

ASN1C

ASN.1 Compiler
Version 7.2
C++ Runtime
Reference Manual

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

Copyright Notice

Copyright ©1997–2018 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 ASN1C may be submitted via electronic mail to info@obj-sys.com.

Contents

1	Main Page	1
2	Module Documentation	3
2.1	C++ Run-Time Classes	3
2.1.1	Detailed Description	3
2.2	OSRT Message Buffer Classes	4
2.2.1	Detailed Description	4
2.3	Control (ASN1C_) Base Classes	5
2.3.1	Detailed Description	5
2.4	ASN.1 Type (ASN1T_) Base Classes	6
2.4.1	Detailed Description	7
2.4.2	Function Documentation	7
2.4.2.1	operator"!=() [1/8]	7
2.4.2.2	operator"!=() [2/8]	8
2.4.2.3	operator"!=() [3/8]	8
2.4.2.4	operator"!=() [4/8]	9
2.4.2.5	operator"!=() [5/8]	9
2.4.2.6	operator"!=() [6/8]	10
2.4.2.7	operator"!=() [7/8]	10
2.4.2.8	operator"!=() [8/8]	10
2.4.2.9	operator+()	11

2.4.2.10	operator<() [1/8]	11
2.4.2.11	operator<() [2/8]	12
2.4.2.12	operator<() [3/8]	12
2.4.2.13	operator<() [4/8]	12
2.4.2.14	operator<() [5/8]	13
2.4.2.15	operator<() [6/8]	13
2.4.2.16	operator<() [7/8]	14
2.4.2.17	operator<() [8/8]	14
2.4.2.18	operator<=() [1/8]	15
2.4.2.19	operator<=() [2/8]	15
2.4.2.20	operator<=() [3/8]	16
2.4.2.21	operator<=() [4/8]	16
2.4.2.22	operator<=() [5/8]	16
2.4.2.23	operator<=() [6/8]	17
2.4.2.24	operator<=() [7/8]	17
2.4.2.25	operator<=() [8/8]	18
2.4.2.26	operator==() [1/6]	18
2.4.2.27	operator==() [2/6]	19
2.4.2.28	operator==() [3/6]	19
2.4.2.29	operator==() [4/6]	19
2.4.2.30	operator==() [5/6]	20
2.4.2.31	operator==() [6/6]	20
2.4.2.32	operator>() [1/8]	21
2.4.2.33	operator>() [2/8]	21
2.4.2.34	operator>() [3/8]	22
2.4.2.35	operator>() [4/8]	22
2.4.2.36	operator>() [5/8]	22
2.4.2.37	operator>() [6/8]	23

2.4.2.38	operator>() [7/8]	23
2.4.2.39	operator>() [8/8]	24
2.4.2.40	operator>=() [1/8]	24
2.4.2.41	operator>=() [2/8]	25
2.4.2.42	operator>=() [3/8]	25
2.4.2.43	operator>=() [4/8]	25
2.4.2.44	operator>=() [5/8]	26
2.4.2.45	operator>=() [6/8]	26
2.4.2.46	operator>=() [7/8]	27
2.4.2.47	operator>=() [8/8]	27
2.5	Context Management Classes	28
2.5.1	Detailed Description	28
2.6	Date and Time Runtime Classes	29
2.6.1	Detailed Description	29
2.6.2	Macro Definition Documentation	29
2.6.2.1	LOG_TMERR	29
2.7	ASN.1 Stream Classes	30
2.7.1	Detailed Description	30
2.8	Generic Input Stream Classes	31
2.8.1	Detailed Description	31
2.9	Generic Output Stream Classes	32
2.9.1	Detailed Description	32
2.10	TCP/IP or UDP Socket Classes	33
2.10.1	Detailed Description	33
2.11	Named Event Handlers	34
2.11.1	Detailed Description	34
2.11.2	Macro Definition Documentation	34
2.11.2.1	OS_UNUSED_ARG	34

2.11.3	Variable Documentation	34
2.11.3.1	ASN1MessageBuffer	35
2.12	Run-time error status codes.	36
2.12.1	Detailed Description	36
2.12.2	Macro Definition Documentation	36
2.12.2.1	ASN_E_BAD_ALIGN	36
2.12.2.2	ASN_E_BADTAG	37
2.12.2.3	ASN_E_BASE	37
2.12.2.4	ASN_E_CONCMODF	37
2.12.2.5	ASN_E_ILLSTATE	37
2.12.2.6	ASN_E_INVBINS	37
2.12.2.7	ASN_E_INVINDEX	37
2.12.2.8	ASN_E_INVLEN	38
2.12.2.9	ASN_E_INVOBJID	38
2.12.2.10	ASN_E_INVPERENC	38
2.12.2.11	ASN_E_INVTCVAL	38
2.12.2.12	ASN_E_NOTCANON	38
2.12.2.13	ASN_E_NOTINSEQ	38
2.12.2.14	ASN_E_NOTPDU	39
2.12.2.15	ASN_E_UNDEFTYP	39
2.12.2.16	ASN_E_UNKNOWNPDU	39
2.12.2.17	ASN_OK_FRAG	39

3	Class Documentation	41
3.1	ASN1CBitStr Class Reference	41
3.1.1	Detailed Description	43
3.1.2	Constructor & Destructor Documentation	43
3.1.2.1	ASN1CBitStr() [1/2]	43
3.1.2.2	ASN1CBitStr() [2/2]	43
3.1.3	Member Function Documentation	44
3.1.3.1	cardinality()	44
3.1.3.2	change()	44
3.1.3.3	clear() [1/3]	45
3.1.3.4	clear() [2/3]	45
3.1.3.5	clear() [3/3]	46
3.1.3.6	doAnd() [1/3]	46
3.1.3.7	doAnd() [2/3]	47
3.1.3.8	doAnd() [3/3]	47
3.1.3.9	doAndNot() [1/3]	48
3.1.3.10	doAndNot() [2/3]	48
3.1.3.11	doAndNot() [3/3]	49
3.1.3.12	doOr() [1/3]	49
3.1.3.13	doOr() [2/3]	50
3.1.3.14	doOr() [3/3]	50
3.1.3.15	doXor() [1/3]	50
3.1.3.16	doXor() [2/3]	52
3.1.3.17	doXor() [3/3]	52
3.1.3.18	get() [1/2]	53
3.1.3.19	get() [2/2]	53
3.1.3.20	getBytes()	54
3.1.3.21	invert() [1/2]	54

3.1.3.22	invert() [2/2]	55
3.1.3.23	isEmpty()	55
3.1.3.24	isSet()	56
3.1.3.25	length()	56
3.1.3.26	operator ASN1TDynBitStr()	56
3.1.3.27	operator ASN1TDynBitStr *()	57
3.1.3.28	set() [1/2]	57
3.1.3.29	set() [2/2]	58
3.1.3.30	shiftLeft()	58
3.1.3.31	shiftRight()	59
3.1.3.32	size()	59
3.1.3.33	unusedBitsInLastUnit()	60
3.2	ASN1CBitStrSizeHolder Class Reference	60
3.2.1	Detailed Description	60
3.3	ASN1CBitStrSizeHolder16 Class Reference	61
3.3.1	Detailed Description	61
3.4	ASN1CBitStrSizeHolder32 Class Reference	61
3.4.1	Detailed Description	62
3.5	ASN1CBitStrSizeHolder8 Class Reference	62
3.5.1	Detailed Description	63
3.6	ASN1CGeneralizedTime Class Reference	63
3.6.1	Detailed Description	64
3.6.2	Constructor & Destructor Documentation	64
3.6.2.1	ASN1CGeneralizedTime() [1/5]	64
3.6.2.2	ASN1CGeneralizedTime() [2/5]	64
3.6.2.3	ASN1CGeneralizedTime() [3/5]	65
3.6.2.4	ASN1CGeneralizedTime() [4/5]	65
3.6.2.5	ASN1CGeneralizedTime() [5/5]	66

3.6.3	Member Function Documentation	66
3.6.3.1	compileString()	66
3.6.3.2	getCentury()	66
3.6.3.3	setCentury()	67
3.6.3.4	setTime()	67
3.7	ASN1Context Class Reference	68
3.7.1	Detailed Description	68
3.7.2	Constructor & Destructor Documentation	68
3.7.2.1	ASN1Context()	69
3.7.3	Member Function Documentation	69
3.7.3.1	setRunTimeKey()	69
3.8	ASN1CSeqOfList Class Reference	69
3.8.1	Detailed Description	71
3.8.2	Constructor & Destructor Documentation	71
3.8.2.1	ASN1CSeqOfList() [1/6]	71
3.8.2.2	ASN1CSeqOfList() [2/6]	71
3.8.2.3	ASN1CSeqOfList() [3/6]	72
3.8.2.4	ASN1CSeqOfList() [4/6]	72
3.8.2.5	ASN1CSeqOfList() [5/6]	72
3.8.2.6	ASN1CSeqOfList() [6/6]	74
3.8.3	Member Function Documentation	74
3.8.3.1	append()	74
3.8.3.2	appendArray()	75
3.8.3.3	appendArrayCopy()	75
3.8.3.4	clear()	76
3.8.3.5	contains()	76
3.8.3.6	freeMemory()	76
3.8.3.7	get()	76

3.8.3.8	getFirst()	77
3.8.3.9	getLast()	77
3.8.3.10	indexOf()	77
3.8.3.11	init()	78
3.8.3.12	insert()	78
3.8.3.13	isEmpty()	78
3.8.3.14	iterator()	79
3.8.3.15	iteratorFrom()	79
3.8.3.16	iteratorFromLast()	79
3.8.3.17	operator[]()	80
3.8.3.18	remove() [1/2]	80
3.8.3.19	remove() [2/2]	80
3.8.3.20	removeFirst()	81
3.8.3.21	removeLast()	81
3.8.3.22	set()	81
3.8.3.23	size()	82
3.8.3.24	toArray() [1/2]	82
3.8.3.25	toArray() [2/2]	83
3.9	ASN1CSeqOfListIterator Class Reference	83
3.9.1	Detailed Description	84
3.9.2	Member Function Documentation	84
3.9.2.1	hasNext()	84
3.9.2.2	hasPrev()	85
3.9.2.3	insert()	85
3.9.2.4	next()	85
3.9.2.5	prev()	86
3.9.2.6	remove()	86
3.9.2.7	set()	87

3.10 ASN1CTime Class Reference	87
3.10.1 Detailed Description	89
3.10.2 Constructor & Destructor Documentation	89
3.10.2.1 ASN1CTime() [1/5]	89
3.10.2.2 ASN1CTime() [2/5]	90
3.10.2.3 ASN1CTime() [3/5]	90
3.10.2.4 ASN1CTime() [4/5]	91
3.10.2.5 ASN1CTime() [5/5]	91
3.10.2.6 ~ASN1CTime()	91
3.10.3 Member Function Documentation	91
3.10.3.1 clear()	92
3.10.3.2 compileString()	92
3.10.3.3 equals()	92
3.10.3.4 getDay()	92
3.10.3.5 getDiff()	93
3.10.3.6 getDiffHour()	93
3.10.3.7 getDiffMinute()	94
3.10.3.8 getFraction()	94
3.10.3.9 getFractionAsDouble()	95
3.10.3.10 getFractionLen()	95
3.10.3.11 getFractionStr()	95
3.10.3.12 getHour()	95
3.10.3.13 getMinute()	96
3.10.3.14 getMonth()	96
3.10.3.15 getSecond()	97
3.10.3.16 getTime()	97
3.10.3.17 getTimeString()	98
3.10.3.18 getTimeStringLen()	98

3.10.3.19	getUTC()	98
3.10.3.20	getYear()	99
3.10.3.21	operator"!=(())	99
3.10.3.22	operator<()	99
3.10.3.23	operator<=()	100
3.10.3.24	operator=()	100
3.10.3.25	operator==(())	100
3.10.3.26	operator>()	101
3.10.3.27	operator>=()	101
3.10.3.28	parseString()	101
3.10.3.29	setDay()	102
3.10.3.30	setDER()	102
3.10.3.31	setDiff() [1/2]	102
3.10.3.32	setDiff() [2/2]	103
3.10.3.33	setDiffHour()	103
3.10.3.34	setFraction() [1/3]	104
3.10.3.35	setFraction() [2/3]	104
3.10.3.36	setFraction() [3/3]	105
3.10.3.37	setHour()	105
3.10.3.38	setMinute()	106
3.10.3.39	setMonth()	106
3.10.3.40	setSecond()	107
3.10.3.41	setTime()	107
3.10.3.42	setUTC()	108
3.10.3.43	setYear()	108
3.11	ASN1CType Class Reference	109
3.11.1	Detailed Description	110
3.11.2	Constructor & Destructor Documentation	110

3.11.2.1	ASN1CType() [1/4]	110
3.11.2.2	ASN1CType() [2/4]	110
3.11.2.3	ASN1CType() [3/4]	111
3.11.2.4	ASN1CType() [4/4]	111
3.11.2.5	~ASN1CType()	111
3.11.3	Member Function Documentation	111
3.11.3.1	append()	111
3.11.3.2	Decode()	112
3.11.3.3	DecodeFrom()	112
3.11.3.4	Encode()	113
3.11.3.5	EncodeTo()	113
3.11.3.6	getContext()	114
3.11.3.7	getCtxtPtr()	114
3.11.3.8	getErrorText()	114
3.11.3.9	getStatus()	115
3.11.3.10	memAlloc()	115
3.11.3.11	memAllocZ()	115
3.11.3.12	memFreeAll()	116
3.11.3.13	memFreePtr()	116
3.11.3.14	memRealloc()	116
3.11.3.15	memReset()	117
3.11.3.16	printErrorInfo()	117
3.11.3.17	resetError()	117
3.11.3.18	setDiag()	117
3.11.3.19	setRunTimeKey()	118
3.11.4	Member Data Documentation	118
3.11.4.1	mpContext	118
3.11.4.2	mpMsgBuf	119

3.12 ASN1CUTCTime Class Reference	119
3.12.1 Detailed Description	120
3.12.2 Constructor & Destructor Documentation	120
3.12.2.1 ASN1CUTCTime() [1/2]	120
3.12.2.2 ASN1CUTCTime() [2/2]	120
3.12.3 Member Function Documentation	121
3.12.3.1 compileString()	121
3.12.3.2 getFraction()	121
3.12.3.3 setTime()	122
3.13 Asn1ErrorHandler Class Reference	122
3.13.1 Detailed Description	123
3.13.2 Member Function Documentation	123
3.13.2.1 error()	123
3.13.2.2 setErrorHandler()	123
3.14 ASN1MessageBuffer Class Reference	124
3.14.1 Detailed Description	125
3.14.2 Constructor & Destructor Documentation	125
3.14.2.1 ASN1MessageBuffer() [1/2]	125
3.14.2.2 ASN1MessageBuffer() [2/2]	125
3.14.2.3 ~ASN1MessageBuffer()	126
3.14.3 Member Function Documentation	126
3.14.3.1 addEventHandler()	126
3.14.3.2 CStringToBMPString()	126
3.14.3.3 getAppInfo()	127
3.14.3.4 getBitOffset()	127
3.14.3.5 getMsgLen()	127
3.14.3.6 initBuffer() [1/2]	127
3.14.3.7 initBuffer() [2/2]	128

3.14.3.8	isA()	128
3.14.3.9	removeEventHandler()	129
3.14.3.10	resetErrorInfo()	129
3.14.3.11	setAppInfo()	129
3.14.3.12	setErrorHandler()	129
3.14.3.13	setRunTimeKey()	130
3.14.3.14	setStatus()	130
3.15	Asn1NamedEventHandler Class Reference	131
3.15.1	Detailed Description	132
3.15.2	Member Function Documentation	132
3.15.2.1	addEventHandler()	132
3.15.2.2	bitStrValue()	132
3.15.2.3	boolValue()	133
3.15.2.4	charStrValue() [1/4]	133
3.15.2.5	charStrValue() [2/4]	134
3.15.2.6	charStrValue() [3/4]	134
3.15.2.7	charStrValue() [4/4]	135
3.15.2.8	endElement()	135
3.15.2.9	enumValue()	136
3.15.2.10	int64Value()	136
3.15.2.11	intValue()	137
3.15.2.12	invokeBitStrValue()	137
3.15.2.13	invokeBoolValue()	137
3.15.2.14	invokeCharStrValue() [1/4]	138
3.15.2.15	invokeCharStrValue() [2/4]	138
3.15.2.16	invokeCharStrValue() [3/4]	138
3.15.2.17	invokeCharStrValue() [4/4]	139
3.15.2.18	invokeEndElement()	139

3.15.2.19	invokeEnumValue()	140
3.15.2.20	invokeInt64Value()	140
3.15.2.21	invokeIntValue()	140
3.15.2.22	invokeNullValue()	141
3.15.2.23	invokeOctStrValue()	141
3.15.2.24	invokeOidValue()	141
3.15.2.25	invokeOpenTypeValue()	142
3.15.2.26	invokeRealValue()	142
3.15.2.27	invokeStartElement()	142
3.15.2.28	invokeUInt64Value()	143
3.15.2.29	invokeUIntValue()	143
3.15.2.30	nullValue()	143
3.15.2.31	octStrValue()	144
3.15.2.32	oidValue()	144
3.15.2.33	openTypeValue()	145
3.15.2.34	realValue()	145
3.15.2.35	removeEventHandler()	146
3.15.2.36	startElement()	146
3.15.2.37	uInt64Value()	146
3.15.2.38	uIntValue()	147
3.16	Asn1NullEventHandler Class Reference	147
3.16.1	Detailed Description	148
3.16.2	Member Function Documentation	148
3.16.2.1	endElement()	148
3.16.2.2	startElement()	148
3.17	ASN1TBitStr32 Struct Reference	149
3.17.1	Detailed Description	149
3.17.2	Constructor & Destructor Documentation	149

3.17.2.1	ASN1TBitStr32() [1/3]	149
3.17.2.2	ASN1TBitStr32() [2/3]	149
3.17.2.3	ASN1TBitStr32() [3/3]	150
3.18	ASN1TBMPString Struct Reference	150
3.18.1	Detailed Description	150
3.18.2	Constructor & Destructor Documentation	151
3.18.2.1	ASN1TBMPString()	151
3.19	ASN1TDynBitStr Struct Reference	151
3.19.1	Detailed Description	151
3.19.2	Constructor & Destructor Documentation	151
3.19.2.1	ASN1TDynBitStr() [1/3]	152
3.19.2.2	ASN1TDynBitStr() [2/3]	152
3.19.2.3	ASN1TDynBitStr() [3/3]	153
3.20	ASN1TDynBitStr64 Struct Reference	153
3.20.1	Detailed Description	153
3.20.2	Constructor & Destructor Documentation	154
3.20.2.1	ASN1TDynBitStr64() [1/3]	154
3.20.2.2	ASN1TDynBitStr64() [2/3]	154
3.20.2.3	ASN1TDynBitStr64() [3/3]	154
3.21	ASN1TDynOctStr Struct Reference	154
3.21.1	Detailed Description	155
3.21.2	Constructor & Destructor Documentation	155
3.21.2.1	ASN1TDynOctStr() [1/4]	155
3.21.2.2	ASN1TDynOctStr() [2/4]	155
3.21.2.3	ASN1TDynOctStr() [3/4]	156
3.21.2.4	ASN1TDynOctStr() [4/4]	156
3.21.3	Member Function Documentation	156
3.21.3.1	nCompare()	156

3.21.3.2	operator=() [1/2]	157
3.21.3.3	operator=() [2/2]	157
3.21.3.4	toHexString()	157
3.21.3.5	toString()	159
3.22	ASN1TGeneralizedTime Class Reference	159
3.22.1	Detailed Description	160
3.22.2	Constructor & Destructor Documentation	160
3.22.2.1	ASN1TGeneralizedTime() [1/4]	160
3.22.2.2	ASN1TGeneralizedTime() [2/4]	160
3.22.2.3	ASN1TGeneralizedTime() [3/4]	160
3.22.2.4	ASN1TGeneralizedTime() [4/4]	161
3.22.3	Member Function Documentation	161
3.22.3.1	compileString()	161
3.22.3.2	getCentury()	161
3.22.3.3	parseString()	162
3.22.3.4	setCentury()	162
3.22.3.5	setTime()	163
3.23	Asn1TObject Struct Reference	163
3.23.1	Detailed Description	164
3.23.2	Constructor & Destructor Documentation	164
3.23.2.1	Asn1TObject()	164
3.24	ASN1TObjId Struct Reference	164
3.24.1	Detailed Description	165
3.24.2	Constructor & Destructor Documentation	165
3.24.2.1	ASN1TObjId() [1/5]	165
3.24.2.2	~ASN1TObjId()	165
3.24.2.3	ASN1TObjId() [2/5]	165
3.24.2.4	ASN1TObjId() [3/5]	165

3.24.2.5	ASN1ObjId() [4/5]	166
3.24.2.6	ASN1ObjId() [5/5]	166
3.24.3	Member Function Documentation	166
3.24.3.1	nCompare()	166
3.24.3.2	operator+=() [1/3]	167
3.24.3.3	operator+=() [2/3]	167
3.24.3.4	operator+=() [3/3]	168
3.24.3.5	operator=() [1/3]	168
3.24.3.6	operator=() [2/3]	168
3.24.3.7	operator=() [3/3]	169
3.24.3.8	RnCompare()	169
3.24.3.9	set_data()	169
3.24.3.10	toString()	170
3.24.3.11	trim()	170
3.25	ASN1OpenType Struct Reference	170
3.25.1	Detailed Description	171
3.25.2	Constructor & Destructor Documentation	171
3.25.2.1	ASN1OpenType()	171
3.26	ASN1TPDU Struct Reference	171
3.26.1	Detailed Description	172
3.26.2	Constructor & Destructor Documentation	172
3.26.2.1	~ASN1TPDU()	172
3.26.3	Member Function Documentation	172
3.26.3.1	setContext()	172
3.26.4	Member Data Documentation	172
3.26.4.1	mpContext	173
3.27	ASN1TPDUSeqOfList Struct Reference	173
3.27.1	Detailed Description	173

3.27.2	Constructor & Destructor Documentation	173
3.27.2.1	ASN1TPDUSeqOfList()	174
3.28	ASN1TSeqExt Struct Reference	174
3.28.1	Detailed Description	174
3.28.2	Constructor & Destructor Documentation	174
3.28.2.1	ASN1TSeqExt()	174
3.29	ASN1TSeqOfList Struct Reference	175
3.29.1	Detailed Description	175
3.29.2	Constructor & Destructor Documentation	175
3.29.2.1	ASN1TSeqOfList()	175
3.30	ASN1TTime Class Reference	176
3.30.1	Detailed Description	178
3.30.2	Constructor & Destructor Documentation	178
3.30.2.1	ASN1TTime() [1/3]	178
3.30.2.2	ASN1TTime() [2/3]	178
3.30.2.3	ASN1TTime() [3/3]	178
3.30.2.4	~ASN1TTime()	179
3.30.3	Member Function Documentation	179
3.30.3.1	clear()	179
3.30.3.2	compileString()	179
3.30.3.3	equals()	180
3.30.3.4	getDay()	180
3.30.3.5	getDiff()	180
3.30.3.6	getDiffHour()	181
3.30.3.7	getDiffMinute()	181
3.30.3.8	getFraction()	181
3.30.3.9	getFractionAsDouble()	182
3.30.3.10	getFractionLen()	182

3.30.3.11	getFractionStr()	182
3.30.3.12	getHour()	183
3.30.3.13	getMinute()	183
3.30.3.14	getMonth()	183
3.30.3.15	getSecond()	184
3.30.3.16	getTime()	184
3.30.3.17	getUTC()	184
3.30.3.18	getYear()	185
3.30.3.19	parseString()	185
3.30.3.20	setDay()	185
3.30.3.21	setDER()	186
3.30.3.22	setDiff() [1/2]	186
3.30.3.23	setDiff() [2/2]	187
3.30.3.24	setDiffHour()	187
3.30.3.25	setFraction() [1/3]	188
3.30.3.26	setFraction() [2/3]	188
3.30.3.27	setFraction() [3/3]	189
3.30.3.28	setHour()	189
3.30.3.29	setMinute()	190
3.30.3.30	setMonth()	190
3.30.3.31	setSecond()	191
3.30.3.32	setTime()	191
3.30.3.33	setUTC()	192
3.30.3.34	setYear()	193
3.30.3.35	toString() [1/3]	193
3.30.3.36	toString() [2/3]	194
3.30.3.37	toString() [3/3]	194
3.30.4	Member Data Documentation	194

3.30.4.1	mbDerRules	194
3.30.4.2	mbUtcFlag	194
3.30.4.3	mDay	195
3.30.4.4	mDiffHour	195
3.30.4.5	mDiffMin	195
3.30.4.6	mHour	195
3.30.4.7	mMinute	195
3.30.4.8	mMonth	195
3.30.4.9	mSecFracLen	195
3.30.4.10	mSecFraction	196
3.30.4.11	mSecond	196
3.30.4.12	mStatus	196
3.30.4.13	mYear	196
3.31	ASN1TUniversalString Struct Reference	196
3.31.1	Detailed Description	197
3.31.2	Constructor & Destructor Documentation	197
3.31.2.1	ASN1TUniversalString()	197
3.32	ASN1TUTCTime Class Reference	197
3.32.1	Detailed Description	198
3.32.2	Constructor & Destructor Documentation	198
3.32.2.1	ASN1TUTCTime() [1/4]	198
3.32.2.2	ASN1TUTCTime() [2/4]	198
3.32.2.3	ASN1TUTCTime() [3/4]	198
3.32.2.4	ASN1TUTCTime() [4/4]	199
3.32.3	Member Function Documentation	199
3.32.3.1	clear()	199
3.32.3.2	compileString()	199
3.32.3.3	getFraction()	200

3.32.3.4	parseString()	200
3.32.3.5	setFraction()	201
3.32.3.6	setTime()	201
3.32.3.7	setUTC()	202
3.32.3.8	setYear()	202
3.33	OSRTBaseType Class Reference	203
3.33.1	Detailed Description	203
3.34	OSRTContext Class Reference	203
3.34.1	Detailed Description	204
3.34.2	Constructor & Destructor Documentation	204
3.34.2.1	OSRTContext()	204
3.34.2.2	~OSRTContext()	205
3.34.3	Member Function Documentation	205
3.34.3.1	_ref()	205
3.34.3.2	_unref()	205
3.34.3.3	getErrorInfo() [1/3]	205
3.34.3.4	getErrorInfo() [2/3]	205
3.34.3.5	getErrorInfo() [3/3]	206
3.34.3.6	getPtr()	206
3.34.3.7	getRefCount()	207
3.34.3.8	getStatus()	207
3.34.3.9	isInitialized()	207
3.34.3.10	memAlloc()	207
3.34.3.11	memAllocZ()	208
3.34.3.12	memFreeAll()	208
3.34.3.13	memFreePtr()	208
3.34.3.14	memRealloc()	209
3.34.3.15	memReset()	209

3.34.3.16	printErrorInfo()	209
3.34.3.17	resetErrorInfo()	210
3.34.3.18	setDiag()	210
3.34.3.19	setRunTimeKey()	210
3.34.3.20	setStatus()	211
3.34.4	Member Data Documentation	211
3.34.4.1	mbInitialized	211
3.34.4.2	mCount	211
3.34.4.3	mCtxt	211
3.34.4.4	mStatus	212
3.35	OSRTCtxtPtr Class Reference	212
3.35.1	Detailed Description	212
3.35.2	Constructor & Destructor Documentation	212
3.35.2.1	OSRTCtxtPtr() [1/2]	212
3.35.2.2	OSRTCtxtPtr() [2/2]	213
3.35.2.3	~OSRTCtxtPtr()	213
3.35.3	Member Function Documentation	213
3.35.3.1	getCtxtPtr()	213
3.35.3.2	isNull()	214
3.35.3.3	operator OSRTCcontext *()	214
3.35.3.4	operator->()	214
3.35.3.5	operator=() [1/2]	214
3.35.3.6	operator=() [2/2]	214
3.35.3.7	operator==(())	215
3.35.4	Member Data Documentation	215
3.35.4.1	mPointer	215
3.36	OSRTFastString Class Reference	215
3.36.1	Detailed Description	216

3.36.2	Constructor & Destructor Documentation	216
3.36.2.1	OSRTFastString() [1/4]	216
3.36.2.2	OSRTFastString() [2/4]	216
3.36.2.3	OSRTFastString() [3/4]	217
3.36.2.4	OSRTFastString() [4/4]	217
3.36.2.5	~OSRTFastString()	217
3.36.3	Member Function Documentation	217
3.36.3.1	clone()	218
3.36.3.2	getUTF8Value()	218
3.36.3.3	getValue()	218
3.36.3.4	operator=()	218
3.36.3.5	print()	218
3.36.3.6	setValue() [1/2]	219
3.36.3.7	setValue() [2/2]	219
3.37	OSRTFileInputStream Class Reference	220
3.37.1	Detailed Description	220
3.37.2	Constructor & Destructor Documentation	220
3.37.2.1	OSRTFileInputStream() [1/4]	220
3.37.2.2	OSRTFileInputStream() [2/4]	221
3.37.2.3	OSRTFileInputStream() [3/4]	221
3.37.2.4	OSRTFileInputStream() [4/4]	222
3.37.3	Member Function Documentation	222
3.37.3.1	isA()	222
3.38	OSRTFileOutputStream Class Reference	223
3.38.1	Detailed Description	223
3.38.2	Constructor & Destructor Documentation	223
3.38.2.1	OSRTFileOutputStream() [1/4]	223
3.38.2.2	OSRTFileOutputStream() [2/4]	224

3.38.2.3	OSRTFileOutputStream() [3/4]	224
3.38.2.4	OSRTFileOutputStream() [4/4]	225
3.38.3	Member Function Documentation	225
3.38.3.1	isA()	225
3.39	OSRTHexTextInputStream Class Reference	226
3.39.1	Detailed Description	227
3.39.2	Constructor & Destructor Documentation	227
3.39.2.1	OSRTHexTextInputStream()	227
3.39.2.2	~OSRTHexTextInputStream()	227
3.39.3	Member Function Documentation	227
3.39.3.1	isA()	227
3.39.3.2	setOwnUnderStream()	228
3.40	OSRTInputStream Class Reference	228
3.40.1	Detailed Description	229
3.40.2	Constructor & Destructor Documentation	229
3.40.2.1	OSRTInputStream()	229
3.40.2.2	~OSRTInputStream()	230
3.40.3	Member Function Documentation	230
3.40.3.1	close()	230
3.40.3.2	currentPos()	230
3.40.3.3	flush()	231
3.40.3.4	getContext()	231
3.40.3.5	getCtxtPtr()	231
3.40.3.6	getErrorInfo() [1/2]	232
3.40.3.7	getErrorInfo() [2/2]	232
3.40.3.8	getPosition()	232
3.40.3.9	getStatus()	233
3.40.3.10	isA()	233

3.40.3.11	isOpened()	234
3.40.3.12	mark()	234
3.40.3.13	markSupported()	235
3.40.3.14	printErrorInfo()	235
3.40.3.15	read()	235
3.40.3.16	readBlocking()	236
3.40.3.17	reset()	236
3.40.3.18	resetErrorInfo()	237
3.40.3.19	setPosition()	237
3.40.3.20	skip()	237
3.41	OSRTInputStreamIF Class Reference	238
3.41.1	Constructor & Destructor Documentation	239
3.41.1.1	~OSRTInputStreamIF()	239
3.41.2	Member Function Documentation	239
3.41.2.1	currentPos()	239
3.41.2.2	getPosition()	239
3.41.2.3	isA()	240
3.41.2.4	mark()	240
3.41.2.5	markSupported()	241
3.41.2.6	read()	241
3.41.2.7	readBlocking()	242
3.41.2.8	reset()	242
3.41.2.9	setPosition()	243
3.41.2.10	skip()	243
3.42	OSRTInputStreamPtr Class Reference	244
3.43	OSRTMemoryInputStream Class Reference	244
3.43.1	Detailed Description	245
3.43.2	Constructor & Destructor Documentation	245

3.43.2.1	OSRTMemoryInputStream() [1/2]	245
3.43.2.2	OSRTMemoryInputStream() [2/2]	245
3.43.3	Member Function Documentation	246
3.43.3.1	isA()	246
3.44	OSRTMemoryOutputStream Class Reference	246
3.44.1	Detailed Description	247
3.44.2	Constructor & Destructor Documentation	247
3.44.2.1	OSRTMemoryOutputStream() [1/3]	247
3.44.2.2	OSRTMemoryOutputStream() [2/3]	247
3.44.2.3	OSRTMemoryOutputStream() [3/3]	248
3.44.3	Member Function Documentation	248
3.44.3.1	getBuffer()	248
3.44.3.2	isA()	249
3.44.3.3	reset()	249
3.45	OSRTMessageBuffer Class Reference	250
3.45.1	Detailed Description	251
3.45.2	Constructor & Destructor Documentation	251
3.45.2.1	OSRTMessageBuffer()	251
3.45.2.2	~OSRTMessageBuffer()	251
3.45.3	Member Function Documentation	251
3.45.3.1	getAppInfo()	251
3.45.3.2	getByteIndex()	252
3.45.3.3	getContext()	252
3.45.3.4	getCtxtPtr()	252
3.45.3.5	getErrorInfo() [1/2]	252
3.45.3.6	getErrorInfo() [2/2]	252
3.45.3.7	getMsgCopy()	253
3.45.3.8	getMsgPtr()	253

3.45.3.9	getStatus()	253
3.45.3.10	init()	254
3.45.3.11	initBuffer()	254
3.45.3.12	printErrorInfo()	255
3.45.3.13	resetErrorInfo()	255
3.45.3.14	setAppInfo()	255
3.45.3.15	setDiag()	255
3.45.4	Member Data Documentation	256
3.45.4.1	mBufferType	256
3.46	OSRTMessageBufferIF Class Reference	256
3.46.1	Detailed Description	257
3.46.2	Constructor & Destructor Documentation	257
3.46.2.1	~OSRTMessageBufferIF()	257
3.46.3	Member Function Documentation	257
3.46.3.1	getAppInfo()	257
3.46.3.2	getByteIndex()	258
3.46.3.3	getMsgCopy()	258
3.46.3.4	getMsgPtr()	258
3.46.3.5	init()	259
3.46.3.6	initBuffer()	259
3.46.3.7	isA()	259
3.46.3.8	setAppInfo()	260
3.46.3.9	setDiag()	260
3.46.3.10	setNamespace()	260
3.47	OSRTOutputStream Class Reference	261
3.47.1	Detailed Description	262
3.47.2	Constructor & Destructor Documentation	262
3.47.2.1	OSRTOutputStream()	262

3.47.2.2	~OSRTOutputStream()	262
3.47.3	Member Function Documentation	262
3.47.3.1	close()	262
3.47.3.2	flush()	263
3.47.3.3	getContext()	263
3.47.3.4	getCtxtPtr()	263
3.47.3.5	getErrorInfo() [1/2]	264
3.47.3.6	getErrorInfo() [2/2]	264
3.47.3.7	getStatus()	264
3.47.3.8	isA()	265
3.47.3.9	isOpened()	265
3.47.3.10	printErrorInfo()	266
3.47.3.11	resetErrorInfo()	266
3.47.3.12	write() [1/2]	266
3.47.3.13	write() [2/2]	267
3.48	OSRTOutputStreamIF Class Reference	267
3.48.1	Constructor & Destructor Documentation	268
3.48.1.1	~OSRTOutputStreamIF()	268
3.48.2	Member Function Documentation	268
3.48.2.1	isA()	268
3.48.2.2	write()	268
3.49	OSRTOutputStreamPtr Class Reference	269
3.50	OSRTSocket Class Reference	269
3.50.1	Detailed Description	270
3.50.2	Constructor & Destructor Documentation	271
3.50.2.1	OSRTSocket() [1/3]	271
3.50.2.2	OSRTSocket() [2/3]	271
3.50.2.3	OSRTSocket() [3/3]	271

3.50.2.4	~OSRSocket()	271
3.50.3	Member Function Documentation	272
3.50.3.1	accept()	272
3.50.3.2	addrToString()	272
3.50.3.3	bind() [1/3]	273
3.50.3.4	bind() [2/3]	273
3.50.3.5	bind() [3/3]	274
3.50.3.6	bindUrl()	274
3.50.3.7	blockingRead()	275
3.50.3.8	close()	275
3.50.3.9	connect()	276
3.50.3.10	connectTimed()	276
3.50.3.11	connectUrl()	277
3.50.3.12	getOwnership()	277
3.50.3.13	getSocket()	278
3.50.3.14	getStatus()	278
3.50.3.15	listen()	278
3.50.3.16	recv()	279
3.50.3.17	send()	279
3.50.3.18	setOwnership()	280
3.50.3.19	setRetryCount()	280
3.50.3.20	stringToAddr()	280
3.51	OSRSocketInputStream Class Reference	281
3.51.1	Detailed Description	282
3.51.2	Constructor & Destructor Documentation	282
3.51.2.1	OSRSocketInputStream() [1/4]	282
3.51.2.2	OSRSocketInputStream() [2/4]	282
3.51.2.3	OSRSocketInputStream() [3/4]	283

3.51.2.4	OSRSocketInputStream() [4/4]	283
3.51.3	Member Function Documentation	283
3.51.3.1	isA()	284
3.52	OSRSocketOutputStream Class Reference	284
3.52.1	Detailed Description	285
3.52.2	Constructor & Destructor Documentation	285
3.52.2.1	OSRSocketOutputStream() [1/4]	285
3.52.2.2	OSRSocketOutputStream() [2/4]	285
3.52.2.3	OSRSocketOutputStream() [3/4]	286
3.52.2.4	OSRSocketOutputStream() [4/4]	286
3.52.3	Member Function Documentation	287
3.52.3.1	isA()	287
3.53	OSRStream Class Reference	287
3.53.1	Detailed Description	288
3.53.2	Constructor & Destructor Documentation	288
3.53.2.1	OSRStream()	288
3.53.2.2	~OSRStream()	288
3.53.3	Member Function Documentation	288
3.53.3.1	close()	289
3.53.3.2	flush()	289
3.53.3.3	getContext()	290
3.53.3.4	getCtxtPtr()	290
3.53.3.5	getErrorInfo() [1/2]	290
3.53.3.6	getErrorInfo() [2/2]	291
3.53.3.7	getStatus()	291
3.53.3.8	isOpened()	292
3.53.3.9	printErrorInfo()	292
3.53.3.10	resetErrorInfo()	292

3.54 OSRTStreamIF Class Reference	293
3.54.1 Member Function Documentation	293
3.54.1.1 close()	293
3.54.1.2 flush()	294
3.54.1.3 isOpened()	294
3.55 OSRTString Class Reference	295
3.55.1 Detailed Description	295
3.55.2 Constructor & Destructor Documentation	296
3.55.2.1 OSRTString() [1/4]	296
3.55.2.2 OSRTString() [2/4]	296
3.55.2.3 OSRTString() [3/4]	296
3.55.2.4 OSRTString() [4/4]	296
3.55.2.5 ~OSRTString()	297
3.55.3 Member Function Documentation	297
3.55.3.1 clone()	297
3.55.3.2 data()	297
3.55.3.3 getUTF8Value()	297
3.55.3.4 getValue()	298
3.55.3.5 indexOf()	298
3.55.3.6 length()	298
3.55.3.7 operator=()	298
3.55.3.8 print()	298
3.55.3.9 setValue() [1/2]	299
3.55.3.10 setValue() [2/2]	299
3.55.3.11 toInt()	299
3.55.3.12 toSize()	300
3.55.3.13 toUInt()	300
3.55.3.14 toUInt64()	300

3.56 OSRTStringIF Class Reference	301
3.56.1 Detailed Description	301
3.56.2 Constructor & Destructor Documentation	302
3.56.2.1 OSRTStringIF() [1/3]	302
3.56.2.2 OSRTStringIF() [2/3]	302
3.56.2.3 OSRTStringIF() [3/3]	302
3.56.2.4 ~OSRTStringIF()	302
3.56.3 Member Function Documentation	303
3.56.3.1 clone()	303
3.56.3.2 getUTF8Value()	303
3.56.3.3 getValue()	303
3.56.3.4 print()	303
3.56.3.5 setValue() [1/2]	304
3.56.3.6 setValue() [2/2]	304
3.57 OSRTUTF8String Class Reference	305
3.57.1 Detailed Description	305
3.57.2 Constructor & Destructor Documentation	305
3.57.2.1 OSRTUTF8String() [1/4]	305
3.57.2.2 OSRTUTF8String() [2/4]	305
3.57.2.3 OSRTUTF8String() [3/4]	306
3.57.2.4 OSRTUTF8String() [4/4]	306
3.57.2.5 ~OSRTUTF8String()	306
3.57.3 Member Function Documentation	306
3.57.3.1 c_str()	307
3.57.3.2 clone()	307
3.57.3.3 copyValue()	307
3.57.3.4 getValue()	307
3.57.3.5 operator=()	307
3.57.3.6 print()	308
3.57.3.7 setValue()	309

4 File Documentation	311
4.1 ASN1CBitStr.h File Reference	311
4.1.1 Detailed Description	311
4.2 ASN1CGeneralizedTime.h File Reference	311
4.2.1 Detailed Description	312
4.3 ASN1Context.h File Reference	312
4.3.1 Detailed Description	312
4.4 asn1CppEvtHndlr.h File Reference	312
4.4.1 Detailed Description	313
4.5 asn1CppTypes.h File Reference	313
4.5.1 Detailed Description	314
4.5.2 Macro Definition Documentation	314
4.5.2.1 ASN1CATCH	314
4.5.2.2 ASN1RTLTHROW	314
4.5.2.3 ASN1THROW	314
4.5.2.4 ASN1TRY	315
4.6 ASN1CSeqOfList.h File Reference	315
4.6.1 Detailed Description	315
4.7 ASN1CTime.h File Reference	315
4.7.1 Detailed Description	316
4.8 ASN1CUTCTime.h File Reference	316
4.8.1 Detailed Description	316
4.9 asn1ErrCodes.h File Reference	316
4.9.1 Detailed Description	317
4.10 ASN1TObjId.h File Reference	317
4.10.1 Detailed Description	317
4.11 ASN1TOctStr.h File Reference	318
4.11.1 Detailed Description	318

4.12	ASN1TTime.h File Reference	318
4.12.1	Macro Definition Documentation	319
4.12.1.1	LOG_TTMERR	319
4.12.1.2	MAX_TIMESTR_SIZE	319
4.13	OSRTBaseType.h File Reference	319
4.13.1	Detailed Description	320
4.14	OSRTContext.h File Reference	320
4.14.1	Detailed Description	320
4.14.2	Function Documentation	320
4.14.2.1	operator delete()	320
4.14.2.2	operator new()	321
4.15	OSRTFastString.h File Reference	321
4.15.1	Detailed Description	321
4.16	OSRTFileInputStream.h File Reference	321
4.16.1	Detailed Description	321
4.17	OSRTFileOutputStream.h File Reference	321
4.17.1	Detailed Description	322
4.18	OSRTHexTextInputStream.h File Reference	322
4.18.1	Detailed Description	322
4.19	OSRTInputStream.h File Reference	322
4.19.1	Detailed Description	322
4.20	OSRTInputStreamIF.h File Reference	322
4.20.1	Detailed Description	323
4.21	OSRTMemoryInputStream.h File Reference	323
4.21.1	Detailed Description	323
4.22	OSRTMemoryOutputStream.h File Reference	323
4.22.1	Detailed Description	323
4.23	OSRTMsgBuf.h File Reference	323

4.23.1 Detailed Description	324
4.24 OSRTMsgBufIF.h File Reference	324
4.24.1 Detailed Description	324
4.25 OSRTOutputStream.h File Reference	324
4.25.1 Detailed Description	324
4.26 OSRTOutputStreamIF.h File Reference	325
4.26.1 Detailed Description	325
4.27 OSRTSocket.h File Reference	325
4.27.1 Detailed Description	325
4.28 OSRTSocketInputStream.h File Reference	325
4.28.1 Detailed Description	326
4.29 OSRTSocketOutputStream.h File Reference	326
4.29.1 Detailed Description	326
4.30 OSRTStream.h File Reference	326
4.30.1 Detailed Description	326
4.31 OSRTStreamIF.h File Reference	326
4.31.1 Detailed Description	327
4.32 OSRTString.h File Reference	327
4.32.1 Detailed Description	327
4.33 OSRTStringIF.h File Reference	327
4.33.1 Detailed Description	327
4.34 OSRTUTF8String.h File Reference	328
4.34.1 Detailed Description	328

Chapter 1

Main Page

C++ Common Runtime Library Classes

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

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

Chapter 2

Module Documentation

2.1 C++ Run-Time Classes

Modules

- [OSRT Message Buffer Classes](#)
- [Control \(ASN1C_\) Base Classes](#)
- [ASN.1 Type \(ASN1T_\) Base Classes](#)
- [Generic Input Stream Classes](#)
- [Generic Output Stream Classes](#)
- [TCP/IP or UDP Socket Classes](#)
- [Named Event Handlers](#)

2.1.1 Detailed Description

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.

2.2 OSRT Message Buffer Classes

Classes

- class [ASN1MessageBuffer](#)
- class [OSRTMessageBuffer](#)
- class [OSRTMessageBufferIF](#)

2.2.1 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.

2.3 Control (ASN1C_) Base Classes

Modules

- [Date and Time Runtime Classes](#)

Classes

- class [ASN1CType](#)
- class [ASN1CBitStrSizeHolder](#)
- class [ASN1CBitStrSizeHolder8](#)
- class [ASN1CBitStrSizeHolder16](#)
- class [ASN1CBitStrSizeHolder32](#)
- class [ASN1CBitStr](#)
- class [ASN1CSeqOfListIterator](#)
- class [ASN1CSeqOfList](#)

Variables

- class EXTRTCLASS **ASN1CSeqOfList**

2.3.1 Detailed Description

The ASN1C Control Base Classes are used as the base for compiler-generated ASN1C_ classes. These are wrapper classes that can be used to encode/decode PDU types and as helper classes for performing operations on complex data types.

2.4 ASN.1 Type (ASN1T_) Base Classes

Modules

- [Date and Time Runtime Classes](#)

Classes

- struct [ASN1TDynBitStr](#)
- struct [ASN1TDynBitStr64](#)
- struct [ASN1TBitStr32](#)
- struct [ASN1TBMPString](#)
- struct [ASN1TUniversalString](#)
- struct [ASN1TOpenType](#)
- struct [Asn1TObject](#)
- struct [ASN1TSeqExt](#)
- struct [ASN1TPDU](#)
- struct [ASN1TSeqOfList](#)
- struct [ASN1TPDUSeqOfList](#)
- struct [ASN1TObjId](#)
- struct [ASN1TDynOctStr](#)

Typedefs

- typedef [Asn1TObject](#) **ASN1TObject**

Functions

- int [operator==](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator==](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator!=](#) (const [ASN1TObjId](#) &lhs, const [ASN1TObjId](#) &rhs)
- int [operator!=](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator!=](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator!=](#) (const [ASN1TObjId](#) &lhs, const char *dotted_oid_string)
- int [operator<](#) (const [ASN1TObjId](#) &lhs, const [ASN1TObjId](#) &rhs)
- int [operator<](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator<](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator<](#) (const [ASN1TObjId](#) &lhs, const char *dotted_oid_string)
- int [operator<=](#) (const [ASN1TObjId](#) &lhs, const [ASN1TObjId](#) &rhs)
- int [operator<=](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator<=](#) (const [ASN1TObjId](#) &lhs, const char *dotted_oid_string)
- int [operator<=](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator>](#) (const [ASN1TObjId](#) &lhs, const [ASN1TObjId](#) &rhs)
- int [operator>](#) (const [ASN1TObjId](#) &lhs, const char *dotted_oid_string)
- int [operator>](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator>](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator>=](#) (const [ASN1TObjId](#) &lhs, const [ASN1TObjId](#) &rhs)

- int `operator>=` (const [ASN1TObjId](#) &lhs, const char *dotted_oid_string)
- int `operator>=` (const [ASN1OBJID](#) &lhs, const [ASN1OBJID](#) &rhs)
- int `operator>=` (const [ASN1OBJID](#) &lhs, const char *dotted_oid_string)
- [ASN1TObjId](#) `operator+` (const [ASN1TObjId](#) &lhs, const [ASN1TObjId](#) &rhs)
- int `operator==` (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int `operator==` (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int `operator==` (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int `operator==` (const [ASN1DynOctStr](#) &lhs, const char *string)
- int `operator!=` (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int `operator!=` (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int `operator!=` (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int `operator!=` (const [ASN1DynOctStr](#) &lhs, const char *string)
- int `operator<` (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int `operator<` (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int `operator<` (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int `operator<` (const [ASN1DynOctStr](#) &lhs, const char *string)
- int `operator<=` (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int `operator<=` (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int `operator<=` (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int `operator<=` (const [ASN1DynOctStr](#) &lhs, const char *string)
- int `operator>` (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int `operator>` (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int `operator>` (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int `operator>` (const [ASN1DynOctStr](#) &lhs, const char *string)
- int `operator>=` (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int `operator>=` (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int `operator>=` (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int `operator>=` (const [ASN1DynOctStr](#) &lhs, const char *string)

2.4.1 Detailed Description

These classes are used as the base for compiler-generated `ASN1T_C++` data structures. These are enhanced versions of the C structures used for mapping ASN.1 types. The main difference is that constructors and operators have been added to the derived classes.

2.4.2 Function Documentation

2.4.2.1 `operator!=()` [1/8]

```
int operator!=(
    const ASN1TDynOctStr & lhs,
    const ASN1TDynOctStr & rhs )
```

The inequality test operator: compares two dynamic octet strings to each other.

Parameters

<i>lhs</i>	The left-hand ASN1TDynOctStr class.
<i>rhs</i>	The right-hand ASN1TDynOctStr class.

Returns

1 if the two octet strings are equal; 0 otherwise.

Referenced by `ASN1TDynOctStr::operator=()`.

2.4.2.2 `operator!=()` [2/8]

```
int operator!= (
    const ASN1TDynOctStr & lhs,
    const char * string )
```

The inequality test operator: compares a dynamic octet string against a character string.

Parameters

<i>lhs</i>	The left-hand ASN1TDynOctStr class.
<i>string</i>	A character string to be compared against.

Returns

1 if the two strings are equal; 0 otherwise.

2.4.2.3 `operator!=()` [3/8]

```
int operator!= (
    const ASN1DynOctStr & lhs,
    const ASN1DynOctStr & rhs )
```

The inequality test operator: compares two dynamic octet strings to each other.

Parameters

<i>lhs</i>	The left-hand ASN1DynOctStr class.
<i>rhs</i>	The right-hand ASN1DynOctStr structure.

Returns

1 if the two octet strings are equal; 0 otherwise.

2.4.2.4 operator!=(()) [4/8]

```
int operator!= (
    const ASN1DynOctStr & lhs,
    const char * string )
```

The inequality test operator: compares a dynamic octet string against a character string.

Parameters

<i>lhs</i>	The left-hand ASN1DynOctStr class.
<i>string</i>	A character string to be compared against.

Returns

1 if the two strings are equal; 0 otherwise.

2.4.2.5 operator!=(()) [5/8]

```
int operator!= (
    const ASN1ObjId & lhs,
    const ASN1ObjId & rhs )
```

Overloaded not equal operator. This comparison operator allows for comparison of not equality of C++ based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C++ object identifier value.
<i>rhs</i>	- C++ object identifier value

Returns

- True if values are equal.

Referenced by ASN1ObjId::ASN1ObjId(), ASN1TTime::setDER(), and ASN1CTime::setDER().

2.4.2.6 operator!=(()) [6/8]

```
int operator!=(  
    const ASN1OBJID & lhs,  
    const ASN1OBJID & rhs )
```

Overloaded not equal operator. This comparison operator allows for comparison of not equality of C based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C object identifier value.
<i>rhs</i>	- C object identifier value

Returns

- True if values are equal.

2.4.2.7 operator!=(()) [7/8]

```
int operator!=(  
    const ASN1OBJID & lhs,  
    const char * dotted_oid_string )
```

Overloaded not equal operator. This comparison operator allows for comparison of not equality of C based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C object identifier value.
<i>dotted_oid_string</i>	- String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.8 operator!=(()) [8/8]

```
int operator!=(  
    const ASN1ObjId & lhs,  
    const char * dotted_oid_string )
```

Overloaded not equal operator. This comparison operator allows for comparison of not equality of C++ based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C++ object identifier value.
<i>dotted_oid_string</i>	- String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.9 operator+()

```
ASN1ObjId operator+ (
    const ASN1ObjId & lhs,
    const ASN1ObjId & rhs )
```

Overloaded append + operator. This operator allows two Object Identifier values to be concatenated.

Parameters

<i>lhs</i>	- C++ object identifier value.
<i>rhs</i>	- C++ object identifier value.

Referenced by ASN1ObjId::ASN1ObjId().

2.4.2.10 operator<() [1/8]

```
int operator< (
    const ASN1TDynOctStr & lhs,
    const ASN1TDynOctStr & rhs )
```

The less-than test operator: compares two dynamic octet strings to each other.

Parameters

<i>lhs</i>	The left-hand ASN1TDynOctStr class.
<i>rhs</i>	The right-hand ASN1TDynOctStr class.

Returns

- 1 if the two octet strings are equal; 0 otherwise.

Referenced by ASN1TDynOctStr::operator=().

2.4.2.11 operator<() [2/8]

```
int operator< (
    const ASN1TDynOctStr & lhs,
    const char * string )
```

The less-than test operator: compares a dynamic octet string against a character string.

Parameters

<i>lhs</i>	The left-hand ASN1TDynOctStr class.
<i>string</i>	A character string to be compared against.

Returns

1 if the two strings are equal; 0 otherwise.

2.4.2.12 operator<() [3/8]

```
int operator< (
    const ASN1DynOctStr & lhs,
    const ASN1DynOctStr & rhs )
```

The inequality test operator: compares two dynamic octet strings to each other.

Parameters

<i>lhs</i>	The left-hand ASN1DynOctStr class.
<i>rhs</i>	The right-hand ASN1DynOctStr structure.

Returns

1 if the two octet strings are equal; 0 otherwise.

2.4.2.13 operator<() [4/8]

```
int operator< (
    const ASN1ObjId & lhs,
    const ASN1ObjId & rhs )
```

Overloaded less than < operator. This comparison operator allows for comparison of less than of C++ based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C++ object identifier value.
<i>rhs</i>	- C++ object identifier value.

Returns

- True if values are equal.

Referenced by ASN1TObjId::ASN1TObjId(), ASN1TTime::setDER(), and ASN1CTime::setDER().

2.4.2.14 operator<() [5/8]

```
int operator< (
    const ASN1DynOctStr & lhs,
    const char * string )
```

The inequality test operator: compares a dynamic octet string against a character string.

Parameters

<i>lhs</i>	The left-hand ASN1DynOctStr class.
<i>string</i>	A character string to be compared against.

Returns

1 if the two strings are equal; 0 otherwise.

2.4.2.15 operator<() [6/8]

```
int operator< (
    const ASN1OBJID & lhs,
    const ASN1OBJID & rhs )
```

Overloaded less than < operator. This comparison operator allows for comparison of less than of C based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C object identifier value.
<i>rhs</i>	- C object identifier value.

Returns

- True if values are equal.

2.4.2.16 operator<() [7/8]

```
int operator< (
    const ASN1OBJID & lhs,
    const char * dotted_oid_string )
```

Overloaded less than < operator. This comparison operator allows for comparison of less than of C based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C object identifier value.
<i>dotted_oid_string</i>	- String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.17 operator<() [8/8]

```
int operator< (
    const ASN1ObjId & lhs,
    const char * dotted_oid_string )
```

Overloaded less than < operator. This comparison operator allows for comparison of less than of C++ based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C++ object identifier value.
<i>dotted_oid_string</i>	- String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.18 operator<=() [1/8]

```
int operator<= (
    const ASN1TDynOctStr & lhs,
    const ASN1TDynOctStr & rhs )
```

The less-equal test operator: compares two dynamic octet strings to each other.

Parameters

<i>lhs</i>	The left-hand ASN1TDynOctStr class.
<i>rhs</i>	The right-hand ASN1TDynOctStr class.

Returns

- 1 if the two octet strings are equal; 0 otherwise.

Referenced by `ASN1TDynOctStr::operator=()`.

2.4.2.19 operator<=() [2/8]

```
int operator<= (
    const ASN1TDynOctStr & lhs,
    const char * string )
```

The less-than test operator: compares a dynamic octet string against a character string.

Parameters

<i>lhs</i>	The left-hand ASN1TDynOctStr class.
<i>string</i>	A character string to be compared against.

Returns

- 1 if the two strings are equal; 0 otherwise.

2.4.2.20 operator<=() [3/8]

```
int operator<= (
    const ASN1DynOctStr & lhs,
    const ASN1DynOctStr & rhs )
```

The inequality test operator: compares two dynamic octet strings to each other.

Parameters

<i>lhs</i>	The left-hand ASN1DynOctStr class.
<i>rhs</i>	The right-hand ASN1DynOctStr structure.

Returns

1 if the two octet strings are equal; 0 otherwise.

2.4.2.21 operator<=() [4/8]

```
int operator<= (
    const ASN1ObjId & lhs,
    const ASN1ObjId & rhs )
```

Overloaded less than <= operator. This comparison operator allows for comparison of less than of C++ based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C++ object identifier value.
<i>rhs</i>	- C++ object identifier value

Returns

- True if values are equal.

Referenced by ASN1ObjId::ASN1ObjId(), ASN1Time::setDER(), and ASN1CTime::setDER().

2.4.2.22 operator<=() [5/8]

```
int operator<= (
    const ASN1DynOctStr & lhs,
    const char * string )
```

The inequality test operator: compares a dynamic octet string against a character string.

Parameters

<i>lhs</i>	The left-hand ASN1DynOctStr class.
<i>string</i>	A character string to be compared against.

Returns

1 if the two strings are equal; 0 otherwise.

2.4.2.23 operator<=() [6/8]

```
int operator<= (
    const ASN1OBJID & lhs,
    const ASN1OBJID & rhs )
```

Overloaded less than <= operator. This comparison operator allows for comparison of less than of C based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C object identifier value.
<i>rhs</i>	- C object identifier value

Returns

- True if values are equal.

2.4.2.24 operator<=() [7/8]

```
int operator<= (
    const ASN1ObjId & lhs,
    const char * dotted_oid_string )
```

Overloaded less than <= operator. This comparison operator allows for comparison of less than of a C++ based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C++ object identifier value.
<i>dotted_oid_string</i>	- String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.25 operator<=() [8/8]

```
int operator<= (
    const ASN1OBJID & lhs,
    const char * dotted_oid_string )
```

Overloaded less than <= operator. This comparison operator allows for comparison of less than or equal of a C based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C object identifier value.
<i>dotted_oid_string</i>	- String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.26 operator==(1/6]

```
int operator==(
    const ASN1TDynOctStr & lhs,
    const ASN1TDynOctStr & rhs )
```

The equality test operator: compares two dynamic octet strings to each other.

Parameters

<i>lhs</i>	The left-hand ASN1TDynOctStr class.
<i>rhs</i>	The right-hand ASN1TDynOctStr class.

Returns

- 1 if the two octet strings are equal; 0 otherwise.

Referenced by `ASN1TDynOctStr::operator=()`.

2.4.2.27 operator==([2/6])

```
int operator== (
    const ASN1TDynOctStr & lhs,
    const char * string )
```

The equality test operator: compares a dynamic octet string against a character string.

Parameters

<i>lhs</i>	The left-hand ASN1TDynOctStr class.
<i>string</i>	A character string to be compared against.

Returns

1 if the two strings are equal; 0 otherwise.

2.4.2.28 operator==([3/6])

```
int operator== (
    const ASN1DynOctStr & lhs,
    const ASN1DynOctStr & rhs )
```

The equality test operator: compares two dynamic octet strings to each other.

Parameters

<i>lhs</i>	The left-hand ASN1DynOctStr class.
<i>rhs</i>	The right-hand ASN1DynOctStr structure.

Returns

1 if the two octet strings are equal; 0 otherwise.

2.4.2.29 operator==([4/6])

```
int operator== (
    const ASN1DynOctStr & lhs,
    const char * string )
```

The equality test operator: compares a dynamic octet string against a character string.

Parameters

<i>lhs</i>	The left-hand ASN1DynOctStr class.
<i>string</i>	A character string to be compared against.

Returns

1 if the two strings are equal; 0 otherwise.

2.4.2.30 operator==([5/6]

```
int operator== (
    const ASN1OBJID & lhs,
    const ASN1OBJID & rhs )
```

This comparison operator allows for comparison of equality of two C-based object identifier structures.

Parameters

<i>lhs</i>	- C object identifier value.
<i>rhs</i>	- C object identifier value.

Returns

- True if values are equal.

Referenced by ASN1TObjId::ASN1TObjId(), ASN1TTime::setDER(), and ASN1CTime::setDER().

2.4.2.31 operator==([6/6]

```
int operator== (
    const ASN1OBJID & lhs,
    const char * dotted_oid_string )
```

This comparison operator allows for comparison of equality of a C-based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C object identifier value.
<i>dotted_oid_string</i>	- String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.32 operator>() [1/8]

```
int operator> (
    const ASN1TDynOctStr & lhs,
    const ASN1TDynOctStr & rhs )
```

The greater-than test operator: compares two dynamic octet strings to each other.

Parameters

<i>lhs</i>	The left-hand ASN1TDynOctStr class.
<i>rhs</i>	The right-hand ASN1TDynOctStr class.

Returns

- 1 if the two octet strings are equal; 0 otherwise.

Referenced by `ASN1TDynOctStr::operator=()`.

2.4.2.33 operator>() [2/8]

```
int operator> (
    const ASN1TDynOctStr & lhs,
    const char * string )
```

The greater-than test operator: compares a dynamic octet string against a character string.

Parameters

<i>lhs</i>	The left-hand ASN1TDynOctStr class.
<i>string</i>	A character string to be compared against.

Returns

- 1 if the two strings are equal; 0 otherwise.

2.4.2.34 operator>() [3/8]

```
int operator> (
    const ASN1DynOctStr & lhs,
    const ASN1DynOctStr & rhs )
```

The inequality test operator: compares two dynamic octet strings to each other.

Parameters

<i>lhs</i>	The left-hand ASN1DynOctStr class.
<i>rhs</i>	The right-hand ASN1DynOctStr structure.

Returns

1 if the two octet strings are equal; 0 otherwise.

2.4.2.35 operator>() [4/8]

```
int operator> (
    const ASN1ObjId & lhs,
    const ASN1ObjId & rhs )
```

Overloaded greater than > operator. This comparison operator allows for comparison of greater than of C++ based object identifier structures

Parameters

<i>lhs</i>	- C++ object identifier value.
<i>rhs</i>	- C++ object identifier value.

Returns

- True if values are equal.

Referenced by ASN1ObjId::ASN1ObjId(), ASN1Time::setDER(), and ASN1CTime::setDER().

2.4.2.36 operator>() [5/8]

```
int operator> (
    const ASN1DynOctStr & lhs,
    const char * string )
```

The inequality test operator: compares a dynamic octet string against a character string.

Parameters

<i>lhs</i>	The left-hand ASN1DynOctStr class.
<i>string</i>	A character string to be compared against.

Returns

1 if the two strings are equal; 0 otherwise.

2.4.2.37 operator>() [6/8]

```
int operator> (
    const ASN1ObjId & lhs,
    const char * dotted_oid_string )
```

Overloaded greater than > operator. This comparison operator allows for comparison of greater than of a C++ based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C++ object identifier value.
<i>dotted_oid_string</i>	- String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.38 operator>() [7/8]

```
int operator> (
    const ASN1OBJID & lhs,
    const ASN1OBJID & rhs )
```

Overloaded greater than > operator. This comparison operator allows for comparison of greater than of C based object identifier structures.

Parameters

<i>lhs</i>	- C object identifier value.
<i>rhs</i>	- C object identifier value.

Returns

- True if values are equal.

2.4.2.39 operator>() [8/8]

```
int operator> (
    const ASN1OBJID & lhs,
    const char * dotted_oid_string )
```

Overloaded greater than > operator. This comparison operator allows for comparison of greater than of a C based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C object identifier value.
<i>dotted_oid_string</i>	- String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.40 operator>=() [1/8]

```
int operator>= (
    const ASN1TDynOctStr & lhs,
    const ASN1TDynOctStr & rhs )
```

The greater-equal test operator: compares two dynamic octet strings to each other.

Parameters

<i>lhs</i>	The left-hand ASN1TDynOctStr class.
<i>rhs</i>	The right-hand ASN1TDynOctStr class.

Returns

- 1 if the two octet strings are equal; 0 otherwise.

Referenced by `ASN1TDynOctStr::operator=()`.

2.4.2.41 operator>=() [2/8]

```
int operator>= (
    const ASN1TDynOctStr & lhs,
    const char * string )
```

The less-than test operator: compares a dynamic octet string against a character string.

Parameters

<i>lhs</i>	The left-hand ASN1TDynOctStr class.
<i>string</i>	A character string to be compared against.

Returns

1 if the two strings are equal; 0 otherwise.

2.4.2.42 operator>=() [3/8]

```
int operator>= (
    const ASN1DynOctStr & lhs,
    const ASN1DynOctStr & rhs )
```

The inequality test operator: compares two dynamic octet strings to each other.

Parameters

<i>lhs</i>	The left-hand ASN1DynOctStr class.
<i>rhs</i>	The right-hand ASN1DynOctStr structure.

Returns

1 if the two octet strings are equal; 0 otherwise.

2.4.2.43 operator>=() [4/8]

```
int operator>= (
    const ASN1ObjId & lhs,
    const ASN1ObjId & rhs )
```

Overloaded greater than equal >= operator. This comparison operator allows for comparison of greater than or equal of C++ based object identifier structures.

Parameters

<i>lhs</i>	- C++ object identifier value.
<i>rhs</i>	- C++ object identifier value.

Returns

- True if values are equal.

Referenced by ASN1ObjId::ASN1ObjId(), ASN1TTime::setDER(), and ASN1CTime::setDER().

2.4.2.44 operator>=() [5/8]

```
int operator>= (
    const ASN1DynOctStr & lhs,
    const char * string )
```

The inequality test operator: compares a dynamic octet string against a character string.

Parameters

<i>lhs</i>	The left-hand ASN1DynOctStr class.
<i>string</i>	A character string to be compared against.

Returns

- 1 if the two strings are equal; 0 otherwise.

2.4.2.45 operator>=() [6/8]

```
int operator>= (
    const ASN1ObjId & lhs,
    const char * dotted_oid_string )
```

Overloaded greater than equal >= operator. This comparison operator allows for comparison of greater than or equal of a C++ based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C++ object identifier value.
<i>dotted_oid_string</i>	- String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.46 operator>=() [7/8]

```
int operator>= (
    const ASN1OBJID & lhs,
    const ASN1OBJID & rhs )
```

Overloaded greater than equal >= operator. This comparison operator allows for comparison of greater than or equal of C based object identifier structures.

Parameters

<i>lhs</i>	- C object identifier value.
<i>rhs</i>	- C object identifier value.

Returns

- True if values are equal.

2.4.2.47 operator>=() [8/8]

```
int operator>= (
    const ASN1OBJID & lhs,
    const char * dotted_oid_string )
```

Overloaded greater than equal >= operator. This comparison operator allows for comparison of greater than or equal of a C based object identifier structure and a dotted string.

Parameters

<i>lhs</i>	- C object identifier value.
<i>dotted_oid_string</i>	- String containing OID value to compare.

Returns

- True if values are equal.

2.5 Context Management Classes

Classes

- class [ASN1Context](#)

2.5.1 Detailed Description

This group of classes manages an OSCTXT structure. This is the C structure use to keep track of all of the working variables required to encode or decode an ASN.1 message.

2.6 Date and Time Runtime Classes

Classes

- class [ASN1TTime](#)
- class [ASN1TGeneralizedTime](#)
- class [ASN1TUTCTime](#)
- class [ASN1CTime](#)
- class [ASN1CGeneralizedTime](#)
- class [ASN1CUTCTime](#)

Macros

- `#define LOG_TMERR(pctxt, stat) ((pctxt != 0) ? LOG_RTERR (pctxt, stat) : stat)`

2.6.1 Detailed Description

The date and time classes contain methods for doing date/time calculations for the various ASN.1 time types including GeneralizedTime and UTCTime.

2.6.2 Macro Definition Documentation

2.6.2.1 LOG_TMERR

```
#define LOG_TMERR(  
    pctxt,  
    stat ) ((pctxt != 0) ? LOG_RTERR (pctxt, stat) : stat)
```

This macro logs an error in the time functions. It checks to make sure that the context parameter is not null before returning either a logged error (stored in the context) or simply the status itself.

Parameters

<i>pctxt</i>	A pointer to an OSCTXT data structure; it may be null.
<i>stat</i>	The error status.

Returns

The result of calling LOG_RTERR or the status, depending on the value of the pctxt parameter.

2.7 ASN.1 Stream Classes

Classes

- class [OSRTStream](#)
- class [OSRTStreamIF](#)

2.7.1 Detailed Description

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

2.8 Generic Input Stream Classes

Classes

- class [OSRTInputStream](#)
- class [OSRTInputStreamIF](#)
- class [OSRTInputStreamPtr](#)

2.8.1 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.

2.9 Generic Output Stream Classes

Classes

- class [OSRTOutputStream](#)
- class [OSRTOutputStreamIF](#)
- class [OSRTOutputStreamPtr](#)

2.9.1 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.

2.10 TCP/IP or UDP Socket Classes

Classes

- class [OSRSocket](#)

2.10.1 Detailed Description

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

2.11 Named Event Handlers

Classes

- class [Asn1NamedEventHandler](#)
- class [Asn1NullEventHandler](#)
- class [Asn1ErrorHandler](#)

Macros

- #define [OS_UNUSED_ARG](#)(arg) (void)arg

Variables

- class EXTRTCLASS [ASN1MessageBuffer](#)

2.11.1 Detailed Description

Named Event Handler Classes include base classes from which user-defined error handler and event handler classes are derived.

2.11.2 Macro Definition Documentation

2.11.2.1 OS_UNUSED_ARG

```
#define OS_UNUSED_ARG(  
    arg ) (void)arg
```

This macro is used to prevent warnings of unused arguments passed to generated functions and methods.

Parameters

<i>arg</i>	The argument to be passed.
------------	----------------------------

Referenced by [bitStrValue\(\)](#), [boolValue\(\)](#), [charStrValue\(\)](#), [ASN1CType::DecodeFrom\(\)](#), [enumValue\(\)](#), [intValue\(\)](#), [oct↔StrValue\(\)](#), [oidValue\(\)](#), [openTypeValue\(\)](#), [realValue\(\)](#), and [ulntValue\(\)](#).

2.11.3 Variable Documentation

2.11.3.1 ASN1MessageBuffer

```
class EXTRTCLASS ASN1MessageBuffer
```

This definition is a forward reference to the [ASN1MessageBuffer](#) class.

2.12 Run-time error status codes.

Macros

- #define `ASN_OK_FRAG` 2
- #define `ASN_E_BASE` -100
- #define `ASN_E_INVOBJID` (`ASN_E_BASE`)
- #define `ASN_E_INVLEN` (`ASN_E_BASE-1`)
- #define `ASN_E_BADTAG` (`ASN_E_BASE-2`)
- #define `ASN_E_INVBINS` (`ASN_E_BASE-3`)
- #define `ASN_E_INVINDEX` (`ASN_E_BASE-4`)
- #define `ASN_E_INVTCVAL` (`ASN_E_BASE-5`)
- #define `ASN_E_CONCMODF` (`ASN_E_BASE-6`)
- #define `ASN_E_ILLSTATE` (`ASN_E_BASE-7`)
- #define `ASN_E_NOTPDU` (`ASN_E_BASE-8`)
- #define `ASN_E_UNDEFTYP` (`ASN_E_BASE-9`)
- #define `ASN_E_INVPERENC` (`ASN_E_BASE-10`)
- #define `ASN_E_NOTINSEQ` (`ASN_E_BASE-11`)
- #define `ASN_E_BAD_ALIGN` (`ASN_E_BASE-12`)
- #define `ASN_E_UNKNOWNPDU` (`ASN_E_BASE-13`)
- #define `ASN_E_NOTCANON` (`ASN_E_BASE-14`)

2.12.1 Detailed Description

This is a list of status codes that can be returned by the ASN1C run-time functions and generated code. In many cases, additional information and parameters for the different errors are stored in the context structure at the time the error is raised. This additional information can be output using the `rtxErrPrint` or `rtxErrLogUsingCB` run-time functions.

2.12.2 Macro Definition Documentation

2.12.2.1 `ASN_E_BAD_ALIGN`

```
#define ASN_E_BAD_ALIGN (ASN_E_BASE-12)
```

Encoding has bad alignment. This occurs when an encoding is expected to align with a byte boundary and does not. It may happen with reference to the start of the message or with reference to the start of a length-constrained container. Typically this means some bit string element has too few bits, where the encoding of that element is supposed to end the encoding so that padding cannot be used to achieve the desired alignment.

2.12.2.2 ASN_E_BADTAG

```
#define ASN_E_BADTAG (ASN_E_BASE-2)
```

Bad tag value. This error code is returned when a tag value is parsed with an identifier code that is too large to fit in a 32-bit integer variable.

2.12.2.3 ASN_E_BASE

```
#define ASN_E_BASE -100
```

Error base. ASN.1 specific errors start at this base number to distinguish them from common and other error types.

2.12.2.4 ASN_E_CONCMODF

```
#define ASN_E_CONCMODF (ASN_E_BASE-6)
```

Concurrent list modification error. This error is returned from within a list iterator when it is detected that the list was modified outside the control of the iterator.

2.12.2.5 ASN_E_ILLSTATE

```
#define ASN_E_ILLSTATE (ASN_E_BASE-7)
```

Illegal state for operation. This error is returned in places where an operation is attempted but the object is not in a state that would allow the operation to be completed. One example is in a list iterator class when an attempt is made to remove a node but the node does not exist.

2.12.2.6 ASN_E_INVBINS

```
#define ASN_E_INVBINS (ASN_E_BASE-3)
```

Invalid binary string. This error code is returned when decoding XER data and a bit string value is received that contains something other than '1' or '0' characters.

2.12.2.7 ASN_E_INVINDEX

```
#define ASN_E_INVINDEX (ASN_E_BASE-4)
```

Invalid table constraint index. This error code is returned when a value is provided to index into a table and the value does not match any of the defined indexes.

2.12.2.8 ASN_E_INVLEN

```
#define ASN_E_INVLEN (ASN_E_BASE-1)
```

Invalid length. This error code is returned when a length value is parsed that is not consistent with other lengths in a BER/DER message. This typically happens when an inner length within a constructed type is larger than the outer length value.

2.12.2.9 ASN_E_INVOBJID

```
#define ASN_E_INVOBJID (ASN_E_BASE)
```

Invalid object identifier. This error code is returned when an object identifier is encountered that is not valid. Possible reasons for being invalid include invalid first and second arc identifiers (first must be 0, 1, or 2; second must be less than 40), not enough subidentifier values (must be 2 or more), or too many arc values (maximum number is 128).

2.12.2.10 ASN_E_INVPERENC

```
#define ASN_E_INVPERENC (ASN_E_BASE-10)
```

Invalid PER encoding. This occurs when a given element within an ASN.1 specification is configured to have an expected PER encoding and the decoded value does not match this encoding.

2.12.2.11 ASN_E_INVTCVAL

```
#define ASN_E_INVTCVAL (ASN_E_BASE-5)
```

Invalid table constraint value. This error code is returned when a the value for an element in a table-constrained message instance does not match the value for the element defined in the table.

2.12.2.12 ASN_E_NOTCANON

```
#define ASN_E_NOTCANON (ASN_E_BASE-14)
```

Encoded data does not conform to canonical rules for the current encoding rules.

2.12.2.13 ASN_E_NOTINSEQ

```
#define ASN_E_NOTINSEQ (ASN_E_BASE-11)
```

Element not in sequence. This occurs when an element is parsed within a SEQUENCE but is found to not match any of the elements defined for that SEQUENCE.

2.12.2.14 ASN_E_NOTPDU

```
#define ASN_E_NOTPDU (ASN_E_BASE-8)
```

Encode/Decode method called for non-PDU. This error is returned when an Encode or Decode method is called on a non-PDU. Only PDUs have implementations of these methods.

Referenced by ASN1CType::DecodeFrom(), and ASN1CType::EncodeTo().

2.12.2.15 ASN_E_UNDEFTYP

```
#define ASN_E_UNDEFTYP (ASN_E_BASE-9)
```

Element type could not be resolved at run-time. This error is returned when the run-time parser module is used (Asn1RTProd) to decode a type at run-time and the type of the element could not be resolved.

2.12.2.16 ASN_E_UNKNOWNPDU

```
#define ASN_E_UNKNOWNPDU (ASN_E_BASE-13)
```

Unknown PDU type. This error occurs in interpreted BER decoders when the PDU type for the message can't be automatically determined. The user in this case must manually specify the PDU type by some external means. For example, if the application has a command-line interface, it may have a '-pdu' option for specifying the PDU type.

2.12.2.17 ASN_OK_FRAG

```
#define ASN_OK_FRAG 2
```

Fragment decode success status. This is returned when decoding is successful but only a fragment of the item was decoded. User should repeat the decode operation in order to fully decode message.

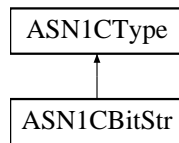
Chapter 3

Class Documentation

3.1 ASN1CBitStr Class Reference

```
#include <ASN1CBitStr.h>
```

Inheritance diagram for ASN1CBitStr:



Public Member Functions

- EXTRTMETHOD [ASN1CBitStr](#) ([OSRTMessageBufferIF](#) &msgbuf, OSUINT32 nbits)
- EXTRTMETHOD [ASN1CBitStr](#) ([OSRTMessageBufferIF](#) &msgbuf, OSOCTET *bitStr, OSUINT32 &numbits, O↔SUINT32 maxNumbits_)
- EXTRTMETHOD **ASN1CBitStr** ([OSRTMessageBufferIF](#) &msgbuf, OSOCTET *bitStr, OSUINT8 &numbits, O↔SUINT32 maxNumbits_)
- EXTRTMETHOD **ASN1CBitStr** ([OSRTMessageBufferIF](#) &msgbuf, OSOCTET *bitStr, OSUINT16 &numbits, O↔SUINT32 maxNumbits_)
- EXTRTMETHOD **ASN1CBitStr** ([OSRTMessageBufferIF](#) &msgbuf, OSOCTET *bitStr, OSUINT32 max↔Numbits_)
- EXTRTMETHOD **ASN1CBitStr** ([OSRTMessageBufferIF](#) &msgBuf, [ASN1TDynBitStr](#) &bitStr)
- EXTRTMETHOD **ASN1CBitStr** ([OSRTContext](#) &ctxt, OSUINT32 nbits)
- EXTRTMETHOD **ASN1CBitStr** ([OSRTContext](#) &ctxt, OSOCTET *bitStr, OSUINT32 &octsNumbits, OSUINT32 maxNumbits_)
- EXTRTMETHOD **ASN1CBitStr** ([OSRTContext](#) &ctxt, OSOCTET *bitStr, OSUINT8 &octsNumbits, OSUINT32 maxNumbits_)
- EXTRTMETHOD **ASN1CBitStr** ([OSRTContext](#) &ctxt, OSOCTET *bitStr, OSUINT16 &octsNumbits, OSUINT32 maxNumbits_)
- EXTRTMETHOD **ASN1CBitStr** ([OSRTContext](#) &ctxt, OSOCTET *bitStr, OSUINT32 maxNumbits_)

- EXTRTMETHOD **ASN1CBitStr** ([OSRTContext](#) &ctxt, [ASN1TDynBitStr](#) &bitStr)
- EXTRTMETHOD **ASN1CBitStr** (const [ASN1CBitStr](#) &bitStr)
- EXTRTMETHOD **ASN1CBitStr** (const [ASN1CBitStr](#) &bitStr, OSBOOL extendable)
- EXTRTMETHOD int **set** (OSUINT32 bitIndex)
- EXTRTMETHOD int **set** (OSUINT32 fromIndex, OSUINT32 toIndex)
- int **change** (OSUINT32 bitIndex, OSBOOL value)
- EXTRTMETHOD int **clear** (OSUINT32 bitIndex)
- EXTRTMETHOD int **clear** (OSUINT32 fromIndex, OSUINT32 toIndex)
- EXTRTMETHOD void **clear** ()
- EXTRTMETHOD int **invert** (OSUINT32 bitIndex)
- EXTRTMETHOD int **invert** (OSUINT32 fromIndex, OSUINT32 toIndex)
- EXTRTMETHOD OSBOOL **get** (OSUINT32 bitIndex)
- OSBOOL **isSet** (OSUINT32 bitIndex)
- OSBOOL **isEmpty** ()
- EXTRTMETHOD OSUINT32 **size** () const
- EXTRTMETHOD OSUINT32 **length** () const
- EXTRTMETHOD OSUINT32 **cardinality** () const
- EXTRTMETHOD int **getBytes** (OSOCKET *pBuf, OSUINT32 bufSz)
- EXTRTMETHOD int **get** (OSUINT32 fromIndex, OSUINT32 toIndex, OSOCKET *pBuf, OSUINT32 bufSz)
- EXTRTMETHOD int **doAnd** (const OSOCKET *pOctstr, OSUINT32 ocsNumbits)
- int **doAnd** (const [ASN1TDynBitStr](#) &bitStr)
- int **doAnd** (const [ASN1CBitStr](#) &bitStr)
- EXTRTMETHOD int **doOr** (const OSOCKET *pOctstr, OSUINT32 ocsNumbits)
- int **doOr** (const [ASN1TDynBitStr](#) &bitStr)
- int **doOr** (const [ASN1CBitStr](#) &bitStr)
- EXTRTMETHOD int **doXor** (const OSOCKET *pOctstr, OSUINT32 ocsNumbits)
- int **doXor** (const [ASN1TDynBitStr](#) &bitStr)
- int **doXor** (const [ASN1CBitStr](#) &bitStr)
- EXTRTMETHOD int **doAndNot** (const OSOCKET *pOctstr, OSUINT32 ocsNumbits)
- int **doAndNot** (const [ASN1TDynBitStr](#) &bitStr)
- int **doAndNot** (const [ASN1CBitStr](#) &bitStr)
- EXTRTMETHOD int **shiftLeft** (OSUINT32 shift)
- EXTRTMETHOD int **shiftRight** (OSUINT32 shift)
- EXTRTMETHOD OSUINT32 **unusedBitsInLastUnit** ()
- EXTRTMETHOD **operator ASN1TDynBitStr** ()
- EXTRTMETHOD **operator ASN1TDynBitStr *** ()

Protected Member Functions

- EXTRTMETHOD **ASN1CBitStr** ([OSRTMessageBufferIF](#) &msgBuf)
- EXTRTMETHOD **ASN1CBitStr** ([OSRTContext](#) &ctxt)
- EXTRTMETHOD **ASN1CBitStr** (OSOCKET *pBits, OSUINT32 &numbits, OSUINT32 maxNumbits)
- EXTRTMETHOD **ASN1CBitStr** (OSOCKET *pBits, OSUINT8 &numbits, OSUINT32 maxNumbits)
- EXTRTMETHOD **ASN1CBitStr** (OSOCKET *pBits, OSUINT16 &numbits, OSUINT32 maxNumbits)
- EXTRTMETHOD **ASN1CBitStr** (OSOCKET *pBits, OSUINT32 maxNumbits)
- EXTRTMETHOD **ASN1CBitStr** ([ASN1TDynBitStr](#) &bitStr)
- void **initBase** (OSOCKET *pBits, OSUINT32 numbits, OSUINT32 maxNumbits)
- EXTRTMETHOD void **init** (OSOCKET *pBits, OSUINT32 &numbits, OSUINT32 maxNumbits)
- EXTRTMETHOD void **init** (OSOCKET *pBits, OSUINT8 &numbits, OSUINT32 maxNumbits)
- EXTRTMETHOD void **init** (OSOCKET *pBits, OSUINT16 &numbits, OSUINT32 maxNumbits)
- EXTRTMETHOD void **init** ([ASN1TDynBitStr](#) &bitStr)

Protected Attributes

- OSOCTET ** **mpUnits**
- OSUINT32 **mMaxNumBits**
- [ASN1CBitStrSizeHolder](#) * **mpNumBits**
- OSUINT32 **mUnitsUsed**
- OSUINT32 **mUnitsAllocated**
- OSBOOL **mDynAlloc**

3.1.1 Detailed Description

ASN.1 bit string control class. The [ASN1CBitStr](#) class is derived from the [ASN1CType](#) base class. It is used as the base class for generated control classes for the ASN.1 BIT STRING type. This class provides utility methods for operating on the bit string referenced by the generated class. This class can also be used inline to operate on bits within generated BIT STRING elements in a SEQUENCE, SET, or CHOICE construct.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 [ASN1CBitStr\(\)](#) [1/2]

```
EXTRTMETHOD ASN1CBitStr::ASN1CBitStr (
    OSRTMessageBufferIF & msgbuf,
    OSUINT32 nbits )
```

This constructor creates an empty bit string. If the *nbits* argument is zero, the bit string is set to be dynamic; otherwise, the capacity is set to *nbits*.

Parameters

<i>msgbuf</i>	- ASN.1 message buffer or stream object.
<i>nbits</i>	- Number of bits this bit string can contain (zero if unbounded).

3.1.2.2 [ASN1CBitStr\(\)](#) [2/2]

```
EXTRTMETHOD ASN1CBitStr::ASN1CBitStr (
    OSRTMessageBufferIF & msgbuf,
    OSOCTET * bitStr,
    OSUINT32 & numbits,
    OSUINT32 maxNumbits_ )
```

This constructor creates a bit string from an array of bits. The constructor does not copy the bit string data, it just references the given data variables. All operations on the bit string cause the referenced items to be updated directly.

Parameters

<i>msgbuf</i>	- ASN.1 message buffer or stream object.
<i>bitStr</i>	- Pointer to static byte array
<i>numbits</i>	- Reference to length of bit string (in bits)
<i>max</i> ↔ <i>Numbits_</i>	- sets maximum length in bits

3.1.3 Member Function Documentation

3.1.3.1 cardinality()

```
EXTRTMETHOD OSUINT32 ASN1CBitStr::cardinality ( ) const
```

This method calculates the cardinality of the target bit string.

Cardinality of the bit string is the number of bits set to 1.

Parameters

-	none
---	------

Returns

The number of bytes of space actually in use by this bit string to represent the bit values.

3.1.3.2 change()

```
int ASN1CBitStr::change (
    OSUINT32 bitIndex,
    OSBOOL value ) [inline]
```

Changes the bit at the specified index to the specified value.

Parameters

<i>bitIndex</i>	Relative index of bit to set in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).
<i>value</i>	Boolean value to which the bit is to be set.

Returns

Completion status of operation: 0 - if succeed

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

3.1.3.3 clear() [1/3]

```
EXTRMETHOD int ASN1CBitStr::clear (
    OSUINT32 bitIndex )
```

This version of the clear method sets the given bit in the target string to zero.

Parameters

<i>bitIndex</i>	Relative index of bit in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).
-----------------	--

Returns

Completion status of operation:

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

3.1.3.4 clear() [2/3]

```
EXTRMETHOD int ASN1CBitStr::clear (
    OSUINT32 fromIndex,
    OSUINT32 toIndex )
```

This version of the clear method sets the bits from the specified fromIndex (inclusive) to the specified toIndex (exclusive) to zero.

Parameters

<i>fromIndex</i>	Relative start index (inclusive) of bits in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).
<i>toIndex</i>	Relative end index (exclusive) of bits in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).

Returns

Completion status of operation:

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

3.1.3.5 clear() [3/3]

```
EXTRTMETHOD void ASN1CBitStr::clear ( )
```

This version of the clear method sets all bits in the bit string to zero.

Parameters

-	none
---	------

Returns

Completion status of operation:

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

3.1.3.6 doAnd() [1/3]

```
EXTRTMETHOD int ASN1CBitStr::doAnd (
    const OSOCTET * pOctstr,
    OSUINT32 octsNumbits )
```

Performs a logical AND of this target bit set with the argument bit set.

Returns: 0 - if succeed, or ASN_E_INVLEN - if 'octsNumbits' is negative, or RTERR_INVPARAM - if pOctstr is the same bit string as this or null, or other error codes (see asn1type.h).

Parameters

<i>pOctstr</i>	A pointer to octets of another bit string for performing logical operation.
<i>octsNumbits</i>	A number of bits in arguent bit string.

Returns

Completion status of operation:

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

3.1.3.7 doAnd() [2/3]

```
int ASN1CBitStr::doAnd (
    const ASN1TDynBitStr & bitStr ) [inline]
```

This method performs a logical AND of the target bit string with the argument bit string.

Parameters

<i>bitStr</i>	A reference to another bit string represented by ASN1TDynBitStr type for performing logical operation.
---------------	--

Returns

Completion status of operation:

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

3.1.3.8 doAnd() [3/3]

```
int ASN1CBitStr::doAnd (
    const ASN1CBitStr & bitStr ) [inline]
```

This method performs a logical AND of the target bit string with the argument bit string.

Parameters

<i>bitStr</i>	A reference to another bit string represented by ASN1CBitStr type for performing logical operation.
---------------	---

Returns

Completion status of operation:

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

References length().

3.1.3.9 doAndNot() [1/3]

```
EXTRTMETHOD int ASN1CBitStr::doAndNot (
    const OSOCTET * pOctstr,
    OSUINT32 octsNumbits )
```

This method performs a logical ANDNOT of the target bit string with the argument bit string.

Logical ANDNOT clears all of the bits in this bit string whose corresponding bit is set in the specified bit string.

Parameters

<i>pOctstr</i>	A pointer to octets of another bit string for performing logical operation.
<i>octsNumbits</i>	A number of bits in argument bit string.

Returns

Completion status of operation:

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

3.1.3.10 doAndNot() [2/3]

```
int ASN1CBitStr::doAndNot (
    const ASN1TDynBitStr & bitStr ) [inline]
```

This method performs a logical ANDNOT of the target bit string with the argument bit string.

Logical ANDNOT clears all of the bits in this bit string whose corresponding bit is set in the specified bit string.

Parameters

<i>bitStr</i>	A reference to another bit string represented by ASN1TDynBitStr type for performing logical operation.
---------------	--

Returns

Completion status of operation:

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

3.1.3.11 doAndNot() [3/3]

```
int ASN1CBitStr::doAndNot (
    const ASN1CBitStr & bitStr ) [inline]
```

This method performs a logical ANDNOT of the target bit string with the argument bit string.

Logical ANDNOT clears all of the bits in this bit string whose corresponding bit is set in the specified bit string.

Parameters

<i>bitStr</i>	A reference to another bit string represented by ASN1CBitStr type for performing logical operation.
---------------	---

Returns

Completion status of operation:

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

References length().

3.1.3.12 doOr() [1/3]

```
EXTRMETHOD int ASN1CBitStr::doOr (
    const OSOCTET * pOctstr,
    OSUINT32 octsNumbits )
```

Performs a logical OR of this target bit set with the argument bit set.

Returns: 0 - if succeed, or ASN_E_INVLEN - if 'octsNumbits' is negative, or RTERR_INVPARAM - if pOctstr is the same bit string as this or null, or other error codes (see [asn1type.h](#)).

Parameters

<i>pOctstr</i>	A pointer to octets of another bit string for performing logical operation.
<i>octsNumbits</i>	A number of bits in arguent bit string.

Returns

Completion status of operation:

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

3.1.3.13 doOr() [2/3]

```
int ASN1CBitStr::doOr (
    const ASN1TDynBitStr & bitStr ) [inline]
```

This method performs a logical OR of the target bit string with the argument bit string.

Parameters

<i>bitStr</i>	A reference to another bit string represented by ASN1TDynBitStr type for performing logical operation.
---------------	--

Returns

Completion status of operation:

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

3.1.3.14 doOr() [3/3]

```
int ASN1CBitStr::doOr (
    const ASN1CBitStr & bitStr ) [inline]
```

This method performs a logical OR of the target bit string with the argument bit string.

Parameters

<i>bitStr</i>	A reference to another bit string represented by ASN1CBitStr type for performing logical operation.
---------------	---

Returns

Completion status of operation:

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

References [length\(\)](#).

3.1.3.15 doXor() [1/3]

```
EXTRTMETHOD int ASN1CBitStr::doXor (
    const OSOCTET * pOctstr,
    OSUINT32 octsNumbits )
```

Performs a logical XOR of this target bit set with the argument bit set.

Returns: 0 - if succeed, or ASN_E_INVLEN - if 'octsNumbits' is negative, or RTERR_INVPARAM - if pOctstr is null, or other error codes (see asn1type.h).

Parameters

<i>pOctstr</i>	A pointer to octets of another bit string for performing logical operation.
<i>octsNumbits</i>	A number of bits in arguent bit string.

Returns

Completion status of operation:

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

3.1.3.16 doXor() [2/3]

```
int ASN1CBitStr::doXor (
    const ASN1TDynBitStr & bitStr ) [inline]
```

This method performs a logical XOR of the target bit string with the argument bit string.

Parameters

<i>bitStr</i>	A reference to another bit string represented by ASN1TDynBitStr type for performing logical operation.
---------------	--

Returns

Completion status of operation:

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

3.1.3.17 doXor() [3/3]

```
int ASN1CBitStr::doXor (
    const ASN1CBitStr & bitStr ) [inline]
```

This method performs a logical OR of the target bit string with the argument bit string.

Parameters

<i>bitStr</i>	A reference to another bit string represented by ASN1CBitStr type for performing logical operation.
---------------	---

Returns

Completion status of operation:

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

References length().

3.1.3.18 get() [1/2]

```
EXTRMETHOD OSBOOL ASN1CBitStr::get (
    OSUINT32 bitIndex )
```

This method returns the value of the bit with the specified index.

Parameters

<i>bitIndex</i>	Relative index of bit in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).
-----------------	--

Returns

Completion status of operation:

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

3.1.3.19 get() [2/2]

```
EXTRMETHOD int ASN1CBitStr::get (
    OSUINT32 fromIndex,
    OSUINT32 toIndex,
    OSOCTET * pBuf,
    OSUINT32 bufSz )
```

This version of the get method copies the bit string composed of bits from this bit string from the specified fromIndex (inclusive) to the specified toIndex (exclusive) into the given buffer.

Parameters

<i>fromIndex</i>	Relative start index (inclusive) of bits in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).
<i>toIndex</i>	Relative end index (exclusive) of bits in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).
<i>pBuf</i>	Pointer to destination buffer where bytes will be copied.
<i>bufSz</i>	Size of the destination buffer. If the size of the buffer is not large enough to receive the entire bit string, a negative status value (RTERR_BUFOVFLOW) will be returned.

Returns

Completion status of operation:

- 0 (0) = success,
- RTERR_OUTOFBND index value is out of bounds
- RTERR_RANGERR fromIndex > toIndex
- other error codes (see asn1type.h).

3.1.3.20 getBytes()

```
EXTRTMETHOD int ASN1CBitStr::getBytes (
    OSOCTET * pBuf,
    OSUINT32 bufSz )
```

This method copies the bit string to the given buffer.

Parameters

<i>pBuf</i>	Pointer to the destination buffer where bytes will be copied.
<i>bufSz</i>	Size of the destination buffer. If the size of the buffer is not large enough to receive the entire bit string, a negative status value (RTERR_BUFOVFLOW) will be returned.

Returns

Completion status of operation:

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

3.1.3.21 invert() [1/2]

```
EXTRTMETHOD int ASN1CBitStr::invert (
    OSUINT32 bitIndex )
```

This version of the invert method inverts the given bit in the target string.

If the bit in the bit string is a zero, it will be set to 1; if the bit is a one, it will be set to 0.

Parameters

<i>bitIndex</i>	Relative index of bit in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).
-----------------	--

Returns

Completion status of operation:

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

3.1.3.22 invert() [2/2]

```
EXTRTMETHOD int ASN1CBitStr::invert (
    OSUINT32 fromIndex,
    OSUINT32 toIndex )
```

This version inverts the bits from the specified fromIndex (inclusive) to the specified toIndex (exclusive).

If the bit in the bit string is a zero, it will be set to 1; if the bit is a one, it will be set to 0.

Parameters

<i>fromIndex</i>	Relative start index (inclusive) of bits in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).
<i>toIndex</i>	Relative end index (exclusive) of bits in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).

Returns

Completion status of operation:

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

3.1.3.23 isEmpty()

```
OSBOOL ASN1CBitStr::isEmpty ( ) [inline]
```

This method returns TRUE if this bit string contains no bits that are set to 1.

Parameters

-	none
---	------

Returns

TRUE, if the bit string contains no bits that are set to 1.

3.1.3.24 isSet()

```
OSBOOL ASN1CBitStr::isSet (
    OSUINT32 bitIndex ) [inline]
```

This method is the same as [ASN1CBitStr::get](#).

See also

[get](#) (OSUINT32 bitIndex)

3.1.3.25 length()

```
EXTRTMETHOD OSUINT32 ASN1CBitStr::length ( ) const
```

This method Calculates the "logical size" of the bith string.

The "logical size" is calculated by noting the index of the highest set bit in the bit string plus one. Zero will be returned if the bit string contains no set bits. The highest bit in the bit string is the LS bit in the last octet set to 1.

Parameters

-	none
---	------

Returns

Returns the "logical size" of this bit string.

Referenced by [doAnd\(\)](#), [doAndNot\(\)](#), [doOr\(\)](#), and [doXor\(\)](#).

3.1.3.26 operator ASN1TDynBitStr()

```
EXTRTMETHOD ASN1CBitStr::operator ASN1TDynBitStr ( )
```

This method returns a filled ANSDITDynBitStr variable.

Memory is not allocated when calling this method; only a pointer is assigned. Thus, the [ASN1TDynBitStr](#) variable is only valid while this [ASN1CBitStr](#) is in scope.

Parameters

-	none
---	------

Returns

Filled [ASN1TDynBitStr](#).

3.1.3.27 operator ASN1TDynBitStr *()

```
EXTRTMETHOD ASN1CBitStr::operator ASN1TDynBitStr * ( )
```

This method returns a pointer to the filled ANSDITDynBitStr variable.

Memory for the ASN1DynBitStr variable is allocated using memory memAlloc and bits are copied into it.

Parameters

-	none
---	------

Returns

Pointer to a filled [ASN1TDynBitStr](#).

3.1.3.28 set() [1/2]

```
EXTRTMETHOD int ASN1CBitStr::set (
    OSUINT32 bitIndex )
```

This version of the set method sets the given bit in the target string.

Parameters

<i>bitIndex</i>	Relative index of the bit to set in the string. The bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from the left to right (MS to LS).
-----------------	--

Returns

Completion status of operation:

- 0 (0) = success,

- negative return value is error.

3.1.3.29 set() [2/2]

```
EXTRTMETHOD int ASN1CBitStr::set (
    OSUINT32 fromIndex,
    OSUINT32 toIndex )
```

This version of the set method sets the bits from the specified fromIndex (inclusive) to the specified toIndex (exclusive) to one.

Parameters

<i>fromIndex</i>	Relative start index (inclusive) of bits in the string. The bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from the left to right (MS to LS).
<i>toIndex</i>	Relative end index (exclusive) of bits in the string. The bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from the left to right (MS to LS).

Returns

Completion status of operation:

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

3.1.3.30 shiftLeft()

```
EXTRTMETHOD int ASN1CBitStr::shiftLeft (
    OSUINT32 shift )
```

This method shifts all bits to the left by the number of specified in the shift operand.

If the bit string can dynamically grow, then the length of the bit string will be decreased by shift bits. Otherwise, bits that are shifted into the bitstring are filled with zeros from the right. Most left bits are lost.

Parameters

<i>shift</i>	Number of bits to be shifted.
--------------	-------------------------------

Returns

Completion status of operation:

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

3.1.3.31 shiftRight()

```
EXTRTMETHOD int ASN1CBitStr::shiftRight (
    OSUINT32 shift )
```

This method shifts all bits to the right by the number of specified in the shift operand.

If the bit string can dynamically grow, then the length of the bit string will be decreased by shift bits. Otherwise, bits that are shifted into the bitstring are filled with zeros from the left. Most right bits are lost.

Parameters

<i>shift</i>	Number of bits to be shifted.
--------------	-------------------------------

Returns

Completion status of operation:

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

3.1.3.32 size()

```
EXTRTMETHOD OSUINT32 ASN1CBitStr::size ( ) const
```

This method returns the number of bytes of space actually in use by this bit string to represent bit values.

Parameters

-	none
---	------

Returns

Number of bytes of space actually in use by this bit string to represent bit values.

3.1.3.33 unusedBitsInLastUnit()

```
EXTRTMETHOD OSUINT32 ASN1CBitStr::unusedBitsInLastUnit ( )
```

This method returns the number of unused bits in the last octet.

Returns

Number of bits in the last octet. It is equal to `length() % 8`.

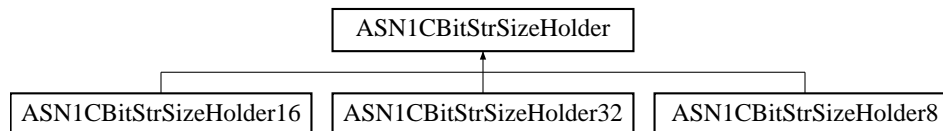
The documentation for this class was generated from the following file:

- [ASN1CBitStr.h](#)

3.2 ASN1CBitStrSizeHolder Class Reference

```
#include <ASN1CBitStr.h>
```

Inheritance diagram for ASN1CBitStrSizeHolder:



Public Member Functions

- virtual `ASN1CBitStrSizeHolder * clone ()=0`
- virtual `OSUINT32 getValue () const =0`
- virtual `int setValue (OSUINT32 value)=0`

3.2.1 Detailed Description

The `ASN1CBitStrSizeHolder` is a class used to hold sizes for the `ASN1CBitStr` control class. This base class is abstract and is implemented by the 8-bit, 16-bit, and 32-bit varieties.

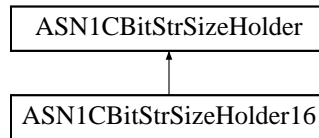
The documentation for this class was generated from the following file:

- [ASN1CBitStr.h](#)

3.3 ASN1CBitStrSizeHolder16 Class Reference

```
#include <ASN1CBitStr.h>
```

Inheritance diagram for ASN1CBitStrSizeHolder16:



Public Member Functions

- **ASN1CBitStrSizeHolder16** (OSUINT16 &value)
- virtual [ASN1CBitStrSizeHolder](#) * **clone** ()
- virtual OSUINT32 **getValue** () const
- virtual int **setValue** (OSUINT32 value)

Protected Attributes

- OSUINT16 & **mSize**

3.3.1 Detailed Description

This is the 16-bit implementation of the [ASN1CBitStrSizeHolder](#) class.

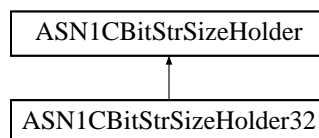
The documentation for this class was generated from the following file:

- [ASN1CBitStr.h](#)

3.4 ASN1CBitStrSizeHolder32 Class Reference

```
#include <ASN1CBitStr.h>
```

Inheritance diagram for ASN1CBitStrSizeHolder32:



Public Member Functions

- **ASN1CBitStrSizeHolder32** (OSUINT32 &value)
- virtual [ASN1CBitStrSizeHolder](#) * **clone** ()
- virtual OSUINT32 **getValue** () const
- virtual int **setValue** (OSUINT32 value)

Protected Attributes

- OSUINT32 & **mSize**

3.4.1 Detailed Description

This is the 32-bit implementation of the [ASN1CBitStrSizeHolder](#) class.

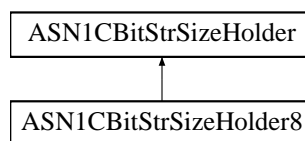
The documentation for this class was generated from the following file:

- [ASN1CBitStr.h](#)

3.5 ASN1CBitStrSizeHolder8 Class Reference

```
#include <ASN1CBitStr.h>
```

Inheritance diagram for ASN1CBitStrSizeHolder8:



Public Member Functions

- **ASN1CBitStrSizeHolder8** (OSUINT8 &value)
- virtual [ASN1CBitStrSizeHolder](#) * **clone** ()
- virtual OSUINT32 **getValue** () const
- virtual int **setValue** (OSUINT32 value)

Protected Attributes

- OSUINT8 & **mSize**

3.5.1 Detailed Description

This is the 8-bit implementation of the [ASN1CBitStrSizeHolder](#) class.

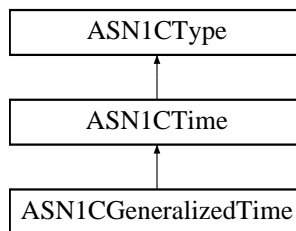
The documentation for this class was generated from the following file:

- [ASN1CBitStr.h](#)

3.6 ASN1CGeneralizedTime Class Reference

```
#include <ASN1CGeneralizedTime.h>
```

Inheritance diagram for ASN1CGeneralizedTime:



Public Member Functions

- EXTRTMETHOD [ASN1CGeneralizedTime](#) ([OSRTMessageBufferIF](#) &msgBuf, char *&buf, int bufSize, OSBOOL useDerRules=FALSE)
- EXTRTMETHOD [ASN1CGeneralizedTime](#) ([OSRTMessageBufferIF](#) &msgBuf, [ASN1GeneralizedTime](#) &buf, OSBOOL useDerRules=FALSE)
- EXTRTMETHOD [ASN1CGeneralizedTime](#) ([OSRTContext](#) &ctxt, char *&buf, int bufSize, OSBOOL useDerRules=FALSE)
- EXTRTMETHOD [ASN1CGeneralizedTime](#) ([OSRTContext](#) &ctxt, [ASN1GeneralizedTime](#) &buf, OSBOOL useDerRules=FALSE)
- [ASN1CGeneralizedTime](#) (const [ASN1CGeneralizedTime](#) &original)
- EXTRTMETHOD int [getCentury](#) ()
- EXTRTMETHOD int [setCentury](#) (short century)
- EXTRTMETHOD int [setTime](#) (time_t time, OSBOOL diffTime)
- const [ASN1CGeneralizedTime](#) & **operator=** (const [ASN1CGeneralizedTime](#) &tm)

Protected Member Functions

- virtual [ASN1TTime](#) & **getTimeObj** ()
- virtual const [ASN1TTime](#) & **getTimeObj** () const
- EXTRTMETHOD **ASN1CGeneralizedTime** (char *&buf, int bufSize, OSBOOL useDerRules=FALSE)
- EXTRTMETHOD **ASN1CGeneralizedTime** ([ASN1GeneralizedTime](#) &buf, OSBOOL useDerRules=FALSE)
- EXTRTMETHOD int [compileString](#) ()

Protected Attributes

- [ASN1GeneralizedTime](#) `timeObj`

Additional Inherited Members

3.6.1 Detailed Description

ASN.1 GeneralizedTime control class. The [ASN1CGeneralizedTime](#) class is derived from the [ASN1CTime](#) base class. It is used as the base class for generated control classes for the ASN.1 Generalized Time ([UNIVERSAL 24] IMPLICIT VisibleString) type. This class provides utility methods for operating on the time information referenced by the generated class. This class can also be used inline to operate on the times within generated time string elements in a SEQUENCE, SET, or CHOICE construct. The time string generally is encoded according to ISO 8601 format with some exceptions (see X.680).

3.6.2 Constructor & Destructor Documentation

3.6.2.1 ASN1CGeneralizedTime() [1/5]

```
EXTRTMETHOD ASN1CGeneralizedTime::ASN1CGeneralizedTime (  
    OSRTMessageBufferIF & msgBuf,  
    char *& buf,  
    int bufSize,  
    OSBOOL useDerRules = FALSE )
```

This constructor creates a time string from a buffer. It does not deep-copy the data, it just assigns the passed array to an internal reference variable. The object will then directly operate on the given data variable.

Parameters

<i>msgBuf</i>	Reference to an OSRTMessage buffer derived object (for example, an ASN1BEREncodeBuffer).
<i>buf</i>	A reference pointer to the time string buffer.
<i>bufSize</i>	The size of the passed buffer, in bytes.
<i>useDerRules</i>	An OSBOOL value.

3.6.2.2 ASN1CGeneralizedTime() [2/5]

```
EXTRTMETHOD ASN1CGeneralizedTime::ASN1CGeneralizedTime (  
    OSRTMessageBufferIF & msgBuf,
```



```
ASN1GeneralizedTime & buf,
OSBOOL useDerRules = FALSE )
```

This constructor creates a time string using the ASN1GeneralizedTime argument. The constructor does not deep-copy the variable, it assigns a reference to it to an internal variable. The object will then directly operate on the given data variable. This form of the constructor is used with a compiler-generated time string variable.

Parameters

<i>msgBuf</i>	Reference to an OSRTMessage buffer derived object (for example, an ASN1BEREncodeBuffer).
<i>buf</i>	A reference pointer to the time string buffer.
<i>useDerRules</i>	An OSBOOL value.

3.6.2.3 ASN1CGeneralizedTime() [3/5]

```
EXTRTMETHOD ASN1CGeneralizedTime::ASN1CGeneralizedTime (
    OSRTContext & ctxt,
    char *& buf,
    int bufSize,
    OSBOOL useDerRules = FALSE )
```

This constructor creates a time string from buffer.

It does not deep-copy the data; it just assigns the passed array to an internal reference variable. The object will then directly operate on the given data variable.

Parameters

<i>ctxt</i>	Reference to an OSRTContext data structure.
<i>buf</i>	Reference to a pointer to a time string buffer.
<i>bufSize</i>	Size of buffer in bytes.
<i>useDerRules</i>	Use the Distinguished Encoding Rules (DER) to operate on this time value.

3.6.2.4 ASN1CGeneralizedTime() [4/5]

```
EXTRTMETHOD ASN1CGeneralizedTime::ASN1CGeneralizedTime (
    OSRTContext & ctxt,
    ASN1GeneralizedTime & buf,
    OSBOOL useDerRules = FALSE )
```

This constructor creates a time string from an ::ASN1GeneralizedTime object.

It does not deep-copy the data; it just assigns the passed array to an internal reference variable. The object will then directly operate on the given data variable.

Parameters

<i>ctxt</i>	Reference to an OSRTContext data structure.
<i>buf</i>	Reference to a pointer to a time string buffer.
<i>useDerRules</i>	Use the Distinguished Encoding Rules (DER) to operate on this time value.

3.6.2.5 ASN1CGeneralizedTime() [5/5]

```
ASN1CGeneralizedTime::ASN1CGeneralizedTime (  
    const ASN1CGeneralizedTime & original ) [inline]
```

The copy constructor. This does not deep-copy the original value. Instead, it assigns references to the internal components.

Parameters

<i>original</i>	The original time string object value.
-----------------	--

References [ASN1CTime::compileString\(\)](#), [ASN1CTime::operator=\(\)](#), and [ASN1CTime::setTime\(\)](#).

3.6.3 Member Function Documentation

3.6.3.1 compileString()

```
EXTRTMETHOD int ASN1CGeneralizedTime::compileString ( ) [protected], [virtual]
```

Compiles new time string according X.680 (clause 41) and ISO 8601. Returns 0, if succeed, or error code, if error.

Returns

0 on success, or an error code otherwise.

Implements [ASN1CTime](#).

3.6.3.2 getCentury()

```
EXTRTMETHOD int ASN1CGeneralizedTime::getCentury ( )
```

This method returns the century part (first two digits) of the year component of the time value.

Parameters

-	none
---	------

Returns

Century part (first two digits) of the year component is returned if the operation is successful. If the operation fails, one of the negative status codes is returned.

3.6.3.3 setCentury()

```
EXTRTMETHOD int ASN1CGeneralizedTime::setCentury (
    short century )
```

This method sets the century part (first two digits) of the year component of the time value.

Parameters

<i>century</i>	Century part (first two digits) of the year component.
----------------	--

Returns

Completion status of operation:

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

3.6.3.4 setTime()

```
EXTRTMETHOD int ASN1CGeneralizedTime::setTime (
    time_t time,
    OSBOOL diffTime ) [virtual]
```

This converts the value of the C built-in type `time_t` to a time string.

The value is the number of seconds from January 1, 1970. Note that the action of this method may differ for different inherited [ASN1CTime](#) Classes.

Parameters

<i>time</i>	The time value, expressed as a number of seconds from January 1, 1970.
<i>diffTime</i>	TRUE means the difference between local time and UTC time will be calculated; in other case, only local time will be stored.

Returns

Completion status of operation:

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

Implements [ASN1CTime](#).

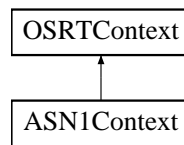
The documentation for this class was generated from the following file:

- [ASN1CGeneralizedTime.h](#)

3.7 ASN1Context Class Reference

```
#include <ASN1Context.h>
```

Inheritance diagram for ASN1Context:



Public Member Functions

- EXTRTMETHOD [ASN1Context](#) ()
- virtual EXTRTMETHOD int [setRunTimeKey](#) (const OSOCTET *key, size_t keylen)
- OSCTXT * **GetPtr** ()
- void **PrintErrorInfo** ()

Additional Inherited Members

3.7.1 Detailed Description

Reference counted ASN.1 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.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 ASN1Context()

```
EXTRTMETHOD ASN1Context::ASN1Context ( )
```

The default constructor initializes the mCtx member variable for ASN.1 encoding/decoding.

3.7.3 Member Function Documentation

3.7.3.1 setRunTimeKey()

```
virtual EXTRTMETHOD int ASN1Context::setRunTimeKey (
    const OSOCTET * key,
    size_t keylen ) [virtual]
```

This method sets run-time key in the context. When using an unlimited runtime, this method will still set the key, but the runtime does not actually check it.

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.

Reimplemented from [OSRTContext](#).

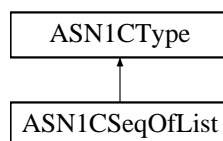
The documentation for this class was generated from the following file:

- [ASN1Context.h](#)

3.8 ASN1CSeqOfList Class Reference

```
#include <ASN1CSeqOfList.h>
```

Inheritance diagram for ASN1CSeqOfList:



Public Member Functions

- EXTRTMETHOD [ASN1CSeqOfList](#) ([OSRTMessageBufferIF](#) &msgBuf, [OSRTDList](#) &lst, OSBOOL initBeforeUse=TRUE)
- EXTRTMETHOD [ASN1CSeqOfList](#) ([OSRTMessageBufferIF](#) &msgBuf)
- EXTRTMETHOD [ASN1CSeqOfList](#) ([ASN1CType](#) &ccobj)
- EXTRTMETHOD [ASN1CSeqOfList](#) ([OSRTMessageBufferIF](#) &msgBuf, [ASN1TSeqOfList](#) &lst)
- EXTRTMETHOD [ASN1CSeqOfList](#) ([ASN1CType](#) &ccobj, [ASN1TSeqOfList](#) &lst)
- EXTRTMETHOD [ASN1CSeqOfList](#) ([OSRTMessageBufferIF](#) &msgBuf, [ASN1TPDUSeqOfList](#) &lst)
- EXTRTMETHOD **[ASN1CSeqOfList](#)** ([OSRTContext](#) &ctxt, [OSRTDList](#) &lst, OSBOOL initBeforeUse=TRUE)
- EXTRTMETHOD **[ASN1CSeqOfList](#)** ([OSRTContext](#) &ctxt)
- EXTRTMETHOD **[ASN1CSeqOfList](#)** ([OSRTContext](#) &ctxt, [ASN1TSeqOfList](#) &lst)
- EXTRTMETHOD **[ASN1CSeqOfList](#)** ([OSRTContext](#) &ctxt, [ASN1TPDUSeqOfList](#) &lst)
- EXTRTMETHOD void [append](#) (void *data)
- EXTRTMETHOD void [appendArray](#) (const void *data, OSSIZE numElems, OSSIZE elemSize)
- EXTRTMETHOD void [appendArrayCopy](#) (const void *data, OSSIZE numElems, OSSIZE elemSize)
- void [init](#) ()
- EXTRTMETHOD void [insert](#) (int index, void *data)
- EXTRTMETHOD void [remove](#) (int index)
- EXTRTMETHOD void [remove](#) (void *data)
- void [removeFirst](#) ()
- void [removeLast](#) ()
- EXTRTMETHOD int [indexOf](#) (void *data) const
- OSBOOL [contains](#) (void *data) const
- EXTRTMETHOD void * [getFirst](#) ()
- EXTRTMETHOD void * [getLast](#) ()
- EXTRTMETHOD void * [get](#) (int index) const
- EXTRTMETHOD void * [set](#) (int index, void *data)
- EXTRTMETHOD void [clear](#) ()
- EXTRTMETHOD void [freeMemory](#) ()
- EXTRTMETHOD OSBOOL [isEmpty](#) () const
- EXTRTMETHOD OSSIZE [size](#) () const
- EXTRTMETHOD [ASN1CSeqOfListIterator](#) * [iterator](#) ()
- EXTRTMETHOD [ASN1CSeqOfListIterator](#) * [iteratorFromLast](#) ()
- EXTRTMETHOD [ASN1CSeqOfListIterator](#) * [iteratorFrom](#) (void *data)
- EXTRTMETHOD void * [toArray](#) (OSSIZE elemSize)
- EXTRTMETHOD void * [toArray](#) (void *pArray, OSSIZE elemSize, OSSIZE allocatedElems)
- void * [operator\[\]](#) (int index) const
- **operator [OSRTDList](#)** * ()

Protected Member Functions

- EXTRTMETHOD **[ASN1CSeqOfList](#)** ([OSRTDList](#) &lst)
- EXTRTMETHOD **[ASN1CSeqOfList](#)** ([ASN1TSeqOfList](#) &lst)
- EXTRTMETHOD **[ASN1CSeqOfList](#)** ([ASN1TPDUSeqOfList](#) &lst)
- EXTRTMETHOD void **[remove](#)** ([OSRTDListNode](#) *node)
- EXTRTMETHOD void **[insertBefore](#)** (void *data, [OSRTDListNode](#) *node)
- EXTRTMETHOD void **[insertAfter](#)** (void *data, [OSRTDListNode](#) *node)

Protected Attributes

- OSRTDList * **pList**
- volatile int **modCount**
- OSBOOL **wasAssigned**

3.8.1 Detailed Description

Doubly-linked list implementation. This class provides all functionality necessary for linked list operations. It is the base class for ASN1C compiler-generated ASN1C_ control classes for SEQUENCE OF and SET OF PDU types.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 ASN1CSeqOfList() [1/6]

```
EXTRTMETHOD ASN1CSeqOfList::ASN1CSeqOfList (
    OSRTMessageBufferIF & msgBuf,
    OSRTDList & lst,
    OSBOOL initBeforeUse = TRUE )
```

This constructor creates a linked list using the OSRTDList argument. The constructor does not deep-copy the variable; it assigns a reference to it to an external variable.

The object will then directly operate on the given list variable.

Parameters

<i>msgBuf</i>	Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).
<i>lst</i>	Reference to a linked list structure.
<i>initBeforeUse</i>	Set to TRUE if the passed linked list needs to be initialized (rtxDListInit to be called).

3.8.2.2 ASN1CSeqOfList() [2/6]

```
EXTRTMETHOD ASN1CSeqOfList::ASN1CSeqOfList (
    OSRTMessageBufferIF & msgBuf )
```

This constructor creates an empty linked list.

Parameters

<i>msgBuf</i>	Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).
---------------	--

3.8.2.3 ASN1CSeqOfList() [3/6]

```
EXTRTMETHOD ASN1CSeqOfList::ASN1CSeqOfList (
    ASN1CType & ccobj )
```

This constructor creates an empty linked list.

Parameters

<i>ccobj</i>	Reference to a control class object (for example, any generated ASN1C_ class object).
--------------	---

3.8.2.4 ASN1CSeqOfList() [4/6]

```
EXTRTMETHOD ASN1CSeqOfList::ASN1CSeqOfList (
    OSRTMessageBufferIF & msgBuf,
    ASN1TSeqOfList & lst )
```

This constructor creates a linked list using the [ASN1TSeqOfList](#) (holder of OSRTDList) argument.

The construction does not deep-copy the variable, it assigns a reference to it to an internal variable. The object will then directly operate on the given list variable. This constructor is used with a compiler-generated linked list variable.

Parameters

<i>msgBuf</i>	Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).
<i>lst</i>	Reference to a linked list holder.

3.8.2.5 ASN1CSeqOfList() [5/6]

```
EXTRTMETHOD ASN1CSeqOfList::ASN1CSeqOfList (
    ASN1CType & ccobj,
    ASN1TSeqOfList & lst )
```

This constructor creates a linked list using the [ASN1TSeqOfList](#) (holder of OSRTDList) argument.

The construction does not deep-copy the variable, it assigns a reference to it to an internal variable. The object will then directly operate on the given list variable. This constructor is used with a compiler-generated linked list variable.

Parameters

<i>ccobj</i>	Reference to a control class object (for example, any generated ASN1C_ class object).
<i>lst</i>	Reference to a linked list holder.

3.8.2.6 ASN1CSeqOfList() [6/6]

```
EXTRTMETHOD ASN1CSeqOfList::ASN1CSeqOfList (
    OSRTMessageBufferIF & msgBuf,
    ASN1TPDUSeqOfList & lst )
```

This constructor creates a linked list using the [ASN1TPDUSeqOfList](#) argument.

The construction does not deep-copy the variable, it assigns a reference to it to an internal variable. The object will then directly operate on the given list variable. This constructor is used with a compiler-generated linked list variable.

Parameters

<i>msgBuf</i>	Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).
<i>lst</i>	Reference to a linked list holder.

3.8.3 Member Function Documentation

3.8.3.1 append()

```
EXTRTMETHOD void ASN1CSeqOfList::append (
    void * data )
```

This method appends an item to the linked list.

This item is represented by a void pointer that can point to an object of any type. The `rtxMemAlloc` function is used to allocate memory for the list node structure, therefore, all internal list memory will be released whenever `rtxMemFree` is called.

Parameters

<i>data</i>	Pointer to a data item to be appended to the list.
-------------	--

Returns

- none

3.8.3.2 appendArray()

```
EXTRTMETHOD void ASN1CSeqOfList::appendArray (
    const void * data,
    OSSIZE numElems,
    OSSIZE elemSize )
```

This method appends array items' pointers to a doubly linked list.

The rtxMemAlloc function is used to allocate memory for the list node structure, therefore all internal list memory will be released whenever the rtxMemFree is called. The data is not copied; it is just assigned to the node.

Parameters

<i>data</i>	Pointer to source array to be appended to the list.
<i>numElems</i>	The number of elements in the source array.
<i>elemSize</i>	The size of one element in the array. Use the <i>sizeof()</i> operator to pass this parameter.

Returns

- none

3.8.3.3 appendArrayCopy()

```
EXTRTMETHOD void ASN1CSeqOfList::appendArrayCopy (
    const void * data,
    OSSIZE numElems,
    OSSIZE elemSize )
```

This method appends array items into a doubly linked list.

The rtxMemAlloc function is used to allocate memory for the list node structure; therefore all internal list memory will be released whenever rtxMemFree is called. The data will be copied; the memory will be allocated using rtxMemAlloc.

Parameters

<i>data</i>	Pointer to source array to be appended to the list.
<i>numElems</i>	The number of elements in the source array.
<i>elemSize</i>	The size of one element in the array. Use the <i>sizeof()</i> operator to pass this parameter.

Returns

- none

3.8.3.4 clear()

```
EXTRTMETHOD void ASN1CSeqOfList::clear ( )
```

This method removes all items from the list.

3.8.3.5 contains()

```
OSBOOL ASN1CSeqOfList::contains (
    void * data ) const [inline]
```

This method returns TRUE if this list contains the specified pointer. Note that a match is not done on the *contents* of each data item (i.e. what is pointed at by the pointer), only the pointer values.

Parameters

<i>data</i>	- Pointer to data item.
-------------	-------------------------

Returns

TRUE if this pointer value found in the list.

3.8.3.6 freeMemory()

```
EXTRTMETHOD void ASN1CSeqOfList::freeMemory ( )
```

This method removes all items from the list and frees the associated memory.

3.8.3.7 get()

```
EXTRTMETHOD void* ASN1CSeqOfList::get (
    int index ) const
```

This method returns the item at the specified position in the list.

Parameters

<i>index</i>	Index of the item to be returned.
--------------	-----------------------------------

Returns

The item at the specified index in the list.

3.8.3.8 `getFirst()`

```
EXTRTMETHOD void* ASN1CSeqOfList::getFirst ( )
```

This method returns the first item from the list or null if there are no elements in the list.

Returns

The first item of the list.

3.8.3.9 `getLast()`

```
EXTRTMETHOD void* ASN1CSeqOfList::getLast ( )
```

This method returns the last item from the list or null if there are no elements in the list.

Returns

The last item in the list.

3.8.3.10 `indexOf()`

```
EXTRTMETHOD int ASN1CSeqOfList::indexOf (
    void * data ) const
```

This method returns the index in this list of the first occurrence of the specified item, or -1 if the list does not contain the item.

Parameters

<i>data</i>	- Pointer to data item to searched.
-------------	-------------------------------------

Returns

The index in this list of the first occurrence of the specified item, or -1 if the list does not contain the item.

3.8.3.11 init()

```
void ASN1CSeqOfList::init ( ) [inline]
```

This method initializes the linked list structure.

References ASN1CSeqOfListIterator::insert().

3.8.3.12 insert()

```
EXTRTMETHOD void ASN1CSeqOfList::insert (
    int index,
    void * data )
```

This method inserts an item into the linked list structure.

The item is represented by a void pointer that can point to an object of any type. The rtxMemAlloc function is used to allocate memory for the list node structure. All internal list memory will be released when the rtxMemFree function is called.

Parameters

<i>index</i>	Index at which the specified item is to be inserted.
<i>data</i>	Pointer to data item to be appended to the list.

Returns

- none

Referenced by ASN1CSeqOfListIterator::hasPrev().

3.8.3.13 isEmpty()

```
EXTRTMETHOD OSBOOL ASN1CSeqOfList::isEmpty ( ) const
```

This method returns TRUE if the list is empty.

Returns

TRUE if this list is empty.

3.8.3.14 iterator()

```
EXTRTMETHOD ASN1CSeqOfListIterator* ASN1CSeqOfList::iterator ( )
```

This method returns an iterator over the elements in the linked list in the sequence from the first to the last.

Returns

The iterator over this linked list.

3.8.3.15 iteratorFrom()

```
EXTRTMETHOD ASN1CSeqOfListIterator* ASN1CSeqOfList::iteratorFrom (
    void * data )
```

This method runs an iterator over the elements in this linked list starting from the specified item in the list.

Parameters

<i>data</i>	The item of the list to be iterated first.
-------------	--

Returns

The iterator over this linked list.

3.8.3.16 iteratorFromLast()

```
EXTRTMETHOD ASN1CSeqOfListIterator* ASN1CSeqOfList::iteratorFromLast ( )
```

This method creates a reverse iterator over the elements in this linked list in the sequence from last to first.

Parameters

-	none
---	------

Returns

The reverse iterator over this linked list.

3.8.3.17 operator[]()

```
void* ASN1CSeqOfList::operator[] (
    int index ) const [inline]
```

This method is the overloaded operator[].

It returns the item at the specified position in the list.

See also

[get](#) (int index)

3.8.3.18 remove() [1/2]

```
EXTRTMETHOD void ASN1CSeqOfList::remove (
    int index )
```

This method removed a node at the specified index from the linked list structure.

The `rtxMemAlloc` function was used to allocate the memory for the list node structure, therefore, all internal list memory will be released whenever the `rtxMemFree` is called.

Parameters

<i>index</i>	Index of the item to be removed.
--------------	----------------------------------

Returns

- none

3.8.3.19 remove() [2/2]

```
EXTRTMETHOD void ASN1CSeqOfList::remove (
    void * data )
```

This method removes the first occurrence of the node with specified data from the linked list structure.

The `rtxMemAlloc` function was used to allocate the memory for the list node structure, therefore, all internal list memory will be released whenever the `rtxMemFree` function is called.

Parameters

<i>data</i>	- Pointer to the data item to be appended to the list.
-------------	--

3.8.3.20 removeFirst()

```
void ASN1CSeqOfList::removeFirst ( ) [inline]
```

This method removes the first node (head) from the linked list structure.

Parameters

-	none
---	------

Returns

- none

3.8.3.21 removeLast()

```
void ASN1CSeqOfList::removeLast ( ) [inline]
```

This method removes the last node (tail) from the linked list structure.

Parameters

-	none
---	------

Returns

- none

3.8.3.22 set()

```
EXTRTMETHOD void* ASN1CSeqOfList::set (
    int index,
    void * data )
```

This method replaces the item at the specified index in this list with the specified item.

Parameters

<i>index</i>	The index of the item to be replaced.
<i>data</i>	The item to be stored at the specified index.

Returns

The item previously at the specified position.

3.8.3.23 size()

```
EXTRTMETHOD OSSIZE ASN1CSeqOfList::size ( ) const
```

This method returns the number of nodes in the list.

Returns

The number of items in this list.

3.8.3.24 toArray() [1/2]

```
EXTRTMETHOD void* ASN1CSeqOfList::toArray (
    OSSIZE elemSize )
```

This method converts the linked list into a new array.

The `rtxMemAlloc` function is used to allocate memory for an array.

Parameters

<i>elemSize</i>	The size of one element in the array. Use the <code>sizeof()</code> operator to pass this parameter.
-----------------	--

Returns

The point to converted array.

3.8.3.25 toArray() [2/2]

```
EXTRTMETHOD void* ASN1CSeqOfList::toArray (
    void * pArray,
    OSSIZE elemSize,
    OSSIZE allocatedElems )
```

This method converts the linked list into an array.

The rtxMemAlloc function is used to allocate memory for the array if the capacity of the specified array is exceeded.

Parameters

<i>pArray</i>	Pointer to destination array.
<i>elemSize</i>	The size of one element in the array. Use the sizeof() operator to pass this parameter.
<i>allocatedElems</i>	The number of elements already allocated in the array. If this number is less than the number of nodes in the list, then a new array is allocated and returned. Memory is allocated using rtxMemAlloc function.

Returns

The pointer to the converted array.

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

- [ASN1CSeqOfList.h](#)

3.9 ASN1CSeqOfListIterator Class Reference

```
#include <ASN1CSeqOfList.h>
```

Public Member Functions

- OSBOOL [hasNext](#) ()
- OSBOOL [hasPrev](#) ()
- EXTRTMETHOD void * [next](#) ()
- EXTRTMETHOD void * [prev](#) ()
- EXTRTMETHOD int [remove](#) ()
- EXTRTMETHOD int [set](#) (void *data)
- EXTRTMETHOD int [insert](#) (void *data)
- int [getState](#) ()

Protected Member Functions

- EXTRTMETHOD **ASN1CSeqOfListIterator** ([ASN1CSeqOfList](#) *list)
- EXTRTMETHOD **ASN1CSeqOfListIterator** ([ASN1CSeqOfList](#) *list, OSRTDListNode *startNode)
- void * **operator new** (size_t, void *data)
- void **operator delete** (void *, void *)
- void **operator delete** (void *, size_t)

Protected Attributes

- [ASN1CSeqOfList](#) * **pSeqList**
- OSRTDListNode * **nextNode**
- OSRTDListNode * **lastNode**
- volatile int **expectedModCount**
- int **stat**

3.9.1 Detailed Description

Linked list iterator class. The [ASN1CSeqOfListIterator](#) class is an iterator for linked lists (represented by [ASN1CSeqOfList](#)) that allows the programmer to traverse the list in either direction and modify the list during iteration. The iterator is fail-fast. This means the list is structurally modified at any time after the [ASN1CSeqOfListIterator](#) class is created, in any way except through the iterator's own remove or insert methods, the iterator's methods next and prev methods will return NULL. The remove, set and insert methods will return the RTERR_CONCMODF error code.

3.9.2 Member Function Documentation

3.9.2.1 hasNext()

```
OSBOOL ASN1CSeqOfListIterator::hasNext ( ) [inline]
```

This method returns TRUE if this iterator has more elements when traversing the list in the forward direction.

In other words, the method returns TRUE if the `next` method would return an element rather than returning a null value.

Returns

TRUE if next would return an element rather than returning a null value.

3.9.2.2 hasPrev()

```
OSBOOL ASN1CSeqOfListIterator::hasPrev ( ) [inline]
```

This method returns TRUE if this iterator has more elements when traversing the list in the reverse direction.

In other words, this method will return TRUE if prev would return an element rather than returning a null value.

Returns

TRUE if next would return an element rather than returning a null value.

References ASN1CSeqOfList::insert().

3.9.2.3 insert()

```
EXTRTMETHOD int ASN1CSeqOfListIterator::insert (
    void * data )
```

This method inserts the specified element into the list.

The element is inserted immediately before the next element that would be returned by the next method, if any, and after the next element would be returned by the prev method, if any. If the list contains no elements, the new element becomes the sole element in the list. The new element is inserted before the implicit cursor: a subsequent call to next would be unaffected, and a subsequent call to prev would return the new element.

Parameters

<i>data</i>	The element to be inserted
-------------	----------------------------

Returns

Completion status of operation:

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

Referenced by ASN1CSeqOfList::init().

3.9.2.4 next()

```
EXTRTMETHOD void* ASN1CSeqOfListIterator::next ( )
```

This method returns the next element in the list.

This method may be called repeatedly to iterate through the list or intermixed with calls to prev to go back and forth.

Returns

The next element in the list. A null value will be returned if the iteration is not successful.

3.9.2.5 prev()

```
EXTRTMETHOD void* ASN1CSeqOfListIterator::prev ( )
```

This method returns the previous element in the list.

This method may be called repeatedly to iterate through the list or intermixed with calls to next to go back and forth.

Parameters

-	none
---	------

Returns

The previous element in the list. A null value will be returned if the iteration is not successful.

3.9.2.6 remove()

```
EXTRTMETHOD int ASN1CSeqOfListIterator::remove ( )
```

This method removes from the list the last element that was returned by the next or prev methods.

This call can only be made once per call to the next or prev methods.

Parameters

-	none
---	------

Returns

Completion status of operation:

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

3.9.2.7 set()

```
EXTRTMETHOD int ASN1CSeqOfListIterator::set (
    void * data )
```

This method replaces the last element returned by the next or prev methods with the specified element.

This call can be made only if neither remove nor insert methods have been called after the last call to next or prev methods.

Parameters

<i>data</i>	The element that replaces the last element returned by the next or prev methods
-------------	---

Returns

Completion status of operation:

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

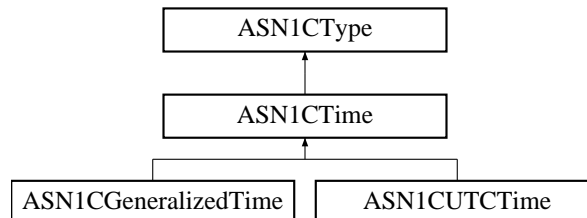
The documentation for this class was generated from the following file:

- [ASN1CSeqOfList.h](#)

3.10 ASN1CTime Class Reference

```
#include <ASN1CTime.h>
```

Inheritance diagram for ASN1CTime:



Public Types

- enum {
January = 1, Jan = 1, February = 2, Feb = 2,
March = 3, Mar = 3, April = 4, Apr = 4,
May = 5, June = 6, Jun = 6, July = 7,
Jul = 7, August = 8, Aug = 8, September = 9,
Sep = 9, October = 10, Oct = 10, November = 11,
Nov = 11, December = 12, Dec = 12 }

Public Member Functions

- EXTRTMETHOD [ASN1CTime](#) ([OSRTMessageBufferIF](#) &msgBuf, char *&buf, int bufSize, OSBOOL useDer←Rules)
- EXTRTMETHOD [ASN1CTime](#) ([OSRTMessageBufferIF](#) &msgBuf, [ASN1VisibleString](#) &buf, OSBOOL useDer←Rules)
- EXTRTMETHOD [ASN1CTime](#) ([OSRTContext](#) &ctxt, char *&buf, int bufSize, OSBOOL useDerRules)
- EXTRTMETHOD [ASN1CTime](#) ([OSRTContext](#) &ctxt, [ASN1VisibleString](#) &buf, OSBOOL useDerRules)
- EXTRTMETHOD [ASN1CTime](#) (const [ASN1CTime](#) &original)
- EXTRTMETHOD [~ASN1CTime](#) ()
- virtual EXTRTMETHOD int [getYear](#) ()
- virtual EXTRTMETHOD int [getMonth](#) ()
- virtual EXTRTMETHOD int [getDay](#) ()
- virtual EXTRTMETHOD int [getHour](#) ()
- virtual EXTRTMETHOD int [getMinute](#) ()
- virtual EXTRTMETHOD int [getSecond](#) ()
- virtual EXTRTMETHOD int [getFraction](#) ()
- virtual EXTRTMETHOD double [getFractionAsDouble](#) ()
- virtual EXTRTMETHOD int [getFractionStr](#) (char *const pBuf, size_t bufSize)
- virtual EXTRTMETHOD int [getFractionLen](#) ()
- virtual EXTRTMETHOD int [getDiffHour](#) ()
- virtual EXTRTMETHOD int [getDiffMinute](#) ()
- virtual EXTRTMETHOD int [getDiff](#) ()
- virtual EXTRTMETHOD OSBOOL [getUTC](#) ()
- virtual EXTRTMETHOD time_t [getTime](#) ()
- void [setDER](#) (OSBOOL bvalue)
- virtual EXTRTMETHOD int [setUTC](#) (OSBOOL utc)
- virtual EXTRTMETHOD int [setYear](#) (short year_)
- virtual EXTRTMETHOD int [setMonth](#) (short month_)
- virtual EXTRTMETHOD int [setDay](#) (short day_)
- virtual EXTRTMETHOD int [setHour](#) (short hour_)
- virtual EXTRTMETHOD int [setMinute](#) (short minute_)
- virtual EXTRTMETHOD int [setSecond](#) (short second_)
- virtual EXTRTMETHOD int [setFraction](#) (int fraction, int fracLen=-1)
- virtual EXTRTMETHOD int [setFraction](#) (double frac, int fracLen)
- virtual EXTRTMETHOD int [setFraction](#) (char const *frac)
- virtual int [setTime](#) (time_t time, OSBOOL diffTime)=0
- virtual EXTRTMETHOD int [setDiffHour](#) (short dhour)
- virtual EXTRTMETHOD int [setDiff](#) (short dhour, short dminute)
- virtual EXTRTMETHOD int [setDiff](#) (short inMinutes)
- virtual EXTRTMETHOD int [parseString](#) (const char *string)
- virtual EXTRTMETHOD void [clear](#) ()
- virtual EXTRTMETHOD int [equals](#) ([ASN1CTime](#) &)
- EXTRTMETHOD size_t [getTimeStringLength](#) ()
- EXTRTMETHOD const char * [getTimeString](#) (char *pbuf, size_t bufsize)
- EXTRTMETHOD const [ASN1CTime](#) & [operator=](#) (const [ASN1CTime](#) &)
- virtual EXTRTMETHOD OSBOOL [operator==](#) ([ASN1CTime](#) &)
- virtual EXTRTMETHOD OSBOOL [operator!=](#) ([ASN1CTime](#) &)
- virtual EXTRTMETHOD OSBOOL [operator>](#) ([ASN1CTime](#) &)
- virtual EXTRTMETHOD OSBOOL [operator<](#) ([ASN1CTime](#) &)
- virtual EXTRTMETHOD OSBOOL [operator>=](#) ([ASN1CTime](#) &)
- virtual EXTRTMETHOD OSBOOL [operator<=](#) ([ASN1CTime](#) &)

Protected Member Functions

- EXTRTMETHOD void **checkCapacity** ()
- EXTRTMETHOD char *& **getTimeStringPtr** ()
- virtual [ASN1TTime](#) & **getTimeObj** ()=0
- virtual const [ASN1TTime](#) & **getTimeObj** () const =0
- EXTRTMETHOD **ASN1CTime** (char *&buf, int bufSize, OSBOOL useDerRules)
- EXTRTMETHOD **ASN1CTime** (ASN1VisibleString &buf, OSBOOL useDerRules)
- virtual int [compileString](#) ()=0

Protected Attributes

- OSBOOL **parsed**
- OSBOOL **derRules**
- char *& **timeStr**
- int **strSize**

3.10.1 Detailed Description

ASN.1 Time control base class. The [ASN1CTime](#) class is derived from the [ASN1CType](#) base class. It is used as the abstract base class for generated control classes for the ASN.1 Generalized Time ([UNIVERSAL 24] IMPLICIT VisibleString) types and Universal Time ([UNIVERSAL 23] IMPLICIT VisibleString) types. This class provides utility methods for operating on the time information referenced by the generated class. This class can also be used inline to operate on the times within generated time string elements in a SEQUENCE, SET, or CHOICE construct. The time string are generally formatted according to ISO 8601 format with some exceptions (X.680).

3.10.2 Constructor & Destructor Documentation

3.10.2.1 ASN1CTime() [1/5]

```
EXTRTMETHOD ASN1CTime::ASN1CTime (  
    OSRTMessageBufferIF & msgBuf,  
    char *& buf,  
    int bufSize,  
    OSBOOL useDerRules )
```

This constructor creates a time string from buffer.

It does not deep-copy the data; it just assigns the passed array to an internal reference variable. The object will then directly operate on the given data variable.

Parameters

<i>msgBuf</i>	Reference to an OSRTMessage buffer derived object (for example, ASNBEREncodeBuffer).
<i>buf</i>	Reference to a pointer to a time string buffer.
<i>bufSize</i>	Size of buffer in bytes. 89
<i>useDerRules</i>	Use the Distinguished Encoding Rules (DER) to operate on this time value.

3.10.2.2 ASN1CTime() [2/5]

```
EXTRTMETHOD ASN1CTime::ASN1CTime (  
    OSRTMessageBufferIF & msgBuf,  
    ASN1VisibleString & buf,  
    OSBOOL useDerRules )
```

This constructor creates a time string from an `ASN1VisibleString` object.

It does not deep-copy the data; it just assigns the passed object to an internal reference variable. The object will then directly operate on the given data variable.

Parameters

<i>msgBuf</i>	Reference to an <code>OSRTMessage</code> buffer derived object (for example, <code>ASNBEREncodeBuffer</code>).
<i>buf</i>	Reference to a visible string object to hold the time data.
<i>useDerRules</i>	Use the Distinguished Encoding Rules (DER) to operate on this time value.

3.10.2.3 ASN1CTime() [3/5]

```
EXTRTMETHOD ASN1CTime::ASN1CTime (  
    OSRTContext & ctxt,  
    char *& buf,  
    int bufSize,  
    OSBOOL useDerRules )
```

This constructor creates a time string from buffer.

It does not deep-copy the data; it just assigns the passed array to an internal reference variable. The object will then directly operate on the given data variable.

Parameters

<i>ctxt</i>	Reference to an <code>OSRTContext</code> data structure.
<i>buf</i>	Reference to a pointer to a time string buffer.
<i>bufSize</i>	Size of buffer in bytes.
<i>useDerRules</i>	Use the Distinguished Encoding Rules (DER) to operate on this time value.

3.10.2.4 ASN1CTime() [4/5]

```
EXTRTMETHOD ASN1CTime::ASN1CTime (
    OSRTContext & ctxt,
    ASN1VisibleString & buf,
    OSBOOL useDerRules )
```

This constructor creates a time string from an ::ASN1VisibleString object.

It does not deep-copy the data; it just assigns the passed array to an internal reference variable. The object will then directly operate on the given data variable.

Parameters

<i>ctxt</i>	Reference to an OSRTContext data structure.
<i>buf</i>	Reference to a pointer to a time string buffer.
<i>useDerRules</i>	Use the Distinguished Encoding Rules (DER) to operate on this time value.

3.10.2.5 ASN1CTime() [5/5]

```
EXTRTMETHOD ASN1CTime::ASN1CTime (
    const ASN1CTime & original )
```

The copy constructor. This does not deep-copy the original value. Instead, it assigns references to the internal components.

Parameters

<i>original</i>	The original time string object value.
-----------------	--

3.10.2.6 ~ASN1CTime()

```
EXTRTMETHOD ASN1CTime::~ASN1CTime ( )
```

The destructor; this cleans up the [ASN1CTime](#) object.

3.10.3 Member Function Documentation

3.10.3.1 clear()

```
virtual EXTRIMETHOD void ASN1CTime::clear ( ) [virtual]
```

This method clears the time string.

Note the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters

-	none
---	------

Returns

- none

3.10.3.2 compileString()

```
virtual int ASN1CTime::compileString ( ) [protected], [pure virtual]
```

Compiles new time string according to X.680 and ISO 8601.

Returns

0 on success, or an error code if there was an error.

Implemented in [ASN1CGeneralizedTime](#), and [ASN1CUTCTime](#).

Referenced by [ASN1CGeneralizedTime::ASN1CGeneralizedTime\(\)](#).

3.10.3.3 equals()

```
virtual EXTRIMETHOD int ASN1CTime::equals (
    ASN1CTime & ) [virtual]
```

This method compares times.

3.10.3.4 getDay()

```
virtual EXTRIMETHOD int ASN1CTime::getDay ( ) [virtual]
```

This method returns the day of month number component of the time value.

The number of the first day in the month is 1; the number of the last day may be in the interval from 28 to 31. Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters

-	none
---	------

Returns

Day of month component (1 - 31) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.10.3.5 getDiff()

```
virtual EXTRIMETHOD int ASN1CTime::getDiff ( ) [virtual]
```

This method returns the difference between the time zone of the object and the Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive of negative time difference. Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters

-	none
---	------

Returns

The negative or positive minute component of the difference between the time zone of the object and the UTC time (-12*60 - +12*60) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.10.3.6 getDiffHour()

```
virtual EXTRIMETHOD int ASN1CTime::getDiffHour ( ) [virtual]
```

This method returns the hour component of the difference between the time zone of the object and the Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive of negative time difference. Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters

-	none
---	------

Returns

The negative or positive hour component of the difference between the time zone of the object and the UTC time (-12 - +12) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.10.3.7 getDiffMinute()

```
virtual EXTRMETHOD int ASN1CTime::getDiffMinute ( ) [virtual]
```

This method returns the minute component of the difference between the time zone of the object and the Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive of negative time difference. Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters

-	none
---	------

Returns

The negative or positive minute component of the difference between the time zone of the object and the UTC time (-59 - +59) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.10.3.8 getFraction()

```
virtual EXTRMETHOD int ASN1CTime::getFraction ( ) [virtual]
```

This method returns the second's decimal component of the time value.

Second's decimal fraction is represented by one or more digits from 0 to 9. Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters

-	none
---	------

Returns

Second's decimal fraction component (0 - 9) is returned if operation is successful. If the operation fails, a negative value is returned.

Reimplemented in [ASN1CUTCTime](#).

3.10.3.9 getFractionAsDouble()

```
virtual EXTRIMETHOD double ASN1CTime::getFractionAsDouble ( ) [virtual]
```

This method returns the second's decimal component of the time value. Second's fraction will be represented as double value more than 0 and less than 1.

Second's decimal fraction is represented by one or more digits from 0 to 9.

Returns

Second's decimal fraction component is returned if operation is successful. If the operation fails, a negative value is returned.

3.10.3.10 getFractionLen()

```
virtual EXTRIMETHOD int ASN1CTime::getFractionLen ( ) [virtual]
```

This method returns the number of digits in second's decimal component of the time value.

Returns

Second's decimal fraction's length is returned if operation is successful. If the operation fails, a negative value is returned.

3.10.3.11 getFractionStr()

```
virtual EXTRIMETHOD int ASN1CTime::getFractionStr (
    char *const pBuf,
    size_t bufSize ) [virtual]
```

This method returns the second's decimal component of the time value. Second's fraction will be represented as string w/o integer part and decimal point.

Returns

Length of the fraction string returned in pBuf, if operation is successful. If the operation fails, a negative value is returned.

3.10.3.12 getHour()

```
virtual EXTRIMETHOD int ASN1CTime::getHour ( ) [virtual]
```

This method returns the hour component of the time value.

As the ISO 8601 is based on the 24-hour timekeeping system, hours are represented by two-digit values from 00 to 23. Note that the return value may differ from different inherited [ASN1CTime](#) classes.

Parameters

-	none
---	------

Returns

Hour component (0 - 23) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.10.3.13 getMinute()

```
virtual EXTRIMETHOD int ASN1CTime::getMinute ( ) [virtual]
```

This method returns the minute component of the time value.

Minutes are represented by the two digits from 00 to 59. Note that the return value may differ from different inherited [ASN1CTime](#) classes.

Parameters

-	none
---	------

Returns

Minute component (0 - 59) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.10.3.14 getMonth()

```
virtual EXTRIMETHOD int ASN1CTime::getMonth ( ) [virtual]
```

This method returns the month number component of the time value.

The number of January is 1, February 2, ... up to December 12. You may also use enumerated valued for decoded months: `ASN1CTime::January`, `ASN1CTime::February`, etc. Also short aliases for months can be used: `ASN1CTime::Jan`, `ASN1CTime::Feb`, etc. Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters

-	none
---	------

Returns

Month component (1 - 12) is returned if operation is successful. If the operation fails, a negative value is returned.

3.10.3.15 getSecond()

```
virtual EXTRIMETHOD int ASN1CTime::getSecond ( ) [virtual]
```

This method returns the second component of the time value.

Seconds are represented by two digits from 00 to 59. Note that the return value may differ from different inherited [ASN1CTime](#) classes.

Parameters

-	none
---	------

Returns

Second component (0 - 59) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.10.3.16 getTime()

```
virtual EXTRIMETHOD time_t ASN1CTime::getTime ( ) [virtual]
```

This method converts the time string to a value of the built-in C type `time_t`.

The value is the number of seconds from January 1, 1970. If the time is represented as UTC time plus or minus a time difference, then the resulting value will be recalculated as local time. For example, if the time string is "19991208120000+0930", then this string will be converted to "19991208213000" and then converted to a `time_t` value. Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters

-	none
---	------

Returns

The time value, expressed as a number of seconds from January 1, 1970. If the operation fails, a negative value is returned.

3.10.3.17 getTimeString()

```
EXTRTMETHOD const char* ASN1CTime::getTimeString (
    char * pbuf,
    size_t bufsize )
```

This method copies the compiled time string into the given buffer.

Parameters

<i>pbuf</i>	The buffer to receive the time string.
<i>bufsize</i>	The size of the buffer.

Returns

A character string containing the compiled time string.

3.10.3.18 getTimeStringLen()

```
EXTRTMETHOD size_t ASN1CTime::getTimeStringLen ( )
```

This method returns the length of the compiled time string.

Returns

The length of the compiled time string.

3.10.3.19 getUTC()

```
virtual EXTRTMETHOD OSBOOL ASN1CTime::getUTC ( ) [virtual]
```

This method returns the UTC flag state.

If the UTC flag is TRUE, then the time is a UTC time and symbol Z is added at the end of the time string. Otherwise, it is local time.

Parameters

-	none
---	------

Returns

UTC flag state is returned.

3.10.3.20 `getYear()`

```
virtual EXTRIMETHOD int ASN1CTime::getYear ( ) [virtual]
```

This method returns the year component of the time value.

Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters

-	none
---	------

Returns

Year component (full 4 digits) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.10.3.21 `operator!=(())`

```
virtual EXTRIMETHOD OSBOOL ASN1CTime::operator!= (
    ASN1CTime & ) [virtual]
```

The [ASN1CTime](#) inequality test.

Returns

TRUE if the two times are not equal.

3.10.3.22 `operator<()`

```
virtual EXTRIMETHOD OSBOOL ASN1CTime::operator< (
    ASN1CTime & ) [virtual]
```

The [ASN1CTime](#) less-than test.

Returns

TRUE if this time is less than the given time.

3.10.3.23 operator<=()

```
virtual EXTRIMETHOD OSBOOL ASN1CTime::operator<= (
    ASN1CTime & ) [virtual]
```

The [ASN1CTime](#) less-equal test.

Returns

TRUE if this time is less-than or equal to the given time.

3.10.3.24 operator=()

```
EXTRIMETHOD const ASN1CTime& ASN1CTime::operator= (
    const ASN1CTime & )
```

This operator assigns this [ASN1CTime](#) object to the given [ASN1CTime](#) reference.

Returns

A constant reference to the given [ASN1CTime](#) object.

Referenced by `ASN1CGeneralizedTime::ASN1CGeneralizedTime()`.

3.10.3.25 operator==()

```
virtual EXTRIMETHOD OSBOOL ASN1CTime::operator== (
    ASN1CTime & ) [virtual]
```

The [ASN1CTime](#) equality test.

Returns

TRUE if the two times are equal.

3.10.3.26 operator>()

```
virtual EXTRIMETHOD OSBOOL ASN1CTime::operator> (
    ASN1CTime & ) [virtual]
```

The [ASN1CTime](#) greater-than test.

Returns

TRUE if this time is greater than the given time.

3.10.3.27 operator>=()

```
virtual EXTRIMETHOD OSBOOL ASN1CTime::operator>= (
    ASN1CTime & ) [virtual]
```

The [ASN1CTime](#) greater-equal test.

Returns

TRUE if this time is greater-than or equal to the given time.

3.10.3.28 parseString()

```
virtual EXTRIMETHOD int ASN1CTime::parseString (
    const char * string ) [virtual]
```

This method parses the given time string.

The string is expected to be in the ASN.1 value notation format for the given ASN.1 time string type. Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters

<i>string</i>	The time string value to be parsed.
---------------	-------------------------------------

Returns

Completion status of operation:

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

3.10.3.29 setDay()

```
virtual EXTRIMETHOD int ASN1CTime::setDay (
    short day_ ) [virtual]
```

This method sets the day of month number component of the time value.

The number of the first day in the month is 1; the number of the last day in the month may be in the interval from 28 to 31. Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters

<i>day</i> ↔	Day of month component (1 - 31).
—	

Returns

Completion status of operation:

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

3.10.3.30 setDER()

```
void ASN1CTime::setDER (
    OSBOOL bvalue ) [inline]
```

This method sets the 'use DER' flag which enforces the DER rules when time strings are constructed or parsed.

References operator!(), operator<(), operator<=(), operator==(), operator>(), and operator>=().

3.10.3.31 setDiff() [1/2]

```
virtual EXTRIMETHOD int ASN1CTime::setDiff (
    short dhour,
    short dminute ) [virtual]
```

This method sets the hours and the minute components of the difference between the time zone of the object and Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters

<i>dhour</i>	The negative or positive hour component of the difference between the time zone of the object and the UTC time (-12 - +12).
<i>dminute</i>	The negative or positive minute component of the difference between the time zone of the object and the UTC time (-59 - +59).

Returns

Completion status of operation:

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

3.10.3.32 setDiff() [2/2]

```
virtual EXTRIMETHOD int ASN1CTime::setDiff (  
    short inMinutes ) [virtual]
```

This method sets the difference between the time zone of the object and Coordinated Universal Time (UTC), in minutes.

The UTC time is the sum of the local time and a positive or negative time difference. Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters

<i>inMinutes</i>	The negative or positive difference, in minutes, between the time zone of the object and the UTC time (-12*60 - +12*60) is returned if the operation is successful.
------------------	---

Returns

Completion status of operation:

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

3.10.3.33 setDiffHour()

```
virtual EXTRIMETHOD int ASN1CTime::setDiffHour (  
    short dhour ) [virtual]
```

This method sets the hour component of the difference between the time zone of the object and the Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the action of this method may differ from different inherited [ASN1CTime](#) classes.

Parameters

<i>dhour</i>	The negative or positive hour component of the difference between the time zone of the object and the UTC time (-12 - +12) is returned if the operation is successful. If the operation fails, a negative value is returned.
--------------	--

Returns

Completion status of operation:

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

3.10.3.34 setFraction() [1/3]

```
virtual EXTRIMETHOD int ASN1CTime::setFraction (  
    int fraction,  
    int fracLen = -1 ) [virtual]
```

This method sets the second's decimal fraction component of the time value.

Second's decimal fraction is represented by one or more digits from 0 to 9. Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters

<i>fraction</i>	Second's decimal fraction component (0 - 9).
<i>fracLen</i>	Optional parameter specifies number of digits in second's fraction.

Returns

Completion status of operation:

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

3.10.3.35 setFraction() [2/3]

```
virtual EXTRIMETHOD int ASN1CTime::setFraction (  
    double frac,  
    int fracLen ) [virtual]
```

This method sets the second's decimal fraction component of the time value. Double value must be greater or equal 0 and less than 1.

Parameters

<i>frac</i>	Second's decimal fraction component.
<i>fracLen</i>	Specifies number of digits in second's fraction.

Returns

Completion status of operation:

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

3.10.3.36 setFraction() [3/3]

```
virtual EXTRIMETHOD int ASN1CTime::setFraction (  
    char const * frac ) [virtual]
```

This method sets the second's decimal fraction component of the time value. Double value must be greater or equal 0 and less than 1.

Parameters

<i>frac</i>	Second's decimal fraction component.
-------------	--------------------------------------

Returns

Completion status of operation:

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

3.10.3.37 setHour()

```
virtual EXTRIMETHOD int ASN1CTime::setHour (  
    short hour_ ) [virtual]
```

This method sets the hour component of the time value.

As the ISO 8601 is based on the 24-hour timekeeping system, hours are represented by two digits from 00 to 23. Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters

<i>hour</i> ↔	Hour component (0 - 23).
—	

Returns

Completion status of operation:

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

3.10.3.38 setMinute()

```
virtual EXTRIMETHOD int ASN1CTime::setMinute (  
    short minute_ ) [virtual]
```

This method sets the minute component of the time value.

Minutes are represented by two digits from 00 to 59. Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters

<i>minute</i> ↔	Minute component (0 - 59).
—	

Returns

Completion status of operation:

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

3.10.3.39 setMonth()

```
virtual EXTRIMETHOD int ASN1CTime::setMonth (  
    short month_ ) [virtual]
```

This method sets the month number component of the time value.

The number of January is 1, February 2, ..., through December 12. You may use enumerated values for months encoding: `ASN1CTime::January`, `ASN1CTime::February`, etc. Also you can use short aliases for months: `ASN1CTime::Jan`, `ASN1CTime::Feb`, etc. Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters

<i>month</i> ↔	Month component (1 - 12).
_	

Returns

Completion status of operation:

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

3.10.3.40 setSecond()

```
virtual EXTRIMETHOD int ASN1CTime::setSecond (  
    short second_ ) [virtual]
```

This method sets the second component of the time value.

Seconds are represented by two digits from 00 to 59. Note that the action of this method may differ from different inherited [ASN1CTime](#) classes.

Parameters

<i>second</i> ↔	Second component (0 - 59).
_	

Returns

Completion status of operation:

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

3.10.3.41 setTime()

```
virtual int ASN1CTime::setTime (  
    time_t time,  
    OSBOOL diffTime ) [pure virtual]
```

This converts the value of the C built-in type `time_t` to a time string.

The value is the number of seconds from January 1, 1970. Note that the action of this method may differ for different inherited [ASN1CTime](#) Classes.

Parameters

<i>time</i>	The time value, expressed as a number of seconds from January 1, 1970.
<i>diffTime</i>	TRUE means the difference between local time and UTC time will be calculated; in other case, only local time will be stored.

Returns

Completion status of operation:

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

Implemented in [ASN1CGeneralizedTime](#), and [ASN1CUTCTime](#).

Referenced by `ASN1CGeneralizedTime::ASN1CGeneralizedTime()`.

3.10.3.42 setUTC()

```
virtual EXTRIMETHOD int ASN1CTime::setUTC (
    OSBOOL utc ) [virtual]
```

This method sets the UTC flag state.

If the UTC flag is TRUE, then the time is a UTC time and symbol 'Z' is added to the end of the string. Otherwise, it is a local time.

Parameters

<i>utc</i>	UTC flag state.
------------	-----------------

Returns

Completion status of operation:

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

3.10.3.43 setYear()

```
virtual EXTRIMETHOD int ASN1CTime::setYear (
    short year_ ) [virtual]
```

This method sets the year component of the time value.

Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters

<code>year↔</code>	Year component (full 4 digits).
<code>-</code>	

Returns

Completion status of operation:

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

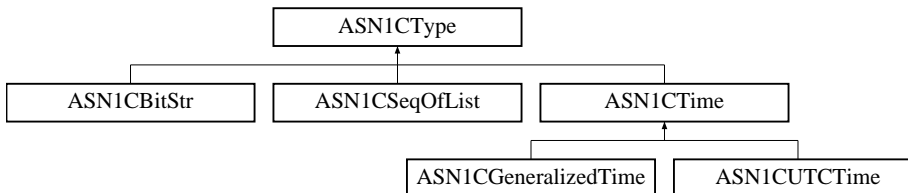
The documentation for this class was generated from the following file:

- [ASN1CTime.h](#)

3.11 ASN1CType Class Reference

```
#include <asn1CppType.h>
```

Inheritance diagram for ASN1CType:



Public Member Functions

- EXTRTMETHOD [ASN1CType](#) ([OSRTMessageBufferIF](#) &msgBuf)
- EXTRTMETHOD [ASN1CType](#) (const [ASN1CType](#) &orig)
- virtual [~ASN1CType](#) ()
- void [append](#) ([OSRTDList](#) &list, void *pdata)
- [OSRTCtxtPtr](#) [getContext](#) ()
- [OSCTXT](#) * [getCtxtPtr](#) ()
- char * [getErrorText](#) (char *textbuf=(char *) 0, [OSSIZE](#) bufsize=0)
- int [getStatus](#) () const
- void * [memAlloc](#) ([OSSIZE](#) numocts)
- void * [memAllocZ](#) ([OSSIZE](#) numocts)
- void [memFreeAll](#) ()
- void * [memRealloc](#) (void *ptr, [OSSIZE](#) numocts)
- void [memReset](#) ()
- void [memFreePtr](#) (void *ptr)
- void [printErrorInfo](#) ()
- void [resetError](#) ()
- [OSBOOL](#) [setDiag](#) ([OSBOOL](#) value)
- virtual EXTRTMETHOD int [Encode](#) ()
- virtual EXTRTMETHOD int [Decode](#) ([OSBOOL](#) free=FALSE)
- virtual int [EncodeTo](#) ([OSRTMessageBufferIF](#) &)
- virtual int [DecodeFrom](#) ([OSRTMessageBufferIF](#) &, [OSBOOL](#) free=TRUE)

Protected Member Functions

- EXTRTMETHOD [ASN1CType](#) ()
- EXTRTMETHOD [ASN1CType](#) ([OSRTCtctxPtr](#) &ctx)
- EXTRTMETHOD int **setMsgBuf** ([OSRTMessageBufferIF](#) &msgBuf, OSBOOL initBuf=FALSE)
- EXTRTMETHOD int **setRunTimeKey** (const OSOCTET *key, OSSIZE keylen)

Protected Attributes

- [OSRTCtctxPtr](#) mpContext
- [OSRTMessageBufferIF](#) * mpMsgBuf

3.11.1 Detailed Description

ASN1C control class base class. This is the main base class for all generated ASN1C_<name> control classes. It holds a variable of a generated data type as well as the associated message buffer or stream class to which a message will be encoded or from which a message will be decoded.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 [ASN1CType](#)() [1/4]

```
EXTRTMETHOD ASN1CType::ASN1CType ( ) [protected]
```

The default constructor sets the message pointer member variable to NULL and creates a new context object.

3.11.2.2 [ASN1CType](#)() [2/4]

```
EXTRTMETHOD ASN1CType::ASN1CType (
    OSRTCtctxPtr & ctx ) [protected]
```

This constructor sets the message pointer member variable to NULL and initializes the context object to point at the given context value.

Parameters

<i>ctx</i>	- Reference to a context object.
------------	----------------------------------

3.11.2.3 ASN1CType() [3/4]

```
EXTRTMETHOD ASN1CType::ASN1CType (
    OSRTMessageBufferIF & msgBuf )
```

This constructor sets the internal message buffer pointer to point at the given message buffer or stream object. The context is set to point at the context contained within the message buffer object. Thus, the message buffer and control class object share the context. It will not be released until both objects are destroyed.

Parameters

<i>msgBuf</i>	- Reference to a message buffer or stream object.
---------------	---

3.11.2.4 ASN1CType() [4/4]

```
EXTRTMETHOD ASN1CType::ASN1CType (
    const ASN1CType & orig )
```

The copy constructor sets the internal message buffer pointer and context to point at the message buffer and context from the original `ASN1CType` object.

Parameters

<i>orig</i>	- Reference to a message buffer or stream object.
-------------	---

3.11.2.5 ~ASN1CType()

```
virtual ASN1CType::~ASN1CType ( ) [inline], [virtual]
```

The virtual destructor does nothing. It is overridden by derived versions of this class.

3.11.3 Member Function Documentation

3.11.3.1 append()

```
void ASN1CType::append (
    OSRTDList & llist,
    void * pdata ) [inline]
```

The append method can be used to append an element to any linked list structure contained within the generated type.

Parameters

<i>llist</i>	Linked list structure.
<i>pdata</i>	Data record to be appended. Note that the pointer value is appended. The data is not copied.

3.11.3.2 Decode()

```
virtual EXTRIMETHOD int ASN1CType::Decode (
    OSBOOL free = FALSE ) [virtual]
```

The `Decode` method decodes the ASN.1 message described by the encapsulated message buffer object by invoking `DecodeFrom`.

Parameters

<i>free</i>	Indicates whether memory for existing objects should be freed prior to decoding as part of object (re)initialization.
-------------	---

Returns

Completion status of operation:

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

3.11.3.3 DecodeFrom()

```
virtual int ASN1CType::DecodeFrom (
    OSRTMessageBufferIF & ,
    OSBOOL free = TRUE ) [inline], [virtual]
```

The `DecodeFrom` method decodes an ASN.1 message from the given message buffer or stream argument.

Parameters

<i>msgBuf</i>	Message buffer or stream containing message to decode.
<i>free</i>	Indicates whether memory for existing objects should be freed prior to decoding as part of object (re)initialization.

Returns

Completion status of operation:

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

References ASN_E_NOTPDU, and OS_UNUSED_ARG.

3.11.3.4 Encode()

```
virtual EXTRMETHOD int ASN1CType::Encode ( ) [virtual]
```

The `Encode` method encodes an ASN.1 message using the encoding rules specified by the derived message buffer object.

Returns

Completion status of operation:

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

3.11.3.5 EncodeTo()

```
virtual int ASN1CType::EncodeTo (
    OSRTMessageBufferIF & ) [inline], [virtual]
```

The `EncodeTo` method encodes an ASN.1 message into the given message buffer or stream argument.

Parameters

<i>msgBuf</i>	Message buffer or stream to which the message is to be encoded.
---------------	---

Returns

Completion status of operation:

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

References ASN_E_NOTPDU.

3.11.3.6 getContext()

```
OSRtCtxPtr ASN1CType::getContext ( ) [inline]
```

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

Returns

Context smart pointer object.

3.11.3.7 getCtxtPtr()

```
OSCTXT* ASN1CType::getCtxtPtr ( ) [inline]
```

The getCtxtPtr method returns the underlying C runtime context. This context can be used in calls to C runtime functions.

References OSRTContext::getPtr(), and OSRtCtxPtr::isNull().

3.11.3.8 getErrorText()

```
char* ASN1CType::getErrorText (
    char * textbuf = (char *) 0,
    OSSIZE bufsize = 0 )
```

This method returns the error text associated with the last run-time error. If buffer pointer and buffer size are specified in parameters (not NULL) then error text will be copied in the passed buffer. Otherwise, this function allocates memory using the 'new' operator. It is the user's responsibility to free this memory using 'delete'.

Parameters

<i>textbuf</i>	A pointer to a destination buffer to hold the error text. If NULL, a dynamic buffer will be allocated.
<i>bufsize</i>	The size of the output buffer size.

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.

3.11.3.9 getStatus()

```
int ASN1CType::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. 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.

References `OSRTContext::getStatus()`, and `OSRTCtxtPtr::isNull()`.

3.11.3.10 memAlloc()

```
void* ASN1CType::memAlloc (
    OSSIZE 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 [ASN1CType](#) 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
----------------	---------------------------------------

Returns

Void pointer to allocated memory or NULL if insufficient memory was available to fulfill the request.

References `OSRTCtxtPtr::isNull()`, and `OSRTContext::memAlloc()`.

3.11.3.11 memAllocZ()

```
void* ASN1CType::memAllocZ (
    OSSIZE numocts ) [inline]
```

The `memAllocZ` method allocates memory using the C runtime memory management functions. The memory is tracked in the underlying context structure. When both this [ASN1CType](#) derived control class object and the message buffer object are destroyed, this memory will be freed. This memory will be zeroed out upon allocation.

Parameters

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

Returns

Void pointer to allocated memory or NULL if insufficient memory was available to fulfill the request.

References OSRTCtxtPtr::isNull(), and OSRTContext::memAllocZ().

3.11.3.12 memFreeAll()

```
void ASN1CType::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.

References OSRTCtxtPtr::isNull(), and OSRTContext::memFreeAll().

3.11.3.13 memFreePtr()

```
void ASN1CType::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>
------------	---

References OSRTCtxtPtr::isNull(), and OSRTContext::memFreePtr().

3.11.3.14 memRealloc()

```
void* ASN1CType::memRealloc (
    void * ptr,
    OSSIZE 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

References OSRTCtxtPtr::isNull(), and OSRTCContext::memRealloc().

3.11.3.15 memReset()

```
void ASN1CType::memReset ( ) [inline]
```

The memReset method resets dynamic memory using the C runtime memory management functions.

References OSRTCtxtPtr::isNull(), and OSRTCContext::memReset().

3.11.3.16 printErrorInfo()

```
void ASN1CType::printErrorInfo ( ) [inline]
```

The PrintErrorInfo method prints information on errors contained within the context.

References OSRTCtxtPtr::isNull(), and OSRTCContext::printErrorInfo().

3.11.3.17 resetError()

```
void ASN1CType::resetError ( ) [inline]
```

This method resets error status and stack trace. This method should be used to continue operations after recovering from the error.

References OSRTCtxtPtr::isNull(), and OSRTCContext::resetErrorInfo().

3.11.3.18 setDiag()

```
OSBOOL ASN1CType::setDiag (
    OSBOOL value ) [inline]
```

This method turns diagnostic tracing on or off.

Parameters

<i>value</i>	Boolean value; TRUE = turn tracing on.
--------------	--

Returns

Previous state.

3.11.3.19 setRunTimeKey()

```
EXTRTMETHOD int ASN1CType::setRunTimeKey (
    const OSOCTET * key,
    OSSIZE keylen ) [protected]
```

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

3.11.4 Member Data Documentation

3.11.4.1 mpContext

```
OSRTtxtPtr ASN1CType::mpContext [protected]
```

The mpContext member variable holds a reference-counted C runtime variable. This context is used in calls to all C runtime functions. The context pointed at by this smart-pointer object is shared with the message buffer object contained within this class.

3.11.4.2 mpMsgBuf

```
OSRTMessageBufferIF* ASN1CType::mpMsgBuf [protected]
```

The mpMsgBuf member variable is a pointer to a derived message buffer or stream class that will manage the ASN.1 message being encoded or decoded.

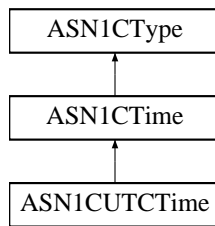
The documentation for this class was generated from the following file:

- [asn1CppTypes.h](#)

3.12 ASN1CUTCTime Class Reference

```
#include <ASN1CUTCTime.h>
```

Inheritance diagram for ASN1CUTCTime:



Public Member Functions

- EXTRTMETHOD [ASN1CUTCTime](#) ([OSRTMessageBufferIF](#) &msgBuf, char *&buf, int bufSize, OSBOOL useDerRules=FALSE)
- EXTRTMETHOD [ASN1CUTCTime](#) ([OSRTMessageBufferIF](#) &msgBuf, [ASN1UTCTime](#) &buf, OSBOOL useDerRules=FALSE)
- EXTRTMETHOD **ASN1CUTCTime** ([OSRTContext](#) &ctx, char *&buf, int bufSize, OSBOOL useDerRules=FALSE)
- EXTRTMETHOD **ASN1CUTCTime** ([OSRTContext](#) &ctx, [ASN1UTCTime](#) &buf, OSBOOL useDerRules=FALSE)
- **ASN1CUTCTime** (const [ASN1CUTCTime](#) &original)
- EXTRTMETHOD int [setTime](#) (time_t time, OSBOOL diffTime)
- const [ASN1CUTCTime](#) & **operator=** (const [ASN1CUTCTime](#) &tm)

Protected Member Functions

- virtual [ASN1TTime](#) & **getTimeObj** ()
- virtual const [ASN1TTime](#) & **getTimeObj** () const
- EXTRTMETHOD **ASN1CUTCTime** (char *&buf, int bufSize, OSBOOL useDerRules=FALSE)
- EXTRTMETHOD **ASN1CUTCTime** ([ASN1UTCTime](#) &buf, OSBOOL useDerRules=FALSE)
- EXTRTMETHOD int [compileString](#) ()
- EXTRTMETHOD int [getFraction](#) ()
- EXTRTMETHOD int **setFraction** (int fraction)

Protected Attributes

- [ASN1TUTCTime](#) `timeObj`

Additional Inherited Members

3.12.1 Detailed Description

ASN.1 UTCTime control class. The ASN1CUTTime class is derived from the [ASN1CTime](#) base class. It used as the bass class for generated control classes for the ASN.1 Universal Time ([UNIVERSAL 23] IMPLICIT VisibleString) type. This class provides utility methods for operating on the time information referenced by the generated class. This class can also be used inline to operate on the time within generated time string elements in a SEQUENCE, SET, or CHOICE construct. The string generally is encoded according to ISO 8601 format with some exceptions (see X.680).

3.12.2 Constructor & Destructor Documentation

3.12.2.1 ASN1CUTTime() [1/2]

```
EXTRTMETHOD ASN1CUTTime::ASN1CUTTime (
    OSRTMessageBufferIF & msgBuf,
    char *& buf,
    int bufSize,
    OSBOOL useDerRules = FALSE )
```

This constructor creates a time string from a buffer.

It does not deep-copy the data, it just assigns the passed array to an internal reference variable. The object will then directly operate on the given data variable.

Parameters

<i>msgBuf</i>	Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).
<i>buf</i>	Reference to a pointer to a time string buffer.
<i>bufSize</i>	Size of passed buffer, in bytes.
<i>useDerRules</i>	Use the Distinguished Encoding Rules to encode or decode the value,

3.12.2.2 ASN1CUTTime() [2/2]

```
EXTRTMETHOD ASN1CUTTime::ASN1CUTTime (
    OSRTMessageBufferIF & msgBuf,
```



```
ASN1UTCTime & buf,
OSBOOL useDerRules = FALSE )
```

This constructor creates a time string using the ASN1UTCTime argument. c The constructor does not deep-copy the variable, it assigns a reference to it to an internal variable. The object will then directly operate on the given data variable. This form of the constructor is used with a compiler-generated time string variable.

Parameters

<i>msgBuf</i>	Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).
<i>buf</i>	Reference to a time string structure.
<i>useDerRules</i>	Use the Distinguished Encoding Rules to encode or decode the value,

3.12.3 Member Function Documentation

3.12.3.1 compileString()

```
EXTRTMETHOD int ASN1CUTCTime::compileString ( ) [protected], [virtual]
```

Compiles new time string according to X.680 and ISO 8601.

Returns

0 on success, or an error code if there was an error.

Implements [ASN1CTime](#).

3.12.3.2 getFraction()

```
EXTRTMETHOD int ASN1CUTCTime::getFraction ( ) [protected], [virtual]
```

This method returns the second's decimal component of the time value.

Second's decimal fraction is represented by one or more digits from 0 to 9. Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters

-	none
---	------

Returns

Second's decimal fraction component (0 - 9) is returned if operation is successful. If the operation fails, a negative value is returned.

Reimplemented from [ASN1CTime](#).

3.12.3.3 setTime()

```
EXTRTMETHOD int ASN1CUTCTime::setTime (  
    time_t time,  
    OSBOOL diffTime ) [virtual]
```

Converts time_t to time string.

Parameters

<i>time</i>	time to convert,
<i>diffTime</i>	TRUE means the difference between local time and UTC will be calculated; in other case only local time will be stored.

Returns

Completion status of operation:

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

Implements [ASN1CTime](#).

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

- [ASN1CUTCTime.h](#)

3.13 Asn1ErrorHandler Class Reference

```
#include <asn1CppEvtHndlr.h>
```

Public Member Functions

- virtual int [error](#) (OSCTXT *pCtxt, ASN1CCB *pCCB, int stat)=0

Static Public Member Functions

- static EXTRTMETHOD int **invoke** (OSCTXT *pCtxt, ASN1CCB *pCCB, int stat)
- static EXTRTMETHOD int **invoke** (OSCTXT *pCtxt, OSOCTET *ptr, int len, int stat)
- static EXTRTMETHOD void **setErrorHandler** (OSCTXT *pCtxt, [Asn1ErrorHandler](#) *pHandler)

3.13.1 Detailed Description

Error handler base class. This is the base class from which user-defined error classes are derived. These classes can be used to provide fault-tolerance when parsing a message. The normal decoder behavior is to stop decoding when it encounters an error. An error handler can be used to ignore or take corrective action that will allow the decoding process to continue.

3.13.2 Member Function Documentation

3.13.2.1 error()

```
virtual int Asn1ErrorHandler::error (
    OSCTXT * pCtxt,
    ASN1CCB * pCCB,
    int stat ) [pure virtual]
```

The error handler callback method. This is the method that the user must override to provide customized error handling.

Parameters

<i>pCtxt</i>	- Pointer to a context block structure.
<i>pCCB</i>	- Pointer to a context control block structure.
<i>stat</i>	- The error status that caused the handler to be invoked.

Returns

- Corrected status. Set to 0 to cause decoding to continue or to a negative status code (most likely *stat*) to cause decoding to terminate.

3.13.2.2 setErrorHandler()

```
static EXTRTMETHOD void Asn1ErrorHandler::setErrorHandler (
    OSCTXT * pCtxt,
    Asn1ErrorHandler * pHandler ) [static]
```

This static method is called to set the error handler within the context structure. Note that unlike event handlers, only a single error handling object can be specified. This must be called by the user to specify the error handling object prior to execution of the main decode function..

Parameters

<i>pCtxt</i>	- Pointer to a context block structure.
<i>pHandler</i>	- Pointer to error handler object to register.

Referenced by ASN1MessageBuffer::setErrorHandler().

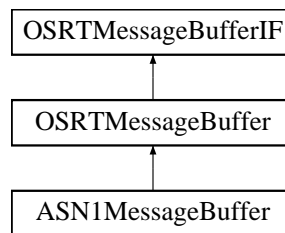
The documentation for this class was generated from the following file:

- [asn1CppEvtHndlr.h](#)

3.14 ASN1MessageBuffer Class Reference

```
#include <asn1CppTypes.h>
```

Inheritance diagram for ASN1MessageBuffer:



Public Member Functions

- virtual `~ASN1MessageBuffer ()`
- virtual void `addEventHandler (Asn1NamedEventHandler *pEventHandler)`
- ASN1BMPString * `CStringToBMPString (const char *cstring, ASN1BMPString *pBMPString, Asn116BitCharSet *pCharSet=0)`
- virtual void * `getAppInfo ()`
- virtual size_t `getMsgLen ()`
- virtual EXTRTMETHOD int `initBuffer (OSRTMEMBUF &membuf)`
- virtual EXTRTMETHOD int `initBuffer (OSUNICHAR *unistr)`
- virtual int `initBuffer (const OSUTF8CHAR *)`
- virtual OSBOOL `isA (Type)`
- virtual void `removeEventHandler (Asn1NamedEventHandler *pEventHandler)`
- virtual void `resetErrorInfo ()`
- virtual void `setAppInfo (void *)`
- virtual void `setErrorHandler (Asn1ErrorHandler *pErrorHandler)`
- EXTRTMETHOD int `setRunTimeKey (const OSOCTET *key, OSSIZE keylen)`
- size_t `getBitOffset ()`
- OSOCTET * `GetMsgCopy ()`
- const OSOCTET * `GetMsgPtr ()`
- void `PrintErrorInfo ()`

Protected Member Functions

- EXTRTMETHOD [ASN1MessageBuffer](#) (Type *bufferType*)
- EXTRTMETHOD [ASN1MessageBuffer](#) (Type *bufferType*, [OSRTContext](#) **pContext*)
- virtual int [setStatus](#) (int *stat*)

Additional Inherited Members

3.14.1 Detailed Description

Abstract ASN.1 message buffer base class. This class is used to manage a message buffer containing an ASN.1 message. For encoding, this is the buffer the message is being built in. For decoding, it is 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.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 [ASN1MessageBuffer\(\)](#) [1/2]

```
EXTRTMETHOD ASN1MessageBuffer::ASN1MessageBuffer (  
    Type bufferType ) [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, BEREncode).
-------------------	--

3.14.2.2 [ASN1MessageBuffer\(\)](#) [2/2]

```
EXTRTMETHOD ASN1MessageBuffer::ASN1MessageBuffer (  
    Type bufferType,  
    OSRTContext * pContext ) [protected]
```

This constructor sets the buffer class type and also a pointer to a context created by the user.

Parameters

<i>bufferType</i>	Type of message buffer that is being created (for example, BEREncode).
<i>pContext</i>	A pointer to an OSRTContext structure previously created by the user.

3.14.2.3 ~ASN1MessageBuffer()

```
virtual ASN1MessageBuffer::~~ASN1MessageBuffer ( ) [inline], [virtual]
```

The virtual destructor does nothing. It is overridden by derived versions of this class.

3.14.3 Member Function Documentation

3.14.3.1 addEventHandler()

```
virtual void ASN1MessageBuffer::addEventHandler (
    Asn1NamedEventHandler * pEventHandler ) [inline], [virtual]
```

The addEventHandler method is used to register a user-defined named event handler. Methods from within this handler will be invoked when this message buffer is used in the decoding of a message.

Parameters

<i>pEventHandler</i>	- Pointer to named event handler object to register.
----------------------	--

References `Asn1NamedEventHandler::addEventHandler()`, and `OSRTMessageBuffer::getCtxtPtr()`.

3.14.3.2 CStringToBMPString()

```
ASN1BMPString* ASN1MessageBuffer::CStringToBMPString (
    const char * cstring,
    ASN1BMPString * pBMPString,
    Asn116BitCharSet * pCharSet = 0 )
```

The CStringToBMPString method is a utility function for converting a null-terminated ASCII string into a BMP string. A BMP string contains 16-bit Unicode characters.

Parameters

<i>cstring</i>	- Null-terminated character string to convert
<i>pBMPString</i>	- Pointer to BMP string target variable
<i>pCharSet</i>	- Pointer to permitted alphabet character set. If provided, index to character within this set is returned.

3.14.3.3 getAppInfo()

```
virtual void* ASN1MessageBuffer::getAppInfo ( ) [inline], [virtual]
```

Returns a pointer to application-specific information block

Reimplemented from [OSRTMessageBuffer](#).

3.14.3.4 getBitOffset()

```
size_t ASN1MessageBuffer::getBitOffset ( ) [inline]
```

This method returns the current bit offset in the message buffer.

Parameters

-	none
---	------

References [OSRTMessageBuffer::getCtxtPtr\(\)](#), [OSRTMessageBuffer::getMsgCopy\(\)](#), [OSRTMessageBuffer::getMsgPtr\(\)](#), and [OSRTMessageBuffer::printErrorInfo\(\)](#).

3.14.3.5 getMsgLen()

```
virtual size_t ASN1MessageBuffer::getMsgLen ( ) [inline], [virtual]
```

This method returns the length in bytes of an encoded message.

References [OSRTMessageBuffer::initBuffer\(\)](#).

3.14.3.6 initBuffer() [1/2]

```
virtual EXTRIMETHOD int ASN1MessageBuffer::initBuffer (
    OSRTMEMBUF & membuf ) [virtual]
```

This version of the overloaded `initBuffer` method initializes the message buffer to point at the memory contained within the referenced `OSRTMEMBUF` object.

Parameters

<i>membuf</i>	OSRTMEMBUF memory buffer class object reference.
---------------	--

Returns

Completion status of operation:

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

3.14.3.7 initBuffer() [2/2]

```
virtual EXTRMETHOD int ASN1MessageBuffer::initBuffer (  
    OSUNICHAR * unistr ) [virtual]
```

This version of the overloaded initBuffer method initializes the message buffer to point at the given Unicode string. This is used mainly for XER (XML) message decoding.

Parameters

<i>unistr</i>	Pointer to a Unicode character string.
---------------	--

Returns

Completion status of operation:

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

3.14.3.8 isA()

```
virtual OSBOOL ASN1MessageBuffer::isA (  
    Type ) [inline], [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, XEREncode, XERDecode, XMLEncode, XMLDecode, Stream.
-------------------	--

Returns

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

Implements [OSRTMessageBufferIF](#).

3.14.3.9 removeEventHandler()

```
virtual void ASN1MessageBuffer::removeEventHandler (
    Asn1NamedEventHandler * pEventHandler ) [inline], [virtual]
```

The removeEventHandler method is used to de-register a used-defined named event handler.

Parameters

<i>pEventHandler</i>	- Pointer to named event handler object to de-register.
----------------------	---

References [OSRTMessageBuffer::getCxtPtr\(\)](#), and [Asn1NamedEventHandler::removeEventHandler\(\)](#).

3.14.3.10 resetErrorInfo()

```
virtual void ASN1MessageBuffer::resetErrorInfo ( ) [inline], [virtual]
```

The resetErrorInfo method resets information on errors contained within the context.

Reimplemented from [OSRTMessageBuffer](#).

References [OSRTMessageBuffer::resetErrorInfo\(\)](#).

3.14.3.11 setAppInfo()

```
virtual void ASN1MessageBuffer::setAppInfo (
    void * ) [inline], [virtual]
```

Sets the application-specific information block.

Reimplemented from [OSRTMessageBuffer](#).

3.14.3.12 setErrorHandler()

```
virtual void ASN1MessageBuffer::setErrorHandler (
    Asn1ErrorHandler * pErrorHandler ) [inline], [virtual]
```

The setErrorHandler method is used to register a used-defined error handler. Methods from within this handler will be invoked when an error occurs in decoding a message using this message buffer object.

Parameters

<i>pErrorHandler</i>	- Pointer to error handler object to register.
----------------------	--

References OSRTMessageBuffer::getCtxtPtr(), and Asn1ErrorHandler::setErrorHandler().

3.14.3.13 setRunTimeKey()

```
EXTRTMETHOD int ASN1MessageBuffer::setRunTimeKey (
    const OSOCTET * key,
    OSSIZE keylen )
```

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

3.14.3.14 setStatus()

```
virtual int ASN1MessageBuffer::setStatus (
    int stat ) [inline], [protected], [virtual]
```

This method sets error status to the context.

Returns

Error status value being set.

References OSRTMessageBuffer::getContext(), and OSRTContext::setStatus().

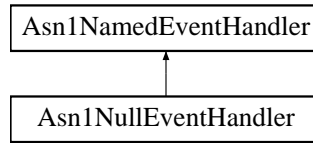
The documentation for this class was generated from the following file:

- [asn1CppTypes.h](#)

3.15 Asn1NamedEventHandler Class Reference

```
#include <asn1CppEvtHndlr.h>
```

Inheritance diagram for Asn1NamedEventHandler:



Public Member Functions

- virtual void [startElement](#) (const char *name, int index)=0
- virtual void [endElement](#) (const char *name, int index)=0
- virtual void [boolValue](#) (OSBOOL value)
- virtual void [intValue](#) (OSINT32 value)
- virtual void [uintValue](#) (OSUINT32 value)
- virtual void [int64Value](#) (OSINT64 value)
- virtual void [uint64Value](#) (OSUINT64 value)
- virtual void [bitStrValue](#) (OSUINT32 numbits, const OSOCTET *data)
- virtual void [octStrValue](#) (OSUINT32 numocts, const OSOCTET *data)
- virtual void [charStrValue](#) (const char *value)
- virtual void [charStrValue](#) (OSUINT32 nchars, const OSUTF8CHAR *value)
- virtual void [charStrValue](#) (OSUINT32 nchars, OSUNICHAR *data)
- virtual void [charStrValue](#) (OSUINT32 nchars, OS32BITCHAR *data)
- virtual void [nullValue](#) ()
- virtual void [oidValue](#) (OSUINT32 numSublds, OSUINT32 *pSublds)
- virtual void [realValue](#) (double value)
- virtual void [enumValue](#) (OSUINT32 value, const OSUTF8CHAR *text)
- virtual void [openTypeValue](#) (OSUINT32 numocts, const OSOCTET *data)

Static Public Member Functions

- static EXTRTMETHOD void [addEventHandler](#) (OSCTXT *pCtxt, [Asn1NamedEventHandler](#) *pHandler)
- static EXTRTMETHOD void [removeEventHandler](#) (OSCTXT *pCtxt, [Asn1NamedEventHandler](#) *pHandler)
- static EXTRTMETHOD void [invokeStartElement](#) (OSCTXT *pCtxt, const char *name, int index)
- static EXTRTMETHOD void [invokeEndElement](#) (OSCTXT *pCtxt, const char *name, int index)
- static EXTRTMETHOD void [invokeBoolValue](#) (OSCTXT *pCtxt, OSBOOL value)
- static EXTRTMETHOD void [invokeIntValue](#) (OSCTXT *pCtxt, OSINT32 value)
- static EXTRTMETHOD void [invokeUIntValue](#) (OSCTXT *pCtxt, OSUINT32 value)
- static EXTRTMETHOD void [invokeInt64Value](#) (OSCTXT *pCtxt, OSINT64 value)
- static EXTRTMETHOD void [invokeUInt64Value](#) (OSCTXT *pCtxt, OSUINT64 value)
- static EXTRTMETHOD void [invokeBitStrValue](#) (OSCTXT *pCtxt, OSUINT32 numbits, const OSOCTET *data)
- static EXTRTMETHOD void [invokeOctStrValue](#) (OSCTXT *pCtxt, OSUINT32 numocts, const OSOCTET *data)
- static EXTRTMETHOD void [invokeCharStrValue](#) (OSCTXT *pCtxt, const char *value)
- static EXTRTMETHOD void [invokeCharStrValue](#) (OSCTXT *pCtxt, OSUINT32 nchars, OSUNICHAR *data)

- static EXTRTMETHOD void [invokeCharStrValue](#) (OSCTXT *pCtxt, OSUINT32 nchars, OS32BITCHAR *data)
- static EXTRTMETHOD void [invokeCharStrValue](#) (OSCTXT *pCtxt, OSUINT32 nchars, const OSUTF8CHAR *data)
- static EXTRTMETHOD void [invokeNullValue](#) (OSCTXT *pCtxt)
- static EXTRTMETHOD void [invokeOidValue](#) (OSCTXT *pCtxt, OSUINT32 numSublds, OSUINT32 *pSublds)
- static EXTRTMETHOD void [invokeRealValue](#) (OSCTXT *pCtxt, double value)
- static EXTRTMETHOD void [invokeEnumValue](#) (OSCTXT *pCtxt, OSUINT32 value, const OSUTF8CHAR *text)
- static EXTRTMETHOD void [invokeOpenTypeValue](#) (OSCTXT *pCtxt, OSUINT32 numocts, const OSOCTET *data)

3.15.1 Detailed Description

Named event handler base class. This is the base class from which user-defined event handler classes are derived. These classes can be used to handle events during the parsing of an ASN.1 message. The event callback methods that can be implemented are startElement, endElement, and contents methods.

3.15.2 Member Function Documentation

3.15.2.1 addEventHandler()

```
static EXTRTMETHOD void Asn1NamedEventHandler::addEventHandler (
    OSCTXT * pCtxt,
    Asn1NamedEventHandler * pHandler ) [static]
```

This method is called to add a new event handler to the context event handler list. It is a static method.

Parameters

<i>pCtxt</i>	A pointer-to an ::OSCTXT data structure.
<i>pHandler</i>	A pointer to an Asn1NamedEventHandler .

Referenced by ASN1MessageBuffer::addEventHandler().

3.15.2.2 bitStrValue()

```
virtual void Asn1NamedEventHandler::bitStrValue (
    OSUINT32 numbits,
    const OSOCTET * data ) [inline], [virtual]
```

This method is invoked from within a decode function when a value of the BIT STRING ASN.1 type is parsed.

Parameters

<i>numbits</i>	- Number of bits in the parsed value.
<i>data</i>	- Pointer to a byte array that contains the bit string data.

Returns

- none

References OS_UNUSED_ARG.

3.15.2.3 boolValue()

```
virtual void Asn1NamedEventHandler::boolValue (
    OSBOOL value ) [inline], [virtual]
```

This method is invoked from within a decode function when a value of the BOOLEAN ASN.1 type is parsed.

Parameters

<i>value</i>	Parsed value.
--------------	---------------

Returns

- none

References OS_UNUSED_ARG.

3.15.2.4 charStrValue() [1/4]

```
virtual void Asn1NamedEventHandler::charStrValue (
    const char * value ) [inline], [virtual]
```

This method is invoked from within a decode function when a value of one of the 8-bit ASN.1 character string types is parsed.

Parameters

<i>value</i>	Null terminated character string value.
--------------	---

Returns

- none

References OS_UNUSED_ARG.

3.15.2.5 charStrValue() [2/4]

```
virtual void Asn1NamedEventHandler::charStrValue (  
    OSUINT32 nchars,  
    const OSUTF8CHAR * value ) [inline], [virtual]
```

This method is invoked from within a decode function when a value of a UTF-8 character string type is parsed.

Parameters

<i>nchars</i>	Number of characters in the parsed value.
<i>value</i>	A UTF-8 character string.

Returns

- none

References OS_UNUSED_ARG.

3.15.2.6 charStrValue() [3/4]

```
virtual void Asn1NamedEventHandler::charStrValue (  
    OSUINT32 nchars,  
    OSUNICHAR * data ) [inline], [virtual]
```

This method is invoked from within a decode function when a value of one of the 16-bit ASN.1 character string types is parsed.

This is used for the ASN.1 BmpString type.

Parameters

<i>nchars</i>	Number of characters in the parsed value.
<i>data</i>	Pointer to an array containing 16-bit values. These are represented using unsigned short integer values.

Returns

- none

References OS_UNUSED_ARG.

3.15.2.7 charStrValue() [4/4]

```
virtual void Asn1NamedEventHandler::charStrValue (
    OSUINT32 nchars,
    OS32BITCHAR * data ) [inline], [virtual]
```

This method is invoked from within a decode function when a value of one of the 32-bit ASN.1 character string types is parsed.

This is used for the ASN.1 UniversalString type.

Parameters

<i>nchars</i>	Number of characters in the parsed value.
<i>data</i>	Pointer to an array containing 32-bit values. Each 32-bit integer value is a universal character.

Returns

- none

References OS_UNUSED_ARG.

3.15.2.8 endElement()

```
virtual void Asn1NamedEventHandler::endElement (
    const char * name,
    int index ) [pure virtual]
```

This method is invoked from within a decode function when parsing is complete on an element of a SEQUENCE, SET, SEQUENCE OF, SET OF, or CHOICE construct.

Parameters

<i>name</i>	For SEQUENCE, SET, or CHOICE, this is the name of the element as defined in the ASN.1 definition. For SEQUENCE OF or SET OF, this is set to the name "element".
<i>index</i>	For SEQUENCE, SET, or CHOICE, this is not used and is set to the value -1. For SEQUENCE OF or SET OF, this contains the zero-based index of the element in the conceptual array associated with the construct.

Returns

- none

Implemented in [Asn1NullEventHandler](#).

3.15.2.9 enumValue()

```
virtual void Asn1NamedEventHandler::enumValue (
    OSUINT32 value,
    const OSUTF8CHAR * text ) [inline], [virtual]
```

This method is invoked from within a decode function when a value of the ENUMERATED ASN.1 type is parsed.

Parameters

<i>value</i>	- Parsed enumerated value
<i>text</i>	- Textual value of enumerated identifier

Returns

- none

References OS_UNUSED_ARG.

3.15.2.10 int64Value()

```
virtual void Asn1NamedEventHandler::int64Value (
    OSINT64 value ) [inline], [virtual]
```

This method is invoked from within a decode function when a value of the 64-bit INTEGER ASN.1 type is parsed.

Parameters

<i>value</i>	Parsed value.
--------------	---------------

Returns

- none

3.15.2.11 intValue()

```
virtual void Asn1NamedEventHandler::intValue (
    OSINT32 value ) [inline], [virtual]
```

This method is invoked from within a decode function when a value of the INTEGER ASN.1 type is parsed.

Parameters

<i>value</i>	Parsed value.
--------------	---------------

Returns

- none

References OS_UNUSED_ARG.

3.15.2.12 invokeBitStrValue()

```
static EXTRTMETHOD void Asn1NamedEventHandler::invokeBitStrValue (
    OSCTXT * pCtxt,
    OSUINT32 numbits,
    const OSOCTET * data ) [static]
```

This method is called by generated code to invoke the event handlers' bit string method. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>numbits</i>	The number of bits in the decoded bit string.
<i>data</i>	The decoded bit string data.

3.15.2.13 invokeBoolValue()

```
static EXTRTMETHOD void Asn1NamedEventHandler::invokeBoolValue (
    OSCTXT * pCtxt,
    OSBOOL value ) [static]
```

This method is called by generated code to invoke the event handlers' boolean method. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>value</i>	A decoded boolean value.

3.15.2.14 invokeCharStrValue() [1/4]

```
static EXTRIMETHOD void Asn1NamedEventHandler::invokeCharStrValue (  
    OSCTXT * pCtxt,  
    const char * value ) [static]
```

This method is called by generated code to invoke the event handlers' character string method. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>value</i>	The decoded character string.

3.15.2.15 invokeCharStrValue() [2/4]

```
static EXTRIMETHOD void Asn1NamedEventHandler::invokeCharStrValue (  
    OSCTXT * pCtxt,  
    OSUINT32 nchars,  
    OSUNICHAR * data ) [static]
```

This method is called by generated code to invoke the event handlers' 16-bit character string method. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>nchars</i>	The number of characters in the string.
<i>value</i>	The decoded 16-bit character string.

3.15.2.16 invokeCharStrValue() [3/4]

```
static EXTRIMETHOD void Asn1NamedEventHandler::invokeCharStrValue (  
    OSCTXT * pCtxt,
```

```

    OSUINT32 nchars,
    OS32BITCHAR * data ) [static]

```

This method is called by generated code to invoke the event handlers' 32-bit character string method. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>nchars</i>	The number of characters in the string.
<i>value</i>	The decoded character string.

3.15.2.17 invokeCharStrValue() [4/4]

```

static EXTRIMETHOD void Asn1NamedEventHandler::invokeCharStrValue (
    OSCTXT * pCtxt,
    OSUINT32 nchars,
    const OSUTF8CHAR * data ) [static]

```

This method is called by generated code to invoke the event handlers' UTF-8 character string method. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>nchars</i>	The number of characters in the string.
<i>value</i>	The decoded character string.

3.15.2.18 invokeEndElement()

```

static EXTRIMETHOD void Asn1NamedEventHandler::invokeEndElement (
    OSCTXT * pCtxt,
    const char * name,
    int index ) [static]

```

This method is called by generated code to invoke the event handlers' end element methods. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>name</i>	The name of the element.
<i>index</i>	The index of the element, if it is in a SEQUENCE or SET OF type.

3.15.2.19 invokeEnumValue()

```
static EXTRIMETHOD void Asn1NamedEventHandler::invokeEnumValue (
    OSCTXT * pCtxt,
    OSUINT32 value,
    const OSUTF8CHAR * text ) [static]
```

This method is called by generated code to invoke the event handlers' enumerated method. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>value</i>	A decoded 64-bit integer value.
<i>text</i>	The character string representation of the value.

3.15.2.20 invokeInt64Value()

```
static EXTRIMETHOD void Asn1NamedEventHandler::invokeInt64Value (
    OSCTXT * pCtxt,
    OSINT64 value ) [static]
```

This method is called by generated code to invoke the event handlers' 64-bit integer method. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>value</i>	A decoded 64-bit integer value.

3.15.2.21 invokeIntValue()

```
static EXTRIMETHOD void Asn1NamedEventHandler::invokeIntValue (
    OSCTXT * pCtxt,
    OSINT32 value ) [static]
```

This method is called by generated code to invoke the event handlers' integer method. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>value</i>	A decoded 32-bit integer value.

3.15.2.22 invokeNullValue()

```
static EXTRTMETHOD void Asn1NamedEventHandler::invokeNullValue (
    OSCTXT * pCtxt ) [static]
```

This method is called by generated code to invoke the event handlers' null method. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
--------------	--

3.15.2.23 invokeOctStrValue()

```
static EXTRTMETHOD void Asn1NamedEventHandler::invokeOctStrValue (
    OSCTXT * pCtxt,
    OSUINT32 numocts,
    const OSOCTET * data ) [static]
```

This method is called by generated code to invoke the event handlers' octet string method. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>numocts</i>	The number of octets in the decoded octet string.
<i>data</i>	The decoded octet string data.

3.15.2.24 invokeOidValue()

```
static EXTRTMETHOD void Asn1NamedEventHandler::invokeOidValue (
    OSCTXT * pCtxt,
    OSUINT32 numSubIds,
    OSUINT32 * pSubIds ) [static]
```

This method is called by generated code to invoke the event handlers' object identifier method. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>numSubIds</i>	The number of OID subids.
<i>value</i>	The decoded OID ids.

3.15.2.25 invokeOpenTypeValue()

```
static EXTRTMETHOD void Asn1NamedEventHandler::invokeOpenTypeValue (
    OSCTXT * pCtxt,
    OSUINT32 numocts,
    const OSOCTET * data ) [static]
```

This method is called by generated code to invoke the event handlers' open type method. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>numocts</i>	The number of octets in the open type.
<i>data</i>	The data octets that comprise the open type value.

3.15.2.26 invokeRealValue()

```
static EXTRTMETHOD void Asn1NamedEventHandler::invokeRealValue (
    OSCTXT * pCtxt,
    double value ) [static]
```

This method is called by generated code to invoke the event handlers' real method. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>value</i>	A decoded real value.

3.15.2.27 invokeStartElement()

```
static EXTRTMETHOD void Asn1NamedEventHandler::invokeStartElement (
    OSCTXT * pCtxt,
    const char * name,
    int index ) [static]
```

This method is called by generated code to invoke the event handlers' start element methods. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>name</i>	The name of the element.
<i>index</i>	The index of the element, if it is in a SEQUENCE or SET OF type.

3.15.2.28 invokeUInt64Value()

```
static EXTRTMETHOD void Asn1NamedEventHandler::invokeUInt64Value (
    OSCTXT * pCtxt,
    OSUINT64 value ) [static]
```

This method is called by generated code to invoke the event handlers' unsigned 64-bit integer method. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>value</i>	A decoded 64-bit integer value.

3.15.2.29 invokeUIntValue()

```
static EXTRTMETHOD void Asn1NamedEventHandler::invokeUIntValue (
    OSCTXT * pCtxt,
    OSUINT32 value ) [static]
```

This method is called by generated code to invoke the event handlers' unsigned integer method. It is static.

Parameters

<i>pCtxt</i>	A pointer to an ::OSCTXT data structure.
<i>value</i>	A decoded unsigned 32-bit integer value.

3.15.2.30 nullValue()

```
virtual void Asn1NamedEventHandler::nullValue ( ) [inline], [virtual]
```

This method is invoked from within a decode function when a value of the NULL ASN.1 type is parsed.

Parameters

-	none
---	------

Returns

- none

3.15.2.31 octStrValue()

```
virtual void Asn1NamedEventHandler::octStrValue (
    OSUINT32 numocts,
    const OSOCTET * data ) [inline], [virtual]
```

This method is invoked from within a decode function when a value of one of the OCTET STRING ASN.1 type is parsed.

Parameters

<i>numocts</i>	Number of octets in the parsed value.
<i>data</i>	Pointer to byte array containing the octet string data.

Returns

- none

References OS_UNUSED_ARG.

3.15.2.32 oidValue()

```
virtual void Asn1NamedEventHandler::oidValue (
    OSUINT32 numSubIds,
    OSUINT32 * pSubIds ) [inline], [virtual]
```

This method is invoked from within a decode function when a value the OBJECT IDENTIFIER ASN.1 type is parsed.

Parameters

<i>numSubIds</i>	Number of subidentifiers in the object identifier.
<i>pSubIds</i>	Pointer to array containing the subidentifier values.

Returns

-none

References OS_UNUSED_ARG.

3.15.2.33 openTypeValue()

```
virtual void Asn1NamedEventHandler::openTypeValue (
    OSUINT32 numocts,
    const OSOCTET * data ) [inline], [virtual]
```

This value is invoked from within a decode function when an ASN.1 open type is parsed.

Parameters

<i>numocts</i>	Number of octets in the parsed value.
<i>data</i>	Pointer to byet array contain in tencoded ASN.1 value.

Returns

- none

References OS_UNUSED_ARG.

3.15.2.34 realValue()

```
virtual void Asn1NamedEventHandler::realValue (
    double value ) [inline], [virtual]
```

This method is invoked from within a decode function when a value the REAL ASN.1 type is parsed.

Parameters

<i>value</i>	Parsed value.
--------------	---------------

Returns

- none

References OS_UNUSED_ARG.

3.15.2.35 removeEventHandler()

```
static EXTRIMETHOD void Asn1NamedEventHandler::removeEventHandler (
    OSCTXT * pCtxt,
    Asn1NamedEventHandler * pHandler ) [static]
```

This method removes the given event handler from the context event handler list. This method does not delete the handler, so memory allocated for it will need to be released elsewhere.

Parameters

<i>pCtxt</i>	A pointer-to an ::OSCTXT data structure.
<i>pHandler</i>	A pointer to an Asn1NamedEventHandler .

Referenced by [ASN1MessageBuffer::removeEventHandler\(\)](#).

3.15.2.36 startElement()

```
virtual void Asn1NamedEventHandler::startElement (
    const char * name,
    int index ) [pure virtual]
```

This method is invoked from within a decode function when an element of a SEQUENCE, SET, SEQUENCE OF, SET OF, or CHOICE construct is parsed.

Parameters

<i>name</i>	For SEQUENCE, SET, or CHOICE, this is the name of the element as defined in the ASN.1 definition. For SEQUENCE OF or SET OF, this is set to the name "element".
<i>index</i>	For SEQUENCE, SET, or CHOICE, this is not used and is set to the value -1. For SEQUENCE OF or SET OF, this contains the zero-based index of the element in the conceptual array associated with the construct.

Returns

- none

Implemented in [Asn1NullEventHandler](#).

3.15.2.37 uInt64Value()

```
virtual void Asn1NamedEventHandler::uInt64Value (
    OSUINT64 value ) [inline], [virtual]
```

This method is invoked from within a decode function when a value of the 64-bit INTEGER ASN.1 type is parsed.

Parameters

<i>value</i>	Parsed value.
--------------	---------------

Returns

- none

3.15.2.38 uIntValue()

```
virtual void Asn1NamedEventHandler::uIntValue (
    OSUINT32 value ) [inline], [virtual]
```

This method is invoked from within a decode function when a value of the INTEGER ASN.1 type is parsed.

In this case, constraints on the integer value forced the use of unsigned integer C type to represent the value.

Parameters

<i>value</i>	Parsed value.
--------------	---------------

Returns

- none

References OS_UNUSED_ARG.

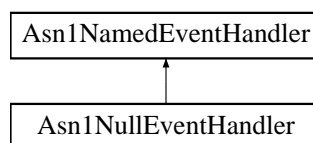
The documentation for this class was generated from the following file:

- [asn1CppEvtHndlr.h](#)

3.16 Asn1NullEventHandler Class Reference

```
#include <asn1CppEvtHndlr.h>
```

Inheritance diagram for Asn1NullEventHandler:



Public Member Functions

- virtual void [startElement](#) (const char *, int)
- virtual void [endElement](#) (const char *, int)

Additional Inherited Members

3.16.1 Detailed Description

The [Asn1NullEventHandler](#) contains a completely empty implementation of all user methods.

3.16.2 Member Function Documentation

3.16.2.1 endElement()

```
virtual void Asn1NullEventHandler::endElement (
    const char * ,
    int ) [inline], [virtual]
```

This endElement method does nothing.

Implements [Asn1NamedEventHandler](#).

3.16.2.2 startElement()

```
virtual void Asn1NullEventHandler::startElement (
    const char * ,
    int ) [inline], [virtual]
```

This startElement method does nothing.

Implements [Asn1NamedEventHandler](#).

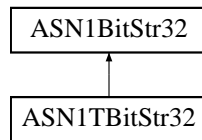
The documentation for this class was generated from the following file:

- [asn1CppEvtHndlr.h](#)

3.17 ASN1BitStr32 Struct Reference

```
#include <asn1CppTypes.h>
```

Inheritance diagram for ASN1BitStr32:



Public Member Functions

- [ASN1TBitStr32](#) ()
- [ASN1TBitStr32](#) (OSUINT32 _numbits, const OSOCTET *_data)
- [ASN1TBitStr32](#) (ASN1BitStr32 &_bs)

3.17.1 Detailed Description

Fixed-size bit string. This is the base class for generated C++ data type classes for sized BIT STRING's with size \leq 32 bits.

3.17.2 Constructor & Destructor Documentation

3.17.2.1 [ASN1TBitStr32\(\)](#) [1/3]

```
ASN1TBitStr32::ASN1TBitStr32 ( ) [inline]
```

The default constructor creates an empty bit string.

3.17.2.2 [ASN1TBitStr32\(\)](#) [2/3]

```
ASN1TBitStr32::ASN1TBitStr32 (
    OSUINT32 _numbits,
    const OSOCTET *_data ) [inline]
```

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

Parameters

<code>_numbits</code>	Number of bits in the bit string.
<code>_data</code>	The binary bit data values.

3.17.2.3 ASN1TBitStr32() [3/3]

```
ASN1TBitStr32::ASN1TBitStr32 (  
    ASN1TBitStr32 & _bs ) [inline]
```

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

Parameters

<code>_bs</code>	- C bit string structure.
------------------	---------------------------

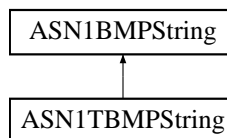
The documentation for this struct was generated from the following file:

- [asn1CppType.h](#)

3.18 ASN1TBMPString Struct Reference

```
#include <asn1CppType.h>
```

Inheritance diagram for ASN1TBMPString:



Public Member Functions

- [ASN1TBMPString \(\)](#)

3.18.1 Detailed Description

BMPString. This is the base class for generated C++ data type classes for BMPString values.

3.18.2 Constructor & Destructor Documentation

3.18.2.1 ASN1TBMPString()

```
ASN1TBMPString::ASN1TBMPString ( ) [inline]
```

The default constructor creates an empty BMPString value.

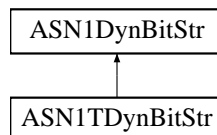
The documentation for this struct was generated from the following file:

- [asn1CppTypes.h](#)

3.19 ASN1TDynBitStr Struct Reference

```
#include <asn1CppTypes.h>
```

Inheritance diagram for ASN1TDynBitStr:



Public Member Functions

- [ASN1TDynBitStr](#) ()
- [ASN1TDynBitStr](#) (OSUINT32 _numbits, const OSOCTET *_data)
- [ASN1TDynBitStr](#) (ASN1DynBitStr &_bs)

3.19.1 Detailed Description

Dynamic bit string. This is the base class for generated C++ data type classes for unsized BIT STRING's.

3.19.2 Constructor & Destructor Documentation

3.19.2.1 ASN1TDynBitStr() [1/3]

```
ASN1TDynBitStr::ASN1TDynBitStr ( ) [inline]
```

The default constructor creates an empty bit string.

3.19.2.2 ASN1TDynBitStr() [2/3]

```
ASN1TDynBitStr::ASN1TDynBitStr (
    OSUINT32 _numbits,
    const OSOCTET * _data ) [inline]
```

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

Parameters

<code>_numbits</code>	- Number of bits in the bit string.
<code>_data</code>	- The binary bit data values.

3.19.2.3 ASN1TDynBitStr() [3/3]

```
ASN1TDynBitStr::ASN1TDynBitStr (  
    ASN1DynBitStr & _bs ) [inline]
```

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

Parameters

<code>_bs</code>	- C bit string structure.
------------------	---------------------------

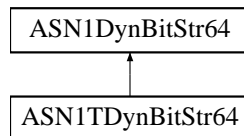
The documentation for this struct was generated from the following file:

- [asn1CppTypes.h](#)

3.20 ASN1TDynBitStr64 Struct Reference

```
#include <asn1CppTypes.h>
```

Inheritance diagram for ASN1TDynBitStr64:



Public Member Functions

- [ASN1TDynBitStr64 \(\)](#)
- [ASN1TDynBitStr64 \(OSSIZE _numbits, const OSOCTET *_data\)](#)
- [ASN1TDynBitStr64 \(ASN1DynBitStr64 &_bs\)](#)

3.20.1 Detailed Description

64-bit dynamic bit string.

3.20.2 Constructor & Destructor Documentation

3.20.2.1 ASN1TDynBitStr64() [1/3]

```
ASN1TDynBitStr64::ASN1TDynBitStr64 ( ) [inline]
```

The default constructor creates an empty bit string.

3.20.2.2 ASN1TDynBitStr64() [2/3]

```
ASN1TDynBitStr64::ASN1TDynBitStr64 (
    OSSIZE _numbits,
    const OSOCTET * _data ) [inline]
```

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

Parameters

<code>_numbits</code>	- Number of bits in the bit string.
<code>_data</code>	- The binary bit data values.

3.20.2.3 ASN1TDynBitStr64() [3/3]

```
ASN1TDynBitStr64::ASN1TDynBitStr64 (
    ASN1DynBitStr64 & _bs ) [inline]
```

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

Parameters

<code>_bs</code>	- C bit string structure.
------------------	---------------------------

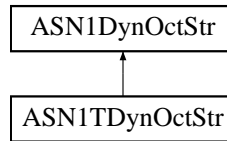
The documentation for this struct was generated from the following file:

- [asn1CppTypes.h](#)

3.21 ASN1TDynOctStr Struct Reference

```
#include <ASN1TOctStr.h>
```

Inheritance diagram for ASN1TDynOctStr:



Public Member Functions

- [ASN1TDynOctStr](#) ()
- [ASN1TDynOctStr](#) (OSUINT32 _numocts, const OSOCTET *_data)
- [ASN1TDynOctStr](#) (ASN1TDynOctStr &_os)
- [ASN1TDynOctStr](#) (const char *cstring)
- [ASN1TDynOctStr](#) & operator= (const char *cstring)
- EXTRTMETHOD [ASN1TDynOctStr](#) & operator= (const [ASN1TDynOctStr](#) &octet)
- EXTRTMETHOD const char * [toString](#) (OSCTXT *pctxt) const
- EXTRTMETHOD const char * [toHexString](#) (OSCTXT *pctxt) const
- EXTRTMETHOD int [nCompare](#) (OSUINT32 n, const [ASN1TDynOctStr](#) &o) const

3.21.1 Detailed Description

Dynamic octet string. This is the base class for generated C++ data type classes for unsized OCTET string's.

3.21.2 Constructor & Destructor Documentation

3.21.2.1 [ASN1TDynOctStr](#)() [1/4]

```
ASN1TDynOctStr::ASN1TDynOctStr ( ) [inline]
```

The default constructor creates an empty octet string.

3.21.2.2 [ASN1TDynOctStr](#)() [2/4]

```
ASN1TDynOctStr::ASN1TDynOctStr (
    OSUINT32 _numocts,
    const OSOCTET *_data ) [inline]
```

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

Parameters

<code>_numocts</code>	- Number of octet in the octet string.
<code>_data</code>	- The binary octet data values.

3.21.2.3 ASN1TDynOctStr() [3/4]

```
ASN1TDynOctStr::ASN1TDynOctStr (  
    ASN1DynOctStr & _os ) [inline]
```

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

Parameters

<code>_os</code>	- C octet string structure.
------------------	-----------------------------

3.21.2.4 ASN1TDynOctStr() [4/4]

```
ASN1TDynOctStr::ASN1TDynOctStr (  
    const char * cstring ) [inline]
```

This constructor initializes the octet string to contain the given data values. In this case, it is initializes the string to contain the characters in a null-terminated C character string.

Parameters

<code>cstring</code>	- C null-terminated string.
----------------------	-----------------------------

3.21.3 Member Function Documentation

3.21.3.1 nCompare()

```
EXTRIMETHOD int ASN1TDynOctStr::nCompare (  
    OSUINT32 n,  
    const ASN1TDynOctStr & o ) const
```

This method compares the first n octets of this octet string with the given octet string.

Parameters

<i>n</i>	- Number of octets to compare
<i>o</i>	- Octet string for comparison

Returns

- 0 if strings are equal, -1 if this octet string is less than the given string, +1 if this string > given string.

3.21.3.2 operator=() [1/2]

```
ASN1TDynOctStr& ASN1TDynOctStr::operator= (
    const char * cstring ) [inline]
```

This assignment operator sets the octet string to contain the characters in a null-terminated C character string. For example, `myOctStr = "a char string";`

Parameters

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

References `operator!=()`, `operator<()`, `operator<=()`, `operator==()`, `operator>()`, and `operator>=()`.

3.21.3.3 operator=() [2/2]

```
EXTRTMETHOD ASN1TDynOctStr& ASN1TDynOctStr::operator= (
    const ASN1TDynOctStr & octet )
```

This assignment operator sets the octet string to contain the characters from the given C++ octet string object.

Parameters

<i>octet</i>	- Octet string object reference
--------------	---------------------------------

3.21.3.4 toHexString()

```
EXTRTMETHOD const char* ASN1TDynOctStr::toHexString (
    OSCTXT * pctxt ) const
```

This method converts the binary octet string to a hexadecimal string representation.

Parameters

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

3.21.3.5 toString()

```
EXTRTMETHOD const char* ASN1TDynOctStr::toString (  
    OSCTXT * pctxt ) const
```

This method converts the binary octet string to a human-readable representation. The string is first checked to see if it contains all printable characters. If this is the case, the characters in the string are returned; otherwise, the string contents are converted into a hexadecimal character string.

Parameters

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

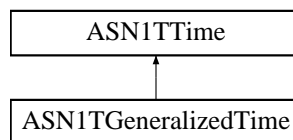
The documentation for this struct was generated from the following file:

- [ASN1TOctStr.h](#)

3.22 ASN1TGeneralizedTime Class Reference

```
#include <ASN1TTime.h>
```

Inheritance diagram for ASN1TGeneralizedTime:



Public Member Functions

- [ASN1TGeneralizedTime](#) ()
- EXTRTMETHOD [ASN1TGeneralizedTime](#) (const char *buf, OSBOOL useDerRules=FALSE)
- [ASN1TGeneralizedTime](#) (OSBOOL useDerRules)
- [ASN1TGeneralizedTime](#) (const [ASN1TGeneralizedTime](#) &original)
- EXTRTMETHOD int [getCentury](#) () const
- EXTRTMETHOD int [setCentury](#) (short century)
- EXTRTMETHOD int [setTime](#) (time_t time, OSBOOL diffTime)
- EXTRTMETHOD int [parseString](#) (const char *string)
- const [ASN1TGeneralizedTime](#) & **operator=** (const [ASN1TGeneralizedTime](#) &tm)
- EXTRTMETHOD int [compileString](#) (char *pbuf, size_t bufsize) const

Additional Inherited Members

3.22.1 Detailed Description

ASN.1 GeneralizedTime utility class. The [ASN1GeneralizedTime](#) class is derived from the [ASN1TTime](#) base class.

3.22.2 Constructor & Destructor Documentation

3.22.2.1 ASN1GeneralizedTime() [1/4]

```
ASN1GeneralizedTime::ASN1GeneralizedTime ( ) [inline]
```

A default constructor.

3.22.2.2 ASN1GeneralizedTime() [2/4]

```
EXTRTMETHOD ASN1GeneralizedTime::ASN1GeneralizedTime (
    const char * buf,
    OSBOOL useDerRules = FALSE )
```

This constructor creates a time object using the specified time string.

Parameters

<i>buf</i>	A pointer to the time string to be parsed.
<i>useDerRules</i>	An OSBOOL value.

3.22.2.3 ASN1GeneralizedTime() [3/4]

```
ASN1GeneralizedTime::ASN1GeneralizedTime (
    OSBOOL useDerRules ) [inline]
```

This constructor creates an empty time object.

Parameters

<i>useDerRules</i>	An OSBOOL value.
--------------------	------------------

3.22.2.4 ASN1TGeneralizedTime() [4/4]

```
ASN1TGeneralizedTime::ASN1TGeneralizedTime (  
    const ASN1TGeneralizedTime & original ) [inline]
```

A copy constructor.

References [ASN1TTime::compileString\(\)](#), [ASN1TTime::parseString\(\)](#), and [ASN1TTime::setTime\(\)](#).

3.22.3 Member Function Documentation

3.22.3.1 compileString()

```
EXTRTMETHOD int ASN1TGeneralizedTime::compileString (  
    char * pbuf,  
    size_t bufsize ) const [virtual]
```

Compiles new time string according X.680 and ISO 8601. Returns 0, if succeed, or error code, if error.

Parameters

<i>pbuf</i>	A pointer to destination buffer.
<i>bufsize</i>	A size of destination buffer.

Returns

Completion status of operation:

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

Implements [ASN1TTime](#).

3.22.3.2 getCentury()

```
EXTRTMETHOD int ASN1TGeneralizedTime::getCentury ( ) const
```

This method returns the century part (first two digits) of the year component of the time value.

Returns

Century part (first two digits) of the year component is returned if the operation is successful. If the operation fails, one of the negative status codes is returned.

3.22.3.3 parseString()

```
EXTRTMETHOD int ASN1TGeneralizedTime::parseString (
    const char * string ) [virtual]
```

Parses sting.

Returns

Completion status of operation:

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

Implements [ASN1TTime](#).

3.22.3.4 setCentury()

```
EXTRTMETHOD int ASN1TGeneralizedTime::setCentury (
    short century )
```

This method sets the centry part (first two digits) of the year component of the time value.

Parameters

<i>century</i>	Century part (first two digits) of the year component.
----------------	--

Returns

Completion status of operation:

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

3.22.3.5 setTime()

```
EXTRTMETHOD int ASN1TGeneralizedTime::setTime (
    time_t time,
    OSBOOL diffTime ) [virtual]
```

This converts the value of the C built-in type `time_t` to a time string.

The value is the number of seconds from January 1, 1970.

Parameters

<i>time</i>	The time value, expressed as a number of seconds from January 1, 1970.
<i>diffTime</i>	TRUE means the difference between local time and UTC time will be calculated; in other case, only local time will be stored.

Returns

Completion status of operation:

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

Implements [ASN1TTime](#).

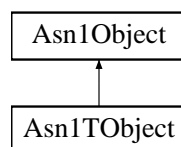
The documentation for this class was generated from the following file:

- [ASN1TTime.h](#)

3.23 Asn1TObject Struct Reference

```
#include <asn1CppType.h>
```

Inheritance diagram for `Asn1TObject`:



Public Member Functions

- [Asn1TObject \(\)](#)

3.23.1 Detailed Description

Open type with table constraint. This is the base class for generated C++ data type classes for open type values with table constraints. It is only used when the `-tables` compiler command line option is specified.

3.23.2 Constructor & Destructor Documentation

3.23.2.1 Asn1TObject()

```
Asn1TObject::Asn1TObject ( ) [inline]
```

The default constructor creates an empty object value.

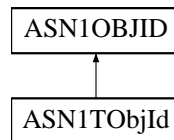
The documentation for this struct was generated from the following file:

- [asn1CppType.h](#)

3.24 ASN1TObjId Struct Reference

```
#include <ASN1TObjId.h>
```

Inheritance diagram for ASN1TObjId:



Public Member Functions

- [ASN1TObjId](#) ()
- virtual EXTRTMETHOD [~ASN1TObjId](#) ()
- EXTRTMETHOD [ASN1TObjId](#) (OSOCKET _numids, const OSUINT32 *_subids)
- EXTRTMETHOD [ASN1TObjId](#) (const ASN1OBJID &oid)
- EXTRTMETHOD [ASN1TObjId](#) (const [ASN1TObjId](#) &oid)
- EXTRTMETHOD [ASN1TObjId](#) (const char *dotted_oid_string)
- EXTRTMETHOD [ASN1TObjId](#) & [operator=](#) (const char *dotted_oid_string)
- EXTRTMETHOD void [operator=](#) (const ASN1OBJID &rhs)
- EXTRTMETHOD void [operator=](#) (const [ASN1TObjId](#) &rhs)
- EXTRTMETHOD [ASN1TObjId](#) & [operator+=](#) (const char *dotted_oid_string)
- EXTRTMETHOD [ASN1TObjId](#) & [operator+=](#) (const OSUINT32 i)
- EXTRTMETHOD [ASN1TObjId](#) & [operator+=](#) (const [ASN1TObjId](#) &o)
- EXTRTMETHOD const char * [toString](#) (OSCTXT *pctx) const
- EXTRTMETHOD void [set_data](#) (const OSUINT32 *raw_oid, OSUINT32 oid_len)
- EXTRTMETHOD int [nCompare](#) (const OSUINT32 n, const [ASN1TObjId](#) &o) const
- EXTRTMETHOD int [RnCompare](#) (const OSUINT32 n, const [ASN1TObjId](#) &o) const
- EXTRTMETHOD void [trim](#) (const OSUINT32 n)

3.24.1 Detailed Description

Object identifier. This is the base class for generated C++ data type classes for object identifier values.

3.24.2 Constructor & Destructor Documentation

3.24.2.1 ASN1ObjId() [1/5]

```
ASN1ObjId::ASN1ObjId ( ) [inline]
```

The default constructor creates an empty object identifier value.

References operator!=(), operator+(), operator<(), operator<=(), operator==(), operator>(), and operator>=().

3.24.2.2 ~ASN1ObjId()

```
virtual EXTRTMETHOD ASN1ObjId::~~ASN1ObjId ( ) [virtual]
```

The Virtual Destructor

3.24.2.3 ASN1ObjId() [2/5]

```
EXTRTMETHOD ASN1ObjId::ASN1ObjId (
    OSOCKET _numids,
    const OSUINT32 * _subids )
```

This constructor initializes the object identifier to contain the given data values.

Parameters

<code>_numids</code>	- Number of subidentifiers in the OID.
<code>_subids</code>	- Array of subidentifier values.

3.24.2.4 ASN1ObjId() [3/5]

```
EXTRTMETHOD ASN1ObjId::ASN1ObjId (
    const ASN1OBJID & oid )
```

This constructor initializes the object identifier to contain the given data values. This can be used to set the value to a compiler-generated OID value.

Parameters

<i>oid</i>	- C object identifier value.
------------	------------------------------

3.24.2.5 ASN1ObjId() [4/5]

```
EXTRTMETHOD ASN1ObjId::ASN1ObjId (  
    const ASN1ObjId & oid )
```

The copy constructor.

Parameters

<i>oid</i>	- C++ object identifier value.
------------	--------------------------------

3.24.2.6 ASN1ObjId() [5/5]

```
EXTRTMETHOD ASN1ObjId::ASN1ObjId (  
    const char * dotted_oid_string )
```

Construct an OID from a dotted string.

Parameters

<i>dotted_oid_string</i>	- for example "1.3.1.6.1.10"
--------------------------	------------------------------

3.24.3 Member Function Documentation

3.24.3.1 nCompare()

```
EXTRTMETHOD int ASN1ObjId::nCompare (  
    const OSUINT32 n,  
    const ASN1ObjId & o ) const
```

Compare the first n sub-ids(left to right) of two object identifiers.

Parameters

<i>n</i>	- Number of subid values to compare.
<i>o</i>	- OID to compare this OID with.

Returns

- 0 if OID's are equal, -1 if this OID less than given OID, +1 if this OID > given OID.

3.24.3.2 operator+=() [1/3]

```
EXTRTMETHOD ASN1ObjId& ASN1ObjId::operator+= (
    const char * dotted_oid_string )
```

Overloaded += operator. This operator allows subidentifiers in the form of a dotted OID string ("n.n.n") to be appended to an existing OID object.

Parameters

<i>dotted_oid_string</i>	- C++ object identifier value.
--------------------------	--------------------------------

Returns

- True if values are equal.

3.24.3.3 operator+=() [2/3]

```
EXTRTMETHOD ASN1ObjId& ASN1ObjId::operator+= (
    const OSUINT32 i )
```

Overloaded += operator. This operator allows a single subidentifier in the form of an integer value to be appended to an existing OID object.

Parameters

<i>i</i>	- Subidentifier to append.
----------	----------------------------

Returns

- True if values are equal.

3.24.3.4 operator+=() [3/3]

```
EXTRTMETHOD ASN1ObjId& ASN1ObjId::operator+= (
    const ASN1ObjId & o )
```

Overloaded += operator. This operator allows one object identifier to be appended to another object identifier.

Parameters

<i>o</i>	- C++ object identifier value.
----------	--------------------------------

Returns

- True if values are equal.

3.24.3.5 operator=() [1/3]

```
EXTRTMETHOD ASN1ObjId& ASN1ObjId::operator= (
    const char * dotted_oid_string )
```

Assignment from a string.

Parameters

<i>dotted_oid_string</i>	- New value (for example "1.3.6.1.6.0");
--------------------------	--

3.24.3.6 operator=() [2/3]

```
EXTRTMETHOD void ASN1ObjId::operator= (
    const ASN1OBJID & rhs )
```

This assignment operator sets the object identifier to contain the OID in the given C structure. This can be used to set the value to a compiler-generated OID value.

Parameters

<i>rhs</i>	- C object identifier value.
------------	------------------------------

3.24.3.7 operator=() [3/3]

```
EXTRTMETHOD void ASN1ObjId::operator= (
    const ASN1ObjId & rhs )
```

This assignment operator sets the object identifier to contain the OID in the given C++ structure.

Parameters

<i>rhs</i>	- C++ object identifier value.
------------	--------------------------------

3.24.3.8 RnCompare()

```
EXTRTMETHOD int ASN1ObjId::RnCompare (
    const OSUINT32 n,
    const ASN1ObjId & o ) const
```

Compare the last n sub-ids(right to left) of two object identifiers.

Parameters

<i>n</i>	- Number of subid values to compare.
<i>o</i>	- OID to compare this OID with.

Returns

- 0 if OID's are equal, -1 if this OID less than given OID, +1 if this OID > given OID.

3.24.3.9 set_data()

```
EXTRTMETHOD void ASN1ObjId::set_data (
    const OSUINT32 * raw_oid,
    OSUINT32 oid_len )
```

Sets the data of an object identifier using a pointer and a length.

Parameters

<i>raw_oid</i>	- Pointer to an array of subidentifier values.
<i>oid_len</i>	- Number of subids in the array,

3.24.3.10 toString()

```
EXTRTMETHOD const char* ASN1ObjId::toString (
    OSCTXT * pctxt ) const
```

Get a printable ASCII string of a part of the value.

Parameters

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

Returns

- Dotted OID string (for example "3.6.1.6")

3.24.3.11 trim()

```
EXTRTMETHOD void ASN1ObjId::trim (
    const OSUINT32 n )
```

Trim the given number of rightmost sub elements from this OID.

Parameters

<i>n</i>	- number of subids to trim from OID
----------	-------------------------------------

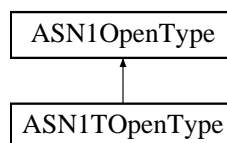
The documentation for this struct was generated from the following file:

- [ASN1ObjId.h](#)

3.25 ASN1OpenType Struct Reference

```
#include <asn1CppTypes.h>
```

Inheritance diagram for ASN1OpenType:



Public Member Functions

- [ASN1OpenType \(\)](#)

3.25.1 Detailed Description

Open type. This is the base class for generated C++ data type classes for open type values.

3.25.2 Constructor & Destructor Documentation

3.25.2.1 ASN1OpenType()

```
ASN1OpenType::ASN1OpenType ( ) [inline]
```

The default constructor creates an empty open type value.

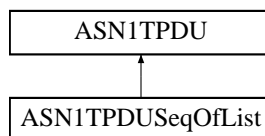
The documentation for this struct was generated from the following file:

- [asn1CppType.h](#)

3.26 ASN1TPDU Struct Reference

```
#include <asn1CppType.h>
```

Inheritance diagram for ASN1TPDU:



Public Member Functions

- void [setContext](#) ([OSRTContext](#) *ctxt)
- virtual [~ASN1TPDU](#) ()

Protected Attributes

- [OSRTCtxtPtr mpContext](#)

3.26.1 Detailed Description

Base class for PDU types. This class is used as the base class for all compiler-generated PDU types. Control classes do not inherit from this class.

3.26.2 Constructor & Destructor Documentation

3.26.2.1 ~ASN1TPDU()

```
virtual ASN1TPDU::~~ASN1TPDU ( ) [inline], [virtual]
```

The virtual destructor does nothing. It is overridden by derived versions of this class.

3.26.3 Member Function Documentation

3.26.3.1 setContext()

```
void ASN1TPDU::setContext (
    OSRContext * ctxt ) [inline]
```

The setContext method allows the context member variable to be set. It is invoked in compiler-generated control class decode and copy methods. This method is invoked to prevent memory freeing of decoded or copied data after a control class or message buffer object goes out of scope. Also, if context is set to [ASN1TPDU](#) then generated destructor of inherited ASN1T_<type> class will invoke generated free routines. Note, it is not obligatory to call generated free routines unless a series of messages is being decoded or control class and message buffer objects go out of scope somewhere. The destructor of the control class or message buffer class will free all dynamically allocated memory. Thus, if performance is a main issue, "setContext (NULL)" may be called after Decode method call. In this case destructor of ASN1T_<type> will do nothing.

Parameters

<i>ctxt</i>	A pointer to reference counted ASN.1 context class instance.
-------------	--

References OSRContextPtr::isNull().

3.26.4 Member Data Documentation

3.26.4.1 mpContext

```
OSRTxtPtr ASN1TPDU::mpContext [protected]
```

The mpContext member variable holds a smart-pointer to the current context variable. This ensures an instance of this PDU type will persist if the control class and message buffer classes used to decode or copy the message are destroyed.

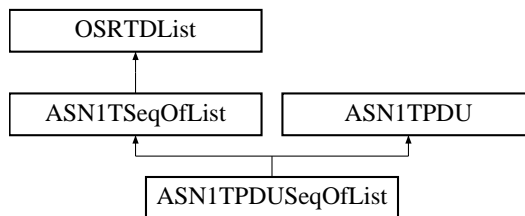
The documentation for this struct was generated from the following file:

- [asn1CppType.h](#)

3.27 ASN1TPDUSeqOfList Struct Reference

```
#include <asn1CppType.h>
```

Inheritance diagram for ASN1TPDUSeqOfList:



Public Member Functions

- [ASN1TPDUSeqOfList \(\)](#)

Additional Inherited Members

3.27.1 Detailed Description

SEQUENCE OF element holder (PDU). This class is used as the base class for compiler-generated SEQUENCE OF linked-list types. In this case, the type has also been determined to be a PDU.

3.27.2 Constructor & Destructor Documentation

3.27.2.1 ASN1TPDUSeqOfList()

```
ASN1TPDUSeqOfList::ASN1TPDUSeqOfList ( ) [inline]
```

The default constructor creates an empty list.

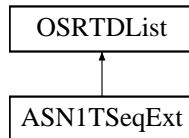
The documentation for this struct was generated from the following file:

- [asn1CppTypes.h](#)

3.28 ASN1TSeqExt Struct Reference

```
#include <asn1CppTypes.h>
```

Inheritance diagram for ASN1TSeqExt:



Public Member Functions

- [ASN1TSeqExt \(\)](#)

3.28.1 Detailed Description

SEQUENCE or SET extension element holder. This is used for the /c extElem1 open extension element in extensible SEQUENCE or SET constructs.

3.28.2 Constructor & Destructor Documentation

3.28.2.1 ASN1TSeqExt()

```
ASN1TSeqExt::ASN1TSeqExt ( ) [inline]
```

The default constructor creates an empty open extension element.

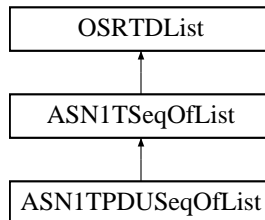
The documentation for this struct was generated from the following file:

- [asn1CppTypes.h](#)

3.29 ASN1TSeqOfList Struct Reference

```
#include <asn1CppTypes.h>
```

Inheritance diagram for ASN1TSeqOfList:



Public Member Functions

- [ASN1TSeqOfList \(\)](#)

3.29.1 Detailed Description

SEQUENCE OF element holder. This class is used as the base class for compiler-generated SEQUENCE OF linked-list types.

3.29.2 Constructor & Destructor Documentation

3.29.2.1 ASN1TSeqOfList()

```
ASN1TSeqOfList::ASN1TSeqOfList ( ) [inline]
```

The default constructor creates an empty list.

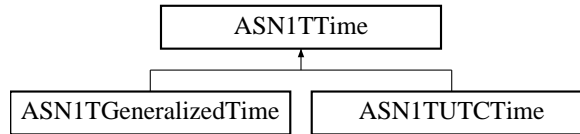
The documentation for this struct was generated from the following file:

- [asn1CppTypes.h](#)

3.30 ASN1Time Class Reference

```
#include <ASN1Time.h>
```

Inheritance diagram for ASN1Time:



Public Types

- enum {
January = 1, **Jan** = 1, **February** = 2, **Feb** = 2,
March = 3, **Mar** = 3, **April** = 4, **Apr** = 4,
May = 5, **June** = 6, **Jun** = 6, **July** = 7,
Jul = 7, **August** = 8, **Aug** = 8, **September** = 9,
Sep = 9, **October** = 10, **Oct** = 10, **November** = 11,
Nov = 11, **December** = 12, **Dec** = 12 }

Public Member Functions

- EXTRTMETHOD [ASN1Time](#) ()
- EXTRTMETHOD [ASN1Time](#) (OSBOOL useDerRules)
- EXTRTMETHOD [ASN1Time](#) (const [ASN1Time](#) &original)
- virtual EXTRTMETHOD [~ASN1Time](#) ()
- virtual EXTRTMETHOD int [getYear](#) () const
- virtual EXTRTMETHOD int [getMonth](#) () const
- virtual EXTRTMETHOD int [getDay](#) () const
- virtual EXTRTMETHOD int [getHour](#) () const
- virtual EXTRTMETHOD int [getMinute](#) () const
- virtual EXTRTMETHOD int [getSecond](#) () const
- virtual EXTRTMETHOD int [getFraction](#) () const
- virtual EXTRTMETHOD double [getFractionAsDouble](#) () const
- virtual EXTRTMETHOD int [getFractionStr](#) (char *const pBuf, size_t bufSize) const
- virtual EXTRTMETHOD int [getFractionLen](#) () const
- virtual EXTRTMETHOD int [getDiffHour](#) () const
- virtual EXTRTMETHOD int [getDiffMinute](#) () const
- virtual EXTRTMETHOD int [getDiff](#) () const
- virtual EXTRTMETHOD OSBOOL [getUTC](#) () const
- virtual EXTRTMETHOD time_t [getTime](#) () const
- void [setDER](#) (OSBOOL bvalue)
- virtual EXTRTMETHOD int [setUTC](#) (OSBOOL utc)
- virtual EXTRTMETHOD int [setYear](#) (short year_)
- virtual EXTRTMETHOD int [setMonth](#) (short month_)
- virtual EXTRTMETHOD int [setDay](#) (short day_)

- virtual EXTRTMETHOD int [setHour](#) (short hour_)
- virtual EXTRTMETHOD int [setMinute](#) (short minute_)
- virtual EXTRTMETHOD int [setSecond](#) (short second_)
- virtual EXTRTMETHOD int [setFraction](#) (int fraction, int fracLen=-1)
- virtual EXTRTMETHOD int [setFraction](#) (double frac, int fracLen)
- virtual EXTRTMETHOD int [setFraction](#) (char const *frac)
- virtual int [setTime](#) (time_t time, OSBOOL diffTime)=0
- virtual EXTRTMETHOD int [setDiffHour](#) (short dhour)
- virtual EXTRTMETHOD int [setDiff](#) (short dhour, short dminute)
- virtual EXTRTMETHOD int [setDiff](#) (short inMinutes)
- virtual int [parseString](#) (const char *string)=0
- virtual EXTRTMETHOD void [clear](#) ()
- virtual int [compileString](#) (char *pbuf, size_t bufsize) const =0
- virtual EXTRTMETHOD int [equals](#) (const [ASN1TTime](#) &) const
- EXTRTMETHOD const char * [toString](#) (char *pbuf, size_t bufsize) const
- EXTRTMETHOD const char * [toString](#) (OSCTXT *pctxt) const
- EXTRTMETHOD const char * [toString](#) () const
- EXTRTMETHOD const [ASN1TTime](#) & **operator=** (const [ASN1TTime](#) &)
- virtual EXTRTMETHOD OSBOOL **operator==** (const [ASN1TTime](#) &) const
- virtual EXTRTMETHOD OSBOOL **operator!=** (const [ASN1TTime](#) &) const
- virtual EXTRTMETHOD OSBOOL **operator>** (const [ASN1TTime](#) &) const
- virtual EXTRTMETHOD OSBOOL **operator<** (const [ASN1TTime](#) &) const
- virtual EXTRTMETHOD OSBOOL **operator>=** (const [ASN1TTime](#) &) const
- virtual EXTRTMETHOD OSBOOL **operator<=** (const [ASN1TTime](#) &) const

Public Attributes

- short [mYear](#)
- short [mMonth](#)
- short [mDay](#)
- short [mHour](#)
- short [mMinute](#)
- short [mSecond](#)
- short [mDiffHour](#)
- short [mDiffMin](#)
- int [mSecFraction](#)
- int [mSecFracLen](#)
- int [mStatus](#)
- OSBOOL [mbUtcFlag](#)
- OSBOOL [mbDerRules](#)

Protected Member Functions

- EXTRTMETHOD void **privateInit** ()
- EXTRTMETHOD int **getDaysNum** () const
- EXTRTMETHOD long **getMillisNum** () const
- int **ncharsToInt** (const char *str, OSSIZE nchars, int &value)

Static Protected Member Functions

- static EXTRTMETHOD int **checkDate** (int day, int month, int year)
- static EXTRTMETHOD void **addMilliseconds** (int deltaMs, short &year, short &month, short &day, short &hour, short &minute, short &second, int &secFraction, int secFracLen)
- static EXTRTMETHOD void **addDays** (int deltaDays, short &year, short &month, short &day)
- static EXTRTMETHOD short **daysInMonth** (int i)
- static EXTRTMETHOD int **daysAfterMonth** (int i)

3.30.1 Detailed Description

ASN.1 Time utility base class.

3.30.2 Constructor & Destructor Documentation

3.30.2.1 ASN1TTime() [1/3]

```
EXTRTMETHOD ASN1TTime::ASN1TTime ( )
```

This constructor creates an empty time class.

3.30.2.2 ASN1TTime() [2/3]

```
EXTRTMETHOD ASN1TTime::ASN1TTime (
    OSBOOL useDerRules )
```

This constructor creates an empty time class.

Parameters

<i>useDerRules</i>	Use the Distinguished Encoding Rules (DER) to operate on this time value.
--------------------	---

3.30.2.3 ASN1TTime() [3/3]

```
EXTRTMETHOD ASN1TTime::ASN1TTime (
    const ASN1TTime & original )
```

The copy constructor.

Parameters

<i>original</i>	The original time string object value.
-----------------	--

3.30.2.4 ~ASN1TTime()

```
virtual EXTRIMETHOD ASN1TTime::~~ASN1TTime ( ) [virtual]
```

The destructor.

3.30.3 Member Function Documentation

3.30.3.1 clear()

```
virtual EXTRIMETHOD void ASN1TTime::clear ( ) [virtual]
```

This method clears the time object.

Note the action of this method may differ for different inherited [ASN1TTime](#) classes.

Reimplemented in [ASN1TUTCTime](#).

Referenced by [ASN1TUTCTime::ASN1TUTCTime\(\)](#).

3.30.3.2 compileString()

```
virtual int ASN1TTime::compileString (
    char * pbuf,
    size_t bufsize ) const [pure virtual]
```

Compiles new time string according X.680 and ISO 8601. Returns 0, if succeed, or error code, if error.

Parameters

<i>pbuf</i>	A pointer to destination buffer.
<i>bufsize</i>	A size of destination buffer.

Returns

Completion status of operation:

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

Implemented in [ASN1TUTCTime](#), and [ASN1TGeneralizedTime](#).

Referenced by [ASN1TGeneralizedTime::ASN1TGeneralizedTime\(\)](#), and [ASN1TUTCTime::ASN1TUTCTime\(\)](#).

3.30.3.3 equals()

```
virtual EXTRIMETHOD int ASN1TTime::equals (
    const ASN1TTime & ) const [virtual]
```

This method compares times.

3.30.3.4 getDay()

```
virtual EXTRIMETHOD int ASN1TTime::getDay ( ) const [virtual]
```

This method returns the day of month number component of the time value.

The number of the first day in the month is 1; the number of the last day may be in the interval from 28 to 31. Note that the return value may differ for different inherited [ASN1TTime](#) classes.

Returns

Day of month component (1 - 31) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.30.3.5 getDiff()

```
virtual EXTRIMETHOD int ASN1TTime::getDiff ( ) const [virtual]
```

This method returns the difference between the time zone of the object and the Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the return value may differ for different inherited [ASN1TTime](#) classes.

Returns

The negative or positive minute component of the difference between the time zone of the object and the UTC time (-12*60 - +12*60) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.30.3.6 `getDiffHour()`

```
virtual EXTRIMETHOD int ASN1TTime::getDiffHour ( ) const [virtual]
```

This method returns the hour component of the difference between the time zone of the object and the Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the return value may differ for different inherited [ASN1TTime](#) classes.

Returns

The negative or positive hour component of the difference between the time zone of the object and the UTC time (-12 - +12) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.30.3.7 `getDiffMinute()`

```
virtual EXTRIMETHOD int ASN1TTime::getDiffMinute ( ) const [virtual]
```

This method returns the minute component of the difference between the time zone of the object and the Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the return value may differ for different inherited [ASN1TTime](#) classes.

Returns

The negative or positive minute component of the difference between the time zone of the object and the UTC time (-59 - +59) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.30.3.8 `getFraction()`

```
virtual EXTRIMETHOD int ASN1TTime::getFraction ( ) const [virtual]
```

This method returns the second's decimal component of the time value.

Second's decimal fraction is represented by one or more digits from 0 to 9.

Returns

Second's decimal fraction component is returned if operation is successful. If the operation fails, a negative value is returned.

Reimplemented in [ASN1TUTCTime](#).

Referenced by `ASN1TUTCTime::ASN1TUTCTime()`.

3.30.3.9 getFractionAsDouble()

```
virtual EXTRIMETHOD double ASN1Time::getFractionAsDouble ( ) const [virtual]
```

This method returns the second's decimal component of the time value. Second's fraction will be represented as double value more than 0 and less than 1.

Second's decimal fraction is represented by one or more digits from 0 to 9.

Returns

Second's decimal fraction component is returned if operation is successful. If the operation fails, a negative value is returned.

3.30.3.10 getFractionLen()

```
virtual EXTRIMETHOD int ASN1Time::getFractionLen ( ) const [virtual]
```

This method returns the number of digits in second's decimal component of the time value.

Returns

Second's decimal fraction's length is returned if operation is successful. If the operation fails, a negative value is returned.

3.30.3.11 getFractionStr()

```
virtual EXTRIMETHOD int ASN1Time::getFractionStr (
    char *const pBuf,
    size_t bufSize ) const [virtual]
```

This method returns the second's decimal component of the time value. Second's fraction will be represented as string w/o integer part and decimal point.

Returns

Length of the fraction string returned in pBuf, if operation is successful. If the operation fails, a negative value is returned.

3.30.3.12 getHour()

```
virtual EXTRIMETHOD int ASN1TTime::getHour ( ) const [virtual]
```

This method returns the hour component of the time value.

As the ISO 8601 is based on the 24-hour timekeeping system, hours are represented by two-digit values from 00 to 23. Note that the return value may differ from different inherited [ASN1TTime](#) classes.

Returns

Hour component (0 - 23) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.30.3.13 getMinute()

```
virtual EXTRIMETHOD int ASN1TTime::getMinute ( ) const [virtual]
```

This method returns the minute component of the time value.

Minutes are represented by the two digits from 00 to 59. Note that the return value may differ from different inherited [ASN1TTime](#) classes.

Returns

Minute component (0 - 59) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.30.3.14 getMonth()

```
virtual EXTRIMETHOD int ASN1TTime::getMonth ( ) const [virtual]
```

This method returns the month number component of the time value.

The number of January is 1, February 2, ... up to December 12. You may also use enumerated valued for decoded months: `ASN1TTime::January`, `ASN1TTime::February`, etc. Also short aliases for months can be used: `ASN1TTime::Jan`, `ASN1TTime::Feb`, etc. Note that the return value may differ for different inherited [ASN1TTime](#) classes.

Returns

Month component (1 - 12) is returned if operation is successful. If the operation fails, a negative value is returned.

3.30.3.15 getSecond()

```
virtual EXTRIMETHOD int ASN1Time::getSecond ( ) const [virtual]
```

This method returns the second component of the time value.

Seconds are represented by two digits from 00 to 59. Note that the return value may differ from different inherited [ASN1Time](#) classes.

Returns

Second component (0 - 59) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.30.3.16 getTime()

```
virtual EXTRIMETHOD time_t ASN1Time::getTime ( ) const [virtual]
```

This method converts the time string to a value of the built-in C type `time_t`.

The value is the number of seconds from January 1, 1970. If the time is represented as UTC time plus or minus a time difference, then the resulting value will be recalculated as local time. For example, if the time string is "19991208120000+0930", then this string will be converted to "19991208213000" and then converted to a `time_t` value. Note that the return value may differ for different inherited [ASN1Time](#) classes.

Returns

The time value, expressed as a number of seconds from January 1, 1970. If the operation fails, a negative value is returned.

3.30.3.17 getUTC()

```
virtual EXTRIMETHOD OSBOOL ASN1Time::getUTC ( ) const [virtual]
```

This method returns the UTC flag state.

If the UTC flag is TRUE, then the time is a UTC time and symbol Z is added at the end of the time string. Otherwise, it is local time.

Returns

UTC flag state is returned.

3.30.3.18 getYear()

```
virtual EXTRIMETHOD int ASN1Time::getYear ( ) const [virtual]
```

This method returns the year component of the time value.

Note that the return value may differ for different inherited [ASN1Time](#) classes.

Returns

Year component (full 4 digits) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.30.3.19 parseString()

```
virtual int ASN1Time::parseString (
    const char * string ) [pure virtual]
```

This method parses the given time string.

The string is expected to be in the ASN.1 value notation format for the given ASN.1 time string type. Note that the action of this method may differ for different inherited [ASN1Time](#) classes.

Parameters

<i>string</i>	The time string value to be parsed.
---------------	-------------------------------------

Returns

Completion status of operation:

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

Implemented in [ASN1TUTCTime](#), and [ASN1TGeneralizedTime](#).

Referenced by [ASN1TGeneralizedTime::ASN1TGeneralizedTime\(\)](#), and [ASN1TUTCTime::ASN1TUTCTime\(\)](#).

3.30.3.20 setDay()

```
virtual EXTRIMETHOD int ASN1Time::setDay (
    short day_ ) [virtual]
```

This method sets the day of month number component of the time value.

The number of the first day in the month is 1; the number of the last day in the month may be in the interval from 28 to 31. Note that the action of this method may differ for different inherited [ASN1Time](#) classes.

Parameters

<i>day</i> ↔	Day of month component (1 - 31).
—	

Returns

Completion status of operation:

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

3.30.3.21 setDER()

```
void ASN1TTime::setDER (
    OSBOOL bvalue ) [inline]
```

This method sets the 'use DER' flag which enforces the DER rules when time strings are constructed or parsed.

References operator!(), operator<(), operator<=(), operator==(), operator>(), and operator>=().

3.30.3.22 setDiff() [1/2]

```
virtual EXTRIMETHOD int ASN1TTime::setDiff (
    short dhour,
    short dminute ) [virtual]
```

This method sets the hours and the minute components of the difference between the time zone of the object and Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the action of this method may differ for different inherited [ASN1TTime](#) classes.

Parameters

<i>dhour</i>	The negative or positive hour component of the difference between the time zone of the object and the UTC time (-12 - +12).
<i>dminute</i>	The negative or positive minute component of the difference between the time zone of the object and the UTC time (-59 - +59).

Returns

Completion status of operation:

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

3.30.3.23 setDiff() [2/2]

```
virtual EXTRIMETHOD int ASN1TTime::setDiff (
    short inMinutes ) [virtual]
```

This method sets the difference between the time zone of the object and Coordinated Universal Time (UTC), in minutes.

The UTC time is the sum of the local time and a positive or negative time difference. Note that the action of this method may differ for different inherited [ASN1TTime](#) classes.

Parameters

<i>inMinutes</i>	The negative or positive difference, in minutes, between the time zone of the object and the UTC time (-12*60 - +12*60) is returned if the operation is successful.
------------------	---

Returns

Completion status of operation:

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

3.30.3.24 setDiffHour()

```
virtual EXTRIMETHOD int ASN1TTime::setDiffHour (
    short dhour ) [virtual]
```

This method sets the hour component of the difference between the time zone of the object and the Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the action of this method may differ from different inherited [ASN1TTime](#) classes.

Parameters

<i>dhour</i>	The negative or positive hour component of the difference between the time zone of the object and the UTC time (-12 - +12) is returned if the operation is successful. If the operation fails, a negative value is returned.
--------------	--

Returns

Completion status of operation:

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

3.30.3.25 setFraction() [1/3]

```
virtual EXTRIMETHOD int ASN1TTime::setFraction (  
    int fraction,  
    int fracLen = -1 ) [virtual]
```

This method sets the second's decimal fraction component of the time value.

Second's decimal fraction is represented by one or more digits from 0 to 9. Note that the action of this method may differ for different inherited [ASN1TTime](#) classes.

Parameters

<i>fraction</i>	Second's decimal fraction component (0 - 9).
<i>fracLen</i>	Optional parameter specifies number of digits in second's fraction.

Returns

Completion status of operation:

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

Reimplemented in [ASN1TUTCTime](#).

Referenced by `ASN1TUTCTime::ASN1TUTCTime()`.

3.30.3.26 setFraction() [2/3]

```
virtual EXTRIMETHOD int ASN1TTime::setFraction (  
    double frac,  
    int fracLen ) [virtual]
```

This method sets the second's decimal fraction component of the time value. Double value must be greater or equal 0 and less than 1.

Parameters

<i>frac</i>	Second's decimal fraction component.
<i>fracLen</i>	Specifies number of digits in second's fraction.

Returns

Completion status of operation:

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

3.30.3.27 setFraction() [3/3]

```
virtual EXTRIMETHOD int ASN1Time::setFraction (  
    char const * frac ) [virtual]
```

This method sets the second's decimal fraction component of the time value. Double value must be greater or equal 0 and less than 1.

Parameters

<i>frac</i>	Second's decimal fraction component.
-------------	--------------------------------------

Returns

Completion status of operation:

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

3.30.3.28 setHour()

```
virtual EXTRIMETHOD int ASN1Time::setHour (  
    short hour_ ) [virtual]
```

This method sets the hour component of the time value.

As the ISO 8601 is based on the 24-hour timekeeping system, hours are represented by two digits from 00 to 23. Note that the action of this method may differ for different inherited [ASN1Time](#) classes.

Parameters

<i>hour</i> ↔	Hour component (0 - 23).
—	

Returns

Completion status of operation:

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

3.30.3.29 setMinute()

```
virtual EXTRIMETHOD int ASN1TTime::setMinute (  
    short minute_ ) [virtual]
```

This method sets the minute component of the time value.

Minutes are represented by two digits from 00 to 59. Note that the action of this method may differ for different inherited [ASN1TTime](#) classes.

Parameters

<i>minute</i> ↔	Minute component (0 - 59).
—	

Returns

Completion status of operation:

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

3.30.3.30 setMonth()

```
virtual EXTRIMETHOD int ASN1TTime::setMonth (  
    short month_ ) [virtual]
```

This method sets the month number component of the time value.

The number of January is 1, February 2, ..., through December 12. You may use enumerated values for months encoding: `ASN1TTime::January`, `ASN1TTime::February`, etc. Also you can use short aliases for months: `ASN1TTime↔::Jan`, `ASN1TTime↔::Feb`, etc. Note that the action of this method may differ for different inherited [ASN1TTime](#) classes.

Parameters

<i>month</i> ↔	Month component (1 - 12).
—	

Returns

Completion status of operation:

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

3.30.3.31 setSecond()

```
virtual EXTRIMETHOD int ASN1TTime::setSecond (  
    short second_ ) [virtual]
```

This method sets the second component of the time value.

Seconds are represented by two digits from 00 to 59. Note that the action of this method may differ from different inherited [ASN1TTime](#) classes.

Parameters

<i>second</i> ↔	Second component (0 - 59).
—	

Returns

Completion status of operation:

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

3.30.3.32 setTime()

```
virtual int ASN1TTime::setTime (  
    time_t time,  
    OSBOOL diffTime ) [pure virtual]
```

This converts the value of the C built-in type `time_t` to a time string.

The value is the number of seconds from January 1, 1970. Note that the action of this method may differ for different inherited [ASN1TTime](#) Classes.

Parameters

<i>time</i>	The time value, expressed as a number of seconds from January 1, 1970.
<i>diffTime</i>	TRUE means the difference between local time and UTC time will be calculated; in other case, only local time will be stored.

Returns

Completion status of operation:

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

Implemented in [ASN1TUTCTime](#), and [ASN1TGeneralizedTime](#).

Referenced by [ASN1TGeneralizedTime::ASN1TGeneralizedTime\(\)](#), and [ASN1TUTCTime::ASN1TUTCTime\(\)](#).

3.30.3.33 setUTC()

```
virtual EXTRIMETHOD int ASN1TTime::setUTC (
    OSBOOL utc ) [virtual]
```

This method sets the UTC flag state.

If the UTC flag is TRUE, then the time is a UTC time and symbol 'Z' is added to the end of the string. Otherwise, it is a local time.

Parameters

<i>utc</i>	UTC flag state.
------------	-----------------

Returns

Completion status of operation:

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

Reimplemented in [ASN1TUTCTime](#).

Referenced by [ASN1TUTCTime::ASN1TUTCTime\(\)](#).

3.30.3.34 setYear()

```
virtual EXTRTMETHOD int ASN1TTime::setYear (
    short year_ ) [virtual]
```

This method sets the year component of the time value.

Note that the action of this method may differ for different inherited [ASN1TTime](#) classes.

Parameters

<i>year</i> ↔	Year component (full 4 digits).
—	

Returns

Completion status of operation:

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

Reimplemented in [ASN1TUTCTime](#).

Referenced by `ASN1TUTCTime::ASN1TUTCTime()`.

3.30.3.35 toString() [1/3]

```
EXTRTMETHOD const char* ASN1TTime::toString (
    char * pbuf,
    size_t bufsize ) const
```

Get a printable ASCII string of the time value into the specified buffer.

Parameters

<i>pbuf</i>	Pointer to a destination buffer.
<i>bufsize</i>	Size of destination buffer.

Returns

Compiled time string. NULL, if error occurs.

3.30.3.36 toString() [2/3]

```
EXTRMETHOD const char* ASN1Time::toString (
    OSCTXT * pctx ) const
```

Get a printable ASCII string of the time value.

Parameters

<i>pctx</i>	Pointer to a context structure.
-------------	---------------------------------

Returns

Compiled time string. NULL, if error occurs.

3.30.3.37 toString() [3/3]

```
EXTRMETHOD const char* ASN1Time::toString ( ) const
```

Get a printable ASCII string of the time value. Memory will be allocated using new[] operator. User is responsible to free it using delete[].

Returns

Compiled time string. NULL, if error occurs.

3.30.4 Member Data Documentation

3.30.4.1 mbDerRules

```
OSBOOL ASN1Time::mbDerRules
```

This member variable tells whether we will encode this time according to the Distinguished Encoding Rules (DER) or not.

3.30.4.2 mbUtcFlag

```
OSBOOL ASN1Time::mbUtcFlag
```

This member variable tells whether this time is UTC time or not.

3.30.4.3 mDay

```
short ASN1TTime::mDay
```

This member variable holds this time's day.

3.30.4.4 mDiffHour

```
short ASN1TTime::mDiffHour
```

This member variable holds this time's hour differential.

3.30.4.5 mDiffMin

```
short ASN1TTime::mDiffMin
```

This member variable holds this time's minute differential.

3.30.4.6 mHour

```
short ASN1TTime::mHour
```

This member variable holds this time's hour.

3.30.4.7 mMinute

```
short ASN1TTime::mMinute
```

This member variable holds this time's minute.

3.30.4.8 mMonth

```
short ASN1TTime::mMonth
```

This member variable holds this time's month.

3.30.4.9 mSecFracLen

```
int ASN1TTime::mSecFracLen
```

This member variable holds this time's fractional seconds length.

3.30.4.10 mSecFraction

```
int ASN1TTime::mSecFraction
```

This member variable holds this time's fractional seconds.

3.30.4.11 mSecond

```
short ASN1TTime::mSecond
```

This member variable holds this time's second.

3.30.4.12 mStatus

```
int ASN1TTime::mStatus
```

This member variable holds this time's status.

3.30.4.13 mYear

```
short ASN1TTime::mYear
```

This member variable holds this time's year.

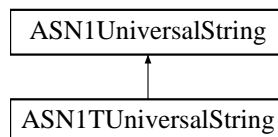
The documentation for this class was generated from the following file:

- [ASN1TTime.h](#)

3.31 ASN1TUniversalString Struct Reference

```
#include <asn1CppTypes.h>
```

Inheritance diagram for ASN1TUniversalString:



Public Member Functions

- [ASN1TUniversalString \(\)](#)

3.31.1 Detailed Description

UniversalString. This is the base class for generated C++ data type classes for UniversalString values.

3.31.2 Constructor & Destructor Documentation

3.31.2.1 ASN1TUniversalString()

```
ASN1TUniversalString::ASN1TUniversalString ( ) [inline]
```

The default constructor creates an empty UniversalString value.

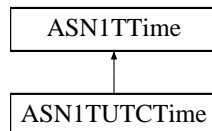
The documentation for this struct was generated from the following file:

- [asn1CppTypes.h](#)

3.32 ASN1TUTCTime Class Reference

```
#include <ASN1TTime.h>
```

Inheritance diagram for ASN1TUTCTime:



Public Member Functions

- EXTRTMETHOD [ASN1TUTCTime](#) ()
- EXTRTMETHOD [ASN1TUTCTime](#) (const char *timeStr, OSBOOL useDerRules=FALSE)
- EXTRTMETHOD [ASN1TUTCTime](#) (OSBOOL useDerRules)
- [ASN1TUTCTime](#) (const [ASN1TUTCTime](#) &original)
- EXTRTMETHOD int [setYear](#) (short year_)
- EXTRTMETHOD int [setTime](#) (time_t time, OSBOOL diffTime)
- EXTRTMETHOD int [setUTC](#) (OSBOOL utc)
- EXTRTMETHOD void [clear](#) ()
- EXTRTMETHOD int [compileString](#) (char *pbuf, size_t bufsize) const
- EXTRTMETHOD int [parseString](#) (const char *string)
- const [ASN1TUTCTime](#) & **operator=** (const [ASN1TUTCTime](#) &tm)

Protected Member Functions

- EXTRTMETHOD int [getFraction](#) () const
- EXTRTMETHOD int [setFraction](#) (int fraction, int fracLen=-1)

Additional Inherited Members

3.32.1 Detailed Description

ASN.1 UTCTime utility class. The ASN1TUTTime class is derived from the [ASN1TTime](#) base class.

3.32.2 Constructor & Destructor Documentation

3.32.2.1 ASN1TUTCTime() [1/4]

```
EXTRTMETHOD ASN1TUTCTime::ASN1TUTCTime ( )
```

A default constructor.

3.32.2.2 ASN1TUTCTime() [2/4]

```
EXTRTMETHOD ASN1TUTCTime::ASN1TUTCTime (
    const char * timeStr,
    OSBOOL useDerRules = FALSE )
```

This constructor creates a time object using the specified time string.

Parameters

<i>timeStr</i>	A pointer to the time string to be parsed.
<i>useDerRules</i>	Create object using DER rules.

3.32.2.3 ASN1TUTCTime() [3/4]

```
EXTRTMETHOD ASN1TUTCTime::ASN1TUTCTime (
    OSBOOL useDerRules )
```

This constructor creates an empty time object.

Parameters

<i>useDerRules</i>	An OSBOOL value.
--------------------	------------------

3.32.2.4 ASN1TUTCTime() [4/4]

```
ASN1TUTCTime::ASN1TUTCTime (
    const ASN1TUTCTime & original ) [inline]
```

A copy constructor.

References [ASN1TTime::clear\(\)](#), [ASN1TTime::compileString\(\)](#), [ASN1TTime::getFraction\(\)](#), [ASN1TTime::parseString\(\)](#), [ASN1TTime::setFraction\(\)](#), [ASN1TTime::setTime\(\)](#), [ASN1TTime::setUTC\(\)](#), and [ASN1TTime::setYear\(\)](#).

3.32.3 Member Function Documentation

3.32.3.1 clear()

```
EXTRTMETHOD void ASN1TUTCTime::clear ( ) [virtual]
```

Clears out the time object.

Reimplemented from [ASN1TTime](#).

3.32.3.2 compileString()

```
EXTRTMETHOD int ASN1TUTCTime::compileString (
    char * pbuf,
    size_t bufsize ) const [virtual]
```

Compiles new time string accoring X.680 and ISO 8601. Returns 0, if succeed, or error code, if error.

Parameters

<i>pbuf</i>	A pointer to destination buffer.
<i>bufsize</i>	A size of destination buffer.

Returns

Completion status of operation:

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

Implements [ASN1TTime](#).

3.32.3.3 getFraction()

```
EXTRTMETHOD int ASN1TUTCTime::getFraction ( ) const [protected], [virtual]
```

This method returns the second's decimal component of the time value.

Second's decimal fraction is represented by one or more digits from 0 to 9.

Returns

Second's decimal fraction component is returned if operation is successful. If the operation fails, a negative value is returned.

Reimplemented from [ASN1TTime](#).

3.32.3.4 parseString()

```
EXTRTMETHOD int ASN1TUTCTime::parseString (
    const char * string ) [virtual]
```

Parses the given time string. The string is assumed to be in standard UTC time format.

Parameters

<i>string</i>	UTC time string to be parsed.
---------------	-------------------------------

Returns

Completion status of operation:

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

Implements [ASN1TTime](#).

3.32.3.5 setFraction()

```
EXTRTMETHOD int ASN1TUTCTime::setFraction (
    int fraction,
    int fracLen = -1 ) [protected], [virtual]
```

This method sets the second's decimal fraction component of the time value.

Second's decimal fraction is represented by one or more digits from 0 to 9. Note that the action of this method may differ for different inherited [ASN1TTime](#) classes.

Parameters

<i>fraction</i>	Second's decimal fraction component (0 - 9).
<i>fracLen</i>	Optional parameter specifies number of digits in second's fraction.

Returns

Completion status of operation:

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

Reimplemented from [ASN1TTime](#).

3.32.3.6 setTime()

```
EXTRTMETHOD int ASN1TUTCTime::setTime (
    time_t time,
    OSBOOL diffTime ) [virtual]
```

Converts *time_t* to time string.

Parameters

<i>time</i>	time to convert,
<i>diffTime</i>	TRUE means the difference between local time and UTC will be calculated; in other case only local time will be stored.

Returns

Completion status of operation:

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

Implements [ASN1TTime](#).

3.32.3.7 setUTC()

```
EXTRTMETHOD int ASN1TUTCTime::setUTC (
    OSBOOL utc ) [virtual]
```

This method sets the UTC flag state.

If the UTC flag is TRUE, then the time is a UTC time and symbol 'Z' is added to the end of the string. Otherwise, it is a local time.

Parameters

<i>utc</i>	UTC flag state.
------------	-----------------

Returns

Completion status of operation:

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

Reimplemented from [ASN1TTime](#).

3.32.3.8 setYear()

```
EXTRTMETHOD int ASN1TUTCTime::setYear (
    short year_ ) [virtual]
```

This method sets the year component of the time value.

The year parameter can be either the two last digits of the year (00 - 99) or the full four digits (0 - 9999). Note: the `getYear` method returns the year in the full four digits, independent of the format of the year parameter used in this method.

Parameters

<i>year</i> ↔	Year component (full four digits or only last two digits).
—	

Returns

Completion status of operation:

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

Reimplemented from [ASN1TTime](#).

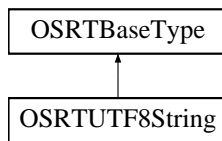
The documentation for this class was generated from the following file:

- [ASN1TTime.h](#)

3.33 OSRTBaseType Class Reference

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

Inheritance diagram for OSRTBaseType:



Public Member Functions

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

3.33.1 Detailed Description

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

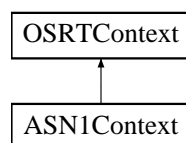
The documentation for this class was generated from the following file:

- [OSRTBaseType.h](#)

3.34 OSRTContext Class Reference

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

Inheritance diagram for OSRTContext:



Public Member Functions

- EXTRTMETHOD [OSRTContext](#) ()
- virtual EXTRTMETHOD [~OSRTContext](#) ()
- OSCTXT * [getPtr](#) ()
- const OSCTXT * [getPtr](#) () const
- EXTRTMETHOD OSUINT32 [getRefCount](#) ()
- int [getStatus](#) () const
- OSBOOL [isInitialized](#) ()
- EXTRTMETHOD void [_ref](#) ()
- EXTRTMETHOD void [_unref](#) ()
- EXTRTMETHOD char * [getErrorInfo](#) ()
- EXTRTMETHOD char * [getErrorInfo](#) (size_t *pBufSize)
- EXTRTMETHOD char * [getErrorInfo](#) (char *pBuf, size_t &bufSize)
- void * [memAlloc](#) (size_t numocts)
- void * [memAllocZ](#) (size_t numocts)
- void [memFreeAll](#) ()
- void [memFreePtr](#) (void *ptr)
- void * [memRealloc](#) (void *ptr, size_t numocts)
- void [memReset](#) ()
- void [printErrorInfo](#) ()
- void [resetErrorInfo](#) ()
- OSBOOL [setDiag](#) (OSBOOL value=TRUE)
- virtual EXTRTMETHOD int [setRunTimeKey](#) (const OSOCTET *key, size_t keylen)
- int [setStatus](#) (int stat)

Protected Attributes

- OSCTXT [mCtxt](#)
- OSUINT32 [mCount](#)
- OSBOOL [mblinitialized](#)
- int [mStatus](#)

3.34.1 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.

3.34.2 Constructor & Destructor Documentation

3.34.2.1 OSRTContext()

```
EXTRTMETHOD OSRTContext::OSRTContext ( )
```

The default constructor initializes the mCtxt member variable and sets the reference count variable (mCount) to zero.

3.34.2.2 ~OSRTContext()

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

The destructor frees all memory held by the context.

3.34.3 Member Function Documentation

3.34.3.1 _ref()

```
EXTRTMETHOD void OSRTContext::_ref ( )
```

The `_ref` method increases the reference count by one.

Referenced by `OSRTCtxtPtr::operator=()`, and `OSRTCtxtPtr::OSRTCtxtPtr()`.

3.34.3.2 _unref()

```
EXTRTMETHOD void OSRTContext::_unref ( )
```

The `_unref` method decreases the reference count by one.

Referenced by `OSRTCtxtPtr::operator=()`, and `OSRTCtxtPtr::~~OSRTCtxtPtr()`.

3.34.3.3 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.

3.34.3.4 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.

3.34.3.5 `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.

3.34.3.6 `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.

Referenced by `OSRTCtxtPtr::getCtxtPtr()`, and `ASN1CType::getCtxtPtr()`.

3.34.3.7 getRefCount()

```
EXTRTMETHOD OSUINT32 OSRTContext::getRefCount ( )
```

The getRefCount method returns the current reference count.

3.34.3.8 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.

Referenced by ASN1CType::getStatus().

3.34.3.9 isInitialized()

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

Returns TRUE, if initialized correctly, FALSE otherwise.

Returns

TRUE, if initialized correctly, FALSE otherwise.

3.34.3.10 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
----------------	---

Referenced by ASN1CType::memAlloc().

3.34.3.11 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
----------------	---

Referenced by ASN1CType::memAllocZ().

3.34.3.12 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.

Referenced by ASN1CType::memFreeAll().

3.34.3.13 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>
------------	---

Referenced by `ASN1CType::memFreePtr()`.

3.34.3.14 `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

Referenced by `ASN1CType::memRealloc()`.

3.34.3.15 `memReset()`

```
void OSRTContext::memReset ( ) [inline]
```

The `memReset` method resets dynamic memory using the C runtime memory management functions.

Referenced by `ASN1CType::memReset()`.

3.34.3.16 `printErrorInfo()`

```
void OSRTContext::printErrorInfo ( ) [inline]
```

The `printErrorInfo` method prints information on errors contained within the context.

Referenced by `ASN1CType::printErrorInfo()`.

3.34.3.17 resetErrorInfo()

```
void OSRTCContext::resetErrorInfo ( ) [inline]
```

The resetErrorInfo method resets information on errors contained within the context.

Referenced by ASN1CType::resetError().

3.34.3.18 setDiag()

```
OSBOOL OSRTCContext::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

3.34.3.19 setRunTimeKey()

```
virtual EXTRMETHOD int OSRTCContext::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.

Reimplemented in [ASN1Context](#).

3.34.3.20 setStatus()

```
int OSRTContext::setStatus (
    int stat )
```

This method sets error status in the context.

Parameters

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

Returns

Error status value being set.

Referenced by `ASN1MessageBuffer::setStatus()`.

3.34.4 Member Data Documentation

3.34.4.1 mbInitialized

```
OSBOOL OSRTContext::mbInitialized [protected]
```

TRUE, if initialized correctly, FALSE otherwise

3.34.4.2 mCount

```
OSUINT32 OSRTContext::mCount [protected]
```

The mCount member variable holds the reference count of this context.

3.34.4.3 mCtxt

```
OSCTXT OSRTContext::mCtxt [protected]
```

The mCtxt member variable is a standard C runtime context variable used in most C runtime function calls.

3.34.4.4 mStatus

```
int OSRTContext::mStatus [protected]
```

The mStatus variable holds the return status from C run-time function calls. The getStatus method will either return this status or the last status on the context error list.

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

- [OSRTContext.h](#)

3.35 OSRTCtxtPtr Class Reference

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

Public Member Functions

- [OSRTCtxtPtr](#) ([OSRTContext](#) *rf=0)
- [OSRTCtxtPtr](#) (const [OSRTCtxtPtr](#) &o)
- virtual [~OSRTCtxtPtr](#) ()
- [OSRTCtxtPtr](#) & [operator=](#) (const [OSRTCtxtPtr](#) &rf)
- [OSRTCtxtPtr](#) & [operator=](#) ([OSRTContext](#) *rf)
- [operator](#) [OSRTContext](#) * ()
- **operator const** [OSRTContext](#) * () const
- [OSRTContext](#) * [operator->](#) ()
- const [OSRTContext](#) * **operator->** () const
- OSBOOL [operator==](#) (const [OSRTContext](#) *o) const
- OSBOOL [isNull](#) () const
- OSCTXT * [getCtxtPtr](#) ()

Protected Attributes

- [OSRTContext](#) * mPointer

3.35.1 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.

3.35.2 Constructor & Destructor Documentation

3.35.2.1 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
-----------	---

References `OSRTContext::_ref()`.

3.35.2.2 `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
----------	--

References `OSRTContext::_ref()`.

3.35.2.3 `~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.

References `OSRTContext::_unref()`.

3.35.3 Member Function Documentation

3.35.3.1 `getCtxtPtr()`

```
OSCTXT* OSRTCtxtPtr::getCtxtPtr ( ) [inline]
```

This method returns the standard context pointer used in C function calls.

References `OSRTContext::getPtr()`.

3.35.3.2 isNull()

```
OSBOOL OSRTCtxtPtr::isNull ( ) const [inline]
```

The isNull method returns TRUE if the underlying context pointer is NULL.

Referenced by ASN1CType::getCtxtPtr(), ASN1CType::getStatus(), ASN1CType::memAlloc(), ASN1CType::memAllocZ(), ASN1CType::memFreeAll(), ASN1CType::memFreePtr(), ASN1CType::memRealloc(), ASN1CType::memReset(), ASN1CType::printErrorInfo(), ASN1CType::resetError(), and ASN1TPDU::setContext().

3.35.3.3 operator OSRTContext *()

```
OSRTCtxtPtr::operator OSRTContext * ( ) [inline]
```

The 'OSRTContext*' operator returns the context object pointer.

3.35.3.4 operator->()

```
OSRTContext* OSRTCtxtPtr::operator-> ( ) [inline]
```

The '->' operator returns the context object pointer.

3.35.3.5 operator=() [1/2]

```
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

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

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

3.35.3.6 operator=() [2/2]

```
OSRTCtxtPtr& OSRTCtxtPtr::operator= (
    OSRTContext * rf ) [inline]
```

This assignment operator assigns does a direct assignment of an [OSRTContext](#) object to this [OSRTCtxPtr](#) object.

References [OSRTContext::_ref\(\)](#), and [OSRTContext::_unref\(\)](#).

3.35.3.7 operator==()

```
OSBOOL OSRTCtxPtr::operator== (
    const OSRTContext * o ) const [inline]
```

The '==' operator compares two [OSRTContext](#) pointer values.

3.35.4 Member Data Documentation

3.35.4.1 mPointer

```
OSRTContext* OSRTCtxPtr::mPointer [protected]
```

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

Referenced by [operator=\(\)](#).

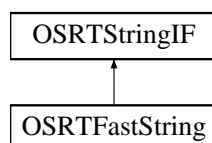
The documentation for this class was generated from the following file:

- [OSRTContext.h](#)

3.36 OSRTFastString Class Reference

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

Inheritance diagram for OSRTFastString:



Public Member Functions

- `OSRTFastString ()`
- `OSRTFastString (const char *strval)`
- `OSRTFastString (const OSUTF8CHAR *strval)`
- `OSRTFastString (const OSRTFastString &str)`
- `virtual ~OSRTFastString ()`
- `virtual OSRTStringIF * clone ()`
- `virtual const char * getValue () const`
- `virtual const OSUTF8CHAR * getUTF8Value () const`
- `virtual void print (const char *name)`
- `virtual void setValue (const char *str)`
- `virtual void setValue (const OSUTF8CHAR *str)`
- `OSRTFastString & operator= (const OSRTFastString &original)`

Protected Attributes

- `const OSUTF8CHAR * mValue`

Additional Inherited Members

3.36.1 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.

3.36.2 Constructor & Destructor Documentation

3.36.2.1 OSRTFastString() [1/4]

```
OSRTFastString::OSRTFastString ( )
```

The default constructor sets the internal string member variable pointer to null.

3.36.2.2 OSRTFastString() [2/4]

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

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

Parameters

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

3.36.2.3 OSRTFastString() [3/4]

```
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
---------------	----------------------------------

3.36.2.4 OSRTFastString() [4/4]

```
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.
------------	-----------------------------------

3.36.2.5 ~OSRTFastString()

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

The destructor does nothing.

3.36.3 Member Function Documentation

3.36.3.1 clone()

```
virtual OSRTStringIF* OSRTFastString::clone ( ) [inline], [virtual]
```

This method creates a copy of the given string object.

Implements [OSRTStringIF](#).

3.36.3.2 getUTF8Value()

```
virtual const OSUTF8CHAR* OSRTFastString::getUTF8Value ( ) const [inline], [virtual]
```

This method returns the pointer to UTF-8 null terminated string as a UTF-8 string.

Implements [OSRTStringIF](#).

3.36.3.3 getValue()

```
virtual const char* OSRTFastString::getValue ( ) const [inline], [virtual]
```

This method returns the pointer to UTF-8 null terminated string as a standard ASCII string.

Implements [OSRTStringIF](#).

3.36.3.4 operator=()

```
OSRTFastString& OSRTFastString::operator= (
    const OSRTFastString & original )
```

Assignment operator.

3.36.3.5 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](#).

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

3.36.3.6 `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](#).

3.36.3.7 `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](#).

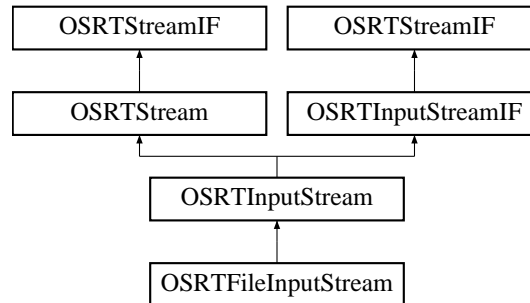
The documentation for this class was generated from the following file:

- [OSRTFastString.h](#)

3.37 OSRTFileInputStream Class Reference

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

Inheritance diagram for OSRTFileInputStream:



Public Member Functions

- EXTRTMETHOD `OSRTFileInputStream` (const char *pFilename)
- EXTRTMETHOD `OSRTFileInputStream` (OSRTContext *pContext, const char *pFilename)
- EXTRTMETHOD `OSRTFileInputStream` (FILE *file)
- EXTRTMETHOD `OSRTFileInputStream` (OSRTContext *pContext, FILE *file)
- virtual OSBOOL `isA` (StreamID id) const

Additional Inherited Members

3.37.1 Detailed Description

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

3.37.2 Constructor & Destructor Documentation

3.37.2.1 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

3.37.2.2 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

3.37.2.3 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

3.37.2.4 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

`::rtxStreamFileAttach`

3.37.3 Member Function Documentation

3.37.3.1 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](#).

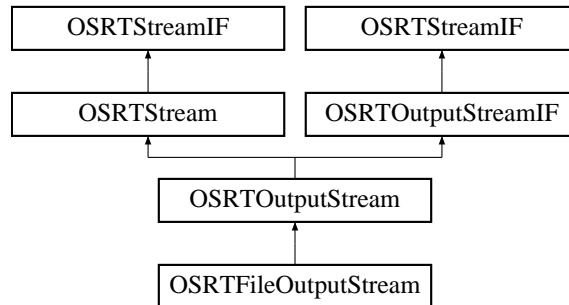
The documentation for this class was generated from the following file:

- [OSRTFileInputStream.h](#)

3.38 OSRTFileOutputStream Class Reference

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

Inheritance diagram for OSRTFileOutputStream:



Public Member Functions

- EXTRTMETHOD [OSRTFileOutputStream](#) (const char *pFilename)
- EXTRTMETHOD [OSRTFileOutputStream](#) (OSRTContext *pContext, const char *pFilename)
- EXTRTMETHOD [OSRTFileOutputStream](#) (FILE *file)
- EXTRTMETHOD [OSRTFileOutputStream](#) (OSRTContext *pContext, FILE *file)
- virtual OSBOOL [isA](#) (StreamID id) const

Additional Inherited Members

3.38.1 Detailed Description

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

3.38.2 Constructor & Destructor Documentation

3.38.2.1 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`

3.38.2.2 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`

3.38.2.3 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

3.38.2.4 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

3.38.3 Member Function Documentation

3.38.3.1 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](#).

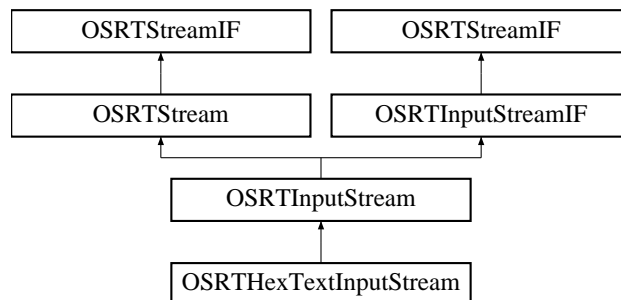
The documentation for this class was generated from the following file:

- [OSRTFileOutputStream.h](#)

3.39 OSRTHexTextInputStream Class Reference

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

Inheritance diagram for OSRTHexTextInputStream:



Public Member Functions

- EXTRTMETHOD [OSRTHexTextInputStream](#) ([OSRTInputStream](#) *pstream)
- EXTRTMETHOD [~OSRTHexTextInputStream](#) ()
- virtual OSBOOL [isA](#) (StreamID id) const
- void [setOwnUnderStream](#) (OSBOOL value=TRUE)

Protected Attributes

- [OSRTInputStream](#) * **mpUnderStream**
- OSBOOL **mbOwnUnderStream**

Additional Inherited Members

3.39.1 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.

3.39.2 Constructor & Destructor Documentation

3.39.2.1 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`

3.39.2.2 ~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.

3.39.3 Member Function Documentation

3.39.3.1 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](#).

References `OSRTInputStream::isA()`.

3.39.3.2 `setOwnUnderStream()`

```
void OSRTHexTextInputStream::setOwnUnderStream (
    OSBOOL value = TRUE ) [inline]
```

This method transfers ownership of the underlying stream to the class.

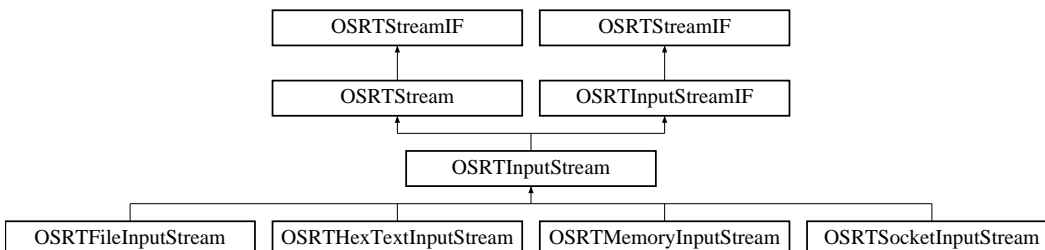
The documentation for this class was generated from the following file:

- [OSRTHexTextInputStream.h](#)

3.40 OSRTInputStream Class Reference

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

Inheritance diagram for OSRTInputStream:



Public Member Functions

- EXTRTMETHOD [OSRTInputStream](#) ()
- EXTRTMETHOD **OSRTInputStream** ([OSRTContext](#) *mpContext, OSBOOL attachStream=FALSE)
- virtual EXTRTMETHOD [~OSRTInputStream](#) ()
- virtual EXTRTMETHOD int [close](#) ()
- virtual EXTRTMETHOD size_t [currentPos](#) ()
- virtual EXTRTMETHOD int [flush](#) ()
- virtual OSBOOL [isA](#) (StreamID id) const
- virtual [OSRTCtxtPtr](#) [getContext](#) ()
- virtual OSCTXT * [getCtxtPtr](#) ()
- virtual char * [getErrorInfo](#) ()
- virtual char * [getErrorInfo](#) (char *pBuf, size_t &bufSize)
- virtual int [getPosition](#) (size_t *ppos)
- virtual int [getStatus](#) () const
- virtual EXTRTMETHOD OSBOOL [isOpened](#) ()
- virtual EXTRTMETHOD OSBOOL [markSupported](#) ()
- virtual EXTRTMETHOD int [mark](#) (size_t readAheadLimit)
- void [printErrorInfo](#) ()
- void [resetErrorInfo](#) ()
- virtual EXTRTMETHOD long [read](#) (OSOCKET *pDestBuf, size_t maxToRead)
- virtual EXTRTMETHOD long [readBlocking](#) (OSOCKET *pDestBuf, size_t toReadBytes)
- virtual EXTRTMETHOD int [reset](#) ()
- virtual int [setPosition](#) (size_t pos)
- virtual EXTRTMETHOD int [skip](#) (size_t n)

Additional Inherited Members

3.40.1 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.

3.40.2 Constructor & Destructor Documentation

3.40.2.1 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.
----------------------------	-------------------------------------

3.40.2.2 ~OSRTInputStream()

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

Virtual destructor. Closes the stream if it was opened.

3.40.3 Member Function Documentation

3.40.3.1 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](#).

3.40.3.2 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.

Implements [OSRTInputStreamIF](#).

3.40.3.3 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](#).

3.40.3.4 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](#).

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

3.40.3.5 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](#).

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

3.40.3.6 `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](#).

References `OSRTStream::getErrorInfo()`.

3.40.3.7 `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](#).

References `OSRTStream::getErrorInfo()`, and `OSRTInputStreamIF::getPosition()`.

3.40.3.8 `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.

Implements [OSRTInputStreamIF](#).

3.40.3.9 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.

References [OSRTStream::getStatus\(\)](#), [OSRTStreamIF::isOpened\(\)](#), [OSRTInputStreamIF::mark\(\)](#), and [OSRTInputStreamIF::markSupported\(\)](#).

3.40.3.10 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

Implements [OSRTInputStreamIF](#).

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

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

3.40.3.11 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

[::rtxStreamIsOpened](#)

Reimplemented from [OSRTStream](#).

3.40.3.12 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`

Implements [OSRTInputStreamIF](#).

3.40.3.13 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`

Implements [OSRTInputStreamIF](#).

3.40.3.14 printErrorInfo()

```
void OSRTInputStream::printErrorInfo ( ) [inline]
```

The `printErrorInfo` method prints information on errors contained within the context.

References `OSRTStream::printErrorInfo()`.

3.40.3.15 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`

Implements [OSRTInputStreamIF](#).

3.40.3.16 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 mamimum 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`

Implements [OSRTInputStreamIF](#).

3.40.3.17 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`

Implements [OSRTInputStreamIF](#).

3.40.3.18 resetErrorInfo()

```
void OSRTInputStream::resetErrorInfo ( ) [inline]
```

The `resetErrorInfo` method resets information on errors contained within the context.

References `OSRTInputStreamIF::read()`, `OSRTInputStreamIF::readBlocking()`, `OSRTInputStreamIF::reset()`, `OSRTStream::resetErrorInfo()`, `OSRTInputStreamIF::setPosition()`, and `OSRTInputStreamIF::skip()`.

3.40.3.19 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.

Implements [OSRTInputStreamIF](#).

3.40.3.20 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

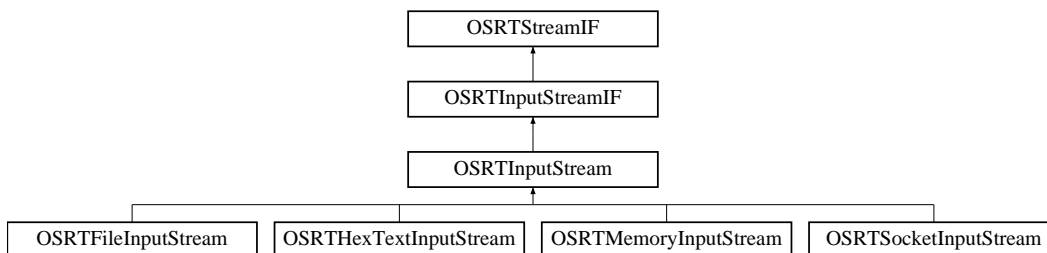
Implements [OSRTInputStreamIF](#).

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

- [OSRTInputStream.h](#)

3.41 OSRTInputStreamIF Class Reference

Inheritance diagram for OSRTInputStreamIF:



Public Types

- enum **StreamID** { **Unknown**, **File**, **Memory**, **Socket** }

Public Member Functions

- virtual EXTRTMETHOD **~OSRTInputStreamIF** ()
- virtual OSBOOL **isA** (StreamID id) const =0
- virtual size_t **currentPos** ()=0
- virtual int **getPosition** (size_t *ppos)=0
- virtual OSBOOL **markSupported** ()=0
- virtual int **mark** (size_t readAheadLimit)=0
- virtual long **read** (OSOCKETET *pDestBuf, size_t maxToRead)=0
- virtual long **readBlocking** (OSOCKETET *pDestBuf, size_t toReadBytes)=0
- virtual int **reset** ()=0
- virtual int **setPosition** (size_t pos)=0
- virtual int **skip** (size_t n)=0

3.41.1 Constructor & Destructor Documentation

3.41.1.1 `~OSRTInputStreamIF()`

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

Virtual destructor. Closes the stream if it was opened.

3.41.2 Member Function Documentation

3.41.2.1 `currentPos()`

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

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

Returns

The number of octets already read from the stream.

Implemented in [OSRTInputStream](#).

3.41.2.2 `getPosition()`

```
virtual int OSRTInputStreamIF::getPosition (
    size_t * ppos ) [pure 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.

Implemented in [OSRTInputStream](#).

Referenced by `OSRTInputStream::getErrorInfo()`.

3.41.2.3 isA()

```
virtual OSBOOL OSRTInputStreamIF::isA (
    StreamID id ) const [pure 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

Implemented in [OSRTSocketInputStream](#), [OSRTInputStream](#), [OSRTFileInputStream](#), [OSRTHexTextInputStream](#), and [OSRTMemoryInputStream](#).

3.41.2.4 mark()

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

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

Parameters

<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`

Implemented in [OSRTInputStream](#).

Referenced by `OSRTInputStream::getStatus()`.

3.41.2.5 markSupported()

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

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

Returns

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

See also

`::rtxStreamIsMarkSupported`

Implemented in [OSRTInputStream](#).

Referenced by `OSRTInputStream::getStatus()`.

3.41.2.6 read()

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

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

Parameters

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

Returns

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

See also

`::rxStreamRead`

Implemented in [OSRTInputStream](#).

Referenced by `OSRTInputStream::resetErrorInfo()`.

3.41.2.7 readBlocking()

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

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

Parameters

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

Returns

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

See also

`::rxStreamRead`

Implemented in [OSRTInputStream](#).

Referenced by `OSRTInputStream::resetErrorInfo()`.

3.41.2.8 reset()

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

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

Returns

Completion status of operation:

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

See also

`::rtxStreamMark`, `::rtxStreamReset`

Implemented in [OSRTInputStream](#).

Referenced by `OSRTInputStream::resetErrorInfo()`.

3.41.2.9 `setPosition()`

```
virtual int OSRTInputStreamIF::setPosition (
    size_t pos ) [pure 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.

Implemented in [OSRTInputStream](#).

Referenced by `OSRTInputStream::resetErrorInfo()`.

3.41.2.10 `skip()`

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

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

Parameters

n	The number of octets to be skipped.
-----	-------------------------------------

Returns

Completion status of operation:

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

See also

`::rtxStreamSkip`

Implemented in [OSRTInputStream](#).

Referenced by `OSRTInputStream::resetErrorInfo()`.

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

- [OSRTInputStreamIF.h](#)

3.42 OSRTInputStreamPtr Class Reference

Public Member Functions

- **OSRTInputStreamPtr** ([OSRTInputStreamIF](#) *ptr)
- **operator OSRTInputStreamIF** * ()
- [OSRTInputStreamIF](#) * **operator->** ()

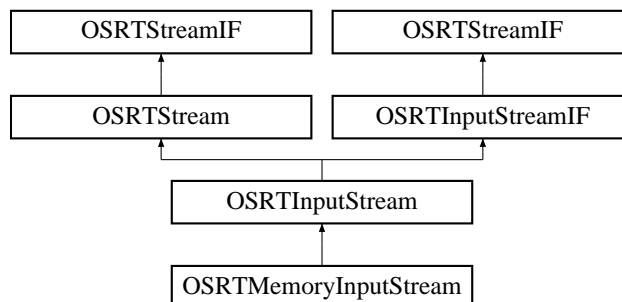
The documentation for this class was generated from the following file:

- [OSRTInputStreamIF.h](#)

3.43 OSRTMemoryInputStream Class Reference

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

Inheritance diagram for OSRTMemoryInputStream:



Public Member Functions

- EXTRTMETHOD [OSRTMemoryInputStream](#) (const OSOCKET *pMemBuf, size_t bufSize)
- EXTRTMETHOD [OSRTMemoryInputStream](#) ([OSRTContext](#) *pContext, const OSOCKET *pMemBuf, size_t bufSize)
- virtual OSBOOL [isA](#) (StreamID id) const

Additional Inherited Members

3.43.1 Detailed Description

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

3.43.2 Constructor & Destructor Documentation

3.43.2.1 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](#)

3.43.2.2 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

`::rxStreamMemoryAttach`

3.43.3 Member Function Documentation

3.43.3.1 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](#).

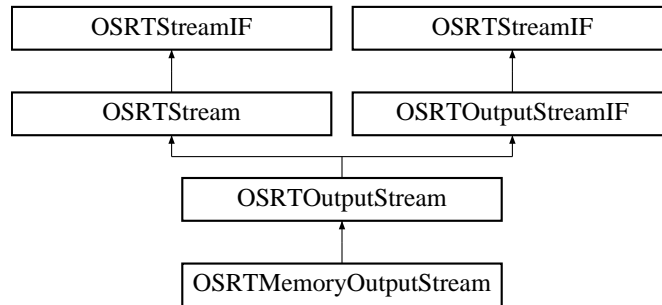
The documentation for this class was generated from the following file:

- [OSRTMemoryInputStream.h](#)

3.44 OSRTMemoryOutputStream Class Reference

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

Inheritance diagram for OSRTMemoryOutputStream:



Public Member Functions

- EXTRTMETHOD `OSRTMemoryOutputStream()`
- EXTRTMETHOD `OSRTMemoryOutputStream(OSOCTET *pMemBuf, size_t bufSize)`
- EXTRTMETHOD `OSRTMemoryOutputStream(OSRTContext *pContext, OSOCTET *pMemBuf, size_t bufSize)`
- EXTRTMETHOD `OSOCTET *getBuffer(size_t *pSize=0)`
- virtual `OSBOOL isA(StreamID id) const`
- `int reset()`

Additional Inherited Members

3.44.1 Detailed Description

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

3.44.2 Constructor & Destructor Documentation

3.44.2.1 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`

3.44.2.2 OSRTMemoryOutputStream() [2/3]

```
EXTRTMETHOD OSRTMemoryOutputStream::OSRTMemoryOutputStream (
    OSOCTET * 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`

3.44.2.3 OSRTMemoryOutputStream() [3/3]

```
EXTRTMETHOD OSRTMemoryOutputStream::OSRTMemoryOutputStream (
    OSRTCContext * 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`

3.44.3 Member Function Documentation

3.44.3.1 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.

3.44.3.2 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](#).

3.44.3.3 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.

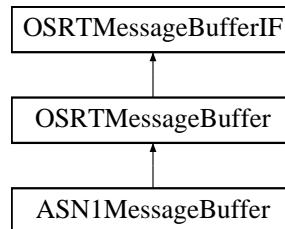
The documentation for this class was generated from the following file:

- [OSRTMemoryOutputStream.h](#)

3.45 OSRTMessageBuffer Class Reference

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

Inheritance diagram for OSRTMessageBuffer:



Public Member Functions

- virtual `~OSRTMessageBuffer ()`
- virtual `void * getAppInfo ()`
- virtual `size_t getByteIndex ()`
- virtual `OSRTCtxtPtr getContext ()`
- virtual `OSCTXT * getCtxtPtr ()`
- virtual `char * getErrorInfo ()`
- virtual `char * getErrorInfo (char *pBuf, size_t &bufSize)`
- virtual `OSOCKET * getMsgCopy ()`
- virtual `const OSOCKET * getMsgPtr ()`
- `int getStatus () const`
- virtual `int init ()`
- virtual `EXTRTMETHOD int initBuffer (OSOCKET *pMsgBuf, size_t msgBufLen)`
- virtual `void printErrorInfo ()`
- virtual `void resetErrorInfo ()`
- virtual `void setAppInfo (void *)`
- virtual `EXTRTMETHOD void setDiag (OSBOOL value=TRUE)`

Protected Member Functions

- `EXTRTMETHOD OSRTMessageBuffer (Type bufferType, OSRTContext *pContext=0)`

Protected Attributes

- OSRTCtxtHolder **mCtxtHolder**
- Type **mBufferType**

Additional Inherited Members

3.45.1 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.

3.45.2 Constructor & Destructor Documentation

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

3.45.2.2 ~OSRTMessageBuffer()

```
virtual OSRTMessageBuffer::~OSRTMessageBuffer ( ) [inline], [virtual]
```

The virtual destructor does nothing. It is overridden by derived versions of this class.

3.45.3 Member Function Documentation

3.45.3.1 getAppInfo()

```
virtual void* OSRTMessageBuffer::getAppInfo ( ) [inline], [virtual]
```

Returns a pointer to application-specific information block

Implements [OSRTMessageBufferIF](#).

Reimplemented in [ASN1MessageBuffer](#).

3.45.3.2 `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](#).

3.45.3.3 `getContext()`

```
virtual OSRCTxtPtr OSRTMessageBuffer::getContext ( ) [inline], [virtual]
```

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

Implements [OSRTMessageBufferIF](#).

Referenced by `ASN1MessageBuffer::setStatus()`.

3.45.3.4 `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](#).

Referenced by `ASN1MessageBuffer::addEventHandler()`, `ASN1MessageBuffer::getBitOffset()`, `ASN1MessageBuffer::removeEventHandler()`, and `ASN1MessageBuffer::setErrorHandler()`.

3.45.3.5 `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.

3.45.3.6 `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.

3.45.3.7 getMsgCopy()

```
virtual OSOCTET* OSRTMessageBuffer::getMsgCopy ( ) [inline], [virtual]
```

The getMsgCopy method will return a copy of the encoded message managed by the object.

Implements [OSRTMessageBufferIF](#).

Referenced by ASN1MessageBuffer::getBitOffset().

3.45.3.8 getMsgPtr()

```
virtual const OSOCTET* OSRTMessageBuffer::getMsgPtr ( ) [inline], [virtual]
```

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

Implements [OSRTMessageBufferIF](#).

Referenced by ASN1MessageBuffer::getBitOffset().

3.45.3.9 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.

3.45.3.10 `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](#).

References [OSRTMessageBufferIF::initBuffer\(\)](#).

3.45.3.11 `initBuffer()`

```
virtual EXTRIMETHOD 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](#).

Referenced by [ASN1MessageBuffer::getMsgLen\(\)](#).

3.45.3.12 printErrorInfo()

```
virtual void OSRTMessageBuffer::printErrorInfo ( ) [inline], [virtual]
```

The printErrorInfo method prints information on errors contained within the context.

Referenced by ASN1MessageBuffer::getBitOffset().

3.45.3.13 resetErrorInfo()

```
virtual void OSRTMessageBuffer::resetErrorInfo ( ) [inline], [virtual]
```

The resetErrorInfo method resets information on errors contained within the context.

Reimplemented in [ASN1MessageBuffer](#).

Referenced by ASN1MessageBuffer::resetErrorInfo().

3.45.3.14 setAppInfo()

```
virtual void OSRTMessageBuffer::setAppInfo (
    void * ) [inline], [virtual]
```

Sets the application-specific information block.

Implements [OSRTMessageBufferIF](#).

Reimplemented in [ASN1MessageBuffer](#).

References OSRTMessageBufferIF::setDiag().

3.45.3.15 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](#).

3.45.4 Member Data Documentation

3.45.4.1 mBufferType

```
Type OSRTMessageBuffer::mBufferType [protected]
```

The mBufferType member variable holds information on the derived message buffer class type (for example, XML↔Encode).

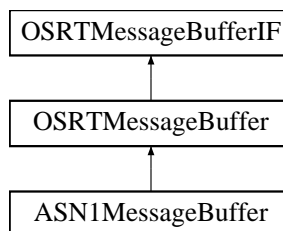
The documentation for this class was generated from the following file:

- [OSRTMsgBuf.h](#)

3.46 OSRTMessageBufferIF Class Reference

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

Inheritance diagram for OSRTMessageBufferIF:



Public Types

- enum **Type** {
BEREncode, BERDecode, PEREncode, PERDecode,
XEREncode, XERDecode, XMLEncode, XMLDecode,
JSONEncode, JSONDecode, Stream, OEREncode,
OERDecode }

Public Member Functions

- virtual void * [getAppInfo](#) ()=0
- virtual size_t [getByteIndex](#) ()=0
- virtual [OSRTCtxtPtr](#) [getContext](#) ()=0
- virtual OSCTXT * [getCtxtPtr](#) ()=0
- virtual OSOCTET * [getMsgCopy](#) ()=0
- virtual const OSOCTET * [getMsgPtr](#) ()=0
- virtual int [init](#) ()=0
- virtual int [initBuffer](#) (OSOCTET *pMsgBuf, size_t msgBufLen)=0
- virtual OSBOOL [isA](#) (Type bufferType)=0
- virtual void [setAppInfo](#) (void *pAppInfo)=0
- virtual void [setNamespace](#) (const OSUTF8CHAR *, const OSUTF8CHAR *, OSRTDList *=0)
- virtual void [setDiag](#) (OSBOOL value=TRUE)=0

Protected Member Functions

- virtual [~OSRTMessageBufferIF](#) ()

3.46.1 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.

3.46.2 Constructor & Destructor Documentation

3.46.2.1 [~OSRTMessageBufferIF](#)()

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

The virtual destructor does nothing. It is overridden by derived versions of this class.

3.46.3 Member Function Documentation

3.46.3.1 [getAppInfo](#)()

```
virtual void* OSRTMessageBufferIF::getAppInfo ( ) [pure virtual]
```

Returns a pointer to application-specific information block

Implemented in [ASN1MessageBuffer](#), and [OSRTMessageBuffer](#).

3.46.3.2 `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](#).

3.46.3.3 `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](#).

3.46.3.4 `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](#).

3.46.3.5 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](#).

3.46.3.6 initBuffer()

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

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

Parameters

<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().

3.46.3.7 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, XEREncode, XERDecode, XMLEncode, XMLDecode, Stream.
-------------------	--

Returns

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

Implemented in [ASN1MessageBuffer](#).

3.46.3.8 setAppInfo()

```
virtual void OSRTMessageBufferIF::setAppInfo (
    void * pAppInfo ) [pure virtual]
```

Sets the application-specific information block.

Implemented in [ASN1MessageBuffer](#), and [OSRTMessageBuffer](#).

3.46.3.9 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\(\)](#).

3.46.3.10 setNamespace()

```
virtual void OSRTMessageBufferIF::setNamespace (
    const OSUTF8CHAR * ,
```

```

const OSUTF8CHAR * ,
OSRDLList * = 0 ) [inline], [virtual]

```

Sets the namespace information.

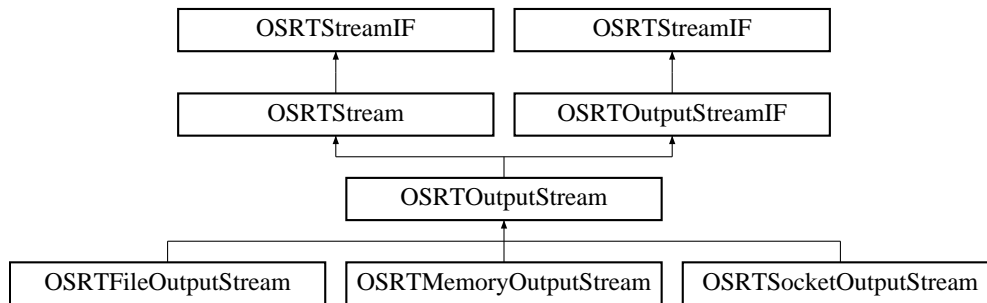
The documentation for this class was generated from the following file:

- [OSRTMsgBufIF.h](#)

3.47 OSRTOutputStream Class Reference

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

Inheritance diagram for OSRTOutputStream:



Public Member Functions

- EXTRTMETHOD [OSRTOutputStream](#) ()
- EXTRTMETHOD **OSRTOutputStream** ([OSRTContext](#) *mpContext, OSBOOL attachStream=FALSE)
- virtual EXTRTMETHOD [~OSRTOutputStream](#) ()
- virtual EXTRTMETHOD int [close](#) ()
- virtual EXTRTMETHOD int [flush](#) ()
- virtual [OSRTCtxtPtr](#) [getContext](#) ()
- virtual OSCTXT * [getCtxtPtr](#) ()
- virtual char * [getErrorInfo](#) ()
- virtual char * [getErrorInfo](#) (char *pBuf, size_t &bufSize)
- virtual int [getStatus](#) () const
- virtual OSBOOL [isA](#) (StreamID id) const
- virtual EXTRTMETHOD OSBOOL [isOpened](#) ()
- void [printErrorInfo](#) ()
- void [resetErrorInfo](#) ()
- virtual EXTRTMETHOD long [write](#) (const OSOCKET *pdata, size_t size)
- virtual EXTRTMETHOD long [write](#) (const char *pdata)

Additional Inherited Members

3.47.1 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.

3.47.2 Constructor & Destructor Documentation

3.47.2.1 OSRTOutputStream()

```
EXTRIMETHOD 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.

3.47.2.2 ~OSRTOutputStream()

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

Virtual destructor. Closes the stream if it was opened.

3.47.3 Member Function Documentation

3.47.3.1 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](#).

3.47.3.2 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

[::rxStreamFlush](#)

Reimplemented from [OSRTStream](#).

3.47.3.3 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](#).

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

3.47.3.4 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](#).

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

3.47.3.5 `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](#).

References `OSRTStream::getErrorInfo()`.

3.47.3.6 `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](#).

References `OSRTStream::getErrorInfo()`.

3.47.3.7 `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.

References `OSRTStream::getStatus()`.

3.47.3.8 `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

Implements [OSRTOutputStreamIF](#).

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

References `OSRTStreamIF::isOpened()`.

3.47.3.9 `isOpened()`

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

Checks if the stream open or not.

Returns

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

See also

`::rtxStreamsIsOpened`

Reimplemented from [OSRTStream](#).

3.47.3.10 printErrorInfo()

```
void OSRTOutputStream::printErrorInfo ( ) [inline]
```

The `printErrorInfo` method prints information on errors contained within the context.

References `OSRTStream::printErrorInfo()`.

3.47.3.11 resetErrorInfo()

```
void OSRTOutputStream::resetErrorInfo ( ) [inline]
```

The `resetErrorInfo` method resets information on errors contained within the context.

References `OSRTStream::resetErrorInfo()`, and `OSRTOutputStreamIF::write()`.

3.47.3.12 write() [1/2]

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

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

Parameters

<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

`::rxStreamWrite`

Implements [OSRTOutputStreamIF](#).

3.47.3.13 write() [2/2]

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

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

Parameters

<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

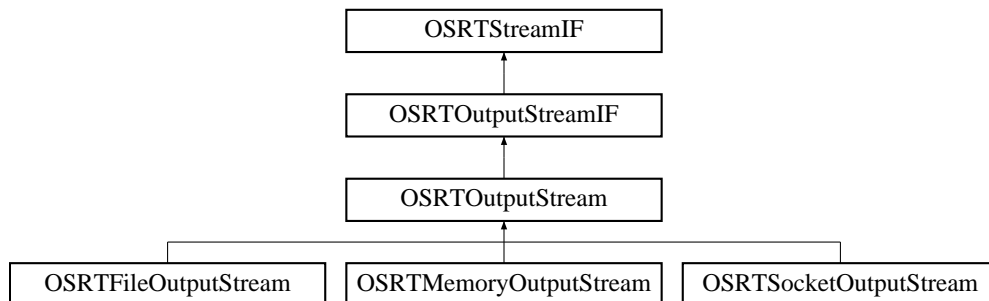
`::rtxStreamWrite`

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

- [OSRTOutputStream.h](#)

3.48 OSRTOutputStreamIF Class Reference

Inheritance diagram for OSRTOutputStreamIF:



Public Types

- enum **StreamID** { **Unknown**, **File**, **Memory**, **Socket** }

Public Member Functions

- virtual EXTRTMETHOD `~OSRTOutputStreamIF` ()
- virtual OSBOOL `isA` (StreamID id) const =0
- virtual long `write` (const OSOCKETET *pdata, size_t size)=0

3.48.1 Constructor & Destructor Documentation

3.48.1.1 ~OSRTOutputStreamIF()

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

Virtual destructor. Closes the stream if it was opened.

3.48.2 Member Function Documentation

3.48.2.1 isA()

```
virtual OSBOOL OSRTOutputStreamIF::isA (
    StreamID id ) const [pure 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

Implemented in [OSRTOutputStream](#), [OSRTSocketOutputStream](#), [OSRTMemoryOutputStream](#), and [OSRTFileOutput↔Stream](#).

3.48.2.2 write()

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

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

Parameters

<i>pdata</i>	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

`::rxStreamWrite`

Implemented in [OSRTOutputStream](#).

Referenced by `OSRTOutputStream::resetErrorInfo()`.

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

- [OSRTOutputStreamIF.h](#)

3.49 OSRTOutputStreamPtr Class Reference

Public Member Functions

- **OSRTOutputStreamPtr** ([OSRTOutputStreamIF](#) *ptr)
- **operator OSRTOutputStreamIF** * ()
- [OSRTOutputStreamIF](#) * **operator->** ()

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

- [OSRTOutputStreamIF.h](#)

3.50 OSRTSocket Class Reference

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

Public Member Functions

- EXTRTMETHOD [OSRTSocket](#) ()
- EXTRTMETHOD [OSRTSocket](#) (OSRTSOCKET socket, OSBOOL ownership=FALSE, int retryCount=1)
- EXTRTMETHOD [OSRTSocket](#) (const [OSRTSocket](#) &socket)
- EXTRTMETHOD [~OSRTSocket](#) ()
- EXTRTMETHOD [OSRTSocket](#) * [accept](#) (OSIPADDR *destIP=0, int *port=0)
- EXTRTMETHOD int [bind](#) (OSIPADDR addr, int port)
- EXTRTMETHOD int [bindUrl](#) (const char *url)
- EXTRTMETHOD int [bind](#) (const char *pAddrStr, int port)
- int [bind](#) (int port)
- EXTRTMETHOD int [blockingRead](#) (OSOCKET *pbuf, size_t readBytes)
- EXTRTMETHOD int [close](#) ()
- EXTRTMETHOD int [connect](#) (const char *host, int port)
- EXTRTMETHOD int [connectTimed](#) (const char *host, int port, int nsecs)
- EXTRTMETHOD int [connectUrl](#) (const char *url)
- OSBOOL [getOwnership](#) ()
- OSRTSOCKET [getSocket](#) () const
- int [getStatus](#) ()
- EXTRTMETHOD int [listen](#) (int maxConnections)
- EXTRTMETHOD int [recv](#) (OSOCKET *pbuf, size_t bufsize)
- EXTRTMETHOD int [send](#) (const OSOCKET *pdata, size_t size)
- void [setOwnership](#) (OSBOOL ownership)
- void [setRetryCount](#) (int value)

Static Public Member Functions

- static EXTRTMETHOD const char * [addrToString](#) (OSIPADDR ipAddr, char *pAddrStr, size_t bufsize)
- static EXTRTMETHOD OSIPADDR [stringToAddr](#) (const char *pAddrStr)

Protected Member Functions

- OSBOOL [isInitialized](#) ()

Protected Attributes

- OSRTSOCKET [mSocket](#)
- int [mInitStatus](#)
- int [mStatus](#)
- int [mRetryCount](#)
- OSBOOL [mOwner](#)

3.50.1 Detailed Description

Wrapper class for TCP/IP or UDP sockets.

3.50.2 Constructor & Destructor Documentation

3.50.2.1 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.

3.50.2.2 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.

3.50.2.3 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.

3.50.2.4 ~OSRTSocket()

```
EXTRTMETHOD OSRTSocket::~OSRTSocket ( )
```

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

3.50.3 Member Function Documentation

3.50.3.1 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>destIP</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](#)

3.50.3.2 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.

3.50.3.3 `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`

3.50.3.4 `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`

3.50.3.5 `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 \(\)](#)

3.50.3.6 `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

3.50.3.7 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.

3.50.3.8 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

3.50.3.9 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

3.50.3.10 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.

3.50.3.11 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

3.50.3.12 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.

3.50.3.13 getSocket()

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

This method returns the handle of the socket.

Returns

The handle of the socket.

3.50.3.14 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.

3.50.3.15 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

`::rxSocketListen`

3.50.3.16 `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

`::rxSocketRecv`

3.50.3.17 `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 <i>pdata</i> .

Returns

Completion status of operation:

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

See also

`::rtxSocketSend`

3.50.3.18 `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.
------------------	--

3.50.3.19 `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.
--------------	--------------

3.50.3.20 `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.

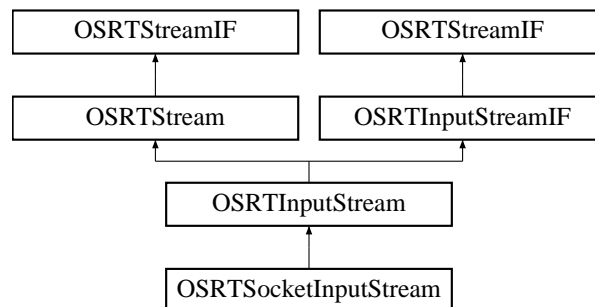
The documentation for this class was generated from the following file:

- [OSRTSocket.h](#)

3.51 OSRTSocketInputStream Class Reference

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

Inheritance diagram for OSRTSocketInputStream:



Public Member Functions

- EXTRTMETHOD [OSRTSocketInputStream](#) ([OSRTSocket](#) &socket)
- EXTRTMETHOD [OSRTSocketInputStream](#) ([OSRTContext](#) *pContext, [OSRTSocket](#) &socket)
- EXTRTMETHOD [OSRTSocketInputStream](#) (OSRTSOCKET socket, OSBOOL ownership=FALSE)
- [OSRTSocketInputStream](#) ([OSRTContext](#) *pContext, OSRTSOCKET socket, OSBOOL ownership=FALSE)
- virtual OSBOOL [isA](#) (StreamID id) const

Protected Attributes

- [OSRTSocket](#) mSocket

Additional Inherited Members

3.51.1 Detailed Description

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

3.51.2 Constructor & Destructor Documentation

3.51.2.1 OSRTSocketInputStream() [1/4]

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

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

Parameters

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

See also

[::rtxStreamSocketAttach](#)

3.51.2.2 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

[::rtxStreamSocketAttach](#)

3.51.2.3 OSRTSocketInputStream() [3/4]

```
EXTRMETHOD 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`

3.51.2.4 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

`::rxStreamSocketAttach`

3.51.3 Member Function Documentation

3.51.3.1 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](#).

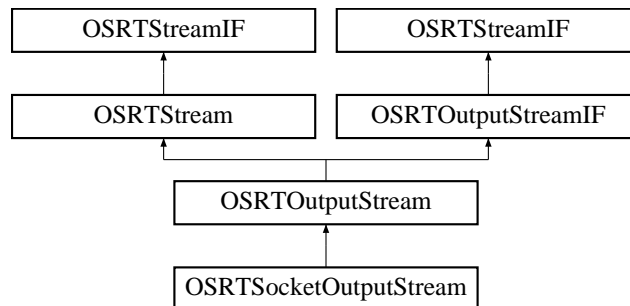
The documentation for this class was generated from the following file:

- [OSRTSocketInputStream.h](#)

3.52 OSRTSocketOutputStream Class Reference

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

Inheritance diagram for OSRTSocketOutputStream:



Public Member Functions

- EXTRMETHOD [OSRTSocketOutputStream](#) ([OSRTSocket](#) &socket)
- EXTRMETHOD [OSRTSocketOutputStream](#) ([OSRTContext](#) *pContext, [OSRTSocket](#) &socket)
- EXTRMETHOD [OSRTSocketOutputStream](#) (OSRTSOCKET socket, OSBOOL ownership=FALSE)
- [OSRTSocketOutputStream](#) ([OSRTContext](#) *pContext, OSRTSOCKET socket, OSBOOL ownership=FALSE)
- virtual OSBOOL [isA](#) (StreamID id) const

Protected Attributes

- [OSRSocket mSocket](#)

Additional Inherited Members

3.52.1 Detailed Description

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

3.52.2 Constructor & Destructor Documentation

3.52.2.1 OSRSocketOutputStream() [1/4]

```
EXTRTMETHOD OSRSocketOutputStream::OSRSocketOutputStream (  
    OSRSocket & socket )
```

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

Parameters

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

See also

[::rtxStreamSocketAttach](#)

3.52.2.2 OSRSocketOutputStream() [2/4]

```
EXTRTMETHOD OSRSocketOutputStream::OSRSocketOutputStream (  
    OSRContext * pContext,  
    OSRSocket & socket )
```

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

Parameters

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

See also

`::rxStreamSocketAttach`

3.52.2.3 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

`::rxStreamSocketAttach`

3.52.2.4 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

`::rxStreamSocketAttach`

3.52.3 Member Function Documentation

3.52.3.1 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](#).

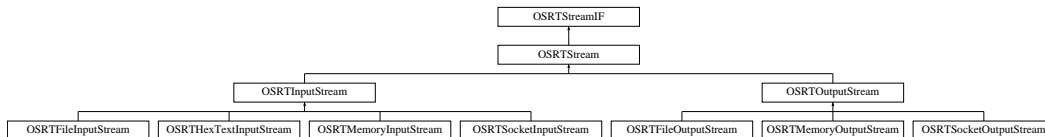
The documentation for this class was generated from the following file:

- [OSRTSocketOutputStream.h](#)

3.53 OSRTStream Class Reference

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

Inheritance diagram for OSRTStream:



Public Member Functions

- virtual EXTRTMETHOD `~OSRTStream ()`
- virtual EXTRTMETHOD `int close ()`
- virtual EXTRTMETHOD `int flush ()`
- virtual `OSRTCtxtPtr getContext ()`
- virtual `OSCTXT * getCtxPtr ()`
- virtual `char * getErrorInfo ()`
- virtual `char * getErrorInfo (char *pBuf, size_t &bufSize)`
- `int getStatus () const`
- `OSBOOL isInitialized ()`
- virtual EXTRTMETHOD `OSBOOL isOpened ()`
- void `printErrorInfo ()`
- void `resetErrorInfo ()`

Protected Member Functions

- EXTRTMETHOD **OSRTStream** ([OSRTContext](#) *pContext, OSBOOL attachStream=FALSE)
- EXTRTMETHOD **OSRTStream** ([OSRTStream](#) &original)
- EXTRTMETHOD [OSRTStream](#) ()
- EXTRTMETHOD char * **getErrorInfo** (size_t *pBufSize)

Protected Attributes

- OSRTCtxtHolder **mCtxtHolder**
- OSBOOL [mbAttached](#)
- int [mStatus](#)
- int [mInitStatus](#)

3.53.1 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.

3.53.2 Constructor & Destructor Documentation

3.53.2.1 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.

3.53.2.2 ~OSRTStream()

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

Virtual destructor. Closes the stream if it was opened.

3.53.3 Member Function Documentation

3.53.3.1 close()

```
virtual EXTRIMETHOD int OSRTStream::close ( ) [virtual]
```

Closes the input or output stream and releases any system resources associated with the stream. For output streams this function also flushes all internal buffers to the stream.

Returns

Completion status of operation:

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

See also

`::rtxStreamClose`

Implements [OSRTStreamIF](#).

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

3.53.3.2 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`

Implements [OSRTStreamIF](#).

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

3.53.3.3 `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.

Implements [OSRTStreamIF](#).

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

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

3.53.3.4 `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.

Implements [OSRTStreamIF](#).

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

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

3.53.3.5 `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](#).

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

3.53.3.6 `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](#).

3.53.3.7 `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.

References `OSRTStreamIF::isOpen()`.

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

3.53.3.8 `isOpen()`

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

Checks, is the stream opened or not.

Returns

TRUE, if the stream is opened, FALSE otherwise.

See also

`::rxStreamIsOpened`

Implements [OSRTStreamIF](#).

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

3.53.3.9 `printErrorInfo()`

```
void OSRTStream::printErrorInfo ( ) [inline]
```

The `printErrorInfo` method prints information on errors contained within the context.

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

3.53.3.10 `resetErrorInfo()`

```
void OSRTStream::resetErrorInfo ( ) [inline]
```

The `resetErrorInfo` method resets information on errors contained within the context.

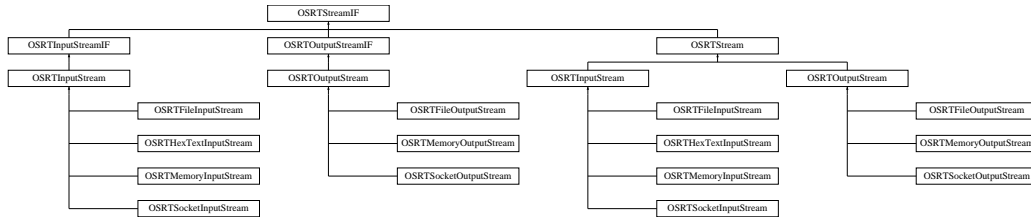
Referenced by [OSRTOutputStream::resetErrorInfo\(\)](#), and [OSRTInputStream::resetErrorInfo\(\)](#).

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

- [OSRTStream.h](#)

3.54 OSRTStreamIF Class Reference

Inheritance diagram for OSRTStreamIF:



Public Member Functions

- virtual int [close](#) ()=0
- virtual int [flush](#) ()=0
- virtual [OSRTCtxtPtr](#) [getContext](#) ()=0
- virtual OSCTXT * [getCtxtPtr](#) ()=0
- virtual OSBOOL [isOpened](#) ()=0

3.54.1 Member Function Documentation

3.54.1.1 [close\(\)](#)

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

Closes the input or output stream and releases any system resources associated with the stream. For output streams this function also flushes all internal buffers to the stream.

Returns

Completion status of operation:

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

See also

[::rtxStreamClose](#)

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

3.54.1.2 flush()

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

Flushes buffered data to the stream.

Returns

Completion status of operation:

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

See also

[::rtxStreamFlush](#)

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

3.54.1.3 isOpened()

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

Checks if the stream is opened or not.

Returns

TRUE, if the stream is opened, FALSE otherwise.

See also

[::rtxStreamIsOpened](#)

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

Referenced by [OSRTStream::getStatus\(\)](#), [OSRTInputStream::getStatus\(\)](#), and [OSRTOutputStream::isA\(\)](#).

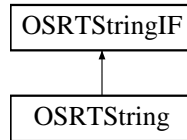
The documentation for this class was generated from the following file:

- [OSRTStreamIF.h](#)

3.55 OSRTString Class Reference

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

Inheritance diagram for OSRTString:



Public Member Functions

- [OSRTString](#) ()
- [OSRTString](#) (const char *strval)
- [OSRTString](#) (const OSUTF8CHAR *strval)
- [OSRTString](#) (const [OSRTString](#) &str)
- virtual [~OSRTString](#) ()
- virtual [OSRTStringIF](#) * [clone](#) ()
- const char * [data](#) () const
- virtual const char * [getValue](#) () const
- virtual const OSUTF8CHAR * [getUTF8Value](#) () const
- int [indexOf](#) (char ch) const
- size_t [length](#) () const
- virtual void [print](#) (const char *name)
- virtual EXTRTMETHOD void [setValue](#) (const char *str)
- virtual EXTRTMETHOD void [setValue](#) (const OSUTF8CHAR *str)
- bool [toInt](#) (OSINT32 &value) const
- bool [toSize](#) (OSSIZE &value) const
- bool [toUInt](#) (OSUINT32 &value) const
- bool [toUInt64](#) (OSUINT64 &value) const
- EXTRTMETHOD [OSRTString](#) & [operator=](#) (const [OSRTString](#) &original)
- **operator const char *** (void) const

Protected Attributes

- char * **mpValue**

Additional Inherited Members

3.55.1 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.

3.55.2 Constructor & Destructor Documentation

3.55.2.1 OSRTString() [1/4]

```
OSRTString::OSRTString ( )
```

The default constructor creates an empty string.

3.55.2.2 OSRTString() [2/4]

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

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

Parameters

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

3.55.2.3 OSRTString() [3/4]

```
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
---------------	----------------------------------

3.55.2.4 OSRTString() [4/4]

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

Copy constructor.

Parameters

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

3.55.2.5 ~OSRTString()

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

The destructor frees string memory using the standard 'delete' operator.

3.55.3 Member Function Documentation

3.55.3.1 clone()

```
virtual OSRTStringIF* OSRTString::clone ( ) [inline], [virtual]
```

This method creates a copy of the given string object.

Implements [OSRTStringIF](#).

3.55.3.2 data()

```
const char* OSRTString::data ( ) const [inline]
```

This method is a synonym for [getValue\(\)](#).

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

3.55.3.3 getUTF8Value()

```
virtual const OSUTF8CHAR* OSRTString::getUTF8Value ( ) const [inline], [virtual]
```

This method returns the pointer to UTF-8 null terminated string as a UTF-8 string.

Implements [OSRTStringIF](#).

3.55.3.4 `getValue()`

```
virtual const char* OSRTString::getValue ( ) const [inline], [virtual]
```

This method returns the pointer to UTF-8 null terminated string as a standard ASCII string.

Implements [OSRTStringIF](#).

3.55.3.5 `indexOf()`

```
int OSRTString::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.

3.55.3.6 `length()`

```
size_t OSRTString::length ( ) const [inline]
```

This method returns the length of the string.

3.55.3.7 `operator=()`

```
EXTRTMETHOD OSRTString& OSRTString::operator= (
    const OSRTString & original )
```

Assignment operator.

3.55.3.8 `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](#).

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

3.55.3.9 setValue() [1/2]

```
virtual EXTRIMETHOD void OSRTString::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](#).

3.55.3.10 setValue() [2/2]

```
virtual EXTRIMETHOD void OSRTString::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](#).

3.55.3.11 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.

3.55.3.12 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.

3.55.3.13 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.

3.55.3.14 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.

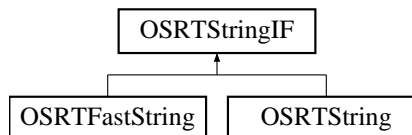
The documentation for this class was generated from the following file:

- [OSRTString.h](#)

3.56 OSRTStringIF Class Reference

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

Inheritance diagram for OSRTStringIF:



Public Member Functions

- virtual `~OSRTStringIF ()`
- virtual `OSRTStringIF * clone ()=0`
- virtual `const char * getValue () const =0`
- virtual `const OSUTF8CHAR * getUTF8Value () const =0`
- virtual `void print (const char *name)=0`
- virtual `void setValue (const char *str)=0`
- virtual `void setValue (const OSUTF8CHAR *utf8str)=0`

Protected Member Functions

- `OSRTStringIF ()`
- `OSRTStringIF (const char *)`
- `OSRTStringIF (const OSUTF8CHAR *)`

3.56.1 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).

3.56.2 Constructor & Destructor Documentation

3.56.2.1 OSRTStringIF() [1/3]

```
OSRTStringIF::OSRTStringIF ( ) [inline], [protected]
```

The default constructor creates an empty string.

3.56.2.2 OSRTStringIF() [2/3]

```
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
---	--------------------------------

3.56.2.3 OSRTStringIF() [3/3]

```
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
---	--------------------------------

3.56.2.4 ~OSRTStringIF()

```
virtual OSRTStringIF::~OSRTStringIF ( ) [inline], [virtual]
```

The destructor frees string memory using the standard 'delete' operator.

3.56.3 Member Function Documentation

3.56.3.1 clone()

```
virtual OSRTStringIF* OSRTStringIF::clone ( ) [pure virtual]
```

This method creates a copy of the given string object.

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

3.56.3.2 getUTF8Value()

```
virtual const OSUTF8CHAR* OSRTStringIF::getUTF8Value ( ) const [pure virtual]
```

This method returns the pointer to UTF-8 null terminated string as a UTF-8 string.

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

3.56.3.3 getValue()

```
virtual const char* OSRTStringIF::getValue ( ) const [pure virtual]
```

This method returns the pointer to UTF-8 null terminated string as a standard ASCII string.

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

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

3.56.3.4 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](#).

3.56.3.5 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\(\)](#).

3.56.3.6 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](#).

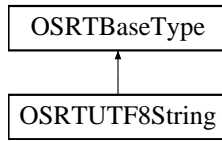
The documentation for this class was generated from the following file:

- [OSRTStringIF.h](#)

3.57 OSRTUTF8String Class Reference

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

Inheritance diagram for OSRTUTF8String:



Public Member Functions

- [OSRTUTF8String](#) ()
- [OSRTUTF8String](#) (const char *strval)
- [OSRTUTF8String](#) (const OSUTF8CHAR *strval)
- [OSRTUTF8String](#) (const [OSRTUTF8String](#) &str)
- virtual [~OSRTUTF8String](#) ()
- [OSRTBaseType](#) * [clone](#) () const
- void [copyValue](#) (const char *str)
- const char * [c_str](#) () const
- const char * [getValue](#) () const
- void [print](#) (const char *name)
- void [setValue](#) (const char *str)
- [OSRTUTF8String](#) & [operator=](#) (const [OSRTUTF8String](#) &original)

3.57.1 Detailed Description

UTF-8 string. This is the base class for generated C++ data type classes for XSD string types (string, token, NMTOKEN, etc.).

3.57.2 Constructor & Destructor Documentation

3.57.2.1 OSRTUTF8String() [1/4]

```
OSRTUTF8String::OSRTUTF8String ( )
```

The default constructor creates an empty string.

3.57.2.2 OSRTUTF8String() [2/4]

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

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

Parameters

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

3.57.2.3 OSRTUTF8String() [3/4]

```
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
---------------	----------------

3.57.2.4 OSRTUTF8String() [4/4]

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

Copy constructor.

Parameters

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

3.57.2.5 ~OSRTUTF8String()

```
virtual OSRTUTF8String::~OSRTUTF8String ( ) [virtual]
```

The destructor frees string memory if the memory ownership flag is set.

3.57.3 Member Function Documentation

3.57.3.1 c_str()

```
const char* OSRTUTF8String::c_str ( ) const [inline]
```

This method returns the pointer to C null terminated string.

3.57.3.2 clone()

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

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

Reimplemented from [OSRTBaseType](#).

3.57.3.3 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.
------------	-----------------------------

3.57.3.4 getValue()

```
const char* OSRTUTF8String::getValue ( ) const [inline]
```

This method returns the pointer to UTF-8 null terminated string.

3.57.3.5 operator=()

```
OSRTUTF8String& OSRTUTF8String::operator= (
    const OSRTUTF8String & original )
```

Assignment operator.

3.57.3.6 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.
-------------	--------------------------------------

3.57.3.7 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](#)

Chapter 4

File Documentation

4.1 ASN1CBitStr.h File Reference

```
#include "rtsrc/asn1CppType.h"
```

Classes

- class [ASN1CBitStrSizeHolder](#)
- class [ASN1CBitStrSizeHolder8](#)
- class [ASN1CBitStrSizeHolder16](#)
- class [ASN1CBitStrSizeHolder32](#)
- class [ASN1CBitStr](#)

4.1.1 Detailed Description

Bit string control class definitions.

4.2 ASN1CGeneralizedTime.h File Reference

```
#include "rtsrc/ASN1CTime.h"
```

Classes

- class [ASN1CGeneralizedTime](#)

4.2.1 Detailed Description

GeneralizedTime control class definition.

4.3 ASN1Context.h File Reference

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

Classes

- class [ASN1Context](#)

4.3.1 Detailed Description

Common C++ type and class definitions.

4.4 asn1CppEvtHndlr.h File Reference

```
#include "rtsrc/asn1type.h"
```

Classes

- class [Asn1NamedEventHandler](#)
- class [Asn1NullEventHandler](#)
- class [Asn1ErrorHandler](#)

Macros

- #define [OS_UNUSED_ARG](#)(arg) (void)arg

Variables

- class EXTRTCLASS [ASN1MessageBuffer](#)

4.4.1 Detailed Description

Named event handler base class. The `Asn1Named Event Handler` class is an abstract base class from which user-defined event handlers are derived. This class contains pure virtual function definitions for all of the methods that must be implemented to create a customized event handler class.

4.5 `asn1CppType.h` File Reference

```
#include <new>
#include "rtxsrc/rtxMemory.h"
#include "rtxsrc/rtxDiag.h"
#include "rtxsrc/rtxError.h"
#include "rtxsrc/rtxMemBuf.h"
#include "rtsrc/asn1CppTypeEvtHndlr64.h"
#include "rtsrc/ASN1Context.h"
#include "rtxsrc/OSRTMsgBuf.h"
#include "rtsrc/ASN1TOctStr.h"
#include "rtsrc/ASN1TOctStr64.h"
#include "rtsrc/ASN1TObjId.h"
```

Classes

- class [ASN1MessageBuffer](#)
- class [ASN1CType](#)
- struct [ASN1TDynBitStr](#)
- struct [ASN1TDynBitStr64](#)
- struct [ASN1TBitStr32](#)
- struct [ASN1TBMPString](#)
- struct [ASN1TUniversalString](#)
- struct [ASN1TOpenType](#)
- struct [Asn1TObject](#)
- struct [ASN1TSeqExt](#)
- struct [ASN1TPDU](#)
- struct [ASN1TSeqOfList](#)
- struct [ASN1TPDUSeqOfList](#)

Macros

- `#define ASN1TRY try`
- `#define ASN1RTLTHROW(stat) exit (-1)`
- `#define ASN1THROW(ex) exit (-1)`
- `#define ASN1CATCH(exType, ex, body) if (0) { body; }`

Typedefs

- typedef [Asn1TObject](#) **ASN1TObject**

4.5.1 Detailed Description

Common C++ type and class definitions.

4.5.2 Macro Definition Documentation

4.5.2.1 ASN1CATCH

```
#define ASN1CATCH(  
    exType,  
    ex,  
    body ) if (0) { body; }
```

This is a no-op, an `if(0) {...}` clause that will never be executed. It is defined for compatibility only.

Parameters

<i>exType</i>	The type of exception.
<i>ex</i>	The exception value.
<i>body</i>	The body of code that is no longer executed.

4.5.2.2 ASN1RTLTHROW

```
#define ASN1RTLTHROW(  
    stat ) exit (-1)
```

Exits the program with a -1 status.

Parameters

<i>stat</i>	This is ignored.
-------------	------------------

4.5.2.3 ASN1THROW

```
#define ASN1THROW(  
    ex ) exit (-1)
```

Exits the program with a -1 status.

Parameters

<i>ex</i>	This is ignored.
-----------	------------------

4.5.2.4 ASN1TRY

```
#define ASN1TRY try
```

Defined as "try" for compatibility only.

4.6 ASN1CSeqOfList.h File Reference

```
#include <stdlib.h>  
#include "rtsrc/asn1CppTypes.h"
```

Classes

- class [ASN1CSeqOfListIterator](#)
- class [ASN1CSeqOfList](#)

Variables

- class EXTRTCLASS **ASN1CSeqOfList**

4.6.1 Detailed Description

[ASN1CSeqOfList](#) linked list control class definition.

4.7 ASN1CTime.h File Reference

```
#include <time.h>  
#include "rtsrc/asn1CppTypes.h"  
#include "rtsrc/ASN1TTime.h"
```

Classes

- class [ASN1CTime](#)

Macros

- #define [LOG_TMERR](#)(pctxt, stat) ((pctxt != 0) ? LOG_RTERR (pctxt, stat) : stat)

4.7.1 Detailed Description

[ASN1CTime](#) abstract class definition. This is used as the base class for other ASN.1 time class definitions.

4.8 ASN1CUTCTime.h File Reference

```
#include "rtsrc/ASN1CTime.h"
```

Classes

- class [ASN1CUTCTime](#)

4.8.1 Detailed Description

[ASN1CUTCTime](#) control class definition.

4.9 asn1ErrCodes.h File Reference

Macros

- #define [ASN_OK_FRAG](#) 2
- #define [ASN_E_BASE](#) -100
- #define [ASN_E_INVOBJID](#) ([ASN_E_BASE](#))
- #define [ASN_E_INVLEN](#) ([ASN_E_BASE](#)-1)
- #define [ASN_E_BADTAG](#) ([ASN_E_BASE](#)-2)
- #define [ASN_E_INVBINS](#) ([ASN_E_BASE](#)-3)
- #define [ASN_E_INVINDEX](#) ([ASN_E_BASE](#)-4)
- #define [ASN_E_INVTCVAL](#) ([ASN_E_BASE](#)-5)
- #define [ASN_E_CONCMODF](#) ([ASN_E_BASE](#)-6)
- #define [ASN_E_ILLSTATE](#) ([ASN_E_BASE](#)-7)
- #define [ASN_E_NOTPDU](#) ([ASN_E_BASE](#)-8)
- #define [ASN_E_UNDEFTYP](#) ([ASN_E_BASE](#)-9)
- #define [ASN_E_INVPERENC](#) ([ASN_E_BASE](#)-10)
- #define [ASN_E_NOTINSEQ](#) ([ASN_E_BASE](#)-11)
- #define [ASN_E_BAD_ALIGN](#) ([ASN_E_BASE](#)-12)
- #define [ASN_E_UNKNOWNPDU](#) ([ASN_E_BASE](#)-13)
- #define [ASN_E_NOTCANON](#) ([ASN_E_BASE](#)-14)

4.9.1 Detailed Description

List of numeric status codes that can be returned by ASN1C run-time functions and generated code.

4.10 ASN1TObjld.h File Reference

```
#include "rtsrc/asn1type.h"
```

Classes

- struct [ASN1TObjld](#)

Functions

- int [operator==](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator==](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator!=](#) (const [ASN1TObjld](#) &lhs, const [ASN1TObjld](#) &rhs)
- int [operator!=](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator!=](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator!=](#) (const [ASN1TObjld](#) &lhs, const char *dotted_oid_string)
- int [operator<](#) (const [ASN1TObjld](#) &lhs, const [ASN1TObjld](#) &rhs)
- int [operator<](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator<](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator<](#) (const [ASN1TObjld](#) &lhs, const char *dotted_oid_string)
- int [operator<=](#) (const [ASN1TObjld](#) &lhs, const [ASN1TObjld](#) &rhs)
- int [operator<=](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator<=](#) (const [ASN1TObjld](#) &lhs, const char *dotted_oid_string)
- int [operator<=](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator>](#) (const [ASN1TObjld](#) &lhs, const [ASN1TObjld](#) &rhs)
- int [operator>](#) (const [ASN1TObjld](#) &lhs, const char *dotted_oid_string)
- int [operator>](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator>](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator>=](#) (const [ASN1TObjld](#) &lhs, const [ASN1TObjld](#) &rhs)
- int [operator>=](#) (const [ASN1TObjld](#) &lhs, const char *dotted_oid_string)
- int [operator>=](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator>=](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- [ASN1TObjld operator+](#) (const [ASN1TObjld](#) &lhs, const [ASN1TObjld](#) &rhs)

4.10.1 Detailed Description

ASN.1 object identifier class definition.

4.11 ASN1TOctStr.h File Reference

```
#include "rtsrc/asn1type.h"
```

Classes

- struct [ASN1TDynOctStr](#)

Functions

- int [operator==](#) (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int [operator==](#) (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int [operator==](#) (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int [operator==](#) (const [ASN1DynOctStr](#) &lhs, const char *string)
- int [operator!=](#) (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int [operator!=](#) (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int [operator!=](#) (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int [operator!=](#) (const [ASN1DynOctStr](#) &lhs, const char *string)
- int [operator<](#) (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int [operator<](#) (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int [operator<](#) (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int [operator<](#) (const [ASN1DynOctStr](#) &lhs, const char *string)
- int [operator<=](#) (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int [operator<=](#) (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int [operator<=](#) (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int [operator<=](#) (const [ASN1DynOctStr](#) &lhs, const char *string)
- int [operator>](#) (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int [operator>](#) (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int [operator>](#) (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int [operator>](#) (const [ASN1DynOctStr](#) &lhs, const char *string)
- int [operator>=](#) (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int [operator>=](#) (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int [operator>=](#) (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int [operator>=](#) (const [ASN1DynOctStr](#) &lhs, const char *string)

4.11.1 Detailed Description

ASN.1 OCTET string class definition.

4.12 ASN1TTime.h File Reference

```
#include <time.h>  
#include "rtsrc/asn1CppType.h"
```

Classes

- class [ASN1TTime](#)
- class [ASN1TGeneralizedTime](#)
- class [ASN1TUTCTime](#)

Macros

- #define [MAX_TIMESTR_SIZE](#) 64
- #define [LOG_TTMERR](#)(stat) (mStatus = stat, stat)

4.12.1 Macro Definition Documentation

4.12.1.1 LOG_TTMERR

```
#define LOG_TTMERR(  
    stat ) (mStatus = stat, stat)
```

A macro for setting the error status of a time structure.

Parameters

<i>stat</i>	The error status.
-------------	-------------------

Returns

The error status.

4.12.1.2 MAX_TIMESTR_SIZE

```
#define MAX_TIMESTR_SIZE 64
```

Sets the max time string size at 64 characters.

4.13 OSRTBaseType.h File Reference

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

Classes

- class [OSRTBaseType](#)

4.13.1 Detailed Description

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

4.14 OSRTContext.h File Reference

```
#include "rtxsrc/rtxContext.h"
#include "rtxsrc/rtxDiag.h"
#include "rtxsrc/rtxError.h"
#include "rtxsrc/rtxMemory.h"
```

Classes

- class [OSRTContext](#)
- class [OSRTCtxtPtr](#)

Functions

- void * [operator new](#) (size_t nbytes, OSCTXT *pctx)
- void [operator delete](#) (void *pmem, OSCTXT *pctx)

4.14.1 Detailed Description

C++ run-time context class definition.

4.14.2 Function Documentation

4.14.2.1 operator delete()

```
void operator delete (
    void * pmem,
    OSCTXT * pctx )
```

Custom placement delete function to free memory using context memory-management functions.

4.14.2.2 operator new()

```
void* operator new (
    size_t nbytes,
    OSCTXT * pctxt )
```

Custom placement new function to allocate memory using context memory-management functions.

4.15 OSRTFastString.h File Reference

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

Classes

- class [OSRTFastString](#)

4.15.1 Detailed Description

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

4.16 OSRTFileInputStream.h File Reference

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

Classes

- class [OSRTFileInputStream](#)

4.16.1 Detailed Description

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

4.17 OSRTFileOutputStream.h File Reference

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

Classes

- class [OSRTFileOutputStream](#)

4.17.1 Detailed Description

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

4.18 OSRTHexTextInputStream.h File Reference

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

Classes

- class [OSRTHexTextInputStream](#)

4.18.1 Detailed Description

C++ hexadecimal text input stream filter class.

4.19 OSRTInputStream.h File Reference

```
#include "rtxsrc/OSRTInputStreamIF.h"  
#include "rtxsrc/OSRTStream.h"
```

Classes

- class [OSRTInputStream](#)

4.19.1 Detailed Description

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

4.20 OSRTInputStreamIF.h File Reference

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

Classes

- class [OSRTInputStreamIF](#)
- class [OSRTInputStreamPtr](#)

4.20.1 Detailed Description

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

4.21 OSRTMemoryInputStream.h File Reference

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

Classes

- class [OSRTMemoryInputStream](#)

4.21.1 Detailed Description

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

4.22 OSRTMemoryOutputStream.h File Reference

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

Classes

- class [OSRTMemoryOutputStream](#)

4.22.1 Detailed Description

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

4.23 OSRTMsgBuf.h File Reference

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

Classes

- class [OSRTMessageBuffer](#)

4.23.1 Detailed Description

C++ run-time message buffer class definition.

4.24 OSRTMsgBufIF.h File Reference

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

Classes

- class [OSRTMessageBufferIF](#)

4.24.1 Detailed Description

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

4.25 OSRTOutputStream.h File Reference

```
#include "rtxsrc/OSRTOutputStreamIF.h"  
#include "rtxsrc/OSRTStream.h"
```

Classes

- class [OSRTOutputStream](#)

4.25.1 Detailed Description

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

4.26 OSRTOutputStreamIF.h File Reference

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

Classes

- class [OSRTOutputStreamIF](#)
- class [OSRTOutputStreamPtr](#)

4.26.1 Detailed Description

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

4.27 OSRTSocket.h File Reference

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

Classes

- class [OSRTSocket](#)

4.27.1 Detailed Description

TCP/IP or UDP socket class definitions.

4.28 OSRTSocketInputStream.h File Reference

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

Classes

- class [OSRTSocketInputStream](#)

4.28.1 Detailed Description

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

4.29 OSRTSocketOutputStream.h File Reference

```
#include "rtxsrc/OSRTSocket.h"  
#include "rtxsrc/OSRTOutputStream.h"
```

Classes

- class [OSRTSocketOutputStream](#)

4.29.1 Detailed Description

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

4.30 OSRTStream.h File Reference

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

Classes

- class [OSRTStream](#)

4.30.1 Detailed Description

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

4.31 OSRTStreamIF.h File Reference

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

Classes

- class [OSRTStreamIF](#)

4.31.1 Detailed Description

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

4.32 OSRTString.h File Reference

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

Classes

- class [OSRTString](#)

4.32.1 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.

4.33 OSRTStringIF.h File Reference

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

Classes

- class [OSRTStringIF](#)

4.33.1 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.

4.34 OSRTUTF8String.h File Reference

```
#include "rtxsrc/OSRTBaseType.h"  
#include "rtxsrc/rtxPrint.h"  
#include "rtxsrc/rtxUTF8.h"
```

Classes

- class [OSRTUTF8String](#)

4.34.1 Detailed Description

C++ UTF-8 string class definition.

Index

- [_ref](#)
 - [OSRTContext, 205](#)
 - [_unref](#)
 - [OSRTContext, 205](#)
 - [~ASN1CTime](#)
 - [ASN1CTime, 91](#)
 - [~ASN1CType](#)
 - [ASN1CType, 111](#)
 - [~ASN1MessageBuffer](#)
 - [ASN1MessageBuffer, 126](#)
 - [~ASN1TObjId](#)
 - [ASN1TObjId, 165](#)
 - [~ASN1TPDU](#)
 - [ASN1TPDU, 172](#)
 - [~ASN1TTime](#)
 - [ASN1TTime, 179](#)
 - [~OSRTContext](#)
 - [OSRTContext, 204](#)
 - [~OSRTCtxtPtr](#)
 - [OSRTCtxtPtr, 213](#)
 - [~OSRTFastString](#)
 - [OSRTFastString, 217](#)
 - [~OSRTHexTextInputStream](#)
 - [OSRTHexTextInputStream, 227](#)
 - [~OSRTInputStream](#)
 - [OSRTInputStream, 230](#)
 - [~OSRTInputStreamIF](#)
 - [OSRTInputStreamIF, 239](#)
 - [~OSRTMessageBuffer](#)
 - [OSRTMessageBuffer, 251](#)
 - [~OSRTMessageBufferIF](#)
 - [OSRTMessageBufferIF, 257](#)
 - [~OSRTOutputStream](#)
 - [OSRTOutputStream, 262](#)
 - [~OSRTOutputStreamIF](#)
 - [OSRTOutputStreamIF, 268](#)
 - [~OSRTSocket](#)
 - [OSRTSocket, 271](#)
 - [~OSRTStream](#)
 - [OSRTStream, 288](#)
 - [~OSRTString](#)
 - [OSRTString, 297](#)
 - [~OSRTStringIF](#)
 - [OSRTStringIF, 302](#)
 - [~OSRTUTF8String](#)
 - [OSRTUTF8String, 306](#)
- [ASN.1 Stream Classes, 30](#)
- [ASN.1 Type \(ASN1T_\) Base Classes, 6](#)
 - [operator!=, 7–10](#)
 - [operator<, 11–14](#)
 - [operator<=, 15–18](#)
 - [operator>, 21–24](#)
 - [operator>=, 24–27](#)
 - [operator+, 11](#)
 - [operator==, 18–20](#)
- [ASN1CATCH](#)
 - [asn1CppType.h, 314](#)
- [ASN1CBitStr, 41](#)
 - [ASN1CBitStr, 43](#)
 - [cardinality, 44](#)
 - [change, 44](#)
 - [clear, 45, 46](#)
 - [doAnd, 46, 47](#)
 - [doAndNot, 47, 48](#)
 - [doOr, 49, 50](#)
 - [doXor, 50, 52](#)
 - [get, 53](#)
 - [getBytes, 54](#)
 - [invert, 54, 55](#)
 - [isEmpty, 55](#)
 - [isSet, 56](#)
 - [length, 56](#)
 - [operator ASN1TDynBitStr, 56](#)
 - [operator ASN1TDynBitStr *, 57](#)
 - [set, 57, 58](#)
 - [shiftLeft, 58](#)
 - [shiftRight, 59](#)
 - [size, 59](#)
 - [unusedBitsInLastUnit, 59](#)
- [ASN1CBitStr.h, 311](#)
- [ASN1CBitStrSizeHolder, 60](#)
- [ASN1CBitStrSizeHolder16, 61](#)
- [ASN1CBitStrSizeHolder32, 61](#)
- [ASN1CBitStrSizeHolder8, 62](#)
- [ASN1CGeneralizedTime, 63](#)
 - [ASN1CGeneralizedTime, 64–66](#)
 - [compileString, 66](#)
 - [getCentury, 66](#)
 - [setCentury, 67](#)
 - [setTime, 67](#)

- ASN1CGeneralizedTime.h, 311
- ASN1CSeqOfList, 69
 - ASN1CSeqOfList, 71, 72, 74
 - append, 74
 - appendArray, 75
 - appendArrayCopy, 75
 - clear, 76
 - contains, 76
 - freeMemory, 76
 - get, 76
 - getFirst, 77
 - getLast, 77
 - indexOf, 77
 - init, 78
 - insert, 78
 - isEmpty, 78
 - iterator, 78
 - iteratorFrom, 79
 - iteratorFromLast, 79
 - operator[], 79
 - remove, 80
 - removeFirst, 81
 - removeLast, 81
 - set, 81
 - size, 82
 - toArray, 82
- ASN1CSeqOfList.h, 315
- ASN1CSeqOfListIterator, 83
 - hasNext, 84
 - hasPrev, 84
 - insert, 85
 - next, 85
 - prev, 86
 - remove, 86
 - set, 86
- ASN1CTime, 87
 - ~ASN1CTime, 91
 - ASN1CTime, 89–91
 - clear, 91
 - compileString, 92
 - equals, 92
 - getDay, 92
 - getDiff, 93
 - getDiffHour, 93
 - getDiffMinute, 94
 - getFraction, 94
 - getFractionAsDouble, 94
 - getFractionLen, 95
 - getFractionStr, 95
 - getHour, 95
 - getMinute, 96
 - getMonth, 96
 - getSecond, 97
 - getTime, 97
 - getTimeString, 97
 - getTimeStringLen, 98
 - getUTC, 98
 - getYear, 99
 - operator!=, 99
 - operator<, 99
 - operator<=, 99
 - operator>, 100
 - operator>=, 101
 - operator=, 100
 - operator==, 100
 - parseString, 101
 - setDER, 102
 - setDay, 101
 - setDiff, 102, 103
 - setDiffHour, 103
 - setFraction, 104, 105
 - setHour, 105
 - setMinute, 106
 - setMonth, 106
 - setSecond, 107
 - setTime, 107
 - setUTC, 108
 - setYear, 108
- ASN1CTime.h, 315
- ASN1CTYPE, 109
 - ~ASN1CTYPE, 111
 - ASN1CTYPE, 110, 111
 - append, 111
 - Decode, 112
 - DecodeFrom, 112
 - Encode, 113
 - EncodeTo, 113
 - getContext, 113
 - getCtxtPtr, 114
 - getErrorText, 114
 - getStatus, 114
 - memAlloc, 115
 - memAllocZ, 115
 - memFreeAll, 116
 - memFreePtr, 116
 - memRealloc, 116
 - memReset, 117
 - mpContext, 118
 - mpMsgBuf, 118
 - printErrorInfo, 117
 - resetError, 117
 - setDiag, 117
 - setRunTimeKey, 118
- ASN1CUTCTime, 119
 - ASN1CUTCTime, 120
 - compileString, 121
 - getFraction, 121
 - setTime, 122

ASN1CUTCTime.h, 316
 ASN1Context, 68
 ASN1Context, 68
 setRunTimeKey, 69
 ASN1Context.h, 312
 ASN1MessageBuffer, 124
 ~ASN1MessageBuffer, 126
 ASN1MessageBuffer, 125
 addEventHandler, 126
 CStringToBMPString, 126
 getAppInfo, 127
 getBitOffset, 127
 getMsgLen, 127
 initBuffer, 127, 128
 isA, 128
 Named Event Handlers, 34
 removeEventHandler, 129
 resetErrorInfo, 129
 setAppInfo, 129
 setErrorHandler, 129
 setRunTimeKey, 130
 setStatus, 130
 ASN1RTLTHROW
 asn1CppType.h, 314
 ASN1TBMPString, 150
 ASN1TBMPString, 151
 ASN1TBitStr32, 149
 ASN1TBitStr32, 149, 150
 ASN1TDynBitStr, 151
 ASN1TDynBitStr, 151–153
 ASN1TDynBitStr64, 153
 ASN1TDynBitStr64, 154
 ASN1TDynOctStr, 154
 ASN1TDynOctStr, 155, 156
 nCompare, 156
 operator=, 157
 toHexString, 157
 toString, 159
 ASN1TGeneralizedTime, 159
 ASN1TGeneralizedTime, 160, 161
 compileString, 161
 getCentury, 161
 parseString, 162
 setCentury, 162
 setTime, 162
 ASN1THROW
 asn1CppType.h, 314
 ASN1TObjId, 164
 ~ASN1TObjId, 165
 ASN1TObjId, 165, 166
 nCompare, 166
 operator+=", 167, 168
 operator=, 168, 169
 RnCompare, 169
 set_data, 169
 toString, 170
 trim, 170
 ASN1TObjId.h, 317
 ASN1TOctStr.h, 318
 ASN1TOpenType, 170
 ASN1TOpenType, 171
 ASN1TPDUSeqOfList, 173
 ASN1TPDUSeqOfList, 173
 ASN1TPDU, 171
 ~ASN1TPDU, 172
 mpContext, 172
 setContext, 172
 ASN1TRY
 asn1CppType.h, 315
 ASN1TSeqExt, 174
 ASN1TSeqExt, 174
 ASN1TSeqOfList, 175
 ASN1TSeqOfList, 175
 ASN1TTime, 176
 ~ASN1TTime, 179
 ASN1TTime, 178
 clear, 179
 compileString, 179
 equals, 180
 getDay, 180
 getDiff, 180
 getDiffHour, 180
 getDiffMinute, 181
 getFraction, 181
 getFractionAsDouble, 181
 getFractionLen, 182
 getFractionStr, 182
 getHour, 182
 getMinute, 183
 getMonth, 183
 getSecond, 183
 getTime, 184
 getUTC, 184
 getYear, 184
 mDay, 194
 mDiffHour, 195
 mDiffMin, 195
 mHour, 195
 mMinute, 195
 mMonth, 195
 mSecFracLen, 195
 mSecFraction, 195
 mSecond, 196
 mStatus, 196
 mYear, 196
 mbDerRules, 194
 mbUtcFlag, 194
 parseString, 185

- setDER, [186](#)
- setDay, [185](#)
- setDiff, [186](#), [187](#)
- setDiffHour, [187](#)
- setFraction, [188](#), [189](#)
- setHour, [189](#)
- setMinute, [190](#)
- setMonth, [190](#)
- setSecond, [191](#)
- setTime, [191](#)
- setUTC, [192](#)
- setYear, [192](#)
- toString, [193](#), [194](#)
- ASN1TTime.h, [318](#)
 - LOG_TTMERR, [319](#)
 - MAX_TIMESTR_SIZE, [319](#)
- ASN1TUTCTime, [197](#)
 - ASN1TUTCTime, [198](#), [199](#)
 - clear, [199](#)
 - compileString, [199](#)
 - getFraction, [200](#)
 - parseString, [200](#)
 - setFraction, [200](#)
 - setTime, [201](#)
 - setUTC, [201](#)
 - setYear, [202](#)
- ASN1TUniversalString, [196](#)
 - ASN1TUniversalString, [197](#)
- ASN_E_BAD_ALIGN
 - Run-time error status codes., [36](#)
- ASN_E_BADTAG
 - Run-time error status codes., [36](#)
- ASN_E_BASE
 - Run-time error status codes., [37](#)
- ASN_E_CONCMODF
 - Run-time error status codes., [37](#)
- ASN_E_ILLSTATE
 - Run-time error status codes., [37](#)
- ASN_E_INVBINS
 - Run-time error status codes., [37](#)
- ASN_E_INVINDEX
 - Run-time error status codes., [37](#)
- ASN_E_INVLEN
 - Run-time error status codes., [37](#)
- ASN_E_INVOBJID
 - Run-time error status codes., [38](#)
- ASN_E_INVPERENC
 - Run-time error status codes., [38](#)
- ASN_E_INVTCVAL
 - Run-time error status codes., [38](#)
- ASN_E_NOTCANON
 - Run-time error status codes., [38](#)
- ASN_E_NOTINSEQ
 - Run-time error status codes., [38](#)
- ASN_E_NOTPDU
 - Run-time error status codes., [38](#)
- ASN_E_UNDEFTYP
 - Run-time error status codes., [39](#)
- ASN_E_UNKNOWNPDU
 - Run-time error status codes., [39](#)
- ASN_OK_FRAG
 - Run-time error status codes., [39](#)
- accept
 - OSRSocket, [272](#)
- addEventHandler
 - ASN1MessageBuffer, [126](#)
 - Asn1NamedEventHandler, [132](#)
- addrToString
 - OSRSocket, [272](#)
- append
 - ASN1CSeqOfList, [74](#)
 - ASN1CType, [111](#)
- appendArray
 - ASN1CSeqOfList, [75](#)
- appendArrayCopy
 - ASN1CSeqOfList, [75](#)
- asn1CppEvtHndlr.h, [312](#)
- asn1CppTypes.h, [313](#)
 - ASN1CATCH, [314](#)
 - ASN1RTLTHROW, [314](#)
 - ASN1THROW, [314](#)
 - ASN1TRY, [315](#)
- asn1ErrCodes.h, [316](#)
- Asn1ErrorHandler, [122](#)
 - error, [123](#)
 - setErrorHandler, [123](#)
- Asn1NamedEventHandler, [131](#)
 - addEventHandler, [132](#)
 - bitStrValue, [132](#)
 - boolValue, [133](#)
 - charStrValue, [133–135](#)
 - endElement, [135](#)
 - enumValue, [136](#)
 - int64Value, [136](#)
 - intValue, [136](#)
 - invokeBitStrValue, [137](#)
 - invokeBoolValue, [137](#)
 - invokeCharStrValue, [138](#), [139](#)
 - invokeEndElement, [139](#)
 - invokeEnumValue, [140](#)
 - invokeInt64Value, [140](#)
 - invokeIntValue, [140](#)
 - invokeNullValue, [141](#)
 - invokeOctStrValue, [141](#)
 - invokeOidValue, [141](#)
 - invokeOpenTypeValue, [142](#)
 - invokeRealValue, [142](#)
 - invokeStartElement, [142](#)

- invokeUInt64Value, [143](#)
- invokeUIntValue, [143](#)
- nullValue, [143](#)
- octStrValue, [144](#)
- oidValue, [144](#)
- openTypeValue, [145](#)
- realValue, [145](#)
- removeEventHandler, [145](#)
- startElement, [146](#)
- uint64Value, [146](#)
- uintValue, [147](#)
- Asn1NullEventHandler, [147](#)
 - endElement, [148](#)
 - startElement, [148](#)
- Asn1TObject, [163](#)
 - Asn1TObject, [164](#)
- bind
 - OSRSocket, [273](#), [274](#)
- bindUrl
 - OSRSocket, [274](#)
- bitStrValue
 - Asn1NamedEventHandler, [132](#)
- blockingRead
 - OSRSocket, [275](#)
- boolValue
 - Asn1NamedEventHandler, [133](#)
- C++ Run-Time Classes, [3](#)
- c_str
 - OSRTUTF8String, [306](#)
- CStringToBMPString
 - ASN1MessageBuffer, [126](#)
- cardinality
 - ASN1CBitStr, [44](#)
- change
 - ASN1CBitStr, [44](#)
- charStrValue
 - Asn1NamedEventHandler, [133–135](#)
- clear
 - ASN1CBitStr, [45](#), [46](#)
 - ASN1CSeqOfList, [76](#)
 - ASN1CTime, [91](#)
 - ASN1TTime, [179](#)
 - ASN1TUTCTime, [199](#)
- clone
 - OSRTFastString, [217](#)
 - OSRTString, [297](#)
 - OSRTStringIF, [303](#)
 - OSRTUTF8String, [307](#)
- close
 - OSRTInputStream, [230](#)
 - OSRTOutputStream, [262](#)
 - OSRSocket, [275](#)
 - OSRTStream, [288](#)
 - OSRTStreamIF, [293](#)
- compileString
 - ASN1CGeneralizedTime, [66](#)
 - ASN1CTime, [92](#)
 - ASN1CUTCTime, [121](#)
 - ASN1TGeneralizedTime, [161](#)
 - ASN1TTime, [179](#)
 - ASN1TUTCTime, [199](#)
- connect
 - OSRSocket, [276](#)
- connectTimed
 - OSRSocket, [276](#)
- connectUrl
 - OSRSocket, [277](#)
- contains
 - ASN1CSeqOfList, [76](#)
- Context Management Classes, [28](#)
- Control (ASN1C_) Base Classes, [5](#)
- copyValue
 - OSRTUTF8String, [307](#)
- currentPos
 - OSRTInputStream, [230](#)
 - OSRTInputStreamIF, [239](#)
- data
 - OSRTString, [297](#)
- Date and Time Runtime Classes, [29](#)
 - LOG_TMERR, [29](#)
- Decode
 - ASN1CType, [112](#)
- DecodeFrom
 - ASN1CType, [112](#)
- doAnd
 - ASN1CBitStr, [46](#), [47](#)
- doAndNot
 - ASN1CBitStr, [47](#), [48](#)
- doOr
 - ASN1CBitStr, [49](#), [50](#)
- doXor
 - ASN1CBitStr, [50](#), [52](#)
- Encode
 - ASN1CType, [113](#)
- EncodeTo
 - ASN1CType, [113](#)
- endElement
 - Asn1NamedEventHandler, [135](#)
 - Asn1NullEventHandler, [148](#)
- enumValue
 - Asn1NamedEventHandler, [136](#)
- equals
 - ASN1CTime, [92](#)
 - ASN1TTime, [180](#)
- error
 - Asn1ErrorHandler, [123](#)

- flush
 - OSRTInputStream, [230](#)
 - OSRTOutputStream, [262](#)
 - OSRTStream, [289](#)
 - OSRTStreamIF, [293](#)
- freeMemory
 - ASN1CSeqOfList, [76](#)
- Generic Input Stream Classes, [31](#)
- Generic Output Stream Classes, [32](#)
- get
 - ASN1CBitStr, [53](#)
 - ASN1CSeqOfList, [76](#)
- getAppInfo
 - ASN1MessageBuffer, [127](#)
 - OSRTMessageBuffer, [251](#)
 - OSRTMessageBufferIF, [257](#)
- getBitOffset
 - ASN1MessageBuffer, [127](#)
- getBuffer
 - OSRTMemoryOutputStream, [248](#)
- getByteIndex
 - OSRTMessageBuffer, [251](#)
 - OSRTMessageBufferIF, [257](#)
- getBytes
 - ASN1CBitStr, [54](#)
- getCentury
 - ASN1CGeneralizedTime, [66](#)
 - ASN1TGeneralizedTime, [161](#)
- getContext
 - ASN1CType, [113](#)
 - OSRTInputStream, [231](#)
 - OSRTMessageBuffer, [252](#)
 - OSRTOutputStream, [263](#)
 - OSRTStream, [289](#)
- getCxtPtr
 - ASN1CType, [114](#)
 - OSRTCxtPtr, [213](#)
 - OSRTInputStream, [231](#)
 - OSRTMessageBuffer, [252](#)
 - OSRTOutputStream, [263](#)
 - OSRTStream, [290](#)
- getDay
 - ASN1CTime, [92](#)
 - ASN1TTime, [180](#)
- getDiff
 - ASN1CTime, [93](#)
 - ASN1TTime, [180](#)
- getDiffHour
 - ASN1CTime, [93](#)
 - ASN1TTime, [180](#)
- getDiffMinute
 - ASN1CTime, [94](#)
 - ASN1TTime, [181](#)
- getErrorInfo
 - OSRTContext, [205](#), [206](#)
 - OSRTInputStream, [231](#), [232](#)
 - OSRTMessageBuffer, [252](#)
 - OSRTOutputStream, [263](#), [264](#)
 - OSRTStream, [290](#)
- getErrorText
 - ASN1CType, [114](#)
- getFirst
 - ASN1CSeqOfList, [77](#)
- getFraction
 - ASN1CTime, [94](#)
 - ASN1CUTCTime, [121](#)
 - ASN1TTime, [181](#)
 - ASN1TUTCTime, [200](#)
- getFractionAsDouble
 - ASN1CTime, [94](#)
 - ASN1TTime, [181](#)
- getFractionLen
 - ASN1CTime, [95](#)
 - ASN1TTime, [182](#)
- getFractionStr
 - ASN1CTime, [95](#)
 - ASN1TTime, [182](#)
- getHour
 - ASN1CTime, [95](#)
 - ASN1TTime, [182](#)
- getLast
 - ASN1CSeqOfList, [77](#)
- getMinute
 - ASN1CTime, [96](#)
 - ASN1TTime, [183](#)
- getMonth
 - ASN1CTime, [96](#)
 - ASN1TTime, [183](#)
- getMsgCopy
 - OSRTMessageBuffer, [253](#)
 - OSRTMessageBufferIF, [258](#)
- getMsgLen
 - ASN1MessageBuffer, [127](#)
- getMsgPtr
 - OSRTMessageBuffer, [253](#)
 - OSRTMessageBufferIF, [258](#)
- getOwnership
 - OSRTSocket, [277](#)
- getPosition
 - OSRTInputStream, [232](#)
 - OSRTInputStreamIF, [239](#)
- getPtr
 - OSRTContext, [206](#)
- getRefCount
 - OSRTContext, [206](#)
- getSecond
 - ASN1CTime, [97](#)

- ASN1TTime, [183](#)
- getSocket
 - OSRSocket, [277](#)
- getStatus
 - ASN1CType, [114](#)
 - OSRTContext, [207](#)
 - OSRTInputStream, [233](#)
 - OSRTMessageBuffer, [253](#)
 - OSRTOutputStream, [264](#)
 - OSRSocket, [278](#)
 - OSRTStream, [291](#)
- getTime
 - ASN1CTime, [97](#)
 - ASN1TTime, [184](#)
- getTimeString
 - ASN1CTime, [97](#)
- getTimeStringLen
 - ASN1CTime, [98](#)
- getUTF8Value
 - OSRTFastString, [218](#)
 - OSRTString, [297](#)
 - OSRTStringIF, [303](#)
- getUTC
 - ASN1CTime, [98](#)
 - ASN1TTime, [184](#)
- getValue
 - OSRTFastString, [218](#)
 - OSRTString, [297](#)
 - OSRTStringIF, [303](#)
 - OSRTUTF8String, [307](#)
- getYear
 - ASN1CTime, [99](#)
 - ASN1TTime, [184](#)
- hasNext
 - ASN1CSeqOfListIterator, [84](#)
- hasPrev
 - ASN1CSeqOfListIterator, [84](#)
- indexOf
 - ASN1CSeqOfList, [77](#)
 - OSRTString, [298](#)
- init
 - ASN1CSeqOfList, [78](#)
 - OSRTMessageBuffer, [253](#)
 - OSRTMessageBufferIF, [258](#)
- initBuffer
 - ASN1MessageBuffer, [127](#), [128](#)
 - OSRTMessageBuffer, [254](#)
 - OSRTMessageBufferIF, [259](#)
- insert
 - ASN1CSeqOfList, [78](#)
 - ASN1CSeqOfListIterator, [85](#)
- int64Value
 - Asn1NamedEventHandler, [136](#)
- intValue
 - Asn1NamedEventHandler, [136](#)
- invert
 - ASN1CBitStr, [54](#), [55](#)
- invokeBitStrValue
 - Asn1NamedEventHandler, [137](#)
- invokeBoolValue
 - Asn1NamedEventHandler, [137](#)
- invokeCharStrValue
 - Asn1NamedEventHandler, [138](#), [139](#)
- invokeEndElement
 - Asn1NamedEventHandler, [139](#)
- invokeEnumValue
 - Asn1NamedEventHandler, [140](#)
- invokeInt64Value
 - Asn1NamedEventHandler, [140](#)
- invokeIntValue
 - Asn1NamedEventHandler, [140](#)
- invokeNullValue
 - Asn1NamedEventHandler, [141](#)
- invokeOctStrValue
 - Asn1NamedEventHandler, [141](#)
- invokeOidValue
 - Asn1NamedEventHandler, [141](#)
- invokeOpenTypeValue
 - Asn1NamedEventHandler, [142](#)
- invokeRealValue
 - Asn1NamedEventHandler, [142](#)
- invokeStartElement
 - Asn1NamedEventHandler, [142](#)
- invokeUInt64Value
 - Asn1NamedEventHandler, [143](#)
- invokeUIntValue
 - Asn1NamedEventHandler, [143](#)
- isEmpty
 - ASN1CBitStr, [55](#)
 - ASN1CSeqOfList, [78](#)
- isInitialized
 - OSRTContext, [207](#)
- isNull
 - OSRTCtxtPtr, [213](#)
- isOpened
 - OSRTInputStream, [234](#)
 - OSRTOutputStream, [265](#)
 - OSRTStream, [291](#)
 - OSRTStreamIF, [294](#)
- isSet
 - ASN1CBitStr, [56](#)
- isA
 - ASN1MessageBuffer, [128](#)
 - OSRTFileInputStream, [222](#)
 - OSRTFileOutputStream, [225](#)
 - OSRTHexTextInputStream, [227](#)
 - OSRTInputStream, [233](#)

- OSRTInputStreamIF, [240](#)
- OSRTMemoryInputStream, [246](#)
- OSRTMemoryOutputStream, [249](#)
- OSRTMessageBufferIF, [259](#)
- OSRTOutputStream, [265](#)
- OSRTOutputStreamIF, [268](#)
- OSRTSocketInputStream, [283](#)
- OSRTSocketOutputStream, [287](#)
- iterator
 - ASN1CSeqOfList, [78](#)
- iteratorFrom
 - ASN1CSeqOfList, [79](#)
- iteratorFromLast
 - ASN1CSeqOfList, [79](#)
- LOG_TMERR
 - Date and Time Runtime Classes, [29](#)
- LOG_TTMERR
 - ASN1TTime.h, [319](#)
- length
 - ASN1CBitStr, [56](#)
 - OSRTString, [298](#)
- listen
 - OSRTSocket, [278](#)
- MAX_TIMESTR_SIZE
 - ASN1TTime.h, [319](#)
- mBufferType
 - OSRTMessageBuffer, [256](#)
- mCount
 - OSRTContext, [211](#)
- mCtxt
 - OSRTContext, [211](#)
- mDay
 - ASN1TTime, [194](#)
- mDiffHour
 - ASN1TTime, [195](#)
- mDiffMin
 - ASN1TTime, [195](#)
- mHour
 - ASN1TTime, [195](#)
- mMinute
 - ASN1TTime, [195](#)
- mMonth
 - ASN1TTime, [195](#)
- mPointer
 - OSRTCtxtPtr, [215](#)
- mSecFracLen
 - ASN1TTime, [195](#)
- mSecFraction
 - ASN1TTime, [195](#)
- mSecond
 - ASN1TTime, [196](#)
- mStatus
 - ASN1TTime, [196](#)
- OSRTContext, [211](#)
- mYear
 - ASN1TTime, [196](#)
- mark
 - OSRTInputStream, [234](#)
 - OSRTInputStreamIF, [240](#)
- markSupported
 - OSRTInputStream, [235](#)
 - OSRTInputStreamIF, [241](#)
- mbDerRules
 - ASN1TTime, [194](#)
- mbInitialized
 - OSRTContext, [211](#)
- mbUtcFlag
 - ASN1TTime, [194](#)
- memAlloc
 - ASN1CType, [115](#)
 - OSRTContext, [207](#)
- memAllocZ
 - ASN1CType, [115](#)
 - OSRTContext, [208](#)
- memFreeAll
 - ASN1CType, [116](#)
 - OSRTContext, [208](#)
- memFreePtr
 - ASN1CType, [116](#)
 - OSRTContext, [208](#)
- memRealloc
 - ASN1CType, [116](#)
 - OSRTContext, [209](#)
- memReset
 - ASN1CType, [117](#)
 - OSRTContext, [209](#)
- mpContext
 - ASN1CType, [118](#)
 - ASN1TPDU, [172](#)
- mpMsgBuf
 - ASN1CType, [118](#)
- nCompare
 - ASN1TDynOctStr, [156](#)
 - ASN1TObjId, [166](#)
- Named Event Handlers, [34](#)
 - ASN1MessageBuffer, [34](#)
 - OS_UNUSED_ARG, [34](#)
- next
 - ASN1CSeqOfListIterator, [85](#)
- nullValue
 - Asn1NamedEventHandler, [143](#)
- OS_UNUSED_ARG
 - Named Event Handlers, [34](#)
- OSRT Message Buffer Classes, [4](#)
- OSRTBaseType, [203](#)
- OSRTBaseType.h, [319](#)

- OSRTContext, [203](#)
 - [_ref](#), [205](#)
 - [_unref](#), [205](#)
 - [~OSRTContext](#), [204](#)
 - [getErrorInfo](#), [205](#), [206](#)
 - [getPtr](#), [206](#)
 - [getRefCount](#), [206](#)
 - [getStatus](#), [207](#)
 - [isInitialized](#), [207](#)
 - [mCount](#), [211](#)
 - [mCtxt](#), [211](#)
 - [mStatus](#), [211](#)
 - [mblInitialized](#), [211](#)
 - [memAlloc](#), [207](#)
 - [memAllocZ](#), [208](#)
 - [memFreeAll](#), [208](#)
 - [memFreePtr](#), [208](#)
 - [memRealloc](#), [209](#)
 - [memReset](#), [209](#)
 - [OSRTContext](#), [204](#)
 - [printErrorInfo](#), [209](#)
 - [resetErrorInfo](#), [209](#)
 - [setDiag](#), [210](#)
 - [setRunTimeKey](#), [210](#)
 - [setStatus](#), [211](#)
- OSRTContext.h, [320](#)
 - [operator delete](#), [320](#)
 - [operator new](#), [320](#)
- OSRTCtxtPtr, [212](#)
 - [~OSRTCtxtPtr](#), [213](#)
 - [getCtxtPtr](#), [213](#)
 - [isNull](#), [213](#)
 - [mPointer](#), [215](#)
 - [OSRTCtxtPtr](#), [212](#), [213](#)
 - [operator OSRTContext *](#), [214](#)
 - [operator->](#), [214](#)
 - [operator=](#), [214](#)
 - [operator==](#), [215](#)
- OSRTFastString, [215](#)
 - [~OSRTFastString](#), [217](#)
 - [clone](#), [217](#)
 - [getUTF8Value](#), [218](#)
 - [getValue](#), [218](#)
 - [OSRTFastString](#), [216](#), [217](#)
 - [operator=](#), [218](#)
 - [print](#), [218](#)
 - [setValue](#), [219](#)
- OSRTFastString.h, [321](#)
- OSRTFileInputStream, [220](#)
 - [isA](#), [222](#)
 - [OSRTFileInputStream](#), [220](#), [221](#)
- OSRTFileInputStream.h, [321](#)
- OSRTFileOutputStream, [223](#)
 - [isA](#), [225](#)
 - [OSRTFileOutputStream](#), [223–225](#)
- OSRTFileOutputStream.h, [321](#)
- OSRTHexTextInputStream, [226](#)
 - [~OSRTHexTextInputStream](#), [227](#)
 - [isA](#), [227](#)
 - [OSRTHexTextInputStream](#), [227](#)
 - [setOwnUnderStream](#), [228](#)
- OSRTHexTextInputStream.h, [322](#)
- OSRTInputStream, [228](#)
 - [~OSRTInputStream](#), [230](#)
 - [close](#), [230](#)
 - [currentPos](#), [230](#)
 - [flush](#), [230](#)
 - [getContext](#), [231](#)
 - [getCtxtPtr](#), [231](#)
 - [getErrorInfo](#), [231](#), [232](#)
 - [getPosition](#), [232](#)
 - [getStatus](#), [233](#)
 - [isOpened](#), [234](#)
 - [isA](#), [233](#)
 - [mark](#), [234](#)
 - [markSupported](#), [235](#)
 - [OSRTInputStream](#), [229](#)
 - [printErrorInfo](#), [235](#)
 - [read](#), [235](#)
 - [readBlocking](#), [236](#)
 - [reset](#), [236](#)
 - [resetErrorInfo](#), [237](#)
 - [setPosition](#), [237](#)
 - [skip](#), [237](#)
- OSRTInputStream.h, [322](#)
- OSRTInputStreamIF.h, [322](#)
- OSRTInputStreamIF, [238](#)
 - [~OSRTInputStreamIF](#), [239](#)
 - [currentPos](#), [239](#)
 - [getPosition](#), [239](#)
 - [isA](#), [240](#)
 - [mark](#), [240](#)
 - [markSupported](#), [241](#)
 - [read](#), [241](#)
 - [readBlocking](#), [242](#)
 - [reset](#), [242](#)
 - [setPosition](#), [243](#)
 - [skip](#), [243](#)
- OSRTInputStreamPtr, [244](#)
- OSRTMemoryInputStream, [244](#)
 - [isA](#), [246](#)
 - [OSRTMemoryInputStream](#), [245](#)
- OSRTMemoryInputStream.h, [323](#)
- OSRTMemoryOutputStream, [246](#)
 - [getBuffer](#), [248](#)
 - [isA](#), [249](#)
 - [OSRTMemoryOutputStream](#), [247](#), [248](#)
 - [reset](#), [249](#)

- OSRTMemoryOutputStream.h, [323](#)
- OSRTMessageBuffer, [250](#)
 - ~OSRTMessageBuffer, [251](#)
 - getAppInfo, [251](#)
 - getByteIndex, [251](#)
 - getContext, [252](#)
 - getCtxtPtr, [252](#)
 - getErrorInfo, [252](#)
 - getMsgCopy, [253](#)
 - getMsgPtr, [253](#)
 - getStatus, [253](#)
 - init, [253](#)
 - initBuffer, [254](#)
 - mBufferType, [256](#)
 - OSRTMessageBuffer, [251](#)
 - printErrorInfo, [254](#)
 - resetErrorInfo, [255](#)
 - setAppInfo, [255](#)
 - setDiag, [255](#)
- OSRTMessageBufferIF, [256](#)
 - ~OSRTMessageBufferIF, [257](#)
 - getAppInfo, [257](#)
 - getByteIndex, [257](#)
 - getMsgCopy, [258](#)
 - getMsgPtr, [258](#)
 - init, [258](#)
 - initBuffer, [259](#)
 - isA, [259](#)
 - setAppInfo, [260](#)
 - setDiag, [260](#)
 - setNamespace, [260](#)
- OSRTMsgBuf.h, [323](#)
- OSRTMsgBufIF.h, [324](#)
- OSRTOutputStream, [261](#)
 - ~OSRTOutputStream, [262](#)
 - close, [262](#)
 - flush, [262](#)
 - getContext, [263](#)
 - getCtxtPtr, [263](#)
 - getErrorInfo, [263](#), [264](#)
 - getStatus, [264](#)
 - isOpened, [265](#)
 - isA, [265](#)
 - OSRTOutputStream, [262](#)
 - printErrorInfo, [265](#)
 - resetErrorInfo, [266](#)
 - write, [266](#)
- OSRTOutputStream.h, [324](#)
- OSRTOutputStreamIF.h, [325](#)
- OSRTOutputStreamIF, [267](#)
 - ~OSRTOutputStreamIF, [268](#)
 - isA, [268](#)
 - write, [268](#)
- OSRTOutputStreamPtr, [269](#)
- OSRTSocket, [269](#)
 - ~OSRTSocket, [271](#)
 - accept, [272](#)
 - addrToString, [272](#)
 - bind, [273](#), [274](#)
 - bindUrl, [274](#)
 - blockingRead, [275](#)
 - close, [275](#)
 - connect, [276](#)
 - connectTimed, [276](#)
 - connectUrl, [277](#)
 - getOwnership, [277](#)
 - getSocket, [277](#)
 - getStatus, [278](#)
 - listen, [278](#)
 - OSRTSocket, [271](#)
 - recv, [279](#)
 - send, [279](#)
 - setOwnership, [280](#)
 - setRetryCount, [280](#)
 - stringToAddr, [280](#)
- OSRTSocket.h, [325](#)
- OSRTSocketInputStream, [281](#)
 - isA, [283](#)
 - OSRTSocketInputStream, [282](#), [283](#)
- OSRTSocketInputStream.h, [325](#)
- OSRTSocketOutputStream, [284](#)
 - isA, [287](#)
 - OSRTSocketOutputStream, [285](#), [286](#)
- OSRTSocketOutputStream.h, [326](#)
- OSRTStream, [287](#)
 - ~OSRTStream, [288](#)
 - close, [288](#)
 - flush, [289](#)
 - getContext, [289](#)
 - getCtxtPtr, [290](#)
 - getErrorInfo, [290](#)
 - getStatus, [291](#)
 - isOpened, [291](#)
 - OSRTStream, [288](#)
 - printErrorInfo, [292](#)
 - resetErrorInfo, [292](#)
- OSRTStream.h, [326](#)
- OSRTStreamIF.h, [326](#)
- OSRTStreamIF, [293](#)
 - close, [293](#)
 - flush, [293](#)
 - isOpened, [294](#)
- OSRTString, [295](#)
 - ~OSRTString, [297](#)
 - clone, [297](#)
 - data, [297](#)
 - getUTF8Value, [297](#)
 - getValue, [297](#)

- indexOf, [298](#)
- length, [298](#)
- OSRTString, [296](#)
- operator=, [298](#)
- print, [298](#)
- setValue, [299](#)
- toInt, [299](#)
- toSize, [300](#)
- toUInt, [300](#)
- toUInt64, [300](#)
- OSRTString.h, [327](#)
- OSRTStringIF.h, [327](#)
- OSRTStringIF, [301](#)
 - ~OSRTStringIF, [302](#)
 - clone, [303](#)
 - getUTF8Value, [303](#)
 - getValue, [303](#)
 - OSRTStringIF, [302](#)
 - print, [303](#)
 - setValue, [304](#)
- OSRTUTF8String, [305](#)
 - ~OSRTUTF8String, [306](#)
 - c_str, [306](#)
 - clone, [307](#)
 - copyValue, [307](#)
 - getValue, [307](#)
 - OSRTUTF8String, [305](#), [306](#)
 - operator=, [307](#)
 - print, [307](#)
 - setValue, [309](#)
- OSRTUTF8String.h, [328](#)
- octStringValue
 - Asn1NamedEventHandler, [144](#)
- oidValue
 - Asn1NamedEventHandler, [144](#)
- openTypeValue
 - Asn1NamedEventHandler, [145](#)
- operator ASN1TDynBitStr
 - ASN1CBitStr, [56](#)
- operator ASN1TDynBitStr *
 - ASN1CBitStr, [57](#)
- operator delete
 - OSRTContext.h, [320](#)
- operator new
 - OSRTContext.h, [320](#)
- operator OSRTContext *
 - OSRTCtxtPtr, [214](#)
- operator!=
 - ASN.1 Type (ASN1T_) Base Classes, [7–10](#)
 - ASN1CTime, [99](#)
- operator<
 - ASN.1 Type (ASN1T_) Base Classes, [11–14](#)
 - ASN1CTime, [99](#)
- operator<=
 - ASN.1 Type (ASN1T_) Base Classes, [15–18](#)
 - ASN1CTime, [99](#)
- operator>
 - ASN.1 Type (ASN1T_) Base Classes, [21–24](#)
 - ASN1CTime, [100](#)
- operator>=
 - ASN.1 Type (ASN1T_) Base Classes, [24–27](#)
 - ASN1CTime, [101](#)
- operator+
 - ASN.1 Type (ASN1T_) Base Classes, [11](#)
- operator+=
 - ASN1ObjId, [167](#), [168](#)
- operator->
 - OSRTCtxtPtr, [214](#)
- operator=
 - ASN1CTime, [100](#)
 - ASN1TDynOctStr, [157](#)
 - ASN1ObjId, [168](#), [169](#)
 - OSRTCtxtPtr, [214](#)
 - OSRTFastString, [218](#)
 - OSRTString, [298](#)
 - OSRTUTF8String, [307](#)
- operator==
 - ASN.1 Type (ASN1T_) Base Classes, [18–20](#)
 - ASN1CTime, [100](#)
 - OSRTCtxtPtr, [215](#)
- operator[]
 - ASN1CSeqOfList, [79](#)
- parseString
 - ASN1CTime, [101](#)
 - ASN1TGeneralizedTime, [162](#)
 - ASN1TTime, [185](#)
 - ASN1TUTCTime, [200](#)
- prev
 - ASN1CSeqOfListIterator, [86](#)
- print
 - OSRTFastString, [218](#)
 - OSRTString, [298](#)
 - OSRTStringIF, [303](#)
 - OSRTUTF8String, [307](#)
- printErrorInfo
 - ASN1CType, [117](#)
 - OSRTContext, [209](#)
 - OSRTInputStream, [235](#)
 - OSRTMessageBuffer, [254](#)
 - OSRTOutputStream, [265](#)
 - OSRTStream, [292](#)
- read
 - OSRTInputStream, [235](#)
 - OSRTInputStreamIF, [241](#)
- readBlocking
 - OSRTInputStream, [236](#)
 - OSRTInputStreamIF, [242](#)

- realValue
 - Asn1NamedEventHandler, [145](#)
- recv
 - OSRSocket, [279](#)
- remove
 - ASN1CSeqOfList, [80](#)
 - ASN1CSeqOfListIterator, [86](#)
- removeEventHandler
 - ASN1MessageBuffer, [129](#)
 - Asn1NamedEventHandler, [145](#)
- removeFirst
 - ASN1CSeqOfList, [81](#)
- removeLast
 - ASN1CSeqOfList, [81](#)
- reset
 - OSRTInputStream, [236](#)
 - OSRTInputStreamIF, [242](#)
 - OSRTMemoryOutputStream, [249](#)
- resetError
 - ASN1CType, [117](#)
- resetErrorInfo
 - ASN1MessageBuffer, [129](#)
 - OSRTContext, [209](#)
 - OSRTInputStream, [237](#)
 - OSRTMessageBuffer, [255](#)
 - OSRTOutputStream, [266](#)
 - OSRTStream, [292](#)
- RnCompare
 - ASN1TObjId, [169](#)
- Run-time error status codes., [36](#)
 - ASN_E_BAD_ALIGN, [36](#)
 - ASN_E_BADTAG, [36](#)
 - ASN_E_BASE, [37](#)
 - ASN_E_CONCMODF, [37](#)
 - ASN_E_ILLSTATE, [37](#)
 - ASN_E_INVBINS, [37](#)
 - ASN_E_INVINDEX, [37](#)
 - ASN_E_INVLEN, [37](#)
 - ASN_E_INVOBJID, [38](#)
 - ASN_E_INVPERENC, [38](#)
 - ASN_E_INVTCVAL, [38](#)
 - ASN_E_NOTCANON, [38](#)
 - ASN_E_NOTINSEQ, [38](#)
 - ASN_E_NOTPDU, [38](#)
 - ASN_E_UNDEFTYP, [39](#)
 - ASN_E_UNKNOWNPDU, [39](#)
 - ASN_OK_FRAG, [39](#)
- send
 - OSRSocket, [279](#)
- set
 - ASN1CBitStr, [57](#), [58](#)
 - ASN1CSeqOfList, [81](#)
 - ASN1CSeqOfListIterator, [86](#)
- set_data
 - ASN1TObjId, [169](#)
- setAppInfo
 - ASN1MessageBuffer, [129](#)
 - OSRTMessageBuffer, [255](#)
 - OSRTMessageBufferIF, [260](#)
- setCentury
 - ASN1CGeneralizedTime, [67](#)
 - ASN1TGeneralizedTime, [162](#)
- setContext
 - ASN1TPDU, [172](#)
- setDER
 - ASN1CTime, [102](#)
 - ASN1TTime, [186](#)
- setDay
 - ASN1CTime, [101](#)
 - ASN1TTime, [185](#)
- setDiag
 - ASN1CType, [117](#)
 - OSRTContext, [210](#)
 - OSRTMessageBuffer, [255](#)
 - OSRTMessageBufferIF, [260](#)
- setDiff
 - ASN1CTime, [102](#), [103](#)
 - ASN1TTime, [186](#), [187](#)
- setDiffHour
 - ASN1CTime, [103](#)
 - ASN1TTime, [187](#)
- setErrorHandler
 - ASN1MessageBuffer, [129](#)
 - Asn1ErrorHandler, [123](#)
- setFraction
 - ASN1CTime, [104](#), [105](#)
 - ASN1TTime, [188](#), [189](#)
 - ASN1TUTCTime, [200](#)
- setHour
 - ASN1CTime, [105](#)
 - ASN1TTime, [189](#)
- setMinute
 - ASN1CTime, [106](#)
 - ASN1TTime, [190](#)
- setMonth
 - ASN1CTime, [106](#)
 - ASN1TTime, [190](#)
- setNamespace
 - OSRTMessageBufferIF, [260](#)
- setOwnUnderStream
 - OSRTHexTextInputStream, [228](#)
- setOwnership
 - OSRSocket, [280](#)
- setPosition
 - OSRTInputStream, [237](#)
 - OSRTInputStreamIF, [243](#)
- setRetryCount

- OSRSocket, [280](#)
- setRunTimeKey
 - ASN1CType, [118](#)
 - ASN1Context, [69](#)
 - ASN1MessageBuffer, [130](#)
 - OSRTContext, [210](#)
- setSecond
 - ASN1CTime, [107](#)
 - ASN1TTime, [191](#)
- setStatus
 - ASN1MessageBuffer, [130](#)
 - OSRTContext, [211](#)
- setTime
 - ASN1CGeneralizedTime, [67](#)
 - ASN1CTime, [107](#)
 - ASN1CUTCTime, [122](#)
 - ASN1TGeneralizedTime, [162](#)
 - ASN1TTime, [191](#)
 - ASN1TUTCTime, [201](#)
- setUTC
 - ASN1CTime, [108](#)
 - ASN1TTime, [192](#)
 - ASN1TUTCTime, [201](#)
- setValue
 - OSRTFastString, [219](#)
 - OSRTString, [299](#)
 - OSRTStringIF, [304](#)
 - OSRTUTF8String, [309](#)
- setYear
 - ASN1CTime, [108](#)
 - ASN1TTime, [192](#)
 - ASN1TUTCTime, [202](#)
- shiftLeft
 - ASN1CBitStr, [58](#)
- shiftRight
 - ASN1CBitStr, [59](#)
- size
 - ASN1CBitStr, [59](#)
 - ASN1CSeqOfList, [82](#)
- skip
 - OSRTInputStream, [237](#)
 - OSRTInputStreamIF, [243](#)
- startElement
 - Asn1NamedEventHandler, [146](#)
 - Asn1NullEventHandler, [148](#)
- stringToAddr
 - OSRSocket, [280](#)

- TCP/IP or UDP Socket Classes, [33](#)
- toArray
 - ASN1CSeqOfList, [82](#)
- toHexString
 - ASN1TDynOctStr, [157](#)
- toInt
 - OSRTString, [299](#)
- toSize
 - OSRTString, [300](#)
- toString
 - ASN1TDynOctStr, [159](#)
 - ASN1TObjId, [170](#)
 - ASN1TTime, [193](#), [194](#)
- toUInt
 - OSRTString, [300](#)
- toUInt64
 - OSRTString, [300](#)
- trim
 - ASN1TObjId, [170](#)

- uint64Value
 - Asn1NamedEventHandler, [146](#)
- uintValue
 - Asn1NamedEventHandler, [147](#)
- unusedBitsInLastUnit
 - ASN1CBitStr, [59](#)

- write
 - OSRTOutputStream, [266](#)
 - OSRTOutputStreamIF, [268](#)