

XBinder

XML Schema Compiler
Version 2.7
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 ©1997–2021 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	C++ Common Runtime Library Classes	1
2	Module Index	3
2.1	Modules	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Module Documentation	11
6.1	Generic Input Stream Classes	11
6.1.1	Detailed Description	11
6.2	Message Buffer Classes	12
6.2.1	Detailed Description	12
6.3	Generic Output Stream Classes	13
6.3.1	Detailed Description	13
6.4	TCP/IP or UDP Socket Classes	14
6.4.1	Detailed Description	14
6.5	ASN.1 Stream Classes	15
6.5.1	Detailed Description	15

7	Class Documentation	17
7.1	OSAnyAttrClass Class Reference	17
7.1.1	Detailed Description	18
7.1.2	Constructor & Destructor Documentation	18
7.1.2.1	OSAnyAttrClass() [1/5]	18
7.1.2.2	OSAnyAttrClass() [2/5]	18
7.1.2.3	OSAnyAttrClass() [3/5]	19
7.1.2.4	OSAnyAttrClass() [4/5]	19
7.1.2.5	OSAnyAttrClass() [5/5]	19
7.1.3	Member Function Documentation	20
7.1.3.1	clone()	20
7.1.3.2	copyValue()	20
7.1.3.3	setValue() [1/2]	21
7.1.3.4	setValue() [2/2]	21
7.2	OSAnyElementClass Class Reference	21
7.2.1	Detailed Description	22
7.2.2	Constructor & Destructor Documentation	22
7.2.2.1	OSAnyElementClass() [1/4]	22
7.2.2.2	OSAnyElementClass() [2/4]	23
7.2.2.3	OSAnyElementClass() [3/4]	23
7.2.2.4	OSAnyElementClass() [4/4]	23
7.2.3	Member Function Documentation	25
7.2.3.1	copyValue()	25
7.2.3.2	print()	25
7.2.3.3	setValue()	25
7.3	OSBufferedInputStream Class Reference	26
7.3.1	Detailed Description	26
7.3.2	Constructor & Destructor Documentation	27

7.3.2.1	OSBufferedInputStream()	27
7.3.2.2	~OSBufferedInputStream()	27
7.4	OSDynOctStrClass Class Reference	27
7.4.1	Detailed Description	28
7.4.2	Constructor & Destructor Documentation	28
7.4.2.1	OSDynOctStrClass() [1/4]	29
7.4.2.2	OSDynOctStrClass() [2/4]	30
7.4.2.3	OSDynOctStrClass() [3/4]	30
7.4.2.4	OSDynOctStrClass() [4/4]	30
7.4.3	Member Function Documentation	31
7.4.3.1	clone()	31
7.4.3.2	copyValue()	31
7.4.3.3	setValue() [1/2]	31
7.4.3.4	setValue() [2/2]	32
7.4.3.5	setValueFromBase64()	32
7.5	OSRTBase64TextInputStream Class Reference	33
7.5.1	Detailed Description	33
7.5.2	Constructor & Destructor Documentation	33
7.5.2.1	OSRTBase64TextInputStream()	34
7.5.2.2	~OSRTBase64TextInputStream()	35
7.5.3	Member Function Documentation	35
7.5.3.1	isA()	35
7.6	OSRTBaseType Class Reference	36
7.6.1	Detailed Description	36
7.7	OSRTContext Class Reference	36
7.7.1	Detailed Description	38
7.7.2	Member Function Documentation	38
7.7.2.1	getErrorInfo() [1/3]	38

7.7.2.2	getErrorInfo() [2/3]	38
7.7.2.3	getErrorInfo() [3/3]	39
7.7.2.4	getPtr()	39
7.7.2.5	getStatus()	39
7.7.2.6	isInitialized()	40
7.7.2.7	memAlloc()	40
7.7.2.8	memAllocZ()	40
7.7.2.9	memFreeAll()	41
7.7.2.10	memFreePtr()	41
7.7.2.11	memRealloc()	41
7.7.2.12	setDiag()	42
7.7.2.13	setRunTimeKey()	42
7.7.2.14	setStatus()	43
7.7.3	Member Data Documentation	43
7.7.3.1	mStatus	43
7.8	OSRTCtxtHolder Class Reference	43
7.8.1	Detailed Description	44
7.8.2	Constructor & Destructor Documentation	44
7.8.2.1	OSRTCtxtHolder()	44
7.8.3	Member Function Documentation	45
7.8.3.1	getContext()	45
7.8.3.2	getCtxtPtr()	45
7.8.3.3	getErrorInfo() [1/2]	46
7.8.3.4	getErrorInfo() [2/2]	46
7.8.3.5	getStatus()	47
7.8.4	Member Data Documentation	47
7.8.4.1	mpContext	47
7.9	OSRTCtxtHolderIF Class Reference	47

7.9.1	Detailed Description	48
7.9.2	Constructor & Destructor Documentation	48
7.9.2.1	~OSRTCtxtHolderIF()	48
7.9.3	Member Function Documentation	49
7.9.3.1	getContext()	49
7.9.3.2	getCtxtPtr()	49
7.9.3.3	getErrorInfo() [1/2]	49
7.9.3.4	getErrorInfo() [2/2]	50
7.9.3.5	getStatus()	51
7.10	OSRTCtxtPtr Class Reference	51
7.10.1	Detailed Description	52
7.10.2	Constructor & Destructor Documentation	52
7.10.2.1	OSRTCtxtPtr() [1/2]	52
7.10.2.2	OSRTCtxtPtr() [2/2]	53
7.10.2.3	~OSRTCtxtPtr()	53
7.10.3	Member Function Documentation	53
7.10.3.1	operator=()	53
7.11	OSRTDListBaseClass Class Reference	54
7.11.1	Detailed Description	55
7.11.2	Member Function Documentation	55
7.11.2.1	getCount()	55
7.11.2.2	getList() [1/2]	55
7.11.2.3	getList() [2/2]	56
7.11.2.4	remove()	56
7.12	OSRTDListClass Class Reference	56
7.12.1	Detailed Description	57
7.12.2	Member Function Documentation	57
7.12.2.1	append()	57

7.12.2.2	appendCopy()	58
7.12.2.3	getHead() [1/2]	58
7.12.2.4	getHead() [2/2]	58
7.12.2.5	getItem()	59
7.12.2.6	getTail() [1/2]	59
7.12.2.7	getTail() [2/2]	59
7.12.2.8	insert()	60
7.13	OSRTDListNodeBaseClass Class Reference	61
7.13.1	Detailed Description	61
7.14	OSRTDListNodeClass Class Reference	62
7.14.1	Detailed Description	62
7.14.2	Member Function Documentation	62
7.14.2.1	getData() [1/2]	63
7.14.2.2	getData() [2/2]	63
7.14.2.3	getNext() [1/2]	63
7.14.2.4	getNext() [2/2]	64
7.14.2.5	getPrev() [1/2]	64
7.14.2.6	getPrev() [2/2]	64
7.15	OSRTElemNameGuard Class Reference	65
7.15.1	Detailed Description	65
7.16	OSRTFastString Class Reference	65
7.16.1	Detailed Description	66
7.16.2	Constructor & Destructor Documentation	66
7.16.2.1	OSRTFastString() [1/3]	66
7.16.2.2	OSRTFastString() [2/3]	67
7.16.2.3	OSRTFastString() [3/3]	67
7.16.3	Member Function Documentation	67
7.16.3.1	print()	67

7.16.3.2	setValue() [1/2]	68
7.16.3.3	setValue() [2/2]	68
7.17	OSRTFileInputStream Class Reference	68
7.17.1	Detailed Description	69
7.17.2	Constructor & Destructor Documentation	69
7.17.2.1	OSRTFileInputStream() [1/4]	69
7.17.2.2	OSRTFileInputStream() [2/4]	70
7.17.2.3	OSRTFileInputStream() [3/4]	70
7.17.2.4	OSRTFileInputStream() [4/4]	71
7.17.3	Member Function Documentation	71
7.17.3.1	isA()	71
7.18	OSRTFileOutputStream Class Reference	72
7.18.1	Detailed Description	72
7.18.2	Constructor & Destructor Documentation	72
7.18.2.1	OSRTFileOutputStream() [1/4]	72
7.18.2.2	OSRTFileOutputStream() [2/4]	73
7.18.2.3	OSRTFileOutputStream() [3/4]	73
7.18.2.4	OSRTFileOutputStream() [4/4]	74
7.18.3	Member Function Documentation	74
7.18.3.1	isA()	74
7.19	OSRTHexTextInputStream Class Reference	75
7.19.1	Detailed Description	76
7.19.2	Constructor & Destructor Documentation	76
7.19.2.1	OSRTHexTextInputStream()	76
7.19.2.2	~OSRTHexTextInputStream()	76
7.19.3	Member Function Documentation	76
7.19.3.1	isA()	77
7.20	OSRTInputStream Class Reference	77

7.20.1	Detailed Description	79
7.20.2	Constructor & Destructor Documentation	79
7.20.2.1	OSRTInputStream()	79
7.20.2.2	~OSRTInputStream()	79
7.20.3	Member Function Documentation	79
7.20.3.1	close()	80
7.20.3.2	currentPos()	80
7.20.3.3	flush()	80
7.20.3.4	getContext()	81
7.20.3.5	getCtxtPtr()	81
7.20.3.6	getErrorInfo() [1/2]	82
7.20.3.7	getErrorInfo() [2/2]	82
7.20.3.8	getPosition()	83
7.20.3.9	getStatus()	83
7.20.3.10	isA()	83
7.20.3.11	isOpened()	84
7.20.3.12	mark()	84
7.20.3.13	markSupported()	85
7.20.3.14	read()	85
7.20.3.15	readBlocking()	86
7.20.3.16	reset()	86
7.20.3.17	setPosition()	87
7.20.3.18	skip()	87
7.21	OSRTMemBuf Class Reference	88
7.21.1	Detailed Description	88
7.22	OSRTMemoryInputStream Class Reference	88
7.22.1	Detailed Description	89
7.22.2	Constructor & Destructor Documentation	89

7.22.2.1	OSRTMemoryInputStream() [1/2]	89
7.22.2.2	OSRTMemoryInputStream() [2/2]	89
7.22.3	Member Function Documentation	90
7.22.3.1	isA()	90
7.23	OSRTMemoryOutputStream Class Reference	90
7.23.1	Detailed Description	91
7.23.2	Constructor & Destructor Documentation	91
7.23.2.1	OSRTMemoryOutputStream() [1/3]	91
7.23.2.2	OSRTMemoryOutputStream() [2/3]	91
7.23.2.3	OSRTMemoryOutputStream() [3/3]	92
7.23.3	Member Function Documentation	92
7.23.3.1	getBuffer()	92
7.23.3.2	isA()	93
7.23.3.3	reset()	93
7.24	OSRTMessageBuffer Class Reference	94
7.24.1	Detailed Description	95
7.24.2	Constructor & Destructor Documentation	95
7.24.2.1	OSRTMessageBuffer()	95
7.24.2.2	~OSRTMessageBuffer()	95
7.24.3	Member Function Documentation	96
7.24.3.1	getByteIndex()	96
7.24.3.2	getCtxtPtr()	96
7.24.3.3	getErrorInfo() [1/2]	97
7.24.3.4	getErrorInfo() [2/2]	97
7.24.3.5	getStatus()	98
7.24.3.6	init()	98
7.24.3.7	initBuffer()	98
7.24.3.8	setDiag()	99

7.25 OSRTMessageBufferIF Class Reference	99
7.25.1 Detailed Description	100
7.25.2 Constructor & Destructor Documentation	100
7.25.2.1 ~OSRTMessageBufferIF()	101
7.25.3 Member Function Documentation	101
7.25.3.1 getByteIndex()	101
7.25.3.2 getMsgCopy()	101
7.25.3.3 getMsgPtr()	102
7.25.3.4 init()	102
7.25.3.5 initBuffer()	102
7.25.3.6 isA()	103
7.25.3.7 setDiag()	103
7.26 OSRTObjListClass Class Reference	104
7.26.1 Detailed Description	105
7.26.2 Member Function Documentation	105
7.26.2.1 append()	105
7.26.2.2 appendCopy()	105
7.26.2.3 getHead() [1/2]	105
7.26.2.4 getHead() [2/2]	106
7.26.2.5 getItem()	106
7.26.2.6 getTail() [1/2]	107
7.26.2.7 getTail() [2/2]	107
7.26.2.8 insert()	107
7.26.2.9 operator=()	108
7.27 OSRTObjListNodeClass Class Reference	108
7.27.1 Detailed Description	109
7.27.2 Member Function Documentation	109
7.27.2.1 getData() [1/2]	109

7.27.2.2	getData() [2/2]	109
7.27.2.3	getNext() [1/2]	110
7.27.2.4	getNext() [2/2]	110
7.27.2.5	getPrev() [1/2]	110
7.27.2.6	getPrev() [2/2]	111
7.28	OSRTOutputStream Class Reference	111
7.28.1	Detailed Description	112
7.28.2	Constructor & Destructor Documentation	112
7.28.2.1	OSRTOutputStream()	112
7.28.2.2	~OSRTOutputStream()	113
7.28.3	Member Function Documentation	113
7.28.3.1	close()	113
7.28.3.2	flush()	113
7.28.3.3	getContext()	114
7.28.3.4	getCtxtPtr()	114
7.28.3.5	getErrorInfo() [1/2]	114
7.28.3.6	getErrorInfo() [2/2]	115
7.28.3.7	getStatus()	115
7.28.3.8	isA()	115
7.28.3.9	isOpened()	116
7.28.3.10	write() [1/2]	116
7.28.3.11	write() [2/2]	117
7.29	OSRTSocket Class Reference	117
7.29.1	Detailed Description	119
7.29.2	Constructor & Destructor Documentation	119
7.29.2.1	OSRTSocket() [1/3]	119
7.29.2.2	OSRTSocket() [2/3]	119
7.29.2.3	OSRTSocket() [3/3]	120

7.29.2.4	~OSRSocket()	120
7.29.3	Member Function Documentation	120
7.29.3.1	accept()	120
7.29.3.2	addrToString()	121
7.29.3.3	bind() [1/3]	121
7.29.3.4	bind() [2/3]	122
7.29.3.5	bind() [3/3]	123
7.29.3.6	bindUrl()	123
7.29.3.7	blockingRead()	124
7.29.3.8	close()	124
7.29.3.9	connect()	125
7.29.3.10	connectTimed()	125
7.29.3.11	connectUrl()	126
7.29.3.12	getOwnership()	126
7.29.3.13	getSocket()	127
7.29.3.14	getStatus()	127
7.29.3.15	listen()	127
7.29.3.16	recv()	128
7.29.3.17	send()	128
7.29.3.18	setOwnership()	129
7.29.3.19	setRetryCount()	129
7.29.3.20	stringToAddr()	130
7.30	OSRSocketInputStream Class Reference	130
7.30.1	Detailed Description	131
7.30.2	Constructor & Destructor Documentation	131
7.30.2.1	OSRSocketInputStream() [1/4]	131
7.30.2.2	OSRSocketInputStream() [2/4]	132
7.30.2.3	OSRSocketInputStream() [3/4]	132

7.30.2.4	OSRTSocketInputStream() [4/4]	133
7.30.3	Member Function Documentation	133
7.30.3.1	isA()	133
7.31	OSRTSocketOutputStream Class Reference	134
7.31.1	Detailed Description	134
7.31.2	Constructor & Destructor Documentation	135
7.31.2.1	OSRTSocketOutputStream() [1/4]	135
7.31.2.2	OSRTSocketOutputStream() [2/4]	135
7.31.2.3	OSRTSocketOutputStream() [3/4]	135
7.31.2.4	OSRTSocketOutputStream() [4/4]	137
7.31.3	Member Function Documentation	137
7.31.3.1	isA()	137
7.32	OSRTStream Class Reference	138
7.32.1	Detailed Description	139
7.32.2	Constructor & Destructor Documentation	140
7.32.2.1	OSRTStream()	140
7.32.2.2	~OSRTStream()	140
7.32.3	Member Function Documentation	140
7.32.3.1	close()	140
7.32.3.2	flush()	141
7.32.3.3	getContext()	141
7.32.3.4	getCtxtPtr()	142
7.32.3.5	getErrorInfo() [1/2]	142
7.32.3.6	getErrorInfo() [2/2]	142
7.32.3.7	getStatus()	143
7.32.3.8	isOpened()	143
7.33	OSRTString Class Reference	144
7.33.1	Detailed Description	145

7.33.2	Constructor & Destructor Documentation	145
7.33.2.1	OSRTString() [1/3]	145
7.33.2.2	OSRTString() [2/3]	145
7.33.2.3	OSRTString() [3/3]	147
7.33.3	Member Function Documentation	147
7.33.3.1	print()	147
7.33.3.2	setValue() [1/2]	147
7.33.3.3	setValue() [2/2]	148
7.33.3.4	setValuePtr()	148
7.33.3.5	toInt()	148
7.33.3.6	toSize()	149
7.33.3.7	toUInt()	149
7.33.3.8	toUInt64()	150
7.34	OSRTStringIF Class Reference	150
7.34.1	Detailed Description	151
7.34.2	Constructor & Destructor Documentation	151
7.34.2.1	OSRTStringIF() [1/2]	151
7.34.2.2	OSRTStringIF() [2/2]	151
7.34.3	Member Function Documentation	152
7.34.3.1	print()	152
7.34.3.2	setValue() [1/2]	152
7.34.3.3	setValue() [2/2]	153
7.35	OSRTUTF8String Class Reference	153
7.35.1	Detailed Description	154
7.35.2	Constructor & Destructor Documentation	154
7.35.2.1	OSRTUTF8String() [1/3]	154
7.35.2.2	OSRTUTF8String() [2/3]	154
7.35.2.3	OSRTUTF8String() [3/3]	155

7.35.3	Member Function Documentation	155
7.35.3.1	clone()	155
7.35.3.2	copyValue()	155
7.35.3.3	print()	156
7.35.3.4	setValue()	156
7.36	OSRXMLString Class Reference	156
7.36.1	Detailed Description	158
7.36.2	Constructor & Destructor Documentation	158
7.36.2.1	OSRXMLString() [1/5]	158
7.36.2.2	OSRXMLString() [2/5]	159
7.36.2.3	OSRXMLString() [3/5]	159
7.36.2.4	OSRXMLString() [4/5]	160
7.36.2.5	OSRXMLString() [5/5]	160
7.36.3	Member Function Documentation	160
7.36.3.1	appendValue()	160
7.36.3.2	clone()	161
7.36.3.3	compare()	161
7.36.3.4	copyValue() [1/2]	161
7.36.3.5	copyValue() [2/2]	162
7.36.3.6	decodeXML()	162
7.36.3.7	encodeXML()	162
7.36.3.8	isCDATA()	163
7.36.3.9	print()	163
7.36.3.10	setCDATA()	164
7.36.3.11	setValue() [1/3]	164
7.36.3.12	setValue() [2/3]	164
7.36.3.13	setValue() [3/3]	165
7.37	OSRXMLStringList Class Reference	165
7.37.1	Detailed Description	166
7.37.2	Constructor & Destructor Documentation	166
7.37.2.1	OSRXMLStringList()	166
7.37.3	Member Function Documentation	167
7.37.3.1	append()	167
7.37.3.2	appendCopy()	167
7.37.3.3	clone()	167
7.37.3.4	operator=()	168

8	File Documentation	169
8.1	OSRTBase64TextInputStream.h File Reference	169
8.1.1	Detailed Description	169
8.2	OSRTBaseType.h File Reference	169
8.2.1	Detailed Description	170
8.3	OSRTContext.h File Reference	170
8.3.1	Detailed Description	170
8.4	OSRTCtxtHolder.h File Reference	170
8.4.1	Detailed Description	171
8.5	OSRTCtxtHolderIF.h File Reference	171
8.5.1	Detailed Description	171
8.6	OSRTFastString.h File Reference	171
8.6.1	Detailed Description	172
8.7	OSRTFileInputStream.h File Reference	172
8.7.1	Detailed Description	172
8.8	OSRTFileOutputStream.h File Reference	172
8.8.1	Detailed Description	172
8.9	OSRTHexTextInputStream.h File Reference	173
8.9.1	Detailed Description	173
8.10	OSRTInputStream.h File Reference	173
8.10.1	Detailed Description	173
8.11	OSRTInputStreamIF.h File Reference	173
8.11.1	Detailed Description	174
8.12	OSRTMemBuf.h File Reference	174
8.13	OSRTMemoryInputStream.h File Reference	174
8.13.1	Detailed Description	174
8.14	OSRTMemoryOutputStream.h File Reference	174
8.14.1	Detailed Description	175

8.15 OSRTMsgBuf.h File Reference	175
8.15.1 Detailed Description	175
8.16 OSRTMsgBufIF.h File Reference	175
8.16.1 Detailed Description	176
8.17 OSRTOutputStream.h File Reference	176
8.17.1 Detailed Description	176
8.18 OSRTOutputStreamIF.h File Reference	176
8.18.1 Detailed Description	176
8.19 OSRTSocket.h File Reference	176
8.19.1 Detailed Description	177
8.20 OSRTSocketInputStream.h File Reference	177
8.20.1 Detailed Description	177
8.21 OSRTSocketOutputStream.h File Reference	177
8.21.1 Detailed Description	178
8.22 OSRTStream.h File Reference	178
8.22.1 Detailed Description	178
8.23 OSRTStreamIF.h File Reference	178
8.23.1 Detailed Description	178
8.24 OSRTString.h File Reference	178
8.24.1 Detailed Description	179
8.25 OSRTStringConst.h File Reference	179
8.25.1 Detailed Description	179
8.26 OSRTStringIF.h File Reference	179
8.26.1 Detailed Description	180
8.27 OSRTUTF8String.h File Reference	180
8.27.1 Detailed Description	180
8.28 OSRTVoidPtrList.h File Reference	180
8.28.1 Detailed Description	180

8.29	rtxCppAnyAttr.h File Reference	181
8.29.1	Detailed Description	181
8.30	rtxCppAnyElement.h File Reference	181
8.30.1	Detailed Description	181
8.31	rtxCppBitString.h File Reference	181
8.31.1	Detailed Description	182
8.31.2	Function Documentation	182
8.31.2.1	rtxCppCheckBitBounds()	182
8.31.2.2	rtxCppCheckBitBounds64()	183
8.32	rtxCppBufferedInputStream.h File Reference	183
8.33	rtxCppDateTime.h File Reference	183
8.33.1	Detailed Description	184
8.34	rtxCppDList.h File Reference	184
8.35	rtxCppDynOctStr.h File Reference	184
8.35.1	Detailed Description	184
8.36	rtxCppTypes.h File Reference	185
8.36.1	Detailed Description	185
8.37	rtxCppXmiSTLString.h File Reference	185
8.37.1	Detailed Description	185
8.38	rtxCppXmiSTLStringList.h File Reference	185
8.38.1	Detailed Description	185
8.39	rtxCppXmlString.h File Reference	185
8.39.1	Detailed Description	186
8.40	rtxCppXmlStringList.h File Reference	186
8.40.1	Detailed Description	186

Chapter 1

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 11
- Message Buffer Classes 12
- Generic Output Stream Classes 13
- TCP/IP or UDP Socket Classes 14
- ASN.1 Stream Classes 15

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

OSRTBaseType	36
OSAnyAttrClass	17
OSAnyElementClass	21
OSDynOctStrClass	27
OSRTDListBaseClass	54
OSRTDListClass	56
OSRTObjListClass	104
OSRTUTF8String	153
OSRTXMLString	156
OSRTXMLStringList	165
OSRTContext	36
OSRTCtxtHolderIF	47
OSRTCtxtHolder	43
OSRTCtxtPtr	51
OSRTDListNodeBaseClass	61
OSRTDListNodeClass	62
OSRTObjListNodeClass	108
OSRTElemNameGuard	65
OSRTMemBuf	88
OSRTMessageBufferIF	99
OSRTMessageBuffer	94
OSRTSocket	117
OSRTStream	138
OSRTInputStream	77
OSBufferedInputStream	26
OSRTBase64TextInputStream	33
OSRTFileInputStream	68
OSRTHexTextInputStream	75
OSRTMemoryInputStream	88
OSRTSocketInputStream	130

OSRTOutputStream	111
OSRTFileOutputStream	72
OSRTMemoryOutputStream	90
OSRTSocketOutputStream	134
OSRTStringIF	150
OSRTFastString	65
OSRTString	144

Chapter 4

Class Index

4.1 Class List

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

OSAnyAttrClass	
Any attribute	17
OSAnyElementClass	
Any element	21
OSBufferedInputStream	
The buffered input stream class	26
OSDynOctStrClass	
Dynamic binary string	27
OSRTBase64TextInputStream	
Hexadecimal text input stream filter class	33
OSRTBaseType	
C++ structured type base class	36
OSRTContext	
Reference counted context class	36
OSRTCtxtHolder	
Abstract message buffer or stream interface class	43
OSRTCtxtHolderIF	
Abstract message buffer or stream interface class	47
OSRTCtxtPtr	
Context reference counted pointer class	51
OSRTDListBaseClass	
This class is a base class for C++ representations of a doubly-linked list classes	54
OSRTDListClass	
This class represents a doubly-linked list structure	56
OSRTDListNodeBaseClass	
This class is a base class for C++ representations of a node for the doubly-linked list structure	61
OSRTDListNodeClass	
This class represents a doubly-linked list node structure	62
OSRTElemNameGuard	
Element name guard class	65
OSRTFastString	
C++ fast string class definition	65

OSRTFileInputStream	
Generic file input stream	68
OSRTFileOutputStream	
Generic file output stream	72
OSRTHexTextInputStream	
Hexadecimal text input stream filter class	75
OSRTInputStream	
This is the base class for input streams	77
OSRTMemBuf	
Memory Buffer class	88
OSRTMemoryInputStream	
Generic memory input stream	88
OSRTMemoryOutputStream	
Generic memory output stream	90
OSRTMessageBuffer	
Abstract message buffer base class	94
OSRTMessageBufferIF	
Abstract message buffer or stream interface class	99
OSRTObjListClass	
This class represents a doubly-linked list structure for objects	104
OSRTObjListNodeClass	
This class represents a doubly-linked list node structure for OSRTBaseType instances	108
OSRTOutputStream	
The base class definition for operations with output streams	111
OSRTSocket	
Wrapper class for TCP/IP or UDP sockets	117
OSRTSocketInputStream	
Generic socket input stream	130
OSRTSocketOutputStream	
Generic socket output stream	134
OSRTStream	
The default base class for using I/O streams	138
OSRTString	
C++ string class definition	144
OSRTStringIF	
C++ string class interface	150
OSRTUTF8String	
UTF-8 string	153
OSRTXMLString	
XML string	156
OSRTXMLStringList	
XML list string	165

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	169
OSRTBaseType.h	C++ run-time base class for structured type definitions	169
OSRTContext.h	C++ run-time context class definition	170
OSRTCtxtHolder.h	C++ run-time message buffer interface class definition	170
OSRTCtxtHolderIF.h	C++ run-time message buffer interface class definition	171
OSRTDiag.h	??
OSRTFastString.h	C++ fast string class definition	171
OSRTFileInputStream.h	C++ base class definitions for operations with input file streams	172
OSRTFileOutputStream.h	C++ base class definitions for operations with output file streams	172
OSRTHexTextInputStream.h	C++ hexadecimal text input stream filter class	173
OSRTInputStream.h	C++ base class definitions for operations with input streams	173
OSRTInputStreamIF.h	C++ interface class definitions for operations with input streams	173
OSRTMemBuf.h	174
OSRTMemoryInputStream.h	C++ base class definitions for operations with input memory streams	174
OSRTMemoryOutputStream.h	C++ base class definitions for operations with output memory streams	174
OSRTMsgBuf.h	C++ run-time message buffer class definition	175
OSRTMsgBufIF.h	C++ run-time message buffer interface class definition	175

OSRTOutputStream.h	C++ base class definitions for operations with output streams	176
OSRTOutputStreamIF.h	C++ interface class definitions for operations with output streams	176
OSRTSocket.h	TCP/IP or UDP socket class definitions	176
OSRTSocketInputStream.h	C++ base class definitions for operations with input socket streams	177
OSRTSocketOutputStream.h	C++ base class definitions for operations with output socket streams	177
OSRTStream.h	C++ base class definitions for operations with I/O streams	178
OSRTStreamIF.h	C++ interface class definitions for operations with I/O streams	178
OSRTString.h	C++ string class definition	178
OSRTStringConst.h	C++ string constant class	179
OSRTStringIF.h	C++ string class interface	179
OSRTStringTokenizer.h		??
OSRTUTF8String.h	C++ UTF-8 string class definition	180
OSRTVoidPtrList.h	Void pointer list class definition	180
rtxCppAnyAttr.h	C++ any element class definition	181
rtxCppAnyElement.h	C++ any element class definition	181
rtxCppBitString.h	Contains utility function for sizing a bit string	181
rtxCppBufferedInputStream.h		183
rtxCppDateTime.h	C++ XML schema date/time definition	183
rtxCppDList.h		184
rtxCppDynOctStr.h	C++ dynamic binary string class definition	184
rtxCppTypes.h	C++ common type and class definitions	185
rtxCppXmlSTLString.h	C++ XML STL string class definition	185
rtxCppXmlSTLStringList.h	C++ XML STL string list class definition	185
rtxCppXmlString.h	C++ XML string class definition	185
rtxCppXmlStringList.h	C++ XML string list class definition	186

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

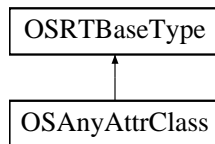
Class Documentation

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 rtxCppAnyAttr.h.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 OSAnyAttrClass() [1/5]

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

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

Parameters

<i>pname</i>	- attribute name
<i>pvalue</i>	- attribute contents

7.1.2.2 OSAnyAttrClass() [2/5]

```
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

<i>pname</i>	- attribute name
<i>pvalue</i>	- attribute contents

7.1.2.3 OSAnyAttrClass() [3/5]

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

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

Parameters

<i>pname</i>	- Attribute name.
<i>pvalue</i>	- Attribute value.

7.1.2.4 OSAnyAttrClass() [4/5]

```
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

<i>os</i>	- C binary string structure.
-----------	------------------------------

7.1.2.5 OSAnyAttrClass() [5/5]

```
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

<i>os</i>	- C++ binary string object reference.
-----------	---------------------------------------

7.1.3 Member Function Documentation

7.1.3.1 clone()

```
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 copyValue()

```
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

<i>pname</i>	- Attribute name.
<i>pvalue</i>	- Attribute value.

7.1.3.3 setValue() [1/2]

```
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

<i>pname</i>	- Attribute name.
<i>pvalue</i>	- Attribute value.

7.1.3.4 setValue() [2/2]

```
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

<i>pname</i>	- Attribute name.
<i>pvalue</i>	- Attribute value.
<i>namebytes</i>	- Attribute name length.
<i>valuebytes</i>	- Attribute value length.

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

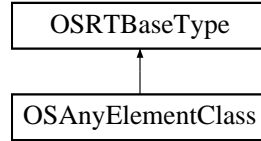
- [rtxCppAnyAttr.h](#)

7.2 OSAnyElementClass Class Reference

Any element.

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

Inheritance diagram for OSAnyElementClass:



Public Member Functions

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

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() [1/4]

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

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

Parameters

<i>pname</i>	- element name
<i>pvalue</i>	- element contents

7.2.2.2 OSAnyElementClass() [2/4]

```
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

<i>pname</i>	- element name
<i>pvalue</i>	- element contents

7.2.2.3 OSAnyElementClass() [3/4]

```
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

<i>os</i>	- C binary string structure.
-----------	------------------------------

7.2.2.4 OSAnyElementClass() [4/4]

```
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

<i>os</i>	- C++ binary string object reference.
-----------	---------------------------------------

7.2.3 Member Function Documentation

7.2.3.1 copyValue()

```
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

<i>pname</i>	- Element name.
<i>pvalue</i>	- Element value.

7.2.3.2 print()

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

This method prints the given element value to standard output.

Parameters

<i>pname</i>	- Name of generated string variable.
--------------	--------------------------------------

Definition at line 109 of file rtxCppAnyElement.h.

7.2.3.3 setValue()

```
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

<i>pname</i>	- Element name.
<i>pvalue</i>	- Element value.

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

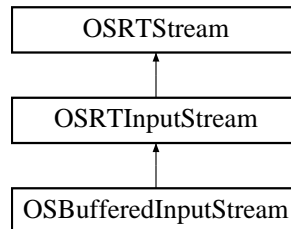
- [rtxCppAnyElement.h](#)

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.

Additional Inherited Members

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::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

<i>OSStreamException</i>	Stream create or initialize failed.
--------------------------	-------------------------------------

7.3.2.2 ~OSBufferedInputStream()

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

Virtual destructor.

Closes the stream if it was opened.

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

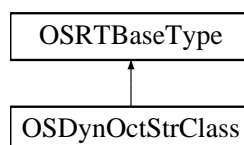
- [rtxCppBufferedInputStream.h](#)

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](#) (OSSIZE 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](#) (OSDynOctStr64 &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](#) (OSSIZE 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](#) (OSSIZE 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() [1/4]

```
OSDynOctStrClass::OSDynOctStrClass (
    OSSIZE numocts_,
    const OSOCTET * data_ )
```

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

Parameters

<i>numocts</i> ↔	- Number of bytes in the binary string.
<i>data_</i>	- The binary string data values.

7.4.2.2 OSDynOctStrClass() [2/4]

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

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

Parameters

<i>os</i>	- C binary string structure.
-----------	------------------------------

7.4.2.3 OSDynOctStrClass() [3/4]

```
OSDynOctStrClass::OSDynOctStrClass (
    OSDynOctStr64 & os )
```

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

Parameters

<i>os</i>	- C binary string structure.
-----------	------------------------------

7.4.2.4 OSDynOctStrClass() [4/4]

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

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

Parameters

<i>os</i>	- C++ binary string object reference.
-----------	---------------------------------------

7.4.3 Member Function Documentation

7.4.3.1 clone()

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

Clone method.

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

Reimplemented from [OSRTBaseType](#).

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

7.4.3.2 copyValue()

```
void OSDynOctStrClass::copyValue (
    OSSIZE 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

<i>numocts_</i> ↔	- Number of bytes in the binary string.
<i>data_</i>	- The binary string data values.

7.4.3.3 setValue() [1/2]

```
void OSDynOctStrClass::setValue (
    OSSIZE 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

<i>numocts</i> ↔	- Number of bytes in the binary string.
<i>data_</i>	- The binary string data values.

7.4.3.4 setValue() [2/2]

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

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

Parameters

<i>hexstr</i>	- Hex char string value.
<i>nchars</i>	- Number of characters in string. If zero, characters are read up to null-terminator.

Returns

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

7.4.3.5 setValueFromBase64()

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

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

Parameters

<i>base64str</i>	- Base64 char string value.
<i>nchars</i>	- 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:

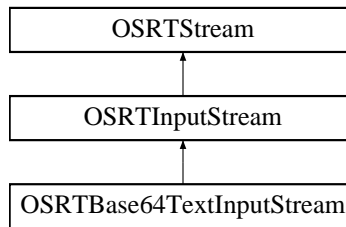
- [rtxCppDynOctStr.h](#)

7.5 OSRTBase64TextInputStream Class Reference

Hexadecimal text input stream filter class.

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

Inheritance diagram for OSRTBase64TextInputStream:



Public Member Functions

- EXTRTMETHOD [OSRTBase64TextInputStream](#) ([OSRTInputStream](#) *pstream)
Initializes the input stream using the existing standard input stream.
- EXTRTMETHOD [~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.

Additional Inherited Members

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 OSRTBase64TextInputStream()

```
EXTRTMETHOD OSRTBase64TextInputStream::OSRTBase64TextInputStream (
    OSRTInputStream * pstream )
```

Initializes the input stream using the existing standard input stream.

Only file and memory underlying stream types are supported.

Parameters

<i>pstream</i>	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 ~OSRTBase64TextInputStream()

```
EXTRIMETHOD 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 isA()

```
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

<i>id</i>	Enumerated stream identifier
-----------	------------------------------

Returns

True if the stream matches the identifier

Reimplemented from [OSRTInputStream](#).

Definition at line 72 of file OSRTBase64TextInputStream.h.

References OSRTInputStream::isA().

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

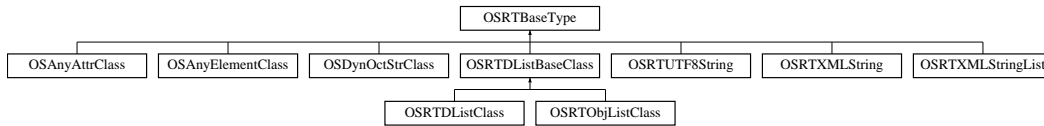
- [OSRTBase64TextInputStream.h](#)

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 mCtxt member variable and sets the reference count variable (mCount) to zero.
- virtual EXTRTMETHOD [~OSRTContext](#) ()
The destructor frees all memory held by the context.
- OSCTX * [getPtr](#) ()
The getPtr method returns a pointer to the mCtxt 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 `mbnInitialized`

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 64 of file OSRTContext.h.

7.7.2 Member Function Documentation

7.7.2.1 `getErrorInfo()` [1/3]

```
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.

7.7.2.2 `getErrorInfo()` [2/3]

```
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

<i>pBufSize</i>	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 `getErrorInfo()` [3/3]

```
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

<i>pBuf</i>	A pointer to a destination buffer to obtain the error text. If NULL, dynamic buffer will be allocated.
<i>bufSize</i>	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.

7.7.2.4 `getPtr()`

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

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

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

Definition at line 109 of file `OSRTContext.h`.

Referenced by `OSRTCtxtPtr::getCtxtPtr()`.

7.7.2.5 `getStatus()`

```
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 124 of file `OSRTContext.h`.

7.7.2.6 isInitialized()

```
OSBOOL OSRTContext::isInitialized ( ) [inline]
```

Returns TRUE, if initialized correctly, FALSE otherwise.

Returns

TRUE, if initialized correctly, FALSE otherwise.

Definition at line 132 of file OSRTContext.h.

7.7.2.7 memAlloc()

```
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

<i>numocts</i>	- Number of bytes of memory to allocate
----------------	---

Definition at line 195 of file OSRTContext.h.

7.7.2.8 memAllocZ()

```
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

<i>numocts</i>	- Number of bytes of memory to allocate
----------------	---

Definition at line 208 of file OSRTContext.h.

7.7.2.9 memFreeAll()

```
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 218 of file OSRTContext.h.

7.7.2.10 memFreePtr()

```
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

<i>ptr</i>	- Pointer to a block of memory allocated with <code>memAlloc</code>
------------	---

Definition at line 230 of file OSRTContext.h.

7.7.2.11 memRealloc()

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

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

Parameters

<i>ptr</i>	- Original pointer containing dynamic memory to be resized.
<i>numocts</i>	- Number of bytes of memory to allocate

Returns

Reallocated memory pointer

Definition at line 243 of file OSRTContext.h.

7.7.2.12 setDiag()

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

The setDiag method will turn diagnostic tracing on or off.

Parameters

<i>value</i>	- 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 setRunTimeKey()

```
virtual EXTRIMETHOD 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

<i>key</i>	- array of octets with the key
<i>keylen</i>	- number of octets in key array.

Returns

Completion status of operation:

- 0 = success,

- negative return value is error.

7.7.2.14 setStatus()

```
int OSRTCtxtHolder::setStatus (
    int stat )
```

This method sets error status in the context.

Parameters

<i>stat</i>	Status value.
-------------	---------------

Returns

Error status value being set.

7.7.3 Member Data Documentation

7.7.3.1 mStatus

```
int OSRTCtxtHolder::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 90 of file OSRTCtxtHolder.h.

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

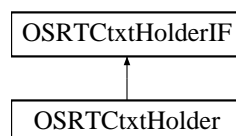
- [OSRTCtxtHolder.h](#)

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.

Additional Inherited Members

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 OSRTCtxtHolder()

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

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

Parameters

<i>pContext</i>	Pointer to a context to use. If NULL, new context will be allocated.
-----------------	--

7.8.3 Member Function Documentation

7.8.3.1 getContext()

```
virtual EXTRIMETHOD OSRTCtxtPtr OSRTCtxtHolder::getContext ( ) [virtual]
```

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

Returns

Context smart pointer object.

Implements [OSRTCtxtHolderIF](#).

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

7.8.3.2 getCtxtPtr()

```
virtual EXTRIMETHOD 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](#).

Referenced by `OSRTMessageBuffer::getCtxtPtr()`, and `OSRTStream::getCtxtPtr()`.

7.8.3.3 `getErrorInfo()` [1/2]

```
virtual EXTRIMETHOD 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](#).

Referenced by `OSRTMessageBuffer::getErrorInfo()`, and `OSRTStream::getErrorInfo()`.

7.8.3.4 `getErrorInfo()` [2/2]

```
virtual EXTRIMETHOD 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

<i>pBuf</i>	A pointer to a destination buffer to obtain the error text. If NULL, dynamic buffer will be allocated.
<i>bufSize</i>	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](#).

7.8.3.5 getStatus()

```
virtual EXTRIMETHOD 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](#).

Referenced by `OSRTMessageBuffer::getStatus()`.

7.8.4 Member Data Documentation

7.8.4.1 mpContext

```
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:

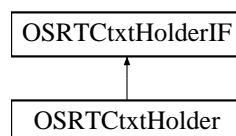
- [OSRTCtxtHolder.h](#)

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 `~OSRTCtxtHolderIF()`

```
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 getContext()

```
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 getCtxtPtr()

```
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 getErrorInfo() [1/2]

```
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](#).

7.9.3.4 `getErrorInfo()` [2/2]

```
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

<i>pBuf</i>	A pointer to a destination buffer to obtain the error text. If NULL, dynamic buffer will be allocated.
<i>bufSize</i>	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](#).

7.9.3.5 getStatus()

```
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 <OSRTCcontext.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 310 of file `OSRTContext.h`.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 `OSRTCtxtPtr()` [1/2]

```
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

<i>rf</i>	- Pointer to OSRTContext object
-----------	---

Definition at line 325 of file OSRTContext.h.

References OSRTContext::_ref().

7.10.2.2 OSRTCtxtPtr() [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

<i>o</i>	- Reference to OSRTCtxtPtr object to be copied
----------	--

Definition at line 335 of file OSRTContext.h.

References OSRTContext::_ref().

7.10.2.3 ~OSRTCtxtPtr()

```
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 344 of file OSRTContext.h.

References OSRTContext::_unref().

7.10.3 Member Function Documentation

7.10.3.1 operator=()

```
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

<code>rf</code>	- Pointer to OSRTCtxtPtr smart-pointer object
-----------------	---

Definition at line 354 of file OSRTContext.h.

References OSRTContext::_ref(), OSRTContext::_unref(), and mPointer.

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

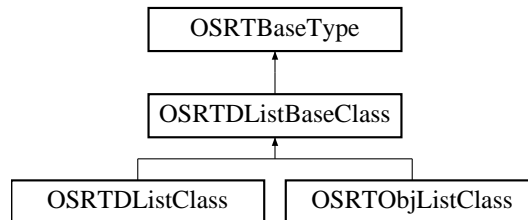
- [OSRTContext.h](#)

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 getCount()

```
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 getList() [1/2]

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

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

Returns

- a pointer to OSRTDList structure for the list instance.

Definition at line 211 of file rtxCppDList.h.

7.11.2.3 `getList()` [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`.

7.11.2.4 `remove()`

```
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

<i>index</i>	- Zero-based index of item to be removed.
--------------	---

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

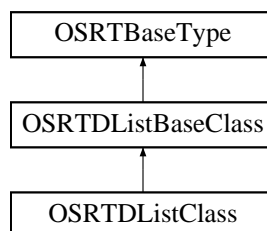
- [rtxCppDList.h](#)

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 `append()`

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

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

Parameters

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

7.12.2.2 appendCopy()

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

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

Parameters

<i>pdata</i>	- 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.
<i>nbytes</i>	- Size of the data pointed to in bytes.

7.12.2.3 getHead() [1/2]

```
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.

7.12.2.4 getHead() [2/2]

```
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.

7.12.2.5 getItem()

```
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

<i>idx</i>	- 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 getTail() [1/2]

```
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.

7.12.2.7 getTail() [2/2]

```
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.

7.12.2.8 insert()

```
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

<i>index</i>	- Zero-based index of insertion point.
<i>pdata</i>	- 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:

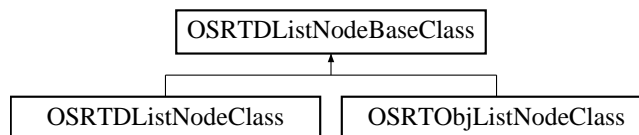
- [rtxCppDList.h](#)

7.13 OSRTDListNodeBaseClass 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 OSRTDListNodeBaseClass:



Friends

- class **OSRTDListBaseClass**
- class **OSRTDListClass**
- 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 OSRTDListNode type.

Definition at line 37 of file rtxCppDList.h.

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

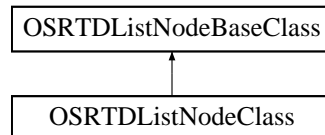
- [rtxCppDList.h](#)

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 `getData()` [1/2]

```
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 `OSRTDListNodeClass::getItem()`.

7.14.2.2 `getData()` [2/2]

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

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

Returns

Node data pointer.

Definition at line 73 of file `rtxCppDList.h`.

7.14.2.3 `getNext()` [1/2]

```
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`.

7.14.2.4 getNext() [2/2]

```
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.

7.14.2.5 getPrev() [1/2]

```
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.

7.14.2.6 getPrev() [2/2]

```
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.

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

- [rtxCppDList.h](#)

7.15 OSRTElemNameGuard Class Reference

Element name guard class.

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

7.15.1 Detailed Description

Element name guard class.

This class pushes and pops element names from the element name stack within the context for diagnostic and error logging.

Definition at line 421 of file OSRTContext.h.

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

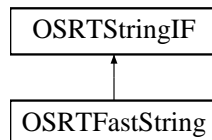
- [OSRTContext.h](#)

7.16 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.

Additional Inherited Members

7.16.1 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`.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 OSRTFastString() [1/3]

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

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

Parameters

<code>strval</code>	- Null-terminated C string value
---------------------	----------------------------------

7.16.2.2 OSRTFastString() [2/3]

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

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

Parameters

<i>strval</i>	- Null-terminated C string value
---------------	----------------------------------

7.16.2.3 OSRTFastString() [3/3]

```
OSRTFastString::OSRTFastString (
    const OSRTFastString & str )
```

Copy constructor.

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

Parameters

<i>str</i>	- C++ string object to be copied.
------------	-----------------------------------

7.16.3 Member Function Documentation

7.16.3.1 print()

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

This method prints the string value to standard output.

Parameters

<i>name</i>	- Name of generated string variable.
-------------	--------------------------------------

Implements [OSRTStringIF](#).

Definition at line 109 of file OSRTFastString.h.

References [OSRTStringIF::setValue\(\)](#).

7.16.3.2 `setValue()` [1/2]

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

This method sets the string value to the given string.

Parameters

<i>str</i>	- C null-terminated string.
------------	-----------------------------

Implements [OSRTStringIF](#).

7.16.3.3 `setValue()` [2/2]

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

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

Parameters

<i>str</i>	- C null-terminated UTF-8 string.
------------	-----------------------------------

Implements [OSRTStringIF](#).

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

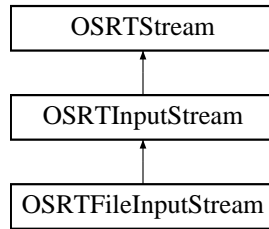
- [OSRTFastString.h](#)

7.17 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.

Additional Inherited Members

7.17.1 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`.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 `OSRTFileInputStream()` [1/4]

```

EXTRTMETHOD OSRTFileInputStream::OSRTFileInputStream (
    const char * pFilename )
  
```

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

Parameters

<i>pFilename</i>	Name of file.
------------------	---------------

See also

::rtxStreamFileOpen

7.17.2.2 OSRTFileInputStream() [2/4]

```
EXTRTMETHOD OSRTFileInputStream::OSRTFileInputStream (
    OSRTContext * pContext,
    const char * pFilename )
```

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

Parameters

<i>pContext</i>	Pointer to a context to use.
<i>pFilename</i>	Name of file.

See also

::rtxStreamFileOpen

7.17.2.3 OSRTFileInputStream() [3/4]

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

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

Parameters

<i>file</i>	Pointer to FILE structure.
-------------	----------------------------

See also

::rtxStreamFileAttach

7.17.2.4 OSRTFileInputStream() [4/4]

```
EXTRTMETHOD OSRTFileInputStream::OSRTFileInputStream (  
    OSRTContext * pContext,  
    FILE * file )
```

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

Parameters

<i>pContext</i>	Pointer to a context to use.
<i>file</i>	Pointer to FILE structure.

See also

`::rxStreamFileAttach`

7.17.3 Member Function Documentation

7.17.3.1 isA()

```
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

<i>id</i>	Enumerated stream identifier
-----------	------------------------------

Returns

True if the stream matches the identifier

Reimplemented from [OSRTInputStream](#).

Definition at line 84 of file OSRTFileInputStream.h.

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

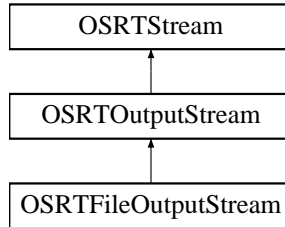
- [OSRTFileInputStream.h](#)

7.18 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.

Additional Inherited Members

7.18.1 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.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 OSRTFileOutputStream() [1/4]

```
EXTRTMETHOD OSRTFileOutputStream::OSRTFileOutputStream (  
    const char * pFilename )
```

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

Parameters

<i>pFilename</i>	Name of file.
------------------	---------------

Exceptions

<i>OSStreamException</i>	Stream create or initialize failed.
--------------------------	-------------------------------------

See also

::rtxStreamFileOpen

7.18.2.2 OSRTFileOutputStream() [2/4]

```
EXTRTMETHOD OSRTFileOutputStream::OSRTFileOutputStream (
    OSRTContext * pContext,
    const char * pFilename )
```

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

Parameters

<i>pContext</i>	Pointer to a context to use.
<i>pFilename</i>	Name of file.

Exceptions

<i>OSStreamException</i>	Stream create or initialize failed.
--------------------------	-------------------------------------

See also

::rtxStreamFileOpen

7.18.2.3 OSRTFileOutputStream() [3/4]

```
EXTRTMETHOD OSRTFileOutputStream::OSRTFileOutputStream (
    FILE * file )
```

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

Parameters

<i>file</i>	Pointer to FILE structure.
-------------	----------------------------

Exceptions

<i>OSStreamException</i>	Stream create or initialize failed.
--------------------------	-------------------------------------

See also

`::rxStreamFileAttach`

7.18.2.4 OSRTFileOutputStream() [4/4]

```
EXTRTMETHOD OSRTFileOutputStream::OSRTFileOutputStream (  
    OSRTContext * pContext,  
    FILE * file )
```

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

Parameters

<i>pContext</i>	Pointer to a context to use.
<i>file</i>	Pointer to FILE structure.

Exceptions

<i>OSStreamException</i>	Stream create or initialize failed.
--------------------------	-------------------------------------

See also

`::rxStreamFileAttach`

7.18.3 Member Function Documentation

7.18.3.1 isA()

```
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

<i>id</i>	Enumerated stream identifier
-----------	------------------------------

Returns

True if the stream matches the identifier

Reimplemented from [OSRTOutputStream](#).

Definition at line 88 of file OSRTFileOutputStream.h.

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

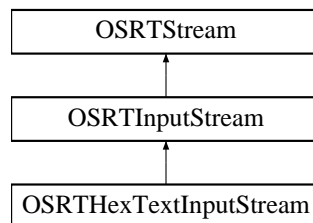
- [OSRTFileOutputStream.h](#)

7.19 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.

Additional Inherited Members

7.19.1 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.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 OSRTHexTextInputStream()

```
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

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

See also

`::rxStreamHexTextAttach`

7.19.2.2 ~OSRTHexTextInputStream()

```
EXTRTMETHOD OSRTHexTextInputStream::~OSRTHexTextInputStream ( )
```

The destructor deletes the underlying stream object.

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

7.19.3 Member Function Documentation

7.19.3.1 isA()

```
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

<i>id</i>	Enumerated stream identifier
-----------	------------------------------

Returns

True if the stream matches the identifier

Reimplemented from [OSRTInputStream](#).

Definition at line 70 of file OSRTHexTextInputStream.h.

References OSRTInputStream::isA().

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

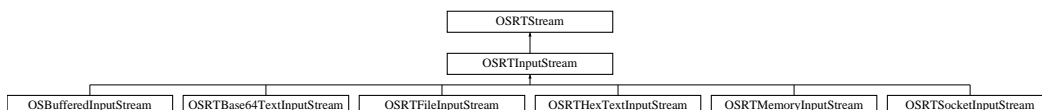
- [OSRTHexTextInputStream.h](#)

7.20 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 [OSRTCtxPtr](#) [getContext](#) ()
This method returns a pointer to the underlying [OSRTContext](#) object.
- virtual OSCTXT * [getCtxtPtr](#) ()
This method returns a pointer to the underlying OSCTXT object.
- virtual char * [getErrorInfo](#) ()
Returns error text in a dynamic memory buffer.
- virtual char * [getErrorInfo](#) (char *pBuf, [size_t](#) &bufSize)
Returns error text in a memory buffer.
- virtual int [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 [isOpen](#) ()
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](#) (OSOCKET *pDestBuf, [size_t](#) maxToRead)
Read data from the stream.
- virtual EXTRTMETHOD long [readBlocking](#) (OSOCKET *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.

Additional Inherited Members

7.20.1 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.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 OSRTInputStream()

```
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

<i>OSRTStreamException</i>	Stream create or initialize failed.
----------------------------	-------------------------------------

7.20.2.2 ~OSRTInputStream()

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

Virtual destructor.

Closes the stream if it was opened.

7.20.3 Member Function Documentation

7.20.3.1 close()

```
virtual EXTRIMETHOD 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

`::rxStreamClose`

Reimplemented from [OSRTStream](#).

7.20.3.2 currentPos()

```
virtual EXTRIMETHOD 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.

7.20.3.3 flush()

```
virtual EXTRIMETHOD 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

`::rxStreamFlush`

Reimplemented from [OSRTStream](#).

7.20.3.4 `getContext()`

```
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()`.

7.20.3.5 `getCtxtPtr()`

```
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()`.

7.20.3.6 `getErrorInfo()` [1/2]

```
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().

7.20.3.7 `getErrorInfo()` [2/2]

```
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

<i>pBuf</i>	A pointer to a destination buffer to obtain the error text. If NULL, dynamic buffer will be allocated.
<i>bufSize</i>	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().

7.20.3.8 getPosition()

```
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

<i>ppos</i>	Pointer to a variable to receive position.
-------------	--

Returns

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

7.20.3.9 getStatus()

```
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.

Definition at line 175 of file `OSRTInputStream.h`.

References `OSRTStream::getStatus()`, and `OSRTStream::isOpened()`.

7.20.3.10 isA()

```
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

<i>id</i>	Enumerated stream identifier
-----------	------------------------------

Returns

True if the stream matches the identifier

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

Definition at line 97 of file OSRTInputStream.h.

Referenced by [OSRTHexTextInputStream::isA\(\)](#), and [OSRTBase64TextInputStream::isA\(\)](#).

7.20.3.11 isOpened()

```
virtual EXTRIMETHOD OSBOOL OSRTInputStream::isOpened ( ) [virtual]
```

Checks, is the stream opened or not.

Returns

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

See also

[::rxStreamIsOpened](#)

Reimplemented from [OSRTStream](#).

7.20.3.12 mark()

```
virtual EXTRIMETHOD 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

<i>readAheadLimit</i>	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

7.20.3.13 markSupported()

```
virtual EXTRIMETHOD 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

7.20.3.14 read()

```
virtual EXTRIMETHOD 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

<i>pDestBuf</i>	Pointer to a buffer to receive a data.
<i>maxToRead</i>	Size of the buffer.

See also

::rtxStreamRead

7.20.3.15 readBlocking()

```
virtual EXTRIMETHOD 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

<i>pDestBuf</i>	Pointer to a buffer to receive a data.
<i>toReadBytes</i>	Number of bytes to be read.

See also

::rtxStreamRead

7.20.3.16 reset()

```
virtual EXTRIMETHOD 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

7.20.3.17 setPosition()

```
virtual int OSRTInputStream::setPosition (
    size_t pos ) [virtual]
```

Sets the current stream position to the given offset.

Parameters

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

Returns

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

7.20.3.18 skip()

```
virtual EXTRIMETHOD int OSRTInputStream::skip (
    size_t n ) [virtual]
```

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

Parameters

<i>n</i>	The number of octets to be skipped.
----------	-------------------------------------

Returns

Completion status of operation:

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

See also

`::rxStreamSkip`

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

- [OSRTInputStream.h](#)

7.21 OSRTMemBuf Class Reference

Memory Buffer class.

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

Inherits OSRTMEMBUF.

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:

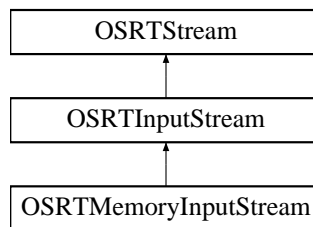
- [OSRTMemBuf.h](#)

7.22 OSRTMemoryInputStream Class Reference

Generic memory input stream.

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

Inheritance diagram for OSRTMemoryInputStream:



Public Member Functions

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

Additional Inherited Members

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 OSRTMemoryInputStream() [1/2]

```
EXTRTMETHOD OSRTMemoryInputStream::OSRTMemoryInputStream (
    const OSOCKET * pMemBuf,
    size_t bufSize )
```

Initializes the memory input stream using the specified memory buffer.

Parameters

<i>pMemBuf</i>	The pointer to the buffer.
<i>bufSize</i>	The size of the buffer.

See also

`::rtxStreamMemoryAttach`

7.22.2.2 OSRTMemoryInputStream() [2/2]

```
EXTRTMETHOD OSRTMemoryInputStream::OSRTMemoryInputStream (
    OSRTContext * pContext,
    const OSOCKET * pMemBuf,
    size_t bufSize )
```

Initializes the memory input stream using the specified memory buffer.

Parameters

<i>pContext</i>	Pointer to a context to use.
<i>pMemBuf</i>	The pointer to the buffer.
<i>bufSize</i>	The size of the buffer.

See also

`::rtxStreamMemoryAttach`

7.22.3 Member Function Documentation

7.22.3.1 `isA()`

```
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

<i>id</i>	Enumerated stream identifier
-----------	------------------------------

Returns

True if the stream matches the identifier

Reimplemented from [OSRTInputStream](#).

Definition at line 68 of file `OSRTMemoryInputStream.h`.

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

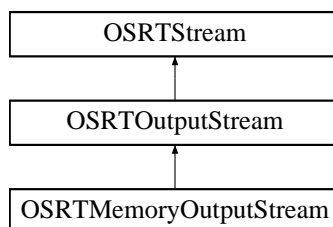
- [OSRTMemoryInputStream.h](#)

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

Additional Inherited Members

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 OSRTMemoryOutputStream() [1/3]

```
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 OSRTMemoryOutputStream() [2/3]

```
EXTRTMETHOD OSRTMemoryOutputStream::OSRTMemoryOutputStream (
    OSOCKETET * 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

<i>pMemBuf</i>	The pointer to the buffer.
<i>bufSize</i>	The size of the buffer.

See also

`::rtxStreamMemoryAttach`

7.23.2.3 OSRTMemoryOutputStream() [3/3]

```
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

<i>pContext</i>	Pointer to a context to use.
<i>pMemBuf</i>	The pointer to the buffer.
<i>bufSize</i>	The size of the buffer.

See also

`::rtxStreamMemoryAttach`

7.23.3 Member Function Documentation

7.23.3.1 getBuffer()

```
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

<i>pSize</i>	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 isA()

```
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

<i>id</i>	Enumerated stream identifier
-----------	------------------------------

Returns

True if the stream matches the identifier

Reimplemented from [OSRTOutputStream](#).

Definition at line 94 of file OSRTMemoryOutputStream.h.

7.23.3.3 reset()

```
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:

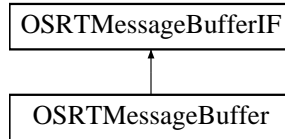
- [OSRTMemoryOutputStream.h](#)

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 * getApplInfo ()`
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 `OSRtCtxPtr getContext ()`
The getContext method returns the underlying context smart-pointer object.
- virtual `OSCTXT * getCtxtPtr ()`
The getCtxtPtr method returns the underlying C runtime context.
- virtual `char * getErrorInfo ()`
Returns error text in a dynamic memory buffer.
- virtual `char * getErrorInfo (char *pBuf, size_t &bufSize)`
Returns error text in a memory buffer.
- virtual `OSOCKET * getMsgCopy ()`
The getMsgCopy method will return a copy of the encoded message managed by the object.
- virtual `const OSOCKET * getMsgPtr ()`
The getMsgPtr method will return a const pointer to the encoded message managed by the object.
- virtual `size_t getMsgLen ()`
This method returns the length in bytes of an encoded message.
- `int getStatus () const`
This method returns the completion status of previous operation.
- virtual `int init ()`
Initializes message buffer.
- virtual `EXTRTMETHOD int initBuffer (OSOCKET *pMsgBuf, size_t msgBufLen)`
This version of the overloaded initBuffer method initializes the message buffer to point at the given null-terminated character string.
- virtual `void printErrorInfo ()`
The printErrorInfo method prints information on errors contained within the context.
- virtual `void resetErrorInfo ()`
The resetErrorInfo method resets information on errors contained within the context.
- virtual `void setApplInfo (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 `OSRTMessageBuffer()`

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

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

Parameters

<i>bufferType</i>	Type of message buffer that is being created (for example, <code>XMLEncode</code>).
<i>pContext</i>	Pointer to a context to use. If NULL, new context will be allocated.

7.24.2.2 `~OSRTMessageBuffer()`

```
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 `getByteIndex()`

```
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 `getCtxtPtr()`

```
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.

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

7.24.3.3 `getErrorInfo()` [1/2]

```
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.

References OSRTCtxtHolder::getErrorInfo().

7.24.3.4 `getErrorInfo()` [2/2]

```
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

<i>pBuf</i>	A pointer to a destination buffer to obtain the error text. If NULL, dynamic buffer will be allocated.
<i>bufSize</i>	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.

References OSRTCtxtHolder::getErrorInfo().

7.24.3.5 getStatus()

```
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 167 of file OSRTMsgBuf.h.

References OSRTCtxtHolder::getStatus().

7.24.3.6 init()

```
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 178 of file OSRTMsgBuf.h.

References OSRTMessageBufferIF::initBuffer().

7.24.3.7 initBuffer()

```
virtual EXTRMETHOD 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

<i>pMsgBuf</i>	Pointer to message buffer.
<i>msgBufLen</i>	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 setDiag()

```
virtual EXTRIMETHOD void OSRTMessageBuffer::setDiag (  
    OSBOOL value = TRUE ) [virtual]
```

The setDiag method will turn diagnostic tracing on or off.

Parameters

<i>value</i>	- Boolean value (default = TRUE = on)
--------------	---------------------------------------

Implements [OSRTMessageBufferIF](#).

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

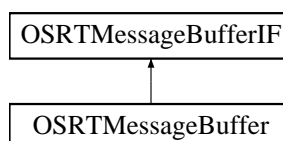
- [OSRTMsgBuf.h](#)

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 bufferSize)=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 ~OSRTMessageBufferIF()

```
virtual OSRTMessageBufferIF::~OSRTMessageBufferIF ( ) [inline], [protected], [virtual]
```

The virtual destructor does nothing.

It is overridden by derived versions of this class.

Definition at line 60 of file OSRTMsgBufIF.h.

7.25.3 Member Function Documentation

7.25.3.1 getByteIndex()

```
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 getMsgCopy()

```
virtual OSOCTET* 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 getMsgPtr()

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

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

Returns

The pointer to the encoded ASN.1 message.

Implemented in [OSRTMessageBuffer](#).

7.25.3.4 init()

```
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 initBuffer()

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

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

Parameters

<i>pMsgBuf</i>	Pointer to message buffer.
<i>msgBufLen</i>	Length of message buffer in bytes. string.

Returns

Completion status of operation:

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

Implemented in [OSRTMessageBuffer](#).

Referenced by OSRTMessageBuffer::init().

7.25.3.6 isA()

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

This method checks the type of the message buffer.

Parameters

<i>bufferType</i>	Enumerated identifier specifying a derived class. Possible values are: BEREncode, BERDecode, PEREncode, PERDecode, XMLEncode, XMLDecode, Stream, OEREncode, OERDecode, CBOREncode, CBORDecode.
-------------------	--

Returns

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

7.25.3.7 setDiag()

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

The setDiag method will turn diagnostic tracing on or off.

Parameters

<i>value</i>	- Boolean value (default = TRUE = on)
--------------	---------------------------------------

Implemented in [OSRTMessageBuffer](#).

Referenced by OSRTMessageBuffer::setAppInfo().

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

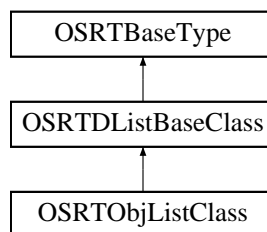
- [OSRTMsgBufIF.h](#)

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 `append()`

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

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

Parameters

<code><i>pdata</i></code>	- Pointer to data item to be appended to list. Note the pointer itself is appended - a copy is not made.
---------------------------	--

7.26.2.2 `appendCopy()`

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

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

Parameters

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

7.26.2.3 `getHead()` [1/2]

```
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.

7.26.2.4 getHead() [2/2]

```
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.

7.26.2.5 getItem()

```
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

<i>idx</i>	- 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 `getTail()` [1/2]

```
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`.

7.26.2.7 `getTail()` [2/2]

```
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`.

7.26.2.8 `insert()`

```
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

<i>index</i>	- Zero-based index of insertion point.
<i>pdata</i>	- Pointer to data item to be inserted into list. Note the pointer itself is inserted - a copy is not made.

7.26.2.9 operator=()

```
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:

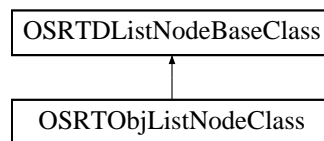
- [rtxCppDList.h](#)

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 `getData()` [1/2]

```
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()`.

7.27.2.2 `getData()` [2/2]

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

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

Returns

Node data pointer.

Definition at line 134 of file `rtxCppDList.h`.

7.27.2.3 getNext() [1/2]

```
OSRObjListNodeClass* OSRObjListNodeClass::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 rxCppDList.h.

7.27.2.4 getNext() [2/2]

```
const OSRObjListNodeClass* OSRObjListNodeClass::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 rxCppDList.h.

7.27.2.5 getPrev() [1/2]

```
OSRObjListNodeClass* OSRObjListNodeClass::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 rxCppDList.h.

7.27.2.6 `getPrev()` [2/2]

```
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`.

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

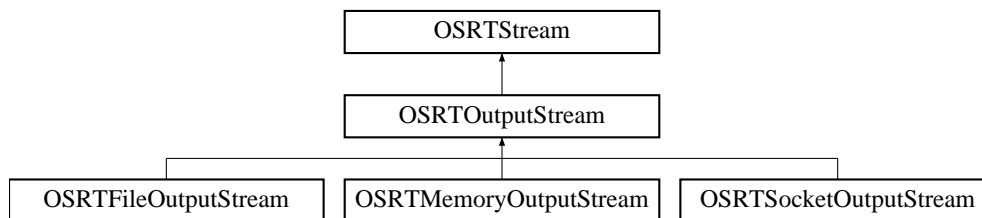
- [rtxCppDList.h](#)

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 [isOpened](#) ()
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 OSOCTET *pdata, size_t size)
Write data to the stream.
- virtual EXTRTMETHOD long [write](#) (const char *pdata)
Write data to the stream.

Additional Inherited Members

7.28.1 Detailed Description

The base class definition for operations with output streams.

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

Definition at line 43 of file OSRTOutputStream.h.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 OSRTOutputStream()

```
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 ~OSRTOutputStream()

```
virtual EXTRIMETHOD OSRTOutputStream::~~OSRTOutputStream ( ) [virtual]
```

Virtual destructor.

Closes the stream if it was opened.

7.28.3 Member Function Documentation

7.28.3.1 close()

```
virtual EXTRIMETHOD 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 flush()

```
virtual EXTRIMETHOD 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 `getContext()`

```
virtual OSRTCtxtPtr 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 `getCtxtPtr()`

```
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 `getErrorInfo()` [1/2]

```
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()`.

7.28.3.6 `getErrorInfo()` [2/2]

```
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

<i>pBuf</i>	A pointer to a destination buffer to obtain the error text. If NULL, dynamic buffer will be allocated.
<i>bufSize</i>	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().

7.28.3.7 `getStatus()`

```
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.

Definition at line 148 of file OSRTOutputStream.h.

References OSRTStream::getStatus().

7.28.3.8 `isA()`

```
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

<i>id</i>	Enumerated stream identifier
-----------	------------------------------

Returns

True if the stream matches the identifier

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

Definition at line 159 of file [OSRTOutputStream.h](#).

References [OSRTStream::isOpen\(\)](#).

7.28.3.9 `isOpen()`

```
virtual EXTRIMETHOD OSBOOL OSRTOutputStream::isOpen ( ) [virtual]
```

Checks if the stream open or not.

Returns

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

See also

[::rxStreamIsOpen](#)

Reimplemented from [OSRTStream](#).

7.28.3.10 `write()` [1/2]

```
virtual EXTRIMETHOD long OSRTOutputStream::write (
    const OSOCKET * 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

<i>pdata</i>	The pointer to the data to be written.
<i>size</i>	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

7.28.3.11 write() [2/2]

```
virtual EXTRIMETHOD 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

<i>pdata</i>	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

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, int retryCount=1)`
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 connectTimed (const char *host, int port, int nsecs)`
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.
- OSRTOCKET `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.
- EXTRTMETHOD `int send (const OSOCKET *pdata, size_t size)`
This method sends data on a connected socket.
- void `setOwnership (OSBOOL ownership)`
Transfers an ownership of the underlying O/S socket to or from the socket object.
- void `setRetryCount (int value)`
This method sets the socket connect retry count.

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
- int [mRetryCount](#)
number of time to retry socker connect
- 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 OSRTSocket() [1/3]

```
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 OSRTSocket() [2/3]

```
EXTRTMETHOD OSRTSocket::OSRTSocket (
    OSRTSOCKET socket,
    OSBOOL ownership = FALSE,
    int retryCount = 1 )
```

This constructor initializes an instance by using an existing socket.

Parameters

<i>socket</i>	An existing socket handle.
<i>ownership</i>	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.
<i>retryCount</i>	Number of times to retry a socket connect operation.

7.29.2.3 OSRTSocket() [3/3]

```
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 ~OSRTSocket()

```
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 accept()

```
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

<i>dest</i> ↔ <i>IP</i>	Optional pointer to a buffer that receives the IP address of the connecting entity. It may be NULL.
<i>port</i>	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 `addrToString()`

```
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

<i>ipAddr</i>	The IP address to be converted.
<i>pAddrStr</i>	Pointer to the buffer to receive a string with the IP address.
<i>bufsize</i>	Size of the buffer.

Returns

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

7.29.3.3 `bind()` [1/3]

```
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

<i>addr</i>	The local IP address to assign to the socket.
<i>port</i>	The local port number to assign to the socket.

Returns

Completion status of operation:

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

See also

::rtxSocketBind

7.29.3.4 bind() [2/3]

```
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

<i>pAddrStr</i>	Null-terminated character string representing a number expressed in the Internet standard "." (dotted) notation.
<i>port</i>	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 `bind()` [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

<i>port</i>	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 187 of file OSRTSocket.h.

7.29.3.6 `bindUrl()`

```
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

<i>url</i>	Univeral resource locator (URL) string.
------------	---

Returns

Completion status of operation:

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

See also

`::rtxSocketBind`

7.29.3.7 blockingRead()

```
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

<i>pbuf</i>	Pointer to the buffer for the incoming data.
<i>readBytes</i>	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 close()

```
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 connect()

```
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

<i>host</i>	Null-terminated character string representing a number expressed in the Internet standard "." (dotted) notation.
<i>port</i>	The destination port to connect.

Returns

Completion status of operation:

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

See also

::rtxSocketConnect

7.29.3.10 connectTimed()

```
EXTRTMETHOD int OSRTSocket::connectTimed (
    const char * host,
    int port,
    int nsecs )
```

This method establishes a connection to this socket.

It is similar to the socketConnect method except that it will only wait the given number of seconds to establish a connection before giving up.

Parameters

<i>host</i>	Null-terminated character string representing a number expressed in the Internet standard "." (dotted) notation.
<i>port</i>	The destination port to connect.
<i>nsecs</i>	Number of seconds to wait before failing.

Returns

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

7.29.3.11 connectUrl()

```
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

<i>url</i>	Univeral resource locator (URL) string.
------------	---

Returns

Completion status of operation:

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

See also

::rtxSocketConnect

7.29.3.12 getOwnership()

```
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 267 of file OSRTSocket.h.

7.29.3.13 getSocket()

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

This method returns the handle of the socket.

Returns

The handle of the socket.

Definition at line 274 of file OSRTSocket.h.

7.29.3.14 getStatus()

```
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 283 of file OSRTSocket.h.

7.29.3.15 listen()

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

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

Parameters

<i>maxConnections</i>	Maximum length of the queue of pending connections.
-----------------------	---

Returns

Completion status of operation:

- 0 = success,

- negative return value is error.

See also

`::rtxSocketListen`

7.29.3.16 recv()

```
EXTRTMETHOD 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

<i>pbuf</i>	Pointer to the buffer for the incoming data.
<i>bufsize</i>	Length of the buffer.

Returns

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

See also

`::rtxSocketRecv`

7.29.3.17 send()

```
EXTRTMETHOD 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

<i>pdata</i>	Buffer containing the data to be transmitted.
<i>size</i>	Length of the data in pdata.

Returns

Completion status of operation:

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

See also

::rxSocketSend

7.29.3.18 setOwnership()

```
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

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

Definition at line 336 of file OSRTSocket.h.

7.29.3.19 setRetryCount()

```
void OSRTSocket::setRetryCount (
    int value ) [inline]
```

This method sets the socket connect retry count.

The connect operation will be retried this many times if the operation fails. By default, the connect operation is not retried.

Parameters

<i>value</i>	Retry count.
--------------	--------------

Definition at line 345 of file OSRTSocket.h.

7.29.3.20 stringToAddr()

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

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

Parameters

<i>pAddrStr</i>	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:

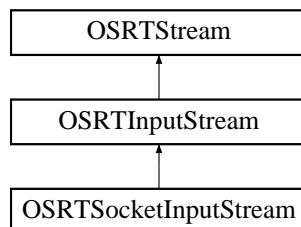
- [OSRTSocket.h](#)

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

Additional Inherited Members

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 [OSRTSocketInputStream\(\)](#) [1/4]

```
EXTRTMETHOD OSRTSocketInputStream::OSRTSocketInputStream (  
    OSRTSocket & socket )
```

Creates and initializes a socket input stream using the [OSRTSocket](#) instance of socket.

Parameters

<code>socket</code>	Reference to OSRTSocket instance.
---------------------	---

See also

`::rxStreamSocketAttach`

7.30.2.2 OSRTSocketInputStream() [2/4]

```
EXTRTMETHOD OSRTSocketInputStream::OSRTSocketInputStream (
    OSRTContext * pContext,
    OSRTSocket & socket )
```

Creates and initializes a socket input stream using the [OSRTSocket](#) instance of socket.

Parameters

<i>pContext</i>	Pointer to a context to use.
<i>socket</i>	Reference to OSRTSocket instance.

See also

`::rxStreamSocketAttach`

7.30.2.3 OSRTSocketInputStream() [3/4]

```
EXTRTMETHOD OSRTSocketInputStream::OSRTSocketInputStream (
    OSRTSOCKET socket,
    OSBOOL ownership = FALSE )
```

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

Parameters

<i>socket</i>	Handle of the socket.
<i>ownership</i>	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

`::rxStreamSocketAttach`

7.30.2.4 OSRTSocketInputStream() [4/4]

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

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

Parameters

<i>pContext</i>	Pointer to a context to use.
<i>socket</i>	Handle of the socket.
<i>ownership</i>	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 isA()

```
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

<i>id</i>	Enumerated stream identifier
-----------	------------------------------

Returns

True if the stream matches the identifier

Reimplemented from [OSRTInputStream](#).

Definition at line 107 of file OSRTSocketInputStream.h.

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

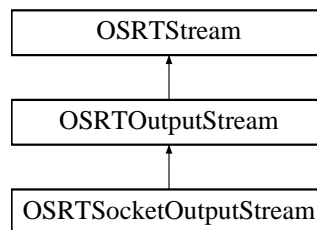
- [OSRTSocketInputStream.h](#)

7.31 OSRTSocketOutputStream Class Reference

Generic socket output stream.

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

Inheritance diagram for OSRTSocketOutputStream:



Public Member Functions

- EXTRMETHOD [OSRTSocketOutputStream](#) ([OSRTSocket](#) &socket)
Creates and initializes a socket output stream using the [OSRTSocket](#) instance of socket.
- EXTRMETHOD [OSRTSocketOutputStream](#) ([OSRTContext](#) *pContext, [OSRTSocket](#) &socket)
Creates and initializes a socket output stream using the [OSRTSocket](#) instance of socket.
- EXTRMETHOD [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

Additional Inherited Members

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 OSRTSocketOutputStream() [1/4]

```
EXTRTMETHOD OSRTSocketOutputStream::OSRTSocketOutputStream (
    OSRTSocket & socket )
```

Creates and initializes a socket output stream using the [OSRTSocket](#) instance of socket.

Parameters

<i>socket</i>	Reference to OSRTSocket instance.
---------------	---

See also

[::rtxStreamSocketAttach](#)

7.31.2.2 OSRTSocketOutputStream() [2/4]

```
EXTRTMETHOD OSRTSocketOutputStream::OSRTSocketOutputStream (
    OSRTContext * pContext,
    OSRTSocket & socket )
```

Creates and initializes a socket output stream using the [OSRTSocket](#) instance of socket.

Parameters

<i>pContext</i>	Pointer to a context to use.
<i>socket</i>	Reference to OSRTSocket instance.

See also

[::rtxStreamSocketAttach](#)

7.31.2.3 OSRTSocketOutputStream() [3/4]

```
EXTRTMETHOD OSRTSocketOutputStream::OSRTSocketOutputStream (
    OSRTSOCKET socket,
    OSBOOL ownership = FALSE )
```

Initializes the socket output stream using the socket handle.

Parameters

<i>socket</i>	Handle of the socket.
<i>ownership</i>	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() [4/4]

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

Initializes the socket output stream using the socket handle.

Parameters

<i>pContext</i>	Pointer to a context to use.
<i>socket</i>	Handle of the socket.
<i>ownership</i>	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 isA()

```
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

<i>id</i>	Enumerated stream identifier
-----------	------------------------------

Returns

True if the stream matches the identifier

Reimplemented from [OSRTOutputStream](#).

Definition at line 105 of file OSRTSocketOutputStream.h.

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

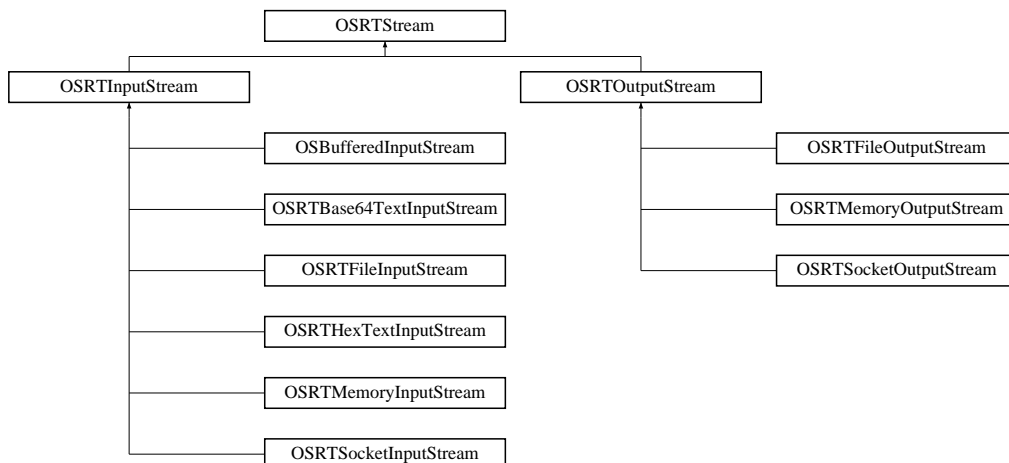
- [OSRTSocketOutputStream.h](#)

7.32 OSRTStream Class Reference

The default base class for using I/O streams.

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

Inheritance diagram for OSRTStream:



Public Member Functions

- virtual EXTRTMETHOD [~OSRTStream](#) ()
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 [OSRTContext](#) object.
- virtual OSCTXT * [getCtxtPtr](#) ()
This method returns a pointer to the underlying OSCTXT object.
- virtual char * [getErrorInfo](#) ()
Returns error text in a dynamic memory buffer.
- virtual char * [getErrorInfo](#) (char *pBuf, size_t &bufSize)
Returns error text in a memory buffer.
- int [getStatus](#) () const
This method returns the completion status of previous operation.
- virtual EXTRTMETHOD OSBOOL [isOpened](#) ()
Checks, is the stream opened or not.
- void [printErrorInfo](#) ()
The [printErrorInfo](#) method prints information on errors contained within the context.
- void [resetErrorInfo](#) ()
The [resetErrorInfo](#) method resets information on errors contained within the context.

Protected Member Functions

- 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 OSRTStream()

```
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 ~OSRTStream()

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

Virtual destructor.

Closes the stream if it was opened.

7.32.3 Member Function Documentation

7.32.3.1 close()

```
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

`::rxStreamClose`

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

7.32.3.2 flush()

```
virtual EXTRIMETHOD 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 getContext()

```
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`.

References `OSRTCtxtHolder::getContext()`.

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

7.32.3.4 `getCtxtPtr()`

```
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.

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

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

7.32.3.5 `getErrorInfo()` [1/2]

```
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.

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

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

7.32.3.6 `getErrorInfo()` [2/2]

```
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

<i>pBuf</i>	A pointer to a destination buffer to obtain the error text. If NULL, dynamic buffer will be allocated.
<i>bufSize</i>	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.

References OSRTCtxtHolder::getErrorInfo().

7.32.3.7 getStatus()

```
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.

Definition at line 155 of file OSRTStream.h.

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

7.32.3.8 isOpened()

```
virtual EXTRIMETHOD OSBOOL OSRTStream::isOpened ( ) [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](#).

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

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

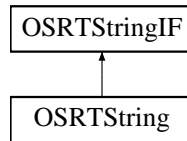
- [OSRTStream.h](#)

7.33 OSRTString Class Reference

C++ string class definition.

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

Inheritance diagram for OSRTString:



Public Member Functions

- **OSRTString** ()
The default constructor creates an empty string.
- **OSRTString** (const char *strval)
This constructor initializes the string to contain the given standard ASCII string value.
- **OSRTString** (const OSUTF8CHAR *strval)
This constructor initializes the string to contain the given UTF-8 string value.
- **OSRTString** (const **OSRTString** &str)
Copy constructor.
- virtual **~OSRTString** ()
The destructor frees string memory using the standard 'delete' operator.
- virtual **OSRTStringIF** * **clone** ()
This method creates a copy of the given string object.
- const char * **data** () const
*This method is a synonym for **getValue()**.*
- 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.
- int **indexOf** (char ch) const
This method returns the index of the first occurrence of the given character within the string or -1 if the character is not found.
- size_t **length** () const
This method returns the length of the string.
- virtual void **print** (const char *name)
This method prints the string value to standard output.
- virtual EXTRTMETHOD void **setValue** (const char *strval)
This method sets the string value to the given string.
- virtual EXTRTMETHOD void **setValue** (const OSUTF8CHAR *strval)
This method sets the string value to the given UTF-8 string value.
- virtual EXTRTMETHOD void **setValuePtr** (char *strval)
This method sets the string value to the given string value pointer.

- bool `toInt` (OSINT32 &value) const
This method converts the string to a signed 32-bit integer value.
- bool `toSize` (OSSIZE &value) const
This method converts the string to a size typed (site_t) value.
- bool `toUInt` (OSUINT32 &value) const
This method converts the string to an unsigned 32-bit integer value.
- bool `toUInt64` (OSUINT64 &value) const
This method converts the string to an unsigned 64-bit integer value.
- EXTRTMETHOD `OSRTString` & `operator=` (const `OSRTString` &original)
Assignment operator.

Additional Inherited Members

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 OSRTString() [1/3]

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

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

Parameters

<code>strval</code>	- Null-terminated C string value
---------------------	----------------------------------

7.33.2.2 OSRTString() [2/3]

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

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

Parameters

<i>strval</i>	- Null-terminated C string value
---------------	----------------------------------

7.33.2.3 OSRTString() [3/3]

```
OSRTString::OSRTString (  
    const OSRTString & str )
```

Copy constructor.

Parameters

<i>str</i>	- C++ string object to be copied.
------------	-----------------------------------

7.33.3 Member Function Documentation

7.33.3.1 print()

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

This method prints the string value to standard output.

Parameters

<i>name</i>	- Name of generated string variable.
-------------	--------------------------------------

Implements [OSRTStringIF](#).

Definition at line 130 of file OSRTString.h.

References [OSRTStringIF::getValue\(\)](#), and [OSRTStringIF::setValue\(\)](#).

7.33.3.2 setValue() [1/2]

```
virtual EXTRIMETHOD void OSRTString::setValue (  
    const char * strval ) [virtual]
```

This method sets the string value to the given string.

Parameters

<i>str</i>	- C null-terminated string.
------------	-----------------------------

Implements [OSRTStringIF](#).

7.33.3.3 setValue() [2/2]

```
virtual EXTRIMETHOD void OSRTString::setValue (
    const OSUTF8CHAR * strval ) [virtual]
```

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

Parameters

<i>str</i>	- C null-terminated UTF-8 string.
------------	-----------------------------------

Implements [OSRTStringIF](#).

7.33.3.4 setValuePtr()

```
virtual EXTRIMETHOD void OSRTString::setValuePtr (
    char * strval ) [virtual]
```

This method sets the string value to the given string value pointer.

This is assumed to be a mutable string allocated with the new operator. This class will assume ownership of the string memory.

Parameters

<i>str</i>	- Mutable null-terminated string allocated with new.
------------	--

7.33.3.5 toInt()

```
bool OSRTString::toInt (
    OSINT32 & value ) const
```

This method converts the string to a signed 32-bit integer value.

Parameters

<i>value</i>	Reference to variable to receive converted integer value.
--------------	---

Returns

Boolean result, true if successful or false if failed.

7.33.3.6 toSize()

```
bool OSRTString::toSize (
    OSSIZE & value ) const
```

This method converts the string to a size typed (site_t) value.

Parameters

<i>value</i>	Reference to variable to receive converted integer value.
--------------	---

Returns

Boolean result, true if successful or false if failed.

7.33.3.7 toUInt()

```
bool OSRTString::toUInt (
    OSUINT32 & value ) const
```

This method converts the string to an unsigned 32-bit integer value.

Parameters

<i>value</i>	Reference to variable to receive converted integer value.
--------------	---

Returns

Boolean result, true if successful or false if failed.

7.33.3.8 toUInt64()

```
bool OSRTString::toUInt64 (
    OSUINT64 & value ) const
```

This method converts the string to an unsigned 64-bit integer value.

Parameters

<i>value</i>	Reference to variable to receive converted integer value.
--------------	---

Returns

Boolean result, true if successful or false if failed.

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

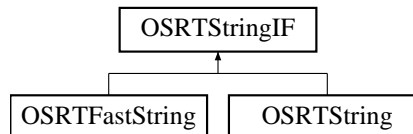
- [OSRTString.h](#)

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() [1/2]

```
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() [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 print()

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

This method prints the string value to standard output.

Parameters

<i>name</i>	- Name of generated string variable.
-------------	--------------------------------------

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

7.34.3.2 setValue() [1/2]

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

This method sets the string value to the given string.

Parameters

<i>str</i>	- C null-terminated string.
------------	-----------------------------

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

Referenced by [OSRTFastString::print\(\)](#), and [OSRTString::print\(\)](#).

7.34.3.3 setValue() [2/2]

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

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

Parameters

<i>utf8str</i>	- C null-terminated UTF-8 string.
----------------	-----------------------------------

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

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

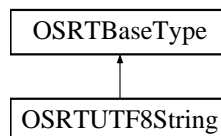
- [OSRTStringIF.h](#)

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() [1/3]

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

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

Parameters

<code>strval</code>	- String value
---------------------	----------------

7.35.2.2 OSRTUTF8String() [2/3]

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

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

Parameters

<i>strval</i>	- String value
---------------	----------------

7.35.2.3 OSRTUTF8String() [3/3]

```
OSRTUTF8String::OSRTUTF8String (
    const OSRTUTF8String & str )
```

Copy constructor.

Parameters

<i>str</i>	- C++ XML string class.
------------	-------------------------

7.35.3 Member Function Documentation

7.35.3.1 clone()

```
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 copyValue()

```
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

<i>str</i>	- C null-terminated string.
------------	-----------------------------

7.35.3.3 print()

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

This method prints the string value to standard output.

Parameters

<i>name</i>	- Name of generated string variable.
-------------	--------------------------------------

Definition at line 111 of file OSRTUTF8String.h.

7.35.3.4 setValue()

```
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

<i>str</i>	- C null-terminated string.
------------	-----------------------------

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

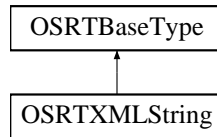
- [OSRTUTF8String.h](#)

7.36 OSRTXMLString Class Reference

XML string.

```
#include <rtxCppXmlString.h>
```

Inheritance diagram for OSRTXMLString:



Public Member Functions

- [OSRTXMLString \(\)](#)
The default constructor creates an empty string.
- [OSRTXMLString \(const OSUTF8CHAR *strval, OSBOOL cdata_=FALSE\)](#)
This constructor initializes the string to contain the value.
- [OSRTXMLString \(const OSUTF8CHAR *strval, size_t nbytes, OSBOOL cdata_=FALSE\)](#)
This constructor initializes the string to contain the value.
- [OSRTXMLString \(const char *strval, OSBOOL cdata_=FALSE\)](#)
This constructor initializes the string to contain the value.
- [OSRTXMLString \(const OSXMLSTRING &str\)](#)
Copy constructor.
- [OSRTXMLString \(const OSRTXMLString &str\)](#)
Copy constructor.
- virtual [~OSRTXMLString \(\)](#)
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 \(OSRTMessageBufferIF &msgbuf, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS\)](#)
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.
- OSRTXMLString & `operator=` (const OSRTXMLString &original)
Assignment operator.
- OSRTXMLString & `operator=` (const char *original)
Assignment operator for C strings.
- OSRTXMLString & `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.

7.36.1 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.

7.36.2 Constructor & Destructor Documentation

7.36.2.1 OSRTXMLString() [1/5]

```
OSRTXMLString::OSRTXMLString (
    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

<i>strval</i>	- String value
<i>CDATA</i> ↔	- Should string be encoded as a CDATA section?
—	

7.36.2.2 OSRXMLString() [2/5]

```
OSRXMLString::OSRXMLString (
    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

<i>strval</i>	- String value
<i>nbytes</i>	- Number of bytes to copy from source string
<i>CDATA</i> ↔	- Should string be encoded as a CDATA section?
—	

7.36.2.3 OSRXMLString() [3/5]

```
OSRXMLString::OSRXMLString (
    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

<i>strval</i>	- String value
<i>CDATA</i> ↔	- Should string be encoded as a CDATA section?
—	

7.36.2.4 OSRXMLString() [4/5]

```
OSRXMLString::OSRXMLString (
    const OSXMLSTRING & str )
```

Copy constructor.

Parameters

<i>str</i>	- C XML string structure.
------------	---------------------------

7.36.2.5 OSRXMLString() [5/5]

```
OSRXMLString::OSRXMLString (
    const OSRXMLString & str )
```

Copy constructor.

Parameters

<i>str</i>	- C++ XML string class.
------------	-------------------------

7.36.3 Member Function Documentation

7.36.3.1 appendValue()

```
void OSRXMLString::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

<i>utf8str</i>	- C null-terminated string.
<i>nbytes</i>	- length of utf8str in bytes.

7.36.3.2 clone()

```
OSRTBaseType* OSRTXMLString::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`.

7.36.3.3 compare()

```
int OSRTXMLString::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`.

7.36.3.4 copyValue() [1/2]

```
void OSRTXMLString::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

<i>utf8str</i>	- C null-terminated string.
<i>nbytes</i>	- length of <i>utf8str</i> in bytes.

7.36.3.5 copyValue() [2/2]

```
void OSRTXMLString::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

<i>cstring</i>	- C null-terminated string.
<i>nbytes</i>	- length of <i>cstring</i> in bytes.

Definition at line 176 of file `rtxCppXmlString.h`.

7.36.3.6 decodeXML()

```
virtual int OSRTXMLString::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

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

Returns

Completion status of operation:

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

7.36.3.7 encodeXML()

```
virtual int OSRTXMLString::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

<i>msgbuf</i>	Message buffer or stream object reference.
<i>elemName</i>	XML element name that should be added to encoded fragment.
<i>pNS</i>	Pointer to namespace structure.

Returns

Completion status of operation:

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

7.36.3.8 isCDATA()

```
OSBOOL OSRTXMLString::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.

7.36.3.9 print()

```
void OSRTXMLString::print (  
    const char * name ) [inline]
```

This method prints the string value to standard output.

Parameters

<i>name</i>	- Name of generated string variable.
-------------	--------------------------------------

Definition at line 253 of file rtxCppXmlString.h.

7.36.3.10 setCDATA()

```
void OSRTXMLString::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

<i>bvalue</i>	- Boolean value.
---------------	------------------

Definition at line 246 of file rtxCppXmlString.h.

7.36.3.11 setValue() [1/3]

```
void OSRTXMLString::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

<i>utf8str</i>	- UTF8 null-terminated string.
<i>nbytes</i>	- Number of bytes to copy from the source string. If zero, bytes are copied up to the null-terminator.

7.36.3.12 setValue() [2/3]

```
void OSRTXMLString::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

<i>cstring</i>	- C null-terminated string.
<i>nbytes</i>	- 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 `rtxCppXmlString.h`.

7.36.3.13 setValue() [3/3]

```
void OSRTXMLString::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

<i>membuf</i>	- Reference to a memory buffer object.
---------------	--

Definition at line 284 of file `rtxCppXmlString.h`.

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

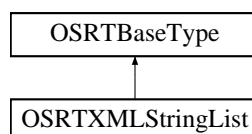
- [rtxCppXmlString.h](#)

7.37 OSRTXMLStringList Class Reference

XML list string.

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

Inheritance diagram for OSRTXMLStringList:



Public Member Functions

- [OSRXMLStringList \(\)](#)
The default constructor creates an empty list.
- [OSRXMLStringList \(const OSRXMLStringList &orig\)](#)
The copy constructor creates a deep-copy of the original list.
- [OSRXMLStringList & operator= \(const OSRXMLStringList &orig\)](#)
The assignment operator frees the existing list and then makes a deep-copy of the original list.
- void [append \(OSRXMLString *pdata\)](#)
The append method adds the given object to the end of the list.
- void [appendCopy \(OSRXMLString *pdata\)](#)
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.
- [OSRXMLString * 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

- [OSRObjListClass mElemList](#)
List of OSRXMLString objects.

7.37.1 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`.

7.37.2 Constructor & Destructor Documentation

7.37.2.1 OSRXMLStringList()

```
OSRXMLStringList::OSRXMLStringList (
    const OSRXMLStringList & orig )
```

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

Parameters

<i>orig</i>	Object to be copied.
-------------	----------------------

7.37.3 Member Function Documentation

7.37.3.1 append()

```
void OSRTXMLStringList::append (  
    OSRTXMLString * 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

<i>pdata</i>	Pointer to object to be appended.
--------------	-----------------------------------

7.37.3.2 appendCopy()

```
void OSRTXMLStringList::appendCopy (  
    OSRTXMLString * 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

<i>pdata</i>	Pointer to object to be appended.
--------------	-----------------------------------

7.37.3.3 clone()

```
virtual OSRTBaseType* OSRTXMLStringList::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](#).

7.37.3.4 operator=()

```
OSRTXMLStringList& OSRTXMLStringList::operator= (  
    const OSRTXMLStringList & orig )
```

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

Parameters

<i>orig</i>	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.

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.

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.
- class [OSRTElemNameGuard](#)
Element name guard class.

Functions

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

8.3.1 Detailed Description

C++ run-time context class definition.

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.

8.5 OSRTCtxtHolderIF.h File Reference

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

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

Classes

- class [OSRTCtxtHolderIF](#)
Abstract message buffer or stream interface class.

8.5.1 Detailed Description

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

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.

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.

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.

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.

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.

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.

8.12 OSRTMemBuf.h File Reference

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

Classes

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

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.

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.

8.15 OSRTMsgBuf.h File Reference

C++ run-time message buffer class definition.

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

Classes

- class [OSRTMessageBuffer](#)
Abstract message buffer base class.

8.15.1 Detailed Description

C++ run-time message buffer class definition.

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.

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.

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.

8.19 OSRTSocket.h File Reference

TCP/IP or UDP socket class definitions.

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

Classes

- class [OSRTSocket](#)
Wrapper class for TCP/IP or UDP sockets.

8.19.1 Detailed Description

TCP/IP or UDP socket class definitions.

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.

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.

8.22 OSRTStream.h File Reference

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

```
#include "rtxsrc/OSRTCtxtHolder.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.

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.

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.

8.25 OSRTStringConst.h File Reference

C++ string constant class.

```
#include <stddef.h>
#include "rtxsrc/osSysTypes.h"
#include "rtxsrc/rtxExternDefs.h"
```

8.25.1 Detailed Description

C++ string constant class.

8.26 OSRTStringIF.h File Reference

C++ string class interface.

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

Classes

- class [OSRTStringIF](#)

C++ string class interface.

8.26.1 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.

8.27 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.

8.27.1 Detailed Description

C++ UTF-8 string class definition.

8.28 OSRTVoidPtrList.h File Reference

Void pointer list class definition.

```
#include "osSysTypes.h"
#include "rtxsrc/rtxExternDefs.h"
```

8.28.1 Detailed Description

Void pointer list class definition.

This is used to hold a list of generic pointers.

8.29 rtxCppAnyAttr.h File Reference

C++ any element class definition.

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

Classes

- class [OSAnyAttrClass](#)
Any attribute.

8.29.1 Detailed Description

C++ any element class definition.

8.30 rtxCppAnyElement.h File Reference

C++ any element class definition.

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

Classes

- class [OSAnyElementClass](#)
Any element.

8.30.1 Detailed Description

C++ any element class definition.

8.31 rtxCppBitString.h File Reference

Contains utility function for sizing a bit string.

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

Functions

- EXTERNRT int [rtxCppCheckBitBounds](#) (OSOCKET *&pBits, OSUIN32 &numocts, size_t minRequiredBits, size_t preferredLimitBits)
Check whether the given bit string is large enough, and expand it if necessary.
- EXTERNRT int [rtxCppCheckBitBounds64](#) (OSOCKET *&pBits, OSSIZE &numocts, size_t minRequiredBits, size_t preferredLimitBits)
This function is identical to rtxCppCheckBitBounds except that it supports a 64-bit integer length on 64-bit systems.

8.31.1 Detailed Description

Contains utility function for sizing a bit string.

8.31.2 Function Documentation

8.31.2.1 rtxCppCheckBitBounds()

```
EXTERNRT int rtxCppCheckBitBounds (  
    OSOCKET *& pBits,  
    OSUIN32 & numocts,  
    size_t minRequiredBits,  
    size_t preferredLimitBits )
```

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

Parameters

<i>pBits</i>	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.
<i>numocts</i>	is the current size of the bit string in octets. If the bit string is expanded, numocts receives the new size.
<i>minRequiredBits</i>	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.
<i>preferredLimitBits</i>	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.

8.31.2.2 rtxCppCheckBitBounds64()

```
EXTERNRT int rtxCppCheckBitBounds64 (
    OSOCKET *& pBits,
    OSSIZE & numocts,
    size_t minRequiredBits,
    size_t preferredLimitBits )
```

This function is identical to rtxCppCheckBitBounds except that it supports a 64-bit integer length on 64-bit systems.

Parameters

<i>pBits</i>	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.
<i>numocts</i>	is the current size of the bit string in octets. If the bit string is expanded, numocts receives the new size.
<i>minRequiredBits</i>	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.
<i>preferredLimitBits</i>	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.

8.32 rtxCppBufferedInputStream.h File Reference

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

Classes

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

8.33 rtxCppDateTime.h File Reference

C++ XML schema date/time definition.

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

8.33.1 Detailed Description

C++ XML schema date/time definition.

8.34 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.

8.35 rtxCppDynOctStr.h File Reference

C++ dynamic binary string class definition.

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

Classes

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

8.35.1 Detailed Description

C++ dynamic binary string class definition.

8.36 rtxCppType.h File Reference

C++ common type and class definitions.

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

8.36.1 Detailed Description

C++ common type and class definitions.

8.37 rtxCppXmlSTLString.h File Reference

C++ XML STL string class definition.

8.37.1 Detailed Description

C++ XML STL string class definition.

8.38 rtxCppXmlSTLStringList.h File Reference

C++ XML STL string list class definition.

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

8.38.1 Detailed Description

C++ XML STL string list class definition.

8.39 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 [OSRTXMLString](#)
XML string.

8.39.1 Detailed Description

C++ XML string class definition.

8.40 rtxCppXmlStringList.h File Reference

C++ XML string list class definition.

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

Classes

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

8.40.1 Detailed Description

C++ XML string list class definition.

Index

- ~OSBufferedInputStream
 - OSBufferedInputStream, 27
- ~OSRTBase64TextInputStream
 - OSRTBase64TextInputStream, 35
- ~OSRTCtxtHolderIF
 - OSRTCtxtHolderIF, 48
- ~OSRTCtxtPtr
 - OSRTCtxtPtr, 53
- ~OSRTHexTextInputStream
 - OSRTHexTextInputStream, 76
- ~OSRTInputStream
 - OSRTInputStream, 79
- ~OSRTMessageBuffer
 - OSRTMessageBuffer, 95
- ~OSRTMessageBufferIF
 - OSRTMessageBufferIF, 100
- ~OSRTOutputStream
 - OSRTOutputStream, 112
- ~OSRTSocket
 - OSRTSocket, 120
- ~OSRTStream
 - OSRTStream, 140
- ASN.1 Stream Classes, 15
- accept
 - OSRTSocket, 120
- addrToString
 - OSRTSocket, 121
- append
 - OSRTDListClass, 57
 - OSRTObjListClass, 105
 - OSRTXMLStringList, 167
- appendCopy
 - OSRTDListClass, 58
 - OSRTObjListClass, 105
 - OSRTXMLStringList, 167
- appendValue
 - OSRTXMLString, 160
- bind
 - OSRTSocket, 121, 122
- bindUrl
 - OSRTSocket, 123
- blockingRead
 - OSRTSocket, 124
- clone
 - OSAnyAttrClass, 20
 - OSDynOctStrClass, 31
 - OSRTUTF8String, 155
 - OSRTXMLString, 161
 - OSRTXMLStringList, 167
- close
 - OSRTInputStream, 79
 - OSRTOutputStream, 113
 - OSRTSocket, 124
 - OSRTStream, 140
- compare
 - OSRTXMLString, 161
- connect
 - OSRTSocket, 124
- connectTimed
 - OSRTSocket, 125
- connectUrl
 - OSRTSocket, 126
- copyValue
 - OSAnyAttrClass, 20
 - OSAnyElementClass, 25
 - OSDynOctStrClass, 31
 - OSRTUTF8String, 155
 - OSRTXMLString, 161, 162
- currentPos
 - OSRTInputStream, 80
- decodeXML
 - OSRTXMLString, 162
- encodeXML
 - OSRTXMLString, 162
- flush
 - OSRTInputStream, 80
 - OSRTOutputStream, 113
 - OSRTStream, 140
- Generic Input Stream Classes, 11
- Generic Output Stream Classes, 13
- getBuffer
 - OSRTMemoryOutputStream, 92
- getByteIndex
 - OSRTMessageBuffer, 96
 - OSRTMessageBufferIF, 101

- getContext
 - OSRTCtxtHolder, 45
 - OSRTCtxtHolderIF, 49
 - OSRTInputStream, 80
 - OSRTOutputStream, 113
 - OSRTStream, 141
- getCount
 - OSRTDListBaseClass, 55
- getCtxtPtr
 - OSRTCtxtHolder, 45
 - OSRTCtxtHolderIF, 49
 - OSRTInputStream, 81
 - OSRTMessageBuffer, 96
 - OSRTOutputStream, 114
 - OSRTStream, 141
- getData
 - OSRTDListNodeClass, 62, 63
 - OSRTObjListNodeClass, 109
- getErrorInfo
 - OSRTCcontext, 38, 39
 - OSRTCtxtHolder, 45, 46
 - OSRTCtxtHolderIF, 49
 - OSRTInputStream, 81, 82
 - OSRTMessageBuffer, 96, 97
 - OSRTOutputStream, 114
 - OSRTStream, 142
- getHead
 - OSRTDListClass, 58
 - OSRTObjListClass, 105, 106
- getItem
 - OSRTDListClass, 58
 - OSRTObjListClass, 106
- getList
 - OSRTDListBaseClass, 55
- getMsgCopy
 - OSRTMessageBufferIF, 101
- getMsgPtr
 - OSRTMessageBufferIF, 101
- getNext
 - OSRTDListNodeClass, 63
 - OSRTObjListNodeClass, 109, 110
- getOwnership
 - OSRTSocket, 126
- getPosition
 - OSRTInputStream, 82
- getPrev
 - OSRTDListNodeClass, 64
 - OSRTObjListNodeClass, 110
- getPtr
 - OSRTCcontext, 39
- getSocket
 - OSRTSocket, 126
- getStatus
 - OSRTCcontext, 39
- OSRTCtxtHolder, 46
- OSRTCtxtHolderIF, 51
- OSRTInputStream, 83
- OSRTMessageBuffer, 97
- OSRTOutputStream, 115
- OSRTSocket, 127
- OSRTStream, 143
- getTail
 - OSRTDListClass, 59
 - OSRTObjListClass, 106, 107
- init
 - OSRTMessageBuffer, 98
 - OSRTMessageBufferIF, 102
- initBuffer
 - OSRTMessageBuffer, 98
 - OSRTMessageBufferIF, 102
- insert
 - OSRTDListClass, 59
 - OSRTObjListClass, 107
- isCDATA
 - OSRTXMLString, 163
- isInitialized
 - OSRTCcontext, 39
- isOpened
 - OSRTInputStream, 84
 - OSRTOutputStream, 116
 - OSRTStream, 143
- isA
 - OSRTBase64TextInputStream, 35
 - OSRTFileInputStream, 71
 - OSRTFileOutputStream, 74
 - OSRTHexTextInputStream, 76
 - OSRTInputStream, 83
 - OSRTMemoryInputStream, 90
 - OSRTMemoryOutputStream, 93
 - OSRTMessageBufferIF, 103
 - OSRTOutputStream, 115
 - OSRTSocketInputStream, 133
 - OSRTSocketOutputStream, 137
- listen
 - OSRTSocket, 127
- mStatus
 - OSRTCcontext, 43
- mark
 - OSRTInputStream, 84
- markSupported
 - OSRTInputStream, 85
- memAlloc
 - OSRTCcontext, 40
- memAllocZ
 - OSRTCcontext, 40
- memFreeAll

- OSRTContext, 41
- memFreePtr
 - OSRTContext, 41
- memRealloc
 - OSRTContext, 41
- Message Buffer Classes, 12
- mpContext
 - OSRTCtxtHolder, 47
- OSAnyAttrClass, 17
 - clone, 20
 - copyValue, 20
 - OSAnyAttrClass, 18, 19
 - setValue, 20, 21
- OSAnyElementClass, 21
 - copyValue, 25
 - OSAnyElementClass, 22, 23
 - print, 25
 - setValue, 25
- OSBufferedInputStream, 26
 - ~OSBufferedInputStream, 27
 - OSBufferedInputStream, 27
- OSDynOctStrClass, 27
 - clone, 31
 - copyValue, 31
 - OSDynOctStrClass, 28, 30
 - setValue, 31, 32
 - setValueFromBase64, 32
- OSRTBase64TextInputStream, 33
 - ~OSRTBase64TextInputStream, 35
 - isA, 35
 - OSRTBase64TextInputStream, 33
- OSRTBase64TextInputStream.h, 169
- OSRTBaseType, 36
- OSRTBaseType.h, 169
- OSRTContext, 36
 - getErrorInfo, 38, 39
 - getPtr, 39
 - getStatus, 39
 - isInitialized, 39
 - mStatus, 43
 - memAlloc, 40
 - memAllocZ, 40
 - memFreeAll, 41
 - memFreePtr, 41
 - memRealloc, 41
 - setDiag, 42
 - setRunTimeKey, 42
 - setStatus, 43
- OSRTContext.h, 170
- OSRTCtxtHolder, 43
 - getContext, 45
 - getCtxtPtr, 45
 - getErrorInfo, 45, 46
 - getStatus, 46
 - mpContext, 47
 - OSRTCtxtHolder, 44
- OSRTCtxtHolder.h, 170
- OSRTCtxtHolderIF.h, 171
- OSRTCtxtHolderIF, 47
 - ~OSRTCtxtHolderIF, 48
 - getContext, 49
 - getCtxtPtr, 49
 - getErrorInfo, 49
 - getStatus, 51
- OSRTCtxtPtr, 51
 - ~OSRTCtxtPtr, 53
 - OSRTCtxtPtr, 52, 53
 - operator=, 53
- OSRTDListBaseClass, 54
 - getCount, 55
 - getList, 55
 - remove, 56
- OSRTDListClass, 56
 - append, 57
 - appendCopy, 58
 - getHead, 58
 - getItem, 58
 - getTail, 59
 - insert, 59
- OSRTDListNodeBaseClass, 61
- OSRTDListNodeClass, 62
 - getData, 62, 63
 - getNext, 63
 - getPrev, 64
- OSRTElemNameGuard, 65
- OSRTFastString, 65
 - OSRTFastString, 66, 67
 - print, 67
 - setValue, 68
- OSRTFastString.h, 171
- OSRTFileInputStream, 68
 - isA, 71
 - OSRTFileInputStream, 69, 70
- OSRTFileInputStream.h, 172
- OSRTFileOutputStream, 72
 - isA, 74
 - OSRTFileOutputStream, 72–74
- OSRTFileOutputStream.h, 172
- OSRTHexTextInputStream, 75
 - ~OSRTHexTextInputStream, 76
 - isA, 76
 - OSRTHexTextInputStream, 76
- OSRTHexTextInputStream.h, 173
- OSRTInputStream, 77
 - ~OSRTInputStream, 79
 - close, 79
 - currentPos, 80

- flush, 80
- getContext, 80
- getCtxtPtr, 81
- getErrorInfo, 81, 82
- getPosition, 82
- getStatus, 83
- isOpened, 84
- isA, 83
- mark, 84
- markSupported, 85
- OSRTInputStream, 79
- read, 85
- readBlocking, 86
- reset, 86
- setPosition, 86
- skip, 87
- OSRTInputStream.h, 173
- OSRTInputStreamIF.h, 173
- OSRTMemBuf, 88
- OSRTMemBuf.h, 174
- OSRTMemoryInputStream, 88
 - isA, 90
 - OSRTMemoryInputStream, 89
- OSRTMemoryInputStream.h, 174
- OSRTMemoryOutputStream, 90
 - getBuffer, 92
 - isA, 93
 - OSRTMemoryOutputStream, 91, 92
 - reset, 93
- OSRTMemoryOutputStream.h, 174
- OSRTMessageBuffer, 94
 - ~OSRTMessageBuffer, 95
 - getByteIndex, 96
 - getCtxtPtr, 96
 - getErrorInfo, 96, 97
 - getStatus, 97
 - init, 98
 - initBuffer, 98
 - OSRTMessageBuffer, 95
 - setDiag, 99
- OSRTMessageBufferIF, 99
 - ~OSRTMessageBufferIF, 100
 - getByteIndex, 101
 - getMsgCopy, 101
 - getMsgPtr, 101
 - init, 102
 - initBuffer, 102
 - isA, 103
 - setDiag, 103
- OSRTMsgBuf.h, 175
- OSRTMsgBufIF.h, 175
- OSRTObjListClass, 104
 - append, 105
 - appendCopy, 105
 - getHead, 105, 106
 - getItem, 106
 - getTail, 106, 107
 - insert, 107
 - operator=, 107
- OSRTObjListNodeClass, 108
 - getData, 109
 - getNext, 109, 110
 - getPrev, 110
- OSRTOutputStream, 111
 - ~OSRTOutputStream, 112
 - close, 113
 - flush, 113
 - getContext, 113
 - getCtxtPtr, 114
 - getErrorInfo, 114
 - getStatus, 115
 - isOpened, 116
 - isA, 115
 - OSRTOutputStream, 112
 - write, 116, 117
- OSRTOutputStream.h, 176
- OSRTOutputStreamIF.h, 176
- OSRTSocket, 117
 - ~OSRTSocket, 120
 - accept, 120
 - addrToString, 121
 - bind, 121, 122
 - bindUrl, 123
 - blockingRead, 124
 - close, 124
 - connect, 124
 - connectTimed, 125
 - connectUrl, 126
 - getOwnership, 126
 - getSocket, 126
 - getStatus, 127
 - listen, 127
 - OSRTSocket, 119, 120
 - recv, 128
 - send, 128
 - setOwnership, 129
 - setRetryCount, 129
 - stringToAddr, 130
- OSRTSocket.h, 176
- OSRTSocketInputStream, 130
 - isA, 133
 - OSRTSocketInputStream, 131, 132
- OSRTSocketInputStream.h, 177
- OSRTSocketOutputStream, 134
 - isA, 137
 - OSRTSocketOutputStream, 135, 137
- OSRTSocketOutputStream.h, 177
- OSRTStream, 138

- ~OSRTStream, 140
- close, 140
- flush, 140
- getContext, 141
- getCtxtPtr, 141
- getErrorInfo, 142
- getStatus, 143
- isOpened, 143
- OSRTStream, 140
- OSRTStream.h, 178
- OSRTStreamIF.h, 178
- OSRTString, 144
 - OSRTString, 145, 147
 - print, 147
 - setValue, 147, 148
 - setValuePtr, 148
 - toInt, 148
 - toSize, 149
 - toUInt, 149
 - toUInt64, 149
- OSRTString.h, 178
- OSRTStringConst.h, 179
- OSRTStringIF.h, 179
- OSRTStringIF, 150
 - OSRTStringIF, 151
 - print, 152
 - setValue, 152
- OSRTUTF8String, 153
 - clone, 155
 - copyValue, 155
 - OSRTUTF8String, 154, 155
 - print, 156
 - setValue, 156
- OSRTUTF8String.h, 180
- OSRTVoidPtrList.h, 180
- OSRTXMLString, 156
 - appendValue, 160
 - clone, 161
 - compare, 161
 - copyValue, 161, 162
 - decodeXML, 162
 - encodeXML, 162
 - isCDATA, 163
 - OSRTXMLString, 158–160
 - print, 163
 - setCDATA, 164
 - setValue, 164, 165
- OSRTXMLStringList, 165
 - append, 167
 - appendCopy, 167
 - clone, 167
 - OSRTXMLStringList, 166
 - operator=, 168
- operator=
 - OSRTCtxtPtr, 53
 - OSRTObjListClass, 107
 - OSRTXMLStringList, 168
- print
 - OSAnyElementClass, 25
 - OSRTFastString, 67
 - OSRTString, 147
 - OSRTStringIF, 152
 - OSRTUTF8String, 156
 - OSRTXMLString, 163
- read
 - OSRTInputStream, 85
- readBlocking
 - OSRTInputStream, 86
- recv
 - OSRTSocket, 128
- remove
 - OSRTDListBaseClass, 56
- reset
 - OSRTInputStream, 86
 - OSRTMemoryOutputStream, 93
- rtxCppAnyAttr.h, 181
- rtxCppAnyElement.h, 181
- rtxCppBitString.h, 181
 - rtxCppCheckBitBounds, 182
 - rtxCppCheckBitBounds64, 182
- rtxCppBufferedInputStream.h, 183
- rtxCppCheckBitBounds
 - rtxCppBitString.h, 182
- rtxCppCheckBitBounds64
 - rtxCppBitString.h, 182
- rtxCppDList.h, 184
- rtxCppDate Time.h, 183
- rtxCppDynOctStr.h, 184
- rtxCppTypes.h, 185
- rtxCppXmlSTLString.h, 185
- rtxCppXmlSTLStringList.h, 185
- rtxCppXmlString.h, 185
- rtxCppXmlStringList.h, 186
- send
 - OSRTSocket, 128
- setCDATA
 - OSRTXMLString, 164
- setDiag
 - OSRTContext, 42
 - OSRTMessageBuffer, 99
 - OSRTMessageBufferIF, 103
- setOwnership
 - OSRTSocket, 129
- setPosition
 - OSRTInputStream, 86
- setRetryCount

- OSRSocket, [129](#)
- setRunTimeKey
 - OSRTContext, [42](#)
- setStatus
 - OSRTContext, [43](#)
- setValue
 - OSAnyAttrClass, [20](#), [21](#)
 - OSAnyElementClass, [25](#)
 - OSDynOctStrClass, [31](#), [32](#)
 - OSRTFastString, [68](#)
 - OSRTString, [147](#), [148](#)
 - OSRTStringIF, [152](#)
 - OSRTUTF8String, [156](#)
 - OSRTXMLString, [164](#), [165](#)
- setValueFromBase64
 - OSDynOctStrClass, [32](#)
- setValuePtr
 - OSRTString, [148](#)
- skip
 - OSRTInputStream, [87](#)
- stringToAddr
 - OSRSocket, [130](#)

- TCP/IP or UDP Socket Classes, [14](#)
- toInt
 - OSRTString, [148](#)
- toSize
 - OSRTString, [149](#)
- toUInt
 - OSRTString, [149](#)
- toUInt64
 - OSRTString, [149](#)

- write
 - OSRTOutputStream, [116](#), [117](#)