedInputStream.h, 167
- rtxCppCheckBitBounds
 - rtxCppBitString.h, 166
- rtxCppDateTime.h, 168
- rtxCppDList.h, 169
- rtxCppDynOctStr.h, 170
- rtxCppException.h, 171
- rtxCppTypes.h, 172
- rtxCppXmlSTLString.h, 173
- rtxCppXmlString.h, 174
- rtxCppXmlStringList.h, 175
- send
 - OSRTSocket, 106
- setCDATA
 - OSXMLStringClass, 134
- setDiag
 - OSRTContext, 33
 - OSRTMessageBuffer, 82
 - OSRTMessageBufferIF, 86
- setOwnership
 - OSRTSocket, 106
- setPosition
 - OSRTInputStream, 69
- setRunTimeKey
 - OSRTContext, 33
- setStatus
 - OSRTContext, 34
- setValue
 - OSAnyAttrClass, 16, 17
 - OSAnyElementClass, 20
 - OSDynOctStrClass, 25

- OSRTFastString, [53](#), [54](#)
- OSRTString, [120](#), [121](#)
- OSRTStringIF, [123](#)
- OSRTUTF8String, [127](#)
- OSXMLStringClass, [134](#), [135](#)
- setValueFromBase64
 - OSDynOctStrClass, [25](#)
- skip
 - OSRTInputStream, [70](#)
- stringToAddr
 - OSRTSocket, [106](#)
- TCP/IP or UDP Socket Classes, [12](#)
- write
 - OSRTOutputStream, [97](#)