

ASN1C

ASN.1 Compiler
Version 6.5
C++ Runtime
Reference Manual

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

Copyright Notice

Copyright ©1997–2012 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	2
2.1	C++ Run-Time Classes	2
2.1.1	Detailed Description	2
2.2	OSRT Message Buffer Classes	3
2.2.1	Detailed Description	3
2.3	Control (ASN1C_) Base Classes	4
2.3.1	Detailed Description	4
2.4	ASN.1 Type (ASN1T_) Base Classes	5
2.4.1	Detailed Description	6
2.4.2	Function Documentation	6
2.4.2.1	operator!=	6
2.4.2.2	operator!=	7
2.4.2.3	operator!=	7
2.4.2.4	operator!=	7
2.4.2.5	operator+	7
2.4.2.6	operator<	8
2.4.2.7	operator<	8
2.4.2.8	operator<	8
2.4.2.9	operator<	8
2.4.2.10	operator<=	9
2.4.2.11	operator<=	9
2.4.2.12	operator<=	9
2.4.2.13	operator<=	9
2.4.2.14	operator==	10
2.4.2.15	operator==	10
2.4.2.16	operator>	10

2.4.2.17	operator>	10
2.4.2.18	operator>	11
2.4.2.19	operator>	11
2.4.2.20	operator>=	11
2.4.2.21	operator>=	11
2.4.2.22	operator>=	12
2.4.2.23	operator>=	12
2.5	Context Management Classes	13
2.5.1	Detailed Description	13
2.6	Date and Time Runtime Classes	14
2.6.1	Detailed Description	14
2.7	ASN.1 Stream Classes	15
2.7.1	Detailed Description	15
2.8	Generic Input Stream Classes	16
2.8.1	Detailed Description	16
2.9	Generic Output Stream Classes	17
2.9.1	Detailed Description	17
2.10	TCP/IP or UDP Socket Classes	18
2.10.1	Detailed Description	18
2.11	Named Event Handlers	19
2.11.1	Detailed Description	19
2.12	Run-time error status codes.	20
2.12.1	Detailed Description	20
2.12.2	Define Documentation	20
2.12.2.1	ASN_E_BADTAG	20
2.12.2.2	ASN_E_BASE	20
2.12.2.3	ASN_E_CONCMODF	20
2.12.2.4	ASN_E_ILLSTATE	20
2.12.2.5	ASN_E_INVBINS	21
2.12.2.6	ASN_E_INVINDEX	21
2.12.2.7	ASN_E_INVLEN	21
2.12.2.8	ASN_E_INVOBJID	21
2.12.2.9	ASN_E_INVPERENC	21
2.12.2.10	ASN_E_INVTCVAL	21
2.12.2.11	ASN_E_NOTINSEQ	21
2.12.2.12	ASN_E_NOTPDU	21
2.12.2.13	ASN_E_UNDEFTYP	21

2.12.2.14	ASN_OK_FRAG	22
3	Class Documentation	23
3.1	ASN1BitStr Class Reference	23
3.1.1	Constructor & Destructor Documentation	25
3.1.1.1	ASN1BitStr	25
3.1.1.2	ASN1BitStr	25
3.1.2	Member Function Documentation	25
3.1.2.1	cardinality	25
3.1.2.2	change	26
3.1.2.3	clear	26
3.1.2.4	clear	26
3.1.2.5	clear	27
3.1.2.6	doAnd	27
3.1.2.7	doAnd	27
3.1.2.8	doAnd	27
3.1.2.9	doAndNot	28
3.1.2.10	doAndNot	28
3.1.2.11	doAndNot	28
3.1.2.12	doOr	29
3.1.2.13	doOr	29
3.1.2.14	doOr	29
3.1.2.15	doXor	30
3.1.2.16	doXor	30
3.1.2.17	doXor	30
3.1.2.18	get	31
3.1.2.19	get	31
3.1.2.20	getBytes	31
3.1.2.21	invert	32
3.1.2.22	invert	32
3.1.2.23	isEmpty	32
3.1.2.24	isSet	33
3.1.2.25	length	33
3.1.2.26	operator ASN1TDynBitStr	33
3.1.2.27	operator ASN1TDynBitStr *	33
3.1.2.28	set	34
3.1.2.29	set	34

3.1.2.30	shiftLeft	34
3.1.2.31	shiftRight	35
3.1.2.32	size	35
3.1.2.33	unusedBitsInLastUnit	35
3.2	ASN1CBitStrSizeHolder Class Reference	36
3.2.1	Detailed Description	36
3.3	ASN1CBitStrSizeHolder16 Class Reference	37
3.4	ASN1CBitStrSizeHolder32 Class Reference	38
3.5	ASN1CBitStrSizeHolder8 Class Reference	39
3.6	ASN1CGeneralizedTime Class Reference	40
3.6.1	Detailed Description	40
3.6.2	Constructor & Destructor Documentation	41
3.6.2.1	ASN1CGeneralizedTime	41
3.6.2.2	ASN1CGeneralizedTime	41
3.6.3	Member Function Documentation	41
3.6.3.1	getCentury	41
3.6.3.2	setCentury	42
3.6.3.3	setTime	42
3.7	ASN1Context Class Reference	43
3.7.1	Detailed Description	43
3.7.2	Constructor & Destructor Documentation	43
3.7.2.1	ASN1Context	43
3.7.3	Member Function Documentation	43
3.7.3.1	setRunTimeKey	43
3.8	ASN1CSeqOfList Class Reference	45
3.8.1	Detailed Description	46
3.8.2	Constructor & Destructor Documentation	46
3.8.2.1	ASN1CSeqOfList	46
3.8.2.2	ASN1CSeqOfList	46
3.8.2.3	ASN1CSeqOfList	46
3.8.2.4	ASN1CSeqOfList	47
3.8.2.5	ASN1CSeqOfList	47
3.8.2.6	ASN1CSeqOfList	47
3.8.3	Member Function Documentation	47
3.8.3.1	append	47
3.8.3.2	appendArray	48
3.8.3.3	appendArrayCopy	48

3.8.3.4	clear	48
3.8.3.5	contains	48
3.8.3.6	free	49
3.8.3.7	get	49
3.8.3.8	getFirst	49
3.8.3.9	getLast	49
3.8.3.10	indexOf	49
3.8.3.11	init	49
3.8.3.12	insert	50
3.8.3.13	isEmpty	50
3.8.3.14	iterator	50
3.8.3.15	iteratorFrom	50
3.8.3.16	iteratorFromLast	50
3.8.3.17	operator[]	51
3.8.3.18	remove	51
3.8.3.19	remove	51
3.8.3.20	removeFirst	51
3.8.3.21	removeLast	51
3.8.3.22	set	52
3.8.3.23	size	52
3.8.3.24	toArray	52
3.8.3.25	toArray	53
3.9	ASN1CSeqOfListIterator Class Reference	54
3.9.1	Detailed Description	54
3.9.2	Member Function Documentation	54
3.9.2.1	hasNext	54
3.9.2.2	hasPrev	55
3.9.2.3	insert	55
3.9.2.4	next	55
3.9.2.5	prev	55
3.9.2.6	remove	56
3.9.2.7	set	56
3.10	ASN1CTime Class Reference	57
3.10.1	Detailed Description	59
3.10.2	Constructor & Destructor Documentation	59
3.10.2.1	ASN1CTime	59
3.10.2.2	ASN1CTime	59

3.10.2.3	ASN1CTime	59
3.10.3	Member Function Documentation	60
3.10.3.1	clear	60
3.10.3.2	equals	60
3.10.3.3	getDay	60
3.10.3.4	getDiff	60
3.10.3.5	getDiffHour	61
3.10.3.6	getDiffMinute	61
3.10.3.7	getFraction	61
3.10.3.8	getFractionAsDouble	61
3.10.3.9	getFractionLen	62
3.10.3.10	getFractionStr	62
3.10.3.11	getHour	62
3.10.3.12	getMinute	62
3.10.3.13	getMonth	63
3.10.3.14	getSecond	63
3.10.3.15	getTime	63
3.10.3.16	getUTC	63
3.10.3.17	getYear	64
3.10.3.18	parseString	64
3.10.3.19	setDay	64
3.10.3.20	setDER	65
3.10.3.21	setDiff	65
3.10.3.22	setDiff	65
3.10.3.23	setDiffHour	65
3.10.3.24	setFraction	66
3.10.3.25	setFraction	66
3.10.3.26	setFraction	66
3.10.3.27	setHour	67
3.10.3.28	setMinute	67
3.10.3.29	setMonth	67
3.10.3.30	setSecond	68
3.10.3.31	setTime	68
3.10.3.32	setUTC	68
3.10.3.33	setYear	69
3.11	ASN1CType Class Reference	70
3.11.1	Detailed Description	71

3.11.2	Constructor & Destructor Documentation	71
3.11.2.1	ASN1Type	71
3.11.2.2	ASN1Type	71
3.11.2.3	ASN1Type	71
3.11.2.4	ASN1Type	71
3.11.2.5	~ASN1Type	71
3.11.3	Member Function Documentation	71
3.11.3.1	append	71
3.11.3.2	Decode	72
3.11.3.3	DecodeFrom	72
3.11.3.4	Encode	72
3.11.3.5	EncodeTo	72
3.11.3.6	getContext	73
3.11.3.7	getCtxtPtr	73
3.11.3.8	getStatus	73
3.11.3.9	memAlloc	73
3.11.3.10	memFreeAll	74
3.11.3.11	memFreePtr	74
3.11.3.12	memRealloc	74
3.11.3.13	memReset	74
3.11.3.14	printErrorInfo	74
3.11.3.15	resetError	74
3.11.3.16	setDiag	74
3.11.3.17	setRunTimeKey	75
3.11.4	Member Data Documentation	75
3.11.4.1	mpContext	75
3.11.4.2	mpMsgBuf	75
3.12	ASN1CUTCTime Class Reference	76
3.12.1	Detailed Description	76
3.12.2	Constructor & Destructor Documentation	77
3.12.2.1	ASN1CUTCTime	77
3.12.2.2	ASN1CUTCTime	77
3.12.3	Member Function Documentation	77
3.12.3.1	getFraction	77
3.12.3.2	setTime	78
3.13	Asn1ErrorHandler Class Reference	79
3.13.1	Detailed Description	79

3.13.2	Member Function Documentation	79
3.13.2.1	error	79
3.13.2.2	setErrorHandler	79
3.14	ASN1MessageBuffer Class Reference	81
3.14.1	Detailed Description	81
3.14.2	Constructor & Destructor Documentation	82
3.14.2.1	ASN1MessageBuffer	82
3.14.2.2	ASN1MessageBuffer	82
3.14.2.3	~ASN1MessageBuffer	82
3.14.3	Member Function Documentation	82
3.14.3.1	addEventHandler	82
3.14.3.2	CStringToBMPString	82
3.14.3.3	getAppInfo	83
3.14.3.4	initBuffer	83
3.14.3.5	initBuffer	83
3.14.3.6	initBuffer	83
3.14.3.7	isA	84
3.14.3.8	removeEventHandler	84
3.14.3.9	resetErrorInfo	84
3.14.3.10	setAppInfo	84
3.14.3.11	setErrorHandler	84
3.14.3.12	setRunTimeKey	85
3.14.3.13	setStatus	85
3.15	Asn1NamedEventHandler Class Reference	86
3.15.1	Detailed Description	87
3.15.2	Member Function Documentation	87
3.15.2.1	bitStrValue	87
3.15.2.2	boolValue	87
3.15.2.3	charStrValue	88
3.15.2.4	charStrValue	88
3.15.2.5	charStrValue	88
3.15.2.6	charStrValue	88
3.15.2.7	endElement	89
3.15.2.8	enumValue	89
3.15.2.9	int64Value	89
3.15.2.10	intValue	90
3.15.2.11	nullValue	90

3.15.2.12 octStrValue	90
3.15.2.13 oidValue	90
3.15.2.14 openTypeValue	91
3.15.2.15 realValue	91
3.15.2.16 startElement	91
3.15.2.17 uInt64Value	91
3.15.2.18 uIntValue	92
3.16 Asn1NullEventHandler Class Reference	93
3.16.1 Member Function Documentation	93
3.16.1.1 endElement	93
3.16.1.2 startElement	93
3.17 ASN1TBitStr32 Struct Reference	95
3.17.1 Detailed Description	95
3.17.2 Constructor & Destructor Documentation	95
3.17.2.1 ASN1TBitStr32	95
3.17.2.2 ASN1TBitStr32	95
3.17.2.3 ASN1TBitStr32	95
3.18 ASN1TBMPString Struct Reference	96
3.18.1 Detailed Description	96
3.18.2 Constructor & Destructor Documentation	96
3.18.2.1 ASN1TBMPString	96
3.19 ASN1TDynBitStr Struct Reference	97
3.19.1 Detailed Description	97
3.19.2 Constructor & Destructor Documentation	97
3.19.2.1 ASN1TDynBitStr	97
3.19.2.2 ASN1TDynBitStr	97
3.19.2.3 ASN1TDynBitStr	97
3.20 ASN1TDynOctStr Struct Reference	98
3.20.1 Detailed Description	98
3.20.2 Constructor & Destructor Documentation	98
3.20.2.1 ASN1TDynOctStr	98
3.20.2.2 ASN1TDynOctStr	98
3.20.2.3 ASN1TDynOctStr	98
3.20.2.4 ASN1TDynOctStr	99
3.20.3 Member Function Documentation	99
3.20.3.1 nCompare	99
3.20.3.2 operator=	99

3.20.3.3	operator=	99
3.20.3.4	toHexString	99
3.20.3.5	toString	100
3.21	ASN1GeneralizedTime Class Reference	101
3.21.1	Detailed Description	101
3.21.2	Constructor & Destructor Documentation	101
3.21.2.1	ASN1GeneralizedTime	101
3.21.2.2	ASN1GeneralizedTime	101
3.21.2.3	ASN1GeneralizedTime	102
3.21.2.4	ASN1GeneralizedTime	102
3.21.3	Member Function Documentation	102
3.21.3.1	compileString	102
3.21.3.2	getCentury	102
3.21.3.3	parseString	102
3.21.3.4	setCentury	103
3.21.3.5	setTime	103
3.22	Asn1TObject Struct Reference	104
3.22.1	Detailed Description	104
3.22.2	Constructor & Destructor Documentation	104
3.22.2.1	Asn1TObject	104
3.23	ASN1ObjId Struct Reference	105
3.23.1	Detailed Description	105
3.23.2	Constructor & Destructor Documentation	105
3.23.2.1	ASN1ObjId	105
3.23.2.2	~ASN1ObjId	105
3.23.2.3	ASN1ObjId	105
3.23.2.4	ASN1ObjId	106
3.23.2.5	ASN1ObjId	106
3.23.2.6	ASN1ObjId	106
3.23.3	Member Function Documentation	106
3.23.3.1	nCompare	106
3.23.3.2	operator+=	106
3.23.3.3	operator+=	107
3.23.3.4	operator+=	107
3.23.3.5	operator=	107
3.23.3.6	operator=	107
3.23.3.7	operator=	107

3.23.3.8	RnCompare	108
3.23.3.9	set_data	108
3.23.3.10	toString	108
3.23.3.11	trim	108
3.24	ASN1ObjId64 Struct Reference	109
3.24.1	Detailed Description	109
3.25	ASN1OpenType Struct Reference	110
3.25.1	Detailed Description	110
3.25.2	Constructor & Destructor Documentation	110
3.25.2.1	ASN1OpenType	110
3.26	ASN1TPDU Struct Reference	111
3.26.1	Detailed Description	111
3.26.2	Member Function Documentation	111
3.26.2.1	setContext	111
3.26.3	Member Data Documentation	111
3.26.3.1	mpContext	111
3.27	ASN1TPDUSeqOfList Struct Reference	113
3.27.1	Detailed Description	113
3.27.2	Constructor & Destructor Documentation	113
3.27.2.1	ASN1TPDUSeqOfList	113
3.28	ASN1TSeqExt Struct Reference	114
3.28.1	Detailed Description	114
3.28.2	Constructor & Destructor Documentation	114
3.28.2.1	ASN1TSeqExt	114
3.29	ASN1TSeqOfList Struct Reference	115
3.29.1	Detailed Description	115
3.29.2	Constructor & Destructor Documentation	115
3.29.2.1	ASN1TSeqOfList	115
3.30	ASN1TTime Class Reference	116
3.30.1	Detailed Description	118
3.30.2	Constructor & Destructor Documentation	118
3.30.2.1	ASN1TTime	118
3.30.2.2	ASN1TTime	118
3.30.2.3	~ASN1TTime	118
3.30.3	Member Function Documentation	118
3.30.3.1	clear	118
3.30.3.2	compileString	118

3.30.3.3	equals	119
3.30.3.4	getDay	119
3.30.3.5	getDiff	119
3.30.3.6	getDiffHour	119
3.30.3.7	getDiffMinute	120
3.30.3.8	getFraction	120
3.30.3.9	getFractionAsDouble	120
3.30.3.10	getFractionLen	120
3.30.3.11	getFractionStr	120
3.30.3.12	getHour	121
3.30.3.13	getMinute	121
3.30.3.14	getMonth	121
3.30.3.15	getSecond	121
3.30.3.16	getTime	122
3.30.3.17	getUTC	122
3.30.3.18	getYear	122
3.30.3.19	parseString	122
3.30.3.20	setDay	123
3.30.3.21	setDER	123
3.30.3.22	setDiff	123
3.30.3.23	setDiff	123
3.30.3.24	setDiffHour	124
3.30.3.25	setFraction	124
3.30.3.26	setFraction	124
3.30.3.27	setFraction	125
3.30.3.28	setHour	125
3.30.3.29	setMinute	125
3.30.3.30	setMonth	126
3.30.3.31	setSecond	126
3.30.3.32	setTime	126
3.30.3.33	setUTC	127
3.30.3.34	setYear	127
3.30.3.35	toString	128
3.30.3.36	toString	128
3.30.3.37	toString	128
3.31	ASN1TUniversalString Struct Reference	129
3.31.1	Detailed Description	129

3.31.2	Constructor & Destructor Documentation	129
3.31.2.1	ASN1TUniversalString	129
3.32	ASN1TUTCTime Class Reference	130
3.32.1	Detailed Description	130
3.32.2	Constructor & Destructor Documentation	130
3.32.2.1	ASN1TUTCTime	130
3.32.2.2	ASN1TUTCTime	130
3.32.2.3	ASN1TUTCTime	131
3.32.2.4	ASN1TUTCTime	131
3.32.3	Member Function Documentation	131
3.32.3.1	clear	131
3.32.3.2	compileString	131
3.32.3.3	getFraction	131
3.32.3.4	parseString	132
3.32.3.5	setFraction	132
3.32.3.6	setTime	132
3.32.3.7	setUTC	133
3.32.3.8	setYear	133
3.33	OSRTBaseType Class Reference	134
3.33.1	Detailed Description	134
3.34	OSRTContext Class Reference	135
3.34.1	Detailed Description	135
3.34.2	Constructor & Destructor Documentation	136
3.34.2.1	OSRTContext	136
3.34.2.2	~OSRTContext	136
3.34.3	Member Function Documentation	136
3.34.3.1	_ref	136
3.34.3.2	_unref	136
3.34.3.3	getErrorInfo	136
3.34.3.4	getErrorInfo	136
3.34.3.5	getErrorInfo	137
3.34.3.6	getPtr	137
3.34.3.7	getRefCount	137
3.34.3.8	getStatus	137
3.34.3.9	isInitialized	137
3.34.3.10	memAlloc	137
3.34.3.11	memFreeAll	137

3.34.3.12	memFreePtr	138
3.34.3.13	memRealloc	138
3.34.3.14	memReset	138
3.34.3.15	printErrorInfo	138
3.34.3.16	resetErrorInfo	138
3.34.3.17	setDiag	138
3.34.3.18	setRunTimeKey	139
3.34.3.19	setStatus	139
3.34.4	Member Data Documentation	139
3.34.4.1	mbInitialized	139
3.34.4.2	mCount	139
3.34.4.3	mCtxt	139
3.34.4.4	mStatus	139
3.35	OSRTCtxtPtr Class Reference	140
3.35.1	Detailed Description	140
3.35.2	Constructor & Destructor Documentation	140
3.35.2.1	OSRTCtxtPtr	140
3.35.2.2	OSRTCtxtPtr	140
3.35.2.3	~OSRTCtxtPtr	141
3.35.3	Member Function Documentation	141
3.35.3.1	getCtxtPtr	141
3.35.3.2	isNull	141
3.35.3.3	operator OSRTContext *	141
3.35.3.4	operator->	141
3.35.3.5	operator=	141
3.35.3.6	operator=	141
3.35.3.7	operator==	141
3.35.4	Member Data Documentation	141
3.35.4.1	mPointer	141
3.36	OSRTFastString Class Reference	143
3.36.1	Detailed Description	143
3.36.2	Constructor & Destructor Documentation	143
3.36.2.1	OSRTFastString	143
3.36.2.2	OSRTFastString	143
3.36.2.3	OSRTFastString	144
3.36.2.4	OSRTFastString	144
3.36.2.5	~OSRTFastString	144

3.36.3	Member Function Documentation	144
3.36.3.1	clone	144
3.36.3.2	getUTF8Value	144
3.36.3.3	getValue	144
3.36.3.4	operator=	144
3.36.3.5	print	144
3.36.3.6	setValue	145
3.36.3.7	setValue	145
3.37	OSRTFileInputStream Class Reference	146
3.37.1	Detailed Description	146
3.37.2	Constructor & Destructor Documentation	146
3.37.2.1	OSRTFileInputStream	146
3.37.2.2	OSRTFileInputStream	146
3.37.2.3	OSRTFileInputStream	147
3.37.2.4	OSRTFileInputStream	147
3.37.3	Member Function Documentation	147
3.37.3.1	isA	147
3.38	OSRTFileOutputStream Class Reference	148
3.38.1	Detailed Description	148
3.38.2	Constructor & Destructor Documentation	148
3.38.2.1	OSRTFileOutputStream	148
3.38.2.2	OSRTFileOutputStream	148
3.38.2.3	OSRTFileOutputStream	149
3.38.2.4	OSRTFileOutputStream	149
3.38.3	Member Function Documentation	149
3.38.3.1	isA	149
3.39	OSRTHexTextInputStream Class Reference	151
3.39.1	Detailed Description	151
3.39.2	Constructor & Destructor Documentation	151
3.39.2.1	OSRTHexTextInputStream	151
3.39.2.2	~OSRTHexTextInputStream	152
3.39.3	Member Function Documentation	152
3.39.3.1	isA	152
3.39.3.2	setOwnUnderStream	152
3.40	OSRTInputStream Class Reference	153
3.40.1	Detailed Description	153
3.40.2	Constructor & Destructor Documentation	154

3.40.2.1	OSRTInputStream	154
3.40.2.2	~OSRTInputStream	154
3.40.3	Member Function Documentation	154
3.40.3.1	close	154
3.40.3.2	currentPos	154
3.40.3.3	flush	154
3.40.3.4	getContext	155
3.40.3.5	getCtxtPtr	155
3.40.3.6	getErrorInfo	155
3.40.3.7	getErrorInfo	156
3.40.3.8	getPosition	156
3.40.3.9	getStatus	156
3.40.3.10	isA	156
3.40.3.11	isOpened	157
3.40.3.12	mark	157
3.40.3.13	markSupported	157
3.40.3.14	printErrorInfo	158
3.40.3.15	read	158
3.40.3.16	readBlocking	158
3.40.3.17	reset	158
3.40.3.18	resetErrorInfo	159
3.40.3.19	setPosition	159
3.40.3.20	skip	159
3.41	OSRTInputStreamIF Class Reference	160
3.41.1	Constructor & Destructor Documentation	160
3.41.1.1	~OSRTInputStreamIF	160
3.41.2	Member Function Documentation	160
3.41.2.1	currentPos	160
3.41.2.2	getPosition	161
3.41.2.3	isA	161
3.41.2.4	mark	161
3.41.2.5	markSupported	162
3.41.2.6	read	162
3.41.2.7	readBlocking	162
3.41.2.8	reset	163
3.41.2.9	setPosition	163
3.41.2.10	skip	163

3.42 OSRTInputStreamPtr Class Reference	164
3.43 OSRTMemoryInputStream Class Reference	165
3.43.1 Detailed Description	165
3.43.2 Constructor & Destructor Documentation	165
3.43.2.1 OSRTMemoryInputStream	165
3.43.2.2 OSRTMemoryInputStream	165
3.43.3 Member Function Documentation	166
3.43.3.1 isA	166
3.44 OSRTMemoryOutputStream Class Reference	167
3.44.1 Detailed Description	167
3.44.2 Constructor & Destructor Documentation	167
3.44.2.1 OSRTMemoryOutputStream	167
3.44.2.2 OSRTMemoryOutputStream	167
3.44.2.3 OSRTMemoryOutputStream	168
3.44.3 Member Function Documentation	168
3.44.3.1 getBuffer	168
3.44.3.2 isA	168
3.44.3.3 reset	169
3.45 OSRTMessageBuffer Class Reference	170
3.45.1 Detailed Description	170
3.45.2 Constructor & Destructor Documentation	171
3.45.2.1 OSRTMessageBuffer	171
3.45.2.2 ~OSRTMessageBuffer	171
3.45.3 Member Function Documentation	171
3.45.3.1 getAppInfo	171
3.45.3.2 getByteIndex	171
3.45.3.3 getContext	171
3.45.3.4 getCtxtPtr	171
3.45.3.5 getErrorInfo	172
3.45.3.6 getErrorInfo	172
3.45.3.7 getMsgCopy	172
3.45.3.8 getMsgPtr	172
3.45.3.9 getStatus	172
3.45.3.10 init	173
3.45.3.11 initBuffer	173
3.45.3.12 printErrorInfo	173
3.45.3.13 resetErrorInfo	173

3.45.3.14	setAppInfo	173
3.45.3.15	setDiag	174
3.45.4	Member Data Documentation	174
3.45.4.1	mBufferType	174
3.46	OSRTMessageBufferIF Class Reference	175
3.46.1	Detailed Description	175
3.46.2	Constructor & Destructor Documentation	176
3.46.2.1	~OSRTMessageBufferIF	176
3.46.3	Member Function Documentation	176
3.46.3.1	getAppInfo	176
3.46.3.2	getByteIndex	176
3.46.3.3	getMsgCopy	176
3.46.3.4	getMsgPtr	176
3.46.3.5	init	176
3.46.3.6	initBuffer	177
3.46.3.7	isA	177
3.46.3.8	setAppInfo	177
3.46.3.9	setDiag	177
3.46.3.10	setNamespace	178
3.47	OSRTOutputStream Class Reference	179
3.47.1	Detailed Description	179
3.47.2	Constructor & Destructor Documentation	179
3.47.2.1	OSRTOutputStream	179
3.47.2.2	~OSRTOutputStream	180
3.47.3	Member Function Documentation	180
3.47.3.1	close	180
3.47.3.2	flush	180
3.47.3.3	getContext	180
3.47.3.4	getCtxtPtr	181
3.47.3.5	getErrorInfo	181
3.47.3.6	getErrorInfo	181
3.47.3.7	getStatus	181
3.47.3.8	isA	182
3.47.3.9	isOpened	182
3.47.3.10	printErrorInfo	182
3.47.3.11	resetErrorInfo	182
3.47.3.12	write	183

3.47.3.13	write	183
3.48	OSRTOutputStreamIF Class Reference	184
3.48.1	Constructor & Destructor Documentation	184
3.48.1.1	~OSRTOutputStreamIF	184
3.48.2	Member Function Documentation	184
3.48.2.1	isA	184
3.48.2.2	write	185
3.49	OSRTOutputStreamPtr Class Reference	186
3.50	OSRTSocket Class Reference	187
3.50.1	Detailed Description	187
3.50.2	Constructor & Destructor Documentation	188
3.50.2.1	OSRTSocket	188
3.50.2.2	OSRTSocket	188
3.50.2.3	OSRTSocket	188
3.50.2.4	~OSRTSocket	188
3.50.3	Member Function Documentation	188
3.50.3.1	accept	188
3.50.3.2	addrToString	189
3.50.3.3	bind	189
3.50.3.4	bind	189
3.50.3.5	bind	190
3.50.3.6	bindUrl	190
3.50.3.7	blockingRead	190
3.50.3.8	close	191
3.50.3.9	connect	191
3.50.3.10	connectUrl	191
3.50.3.11	getOwnership	192
3.50.3.12	getSocket	192
3.50.3.13	getStatus	192
3.50.3.14	listen	192
3.50.3.15	recv	192
3.50.3.16	send	193
3.50.3.17	setOwnership	193
3.50.3.18	stringToAddr	193
3.51	OSRTSocketInputStream Class Reference	195
3.51.1	Detailed Description	195
3.51.2	Constructor & Destructor Documentation	195

3.51.2.1	OSRTSocketInputStream	195
3.51.2.2	OSRTSocketInputStream	196
3.51.2.3	OSRTSocketInputStream	196
3.51.2.4	OSRTSocketInputStream	196
3.51.3	Member Function Documentation	196
3.51.3.1	isA	196
3.52	OSRTSocketOutputStream Class Reference	198
3.52.1	Detailed Description	198
3.52.2	Constructor & Destructor Documentation	198
3.52.2.1	OSRTSocketOutputStream	198
3.52.2.2	OSRTSocketOutputStream	199
3.52.2.3	OSRTSocketOutputStream	199
3.52.2.4	OSRTSocketOutputStream	199
3.52.3	Member Function Documentation	199
3.52.3.1	isA	199
3.53	OSRTStream Class Reference	201
3.53.1	Detailed Description	201
3.53.2	Constructor & Destructor Documentation	202
3.53.2.1	OSRTStream	202
3.53.2.2	~OSRTStream	202
3.53.3	Member Function Documentation	202
3.53.3.1	close	202
3.53.3.2	flush	202
3.53.3.3	getContext	203
3.53.3.4	getCtxtPtr	203
3.53.3.5	getErrorInfo	203
3.53.3.6	getErrorInfo	203
3.53.3.7	getStatus	204
3.53.3.8	isOpened	204
3.53.3.9	printErrorInfo	204
3.53.3.10	resetErrorInfo	204
3.54	OSRTStreamIF Class Reference	205
3.54.1	Member Function Documentation	205
3.54.1.1	close	205
3.54.1.2	flush	205
3.54.1.3	isOpened	206
3.55	OSRTString Class Reference	207

3.55.1	Detailed Description	207
3.55.2	Constructor & Destructor Documentation	207
3.55.2.1	OSRTString	207
3.55.2.2	OSRTString	207
3.55.2.3	OSRTString	208
3.55.2.4	OSRTString	208
3.55.2.5	~OSRTString	208
3.55.3	Member Function Documentation	208
3.55.3.1	clone	208
3.55.3.2	getUTF8Value	208
3.55.3.3	getValue	208
3.55.3.4	operator=	208
3.55.3.5	print	208
3.55.3.6	setValue	209
3.55.3.7	setValue	209
3.56	OSRTStringIF Class Reference	210
3.56.1	Detailed Description	210
3.56.2	Constructor & Destructor Documentation	210
3.56.2.1	OSRTStringIF	210
3.56.2.2	OSRTStringIF	210
3.56.2.3	OSRTStringIF	211
3.56.2.4	~OSRTStringIF	211
3.56.3	Member Function Documentation	211
3.56.3.1	clone	211
3.56.3.2	getUTF8Value	211
3.56.3.3	getValue	211
3.56.3.4	print	211
3.56.3.5	setValue	211
3.56.3.6	setValue	212
3.57	OSRTUTF8String Class Reference	213
3.57.1	Detailed Description	213
3.57.2	Constructor & Destructor Documentation	213
3.57.2.1	OSRTUTF8String	213
3.57.2.2	OSRTUTF8String	213
3.57.2.3	OSRTUTF8String	214
3.57.2.4	OSRTUTF8String	214
3.57.2.5	~OSRTUTF8String	214

3.57.3	Member Function Documentation	214
3.57.3.1	c_str	214
3.57.3.2	clone	214
3.57.3.3	copyValue	214
3.57.3.4	getValue	214
3.57.3.5	operator=	214
3.57.3.6	print	215
3.57.3.7	setValue	215
4	File Documentation	216
4.1	ASN1CBitStr.h File Reference	216
4.1.1	Detailed Description	216
4.2	ASN1CGeneralizedTime.h File Reference	217
4.2.1	Detailed Description	217
4.3	ASN1Context.h File Reference	218
4.3.1	Detailed Description	218
4.4	asn1CppEvtHndlr.h File Reference	219
4.4.1	Detailed Description	219
4.5	asn1CppTypes.h File Reference	220
4.5.1	Detailed Description	220
4.6	ASN1CSeqOfList.h File Reference	221
4.6.1	Detailed Description	221
4.7	ASN1CTime.h File Reference	222
4.7.1	Detailed Description	222
4.8	ASN1CUTCTime.h File Reference	223
4.8.1	Detailed Description	223
4.9	asn1ErrCodes.h File Reference	224
4.9.1	Detailed Description	224
4.10	ASN1TObjId.h File Reference	225
4.10.1	Detailed Description	225
4.11	ASN1TOctStr.h File Reference	226
4.11.1	Detailed Description	226
4.12	ASN1TTime.h File Reference	227
4.12.1	Detailed Description	227
4.13	OSRTBaseType.h File Reference	228
4.13.1	Detailed Description	228
4.14	OSRTCContext.h File Reference	229

4.14.1	Detailed Description	229
4.14.2	Function Documentation	229
4.14.2.1	operator delete	229
4.14.2.2	operator new	229
4.15	OSRTFastString.h File Reference	230
4.15.1	Detailed Description	230
4.16	OSRTFileInputStream.h File Reference	231
4.16.1	Detailed Description	231
4.17	OSRTFileOutputStream.h File Reference	232
4.17.1	Detailed Description	232
4.18	OSRTHexTextInputStream.h File Reference	233
4.18.1	Detailed Description	233
4.19	OSRTInputStream.h File Reference	234
4.19.1	Detailed Description	234
4.20	OSRTInputStreamIF.h File Reference	235
4.20.1	Detailed Description	235
4.21	OSRTMemoryInputStream.h File Reference	236
4.21.1	Detailed Description	236
4.22	OSRTMemoryOutputStream.h File Reference	237
4.22.1	Detailed Description	237
4.23	OSRTMsgBuf.h File Reference	238
4.23.1	Detailed Description	238
4.24	OSRTMsgBufIF.h File Reference	239
4.24.1	Detailed Description	239
4.25	OSRTOutputStream.h File Reference	240
4.25.1	Detailed Description	240
4.26	OSRTOutputStreamIF.h File Reference	241
4.26.1	Detailed Description	241
4.27	OSRTSocket.h File Reference	242
4.27.1	Detailed Description	242
4.28	OSRTSocketInputStream.h File Reference	243
4.28.1	Detailed Description	243
4.29	OSRTSocketOutputStream.h File Reference	244
4.29.1	Detailed Description	244
4.30	OSRTStream.h File Reference	245
4.30.1	Detailed Description	245
4.31	OSRTStreamIF.h File Reference	246

4.31.1 Detailed Description	246
4.32 OSRTString.h File Reference	247
4.32.1 Detailed Description	247
4.33 OSRTStringIF.h File Reference	248
4.33.1 Detailed Description	248
4.34 OSRTUTF8String.h File Reference	249
4.34.1 Detailed Description	249

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

Classes

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

Modules

- [Date and Time Runtime Classes](#)

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

Classes

- struct [ASN1TDynBitStr](#)
- struct [ASN1TBitStr32](#)
- struct [ASN1TBMPString](#)
- struct [ASN1TUniversalString](#)
- struct [ASN1TOpenType](#)
- struct [Asn1TObject](#)
- struct [ASN1TObjId64](#)
- struct [ASN1TSeqExt](#)
- struct [ASN1TPDU](#)
- struct [ASN1TSeqOfList](#)
- struct [ASN1TPDUSeqOfList](#)
- struct [ASN1TObjId](#)
- struct [ASN1TDynOctStr](#)

Modules

- [Date and Time Runtime Classes](#)

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

lhs - C++ object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.2 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

lhs - C object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.3 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

lhs - C object identifier value.

rhs - C object identifier value

Returns

- True if values are equal.

2.4.2.4 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

lhs - C++ object identifier value.

rhs - C++ object identifier value

Returns

- True if values are equal.

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

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

Parameters

lhs - C++ object identifier value.

rhs - C++ object identifier value.

2.4.2.6 `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

lhs - C++ object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.7 `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

lhs - C object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.8 `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

lhs - C object identifier value.

rhs - C object identifier value.

Returns

- True if values are equal.

2.4.2.9 `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

lhs - C++ object identifier value.

rhs - C++ object identifier value.

Returns

- True if values are equal.

2.4.2.10 **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

lhs - C object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.11 **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

lhs - C++ object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.12 **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

lhs - C object identifier value.

rhs - C object identifier value

Returns

- True if values are equal.

2.4.2.13 **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

lhs - C++ object identifier value.

rhs - C++ object identifier value

Returns

- True if values are equal.

2.4.2.14 **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

lhs - C object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.15 **int operator== (const ASN1OBJID & lhs, const ASN1OBJID & rhs)**

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

Parameters

lhs - C object identifier value.

rhs - C object identifier value.

Returns

- True if values are equal.

2.4.2.16 **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

lhs - C object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.17 **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

lhs - C object identifier value.

rhs - C object identifier value.

Returns

- True if values are equal.

2.4.2.18 **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

lhs - C++ object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.19 **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

lhs - C++ object identifier value.

rhs - C++ object identifier value.

Returns

- True if values are equal.

2.4.2.20 **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

lhs - C object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.21 **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

lhs - C object identifier value.

rhs - C object identifier value.

Returns

- True if values are equal.

2.4.2.22 `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

lhs - C++ object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns

- True if values are equal.

2.4.2.23 `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

lhs - C++ object identifier value.

rhs - C++ object identifier value.

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](#)

Defines

- #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.

2.7 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](#)

Defines

- #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.

2.12 Run-time error status codes.

Defines

- #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`)

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 Define Documentation

2.12.2.1 #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.

2.12.2.2 #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.

2.12.2.3 #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.

2.12.2.4 #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.

2.12.2.5 #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.

2.12.2.6 #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.

2.12.2.7 #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.

2.12.2.8 #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).

2.12.2.9 #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.

2.12.2.10 #define ASN_E_INVTCVAL (ASN_E_BASE-5)

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

2.12.2.11 #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.

2.12.2.12 #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().

2.12.2.13 #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 (AsnIRTPProd) to decode a type at run-time and the type of the element could not be resolved.

2.12.2.14 #define ASN_OK_FRAG 2

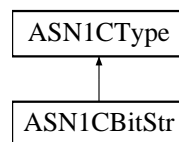
Fragment decode success status. This is returned when decoding is successful but only a fragment of the item was decoded. User should repeat the decode operation in order to fully decode message.

Chapter 3

Class Documentation

3.1 ASN1CBitStr Class Reference

Inheritance diagram for ASN1CBitStr:



Public Member Functions

- EXTRTMETHOD [ASN1CBitStr](#) ([OSRTMessageBufferIF](#) &msgbuf, OSUINT32 nbits)
- EXTRTMETHOD [ASN1CBitStr](#) ([OSRTMessageBufferIF](#) &msgbuf, OSOCTET *bitStr, OSUINT32 &numbits, OSUINT32 maxNumbits_)
- EXTRTMETHOD [ASN1CBitStr](#) ([OSRTMessageBufferIF](#) &msgbuf, OSOCTET *bitStr, OSUINT8 &numbits, OSUINT32 maxNumbits_)
- EXTRTMETHOD [ASN1CBitStr](#) ([OSRTMessageBufferIF](#) &msgbuf, OSOCTET *bitStr, OSUINT16 &numbits, OSUINT32 maxNumbits_)
- EXTRTMETHOD [ASN1CBitStr](#) ([OSRTMessageBufferIF](#) &msgbuf, OSOCTET *bitStr, OSUINT32 maxNumbits_)
- 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 octsNumbits)
- int **doAnd** (const **ASN1TDynBitStr** &bitStr)
- int **doAnd** (const **ASN1CBitStr** &bitStr)
- EXTRTMETHOD int **doOr** (const OSOCKET *pOctstr, OSUINT32 octsNumbits)
- int **doOr** (const **ASN1TDynBitStr** &bitStr)
- int **doOr** (const **ASN1CBitStr** &bitStr)
- EXTRTMETHOD int **doXor** (const OSOCKET *pOctstr, OSUINT32 octsNumbits)
- int **doXor** (const **ASN1TDynBitStr** &bitStr)
- int **doXor** (const **ASN1CBitStr** &bitStr)
- EXTRTMETHOD int **doAndNot** (const OSOCKET *pOctstr, OSUINT32 octsNumbits)
- 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 Constructor & Destructor Documentation

3.1.1.1 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

msgbuf - ASN.1 message buffer or stream object.

nbits - Number of bits this bit string can contain (zero if unbounded).

3.1.1.2 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

msgbuf - ASN.1 message buffer or stream object.

bitStr - Pointer to static byte array

numbits - Reference to length of bit string (in bits)

maxNumbits_ - sets maximum length in bits

3.1.2 Member Function Documentation

3.1.2.1 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.

3.1.2.2 `int ASN1BitStr::change (OSUINT32 bitIndex, OSBOOL value) [inline]`

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

Parameters

bitIndex 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).

value 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.

3.1.2.3 `EXTRTMETHOD void ASN1BitStr::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.

3.1.2.4 `EXTRTMETHOD int ASN1BitStr::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

fromIndex 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).

toIndex 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.

3.1.2.5 EXTRTMETHOD int ASN1CBitStr::clear (OSUINT32 *bitIndex*)

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

Parameters

bitIndex 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.

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

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

Parameters

bitStr 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().

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

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

Parameters

bitStr 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.

3.1.2.8 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

pOctstr A pointer to octets of another bit string for performing logical operation.

octsNumbits A number of bits in arguent bit string.

Returns

Completion status of operation:

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

3.1.2.9 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

bitStr 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().

3.1.2.10 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

bitStr A reference t another bit string represented by [ASN1TDynBitStr](#) type for performing logical operation.

Returns

Completion status of operation:

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

3.1.2.11 EXTRMETHOD 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 clars all of the bits in this bit string whose corresponding bit is set in the specified bit string.

Parameters

pOctstr A pointer to octets of another bit string for performing logical operation.

octsNumbits A number of bits in argument bit string.

Returns

Completion status of operation:

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

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

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

Parameters

bitStr 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().

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

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

Parameters

bitStr 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.

3.1.2.14 `EXTRTMETHOD 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

pOctstr A pointer to octets of another bit string for performing logical operation.

octsNumbits A number of bits in arguent bit string.

Returns

Completion status of operation:

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

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

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

Parameters

bitStr 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().

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

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

Parameters

bitStr A reference t another bit string represented by [ASN1TDynBitStr](#) type for performing logical operation.

Returns

Completion status of operation:

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

3.1.2.17 EXTRTMETHOD int ASN1CBitStr::doXor (const OSOCTET * *pOctstr*, OSUIN32 *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

pOctstr A pointer to octets of another bit string for performing logical operation.

octsNumbits A number of bits in arguent bit string.

Returns

Completion status of operation:

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

3.1.2.18 EXTRTMETHOD 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

fromIndex 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).

toIndex 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).

pBuf Pointer to destination buffer where bytes will be copied.

bufSz 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*).

3.1.2.19 EXTRTMETHOD OSBOOL ASN1CBitStr::get (OSUINT32 *bitIndex*)

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

Parameters

bitIndex 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.

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

This method copies the bit string to the given buffer.

Parameters

pBuf Pointer to the destination buffer where bytes will be copied.

bufSz 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.

3.1.2.21 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

fromIndex 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).

toIndex 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.

3.1.2.22 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

bitIndex 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.

3.1.2.23 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.

3.1.2.24 OSBOOL ASN1CBitStr::isSet (OSUINT32 *bitIndex*) [inline]

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

See also

[get](#) (OSUINT32 *bitIndex*)

3.1.2.25 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\(\)](#).

3.1.2.26 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](#).

3.1.2.27 EXTRTMETHOD ASN1CBitStr::operator ASN1TDynBitStr * ()

This method returns a pointer to the filled ANSDITDynBitStr variable.

Memory for the ASN1DynBitStr variable is alloted using memory [memAlloc](#) and bits are copied into it.

Parameters

- none

Returns

Pointer to a filled [ASN1TDynBitStr](#).

3.1.2.28 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

fromIndex 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).

toIndex 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.

3.1.2.29 EXTRTMETHOD int ASN1CBitStr::set (OSUINT32 *bitIndex*)

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

Parameters

bitIndex 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.

3.1.2.30 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

shift Number of bits to be shifted.

Returns

Completion status of operation:

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

3.1.2.31 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

shift Number of bits to be shifted.

Returns

Completion status of operation:

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

3.1.2.32 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.

3.1.2.33 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`.

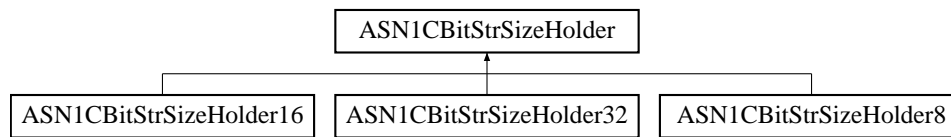
The documentation for this class was generated from the following file:

- [ASN1CBitStr.h](#)

3.2 ASN1BitStrSizeHolder Class Reference

```
#include <ASN1BitStr.h>
```

Inheritance diagram for ASN1BitStrSizeHolder:



Public Member Functions

- virtual `ASN1BitStrSizeHolder * clone ()=0`
- virtual `OSUINT32 getValue () const =0`
- virtual `int setValue (OSUINT32 value)=0`

3.2.1 Detailed Description

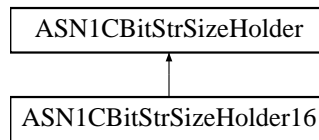
ASN.1 bit string control class. The `ASN1BitStr` class is derived from the `ASN1Type` base class. It is used as the base class for generated control classes for the ASN.1 BIT STRING type. This class provides utility methods for operating on the bit string referenced by the generated class. This class can also be used inline to operate on bits within generated BIT STRING elements in a SEQUENCE, SET, or CHOICE construct.

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

- [ASN1BitStr.h](#)

3.3 ASN1CBitStrSizeHolder16 Class Reference

Inheritance diagram for ASN1CBitStrSizeHolder16:



Public Member Functions

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

Protected Attributes

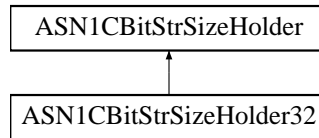
- OSUINT16 & **mSize**

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

- [ASN1CBitStr.h](#)

3.4 ASN1CBitStrSizeHolder32 Class Reference

Inheritance diagram for ASN1CBitStrSizeHolder32:



Public Member Functions

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

Protected Attributes

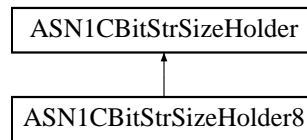
- OSUINT32 & **mSize**

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

- [ASN1CBitStr.h](#)

3.5 ASN1CBitStrSizeHolder8 Class Reference

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**

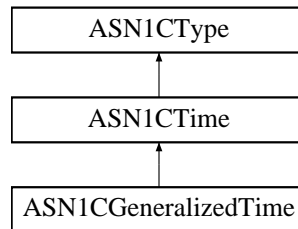
The documentation for this class was generated from the following file:

- [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, OS-
BOOL useDerRules=FALSE)
- EXTRTMETHOD [ASN1CGeneralizedTime](#) ([OSRTMessageBufferIF](#) &msgBuf, [ASN1GeneralizedTime](#) &buf,
OSBOOL useDerRules=FALSE)
- EXTRTMETHOD [ASN1CGeneralizedTime](#) ([OSRTContext](#) &ctxt, char *&buf, int bufSize, OSBOOL useDer-
Rules=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 [ASN1Time](#) & [getTimeObj](#) ()
- virtual const [ASN1Time](#) & [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](#)

3.6.1 Detailed Description

ASN.1 GeneralizedTime control class. The [ASN1CGeneralizedTime](#) class is derived from the [ASN1Time](#) 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 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

msgBuf Reference to an OSRTMessage buffer derived object (for example, an ASN1BEREncodeBuffer).

buf A reference pointer to the time string buffer.

bufSize The size of the passed buffer, in bytes.

useDerRules An OSBOOL value.

3.6.2.2 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

msgBuf Reference to an OSRTMessage buffer derived object (for example, an ASN1BEREncodeBuffer).

buf A reference pointer to the time string buffer.

useDerRules An OSBOOL value.

3.6.3 Member Function Documentation

3.6.3.1 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.

3.6.3.2 EXTRTMETHOD int ASN1CGeneralizedTime::setCentury (short *century*)

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

Parameters

century Century part (first two digits) of the year component.

Returns

Completion status of operation:

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

3.6.3.3 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

time The time value, expressed as a number of seconds from January 1, 1970.

diffTime 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](#).

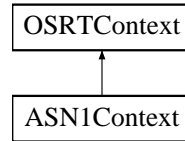
The documentation for this class was generated from the following file:

- [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](#) ()

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 EXTRTMETHOD ASN1Context::ASN1Context ()

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

3.7.3 Member Function Documentation

3.7.3.1 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

key - array of octets with the key

keylen - number of octets in key array.

Returns

Completion status of operation:

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

Reimplemented from [OSRTContext](#).

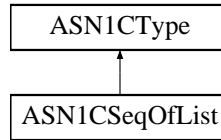
The documentation for this class was generated from the following file:

- [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 [free](#) ()
- 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 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

msgBuf Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).

lst Reference to a linked list structure.

initBeforeUse Set to TRUE if the passed linked list needs to be initialized (rtxDListInit to be called).

3.8.2.2 EXTRTMETHOD ASN1CSeqOfList::ASN1CSeqOfList (OSRTMessageBufferIF & *msgBuf*)

This constructor creates an empty linked list.

Parameters

msgBuf Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).

3.8.2.3 EXTRTMETHOD ASN1CSeqOfList::ASN1CSeqOfList (ASN1CType & *ccobj*)

This constructor creates an empty linked list.

Parameters

ccobj Reference to a control class object (for example, any generated ASN1C_ class object).

3.8.2.4 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

msgBuf Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).

lst Reference to a linked list holder.

3.8.2.5 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

ccobj Reference to a control class object (for example, any generated ASN1C_ class object).

lst Reference to a linked list holder.

3.8.2.6 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

msgBuf Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).

lst Reference to a linked list holder.

3.8.3 Member Function Documentation

3.8.3.1 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

data Pointer to a data item to be appended to the list.

Returns

- none

3.8.3.2 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

data Pointer to source array to be appended to the list.

numElems The number of elements in the source array.

elemSize The size of one element in the array. Use the *sizeof()* operator to pass this parameter.

Returns

- none

3.8.3.3 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

data Pointer to source array to be appended to the list.

numElems The number of elements in the source array.

elemSize The size of one element in the array. Use the *sizeof()* operator to pass this parameter.

Returns

- none

3.8.3.4 EXTRTMETHOD void ASN1CSeqOfList::clear ()

This method removes all items from the list.

3.8.3.5 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

data - Pointer to data item.

Returns

TRUE if this pointer value found in the list.

3.8.3.6 EXTRTMETHOD void ASN1CSeqOfList::free ()

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

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

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

Parameters

index Index of the item to be returned.

Returns

The item at the specified index in the list.

3.8.3.8 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 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 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

data - 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 void ASN1CSeqOfList::init () [inline]

This method initializes the linked list structure.

3.8.3.12 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

index Index at which the specified item is to be inserted.

data Pointer to data item to be appended to the list.

Returns

- none

3.8.3.13 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 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 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

data The item of the list to be iterated first.

Returns

The iterator over this linked list.

3.8.3.16 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 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 EXTRMETHOD 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

data - Pointer to the data item to be appended to the list.

3.8.3.19 EXTRMETHOD 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

index Index of the item to be removed.

Returns

- none

3.8.3.20 void ASN1CSeqOfList::removeFirst () [inline]

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

Parameters

- none

Returns

- none

3.8.3.21 void ASN1CSeqOfList::removeLast () [inline]

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

Parameters

- none

Returns

- none

3.8.3.22 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

index The index of the item to be replaced.

data The item to be stored at the specified index.

Returns

The item previously at the specified position.

3.8.3.23 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 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

pArray Pointer to destination array.

elemSize The size of one element in the array. Use the `sizeof()` operator to pass this parameter.

allocatedElems 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.

3.8.3.25 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

elemSize The size of one element in the array. Use the `sizeof()` operator to pass this parameter.

Returns

The point to 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 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 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.

3.9.2.3 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

data The element to be inserted

Returns

Completion status of operation:

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

3.9.2.4 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 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 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 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

data 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.

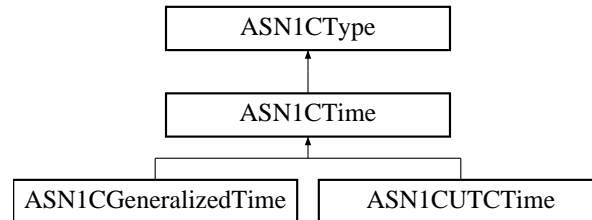
The documentation for this class was generated from the following file:

- [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 useDerRules)
- EXTRTMETHOD [ASN1CTime](#) ([OSRTMessageBufferIF](#) &msgBuf, [ASN1VisibleString](#) &buf, OSBOOL useDerRules)
- EXTRTMETHOD [ASN1CTime](#) ([OSRTContext](#) &ctxt, char *&buf, int bufSize, OSBOOL useDerRules)
- EXTRTMETHOD [ASN1CTime](#) ([OSRTContext](#) &ctxt, [ASN1VisibleString](#) &buf, OSBOOL useDerRules)
- EXTRTMETHOD [ASN1CTime](#) (const [ASN1CTime](#) &original)
- 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 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

msgBuf Reference to an OSRTMessage buffer derived object (for example, ASNBEREncodeBuffer).

buf Reference to a pointer to a time string buffer.

bufSize Size of buffer in bytes.

useDerRules Use the Distinguished Encoding Rules (DER) to operate on this time value.

3.10.2.2 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

msgBuf Reference to an OSRTMessage buffer derived object (for example, ASNBEREncodeBuffer).

buf Reference to a visible string object to hold the time data.

useDerRules Use the Distinguished Encoding Rules (DER) to operate on this time value.

3.10.2.3 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

original The original time string object value.

3.10.3 Member Function Documentation

3.10.3.1 virtual EXTRTMETHOD void ASN1Time::clear () [virtual]

This method clears the time string.

Note the action of this method may differ for different inherited [ASN1Time](#) classes.

Parameters

- none

Returns

- none

3.10.3.2 virtual EXTRTMETHOD int ASN1Time::equals (ASN1Time &) [virtual]

This method compares times.

3.10.3.3 virtual EXTRTMETHOD int ASN1Time::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 [ASN1Time](#) 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.

3.10.3.4 virtual EXTRTMETHOD int ASN1Time::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 or negative time difference. Note that the return value may differ for different inherited [ASN1Time](#) 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.

3.10.3.5 virtual EXTRTMETHOD 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 or 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.

3.10.3.6 virtual EXTRTMETHOD 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 or 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.

3.10.3.7 virtual EXTRTMETHOD 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](#).

3.10.3.8 virtual EXTRTMETHOD 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.

3.10.3.9 virtual EXTRTMETHOD int ASN1Time::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.

3.10.3.10 virtual EXTRTMETHOD int ASN1Time::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.

3.10.3.11 virtual EXTRTMETHOD int ASN1Time::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 [ASN1Time](#) classes.

Parameters

- none

Returns

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

3.10.3.12 virtual EXTRTMETHOD int ASN1Time::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 [ASN1Time](#) classes.

Parameters

- none

Returns

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

3.10.3.13 virtual EXTRTMETHOD 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 values 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.

3.10.3.14 virtual EXTRTMETHOD 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.

3.10.3.15 virtual EXTRTMETHOD 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.

3.10.3.16 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.

3.10.3.17 virtual EXTRMETHOD int ASN1Time::getYear () [virtual]

This method returns the year component of the time value.

Note that the return value may differ for different inherited [ASN1Time](#) 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.

3.10.3.18 virtual EXTRMETHOD int ASN1Time::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 [ASN1Time](#) classes.

Parameters

string The time string value to be parsed.

Returns

Completion status of operation:

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

3.10.3.19 virtual EXTRMETHOD 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

day_ Day of month component (1 - 31).

Returns

Completion status of operation:

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

3.10.3.20 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.

3.10.3.21 virtual EXTRTMETHOD 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

inMinutes 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.

3.10.3.22 virtual EXTRTMETHOD 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

dhour The negative or positive hour component of the difference between the time zone of the object and the UTC time (-12 - +12).

dminute 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.

3.10.3.23 virtual EXTRTMETHOD 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

dhour 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.

3.10.3.24 virtual EXTRTMETHOD 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

frac Second's decimal fraction component.

Returns

Completion status of operation:

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

3.10.3.25 virtual EXTRTMETHOD 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

frac Second's decimal fraction component.

fracLen Specifies number of digits in second's fraction.

Returns

Completion status of operation:

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

3.10.3.26 virtual EXTRTMETHOD 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

fraction Second's decimal fraction component (0 - 9).

fracLen Optional parameter specifies number of digits in second's fraction.

Returns

Completion status of operation:

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

3.10.3.27 virtual EXTRTMETHOD 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

hour_ Hour component (0 - 23).

Returns

Completion status of operation:

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

3.10.3.28 virtual EXTRTMETHOD 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

minute_ Minute component (0 - 59).

Returns

Completion status of operation:

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

3.10.3.29 virtual EXTRTMETHOD 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

month_ Month component (1 - 12).

Returns

Completion status of operation:

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

3.10.3.30 virtual EXTRTMETHOD 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

second_ Second component (0 - 59).

Returns

Completion status of operation:

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

3.10.3.31 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

time The time value, expressed as a number of seconds from January 1, 1970.

diffTime 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](#).

3.10.3.32 virtual EXTRTMETHOD 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

utc UTC flag state.

Returns

Completion status of operation:

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

3.10.3.33 virtual EXTRTMETHOD 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

year_ Year component (full 4 digits).

Returns

Completion status of operation:

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

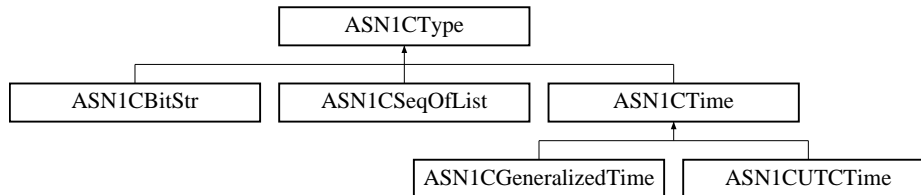
The documentation for this class was generated from the following file:

- [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](#) ()
- int [getStatus](#) () const
- void * [memAlloc](#) (size_t numocts)
- void [memFreeAll](#) ()
- void * [memRealloc](#) (void *ptr, size_t numocts)
- void [memReset](#) ()
- void [memFreePtr](#) (void *ptr)
- void [printErrorInfo](#) ()
- void [resetError](#) ()
- [OSBOOL](#) [setDiag](#) ([OSBOOL](#) value)
- virtual EXTRTMETHOD int [Encode](#) ()
- virtual EXTRTMETHOD int [Decode](#) ()
- virtual int [EncodeTo](#) ([OSRTMessageBufferIF](#) &)
- virtual int [DecodeFrom](#) ([OSRTMessageBufferIF](#) &)

Protected Member Functions

- EXTRTMETHOD [ASN1CType](#) ()
- EXTRTMETHOD [ASN1CType](#) ([OSRTContext](#) &ctxt)
- EXTRTMETHOD int [setMsgBuf](#) ([OSRTMessageBufferIF](#) &msgBuf, [OSBOOL](#) initBuf=FALSE)
- EXTRTMETHOD int [setRunTimeKey](#) (const [OSOCKET](#) *key, size_t keylen)

Protected Attributes

- [OSRTCtxtPtr](#) 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 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 EXTRTMETHOD ASN1CType::ASN1CType (OSRTContext & *ctxt*) [protected]

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

Parameters

ctxt - Reference to a context object.

3.11.2.3 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

msgBuf - Reference to a message buffer or stream object.

3.11.2.4 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

orig - Reference to a message buffer or stream object.

3.11.2.5 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 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

llist Linked list structure.

pdata Data record to be appended. Note that the pointer value is appended. The data is not copied.

3.11.3.2 virtual EXTRTMETHOD int ASN1CType::Decode () [virtual]

The `Decode` method decodes the ASN.1 message described by the encapsulated message buffer object.

Returns

Completion status of operation:

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

3.11.3.3 virtual int ASN1CType::DecodeFrom (OSRTMessageBufferIF &) [inline, virtual]

The `DecodeFrom` method decodes an ASN.1 message from the given message buffer or stream argument.

Parameters

msgBuf Message buffer or stream containing message to decode.

Returns

Completion status of operation:

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

References `ASN_E_NOTPDU`.

3.11.3.4 virtual EXTRTMETHOD 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 virtual int ASN1CType::EncodeTo (OSRTMessageBufferIF &) [inline, virtual]

The `EncodeTo` method encodes an ASN.1 message into the given message buffer or stream argument.

Parameters

msgBuf 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 OSRTCtxtPtr ASN1CType::getContext () [inline]

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

Returns

Context smart pointer object.

3.11.3.7 OSCTXT* ASN1CType::getCtxtPtr () [inline]

The getCtxtPtr method returns the underlying C runtime context. This context can be used in calls to C runtime functions.

3.11.3.8 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.

3.11.3.9 void* ASN1CType::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 [ASN1CType](#) derived control class object and the message buffer object are destroyed, this memory will be freed.

Parameters

numocts Number of bytes of memory to allocate

Returns

Void pointer to allocated memory or NULL if insufficient memory was available to fulfill the request.

3.11.3.10 void ASN1Type::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.

3.11.3.11 void ASN1Type::memFreePtr (void * ptr) [inline]

The memFreePtr method frees the memory at a specific location. This memory must have been allocated using the memAlloc method described earlier.

Parameters

ptr - Pointer to a block of memory allocated with memAlloc

3.11.3.12 void* ASN1Type::memRealloc (void * ptr, size_t numocts) [inline]

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

Parameters

ptr Original pointer containing dynamic memory to be resized.

numocts Number of bytes of memory to allocate

Returns

Reallocated memory pointer

3.11.3.13 void ASN1Type::memReset () [inline]

The memReset method resets dynamic memory using the C runtime memory management functions.

3.11.3.14 void ASN1Type::printErrorInfo () [inline]

The PrintErrorInfo method prints information on errors contained within the context.

3.11.3.15 void ASN1Type::resetError () [inline]

This method resets error status and stack trace. This method should be used to continue operations after recovering from the error.

3.11.3.16 OSBOOL ASN1Type::setDiag (OSBOOL value) [inline]

This method turns diagnostic tracing on or off.

Parameters

value Boolean value; TRUE = turn tracing on.

Returns

Previous state.

3.11.3.17 EXTRTMETHOD int ASN1CType::setRunTimeKey (const OSOCTET * *key*, size_t *keylen*) [protected]

This method sets run-time key to the context. This method does nothing for unlimited redistribution libraries.

Parameters

key - array of octets with the key

keylen - number of octets in key array.

Returns

Completion status of operation:

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

3.11.4 Member Data Documentation

3.11.4.1 OSRTCtxtPtr ASN1CType::mpContext [protected]

The mpContext member variable holds a reference-counted C runtime variable. This context is used in calls to all C run-time functions. The context pointed at by this smart-pointer object is shared with the message buffer object contained within this class.

3.11.4.2 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.

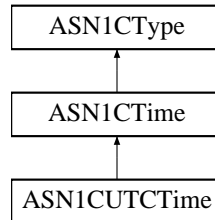
The documentation for this class was generated from the following file:

- [asn1CppType.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](#) &ctxt, char *&buf, int bufSize, OSBOOL useDerRules=FALSE)
- EXTRTMETHOD [ASN1CUTCTime](#) ([OSRTContext](#) &ctxt, [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

- [ASN1UTCTime](#) [timeObj](#)

3.12.1 Detailed Description

ASN.1 UTCTime control class. The ASN1CUTCTime 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 EXTRTMETHOD ASN1CUTCTime::ASN1CUTCTime (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

msgBuf Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).

buf Reference to a pointer to a time string buffer.

bufSize Size of passed buffer, in bytes.

useDerRules Use the Distinguished Encoding Rules to encode or decode the value,

3.12.2.2 EXTRTMETHOD ASN1CUTCTime::ASN1CUTCTime (OSRTMessageBufferIF & *msgBuf*, ASN1UTCTime & *buf*, OSBOOL *useDerRules* = FALSE)

This constructor creates a time string using the ASN1UTCTime 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

msgBuf Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).

buf Reference to a time string structure.

useDerRules Use the Distinguished Encoding Rules to encode or decode the value,

3.12.3 Member Function Documentation

3.12.3.1 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 [ASN1Time](#) 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 [ASN1Time](#).

3.12.3.2 EXTRTMETHOD int ASN1CUTCTime::setTime (time_t *time*, OSBOOL *diffTime*) [virtual]

Converts *time_t* to time string.

Parameters

time time to convert,

diffTime 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 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

- pCtxt* - Pointer to a context block structure.
- pCCB* - Pointer to a context control block structure.
- stat* - 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 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

- pCtxt* - Pointer to a context block structure.

pHandler - Pointer to error handler object to register.

Referenced by ASN1MessageBuffer::setErrorHandler().

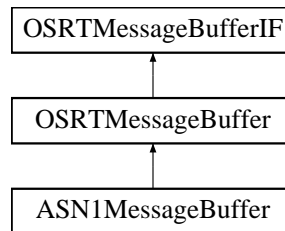
The documentation for this class was generated from the following file:

- [asn1CppEvtHndlr.h](#)

3.14 ASN1MessageBuffer Class Reference

```
#include <asn1CppTypes.h>
```

Inheritance diagram for ASN1MessageBuffer:



Public Member Functions

- virtual `~ASN1MessageBuffer ()`
- virtual void `addEventHandler (Asn1NamedEventHandler *pEventHandler)`
- virtual `ASN1BMPString * CStringToBMPString (const char *cstring, ASN1BMPString *pBMPString, Asn116BitCharSet *pCharSet=0)`
- virtual void `* getAppInfo ()`
- 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, size_t keylen)`
- `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)`

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 EXTRTMETHOD ASN1MessageBuffer::ASN1MessageBuffer (Type *bufferType*) [protected]

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

Parameters

bufferType Type of message buffer that is being created (for example, BEREncode).

3.14.2.2 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

bufferType Type of message buffer that is being created (for example, BEREncode).

pContext A pointer to an [OSRTContext](#) structure previously created by the user.

3.14.2.3 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 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

pEventHandler - Pointer to named event handler object to register.

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

3.14.3.2 virtual ASN1BMPString* ASN1MessageBuffer::CStringToBMPString (const char * *cstring*, ASN1BMPString * *pBMPString*, Asn116BitCharSet * *pCharSet* = 0) [virtual]

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

cstring - Null-terminated character string to convert

pBMPString - Pointer to BMP string target variable

pCharSet - Pointer to permitted alphabet character set. If provided, index to character within this set is returned.

3.14.3.3 `virtual void* ASN1MessageBuffer::getAppInfo () [inline, virtual]`

Returns a pointer to application-specific information block

Reimplemented from [OSRTMessageBuffer](#).

3.14.3.4 `virtual int ASN1MessageBuffer::initBuffer (const OSUTF8CHAR *) [inline, virtual]`

This version of the overloaded `initBuffer` method initializes the message buffer to point at the given null-terminated UTF-8 character string.

Parameters

str Pointer to a null-terminated ASCII character string.

Returns

Completion status of operation:

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

3.14.3.5 `virtual EXTRTMETHOD 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

unistr Pointer to a Unicode character string.

Returns

Completion status of operation:

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

3.14.3.6 `virtual EXTRTMETHOD 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

membuf `OSRTMEMBUF` memory buffer class object reference.

Returns

Completion status of operation:

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

3.14.3.7 virtual OSBOOL ASN1MessageBuffer::isA (Type) [inline, virtual]

This method checks the type of the message buffer.

Parameters

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

Returns

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

3.14.3.8 virtual void ASN1MessageBuffer::removeEventHandler (Asn1NamedEventHandler * pEventHandler) [inline, virtual]

The removeEventHandler method is used to de-register a user-defined named event handler.

Parameters

pEventHandler - Pointer to named event handler object to de-register.

References OSRTMessageBuffer::getCtxtPtr().

3.14.3.9 virtual void ASN1MessageBuffer::resetErrorInfo () [inline, virtual]

The resetErrorInfo method resets information on errors contained within the context.

Reimplemented from [OSRTMessageBuffer](#).

References OSRTMessageBuffer::resetErrorInfo().

3.14.3.10 virtual void ASN1MessageBuffer::setAppInfo (void *) [inline, virtual]

Sets the application-specific information block.

Reimplemented from [OSRTMessageBuffer](#).

3.14.3.11 virtual void ASN1MessageBuffer::setErrorHandler (Asn1ErrorHandler * pErrorHandler) [inline, virtual]

The setErrorHandler method is used to register a user-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

pErrorHandler - Pointer to error handler object to register.

References OSRTMessageBuffer::getCtxtPtr(), and Asn1ErrorHandler::setErrorHandler().

3.14.3.12 **EXTRTMETHOD** int ASN1MessageBuffer::setRunTimeKey (const OSOCTET * *key*, size_t *keylen*)

This method sets run-time key to the context. This method does nothing for unlimited redistribution libraries.

Parameters

key - array of octets with the key

keylen - number of octets in key array.

Returns

Completion status of operation:

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

3.14.3.13 **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().

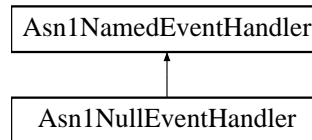
The documentation for this class was generated from the following file:

- [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 numSubIds, OSUINT32 *pSubIds)
- 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 OS-UTF8CHAR *data)
- static EXTRTMETHOD void **invokeNullValue** (OSCTXT *pCtxt)
- static EXTRTMETHOD void **invokeOidValue** (OSCTXT *pCtxt, OSUINT32 numSubIds, OSUINT32 *pSubIds)
- 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 OS-OCTET *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 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

- numbits* - Number of bits in the parsed value.
- data* - Pointer to a byte array that contains the bit string data.

Returns

- none

3.15.2.2 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

- value* Parsed value.

Returns

- none

3.15.2.3 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

nchars Number of characters in the parsed value.

data Pointer to an array containing 32-bit values. Each 32-bit integer value is a universal character.

Returns

- none

3.15.2.4 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

nchars Number of characters in the parsed value.

data Pointer to an array containing 16-bit values. These are represented using unsigned short integer values.

Returns

- none

3.15.2.5 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

nchars Number of characters in the parsed value.

value A UTF-8 character string.

Returns

- none

3.15.2.6 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

value Null terminated character string value.

Returns

- none

3.15.2.7 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

name 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".

index 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](#).

3.15.2.8 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

value - Parsed enumerated value

text - Textual value of enumerated identifier

Returns

- none

3.15.2.9 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

value Parsed value.

Returns

- none

3.15.2.10 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

value Parsed value.

Returns

- none

3.15.2.11 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

**3.15.2.12 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

numocts Number of octets in the parsed value.

data Pointer to byte array containing the octet string data.

Returns

- none

**3.15.2.13 virtual void Asn1NamedEventHandler::oidValue (OSUINT32 *numSubIds*, OSUINT32 **pSubIds*)
[inline, virtual]**

This method is invoked from within a decode function when a value of the OBJECT IDENTIFIER ASN.1 type is parsed.

Parameters

numSubIds Number of subidentifiers in the object identifier.

pSubIds Pointer to array containing the subidentifier values.

Returns

-none

3.15.2.14 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

- numocts* Number of octets in the parsed value.
- data* Pointer to bytet array contain in tencoded ASN.1 value.

Returns

- none

3.15.2.15 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

- value* Parsed value.

Returns

- none

3.15.2.16 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

- name* 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".
- index* 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](#).

3.15.2.17 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

- value* Parsed value.

Returns

- none

3.15.2.18 `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

value Parsed value.

Returns

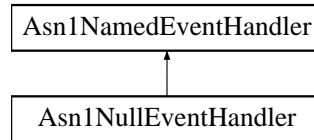
- none

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

- [asn1CppEvtHndlr.h](#)

3.16 Asn1NullEventHandler Class Reference

Inheritance diagram for Asn1NullEventHandler:



Public Member Functions

- virtual void [startElement](#) (const char *, int)
- virtual void [endElement](#) (const char *, int)

3.16.1 Member Function Documentation

3.16.1.1 virtual void Asn1NullEventHandler::endElement (const char * *name*, int *index*) [[inline](#), [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

name 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".

index 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

Implements [Asn1NamedEventHandler](#).

3.16.1.2 virtual void Asn1NullEventHandler::startElement (const char * *name*, int *index*) [[inline](#), [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

name 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".

index 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

Implements [Asn1NamedEventHandler](#).

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

- [asn1CppEvtHndlr.h](#)

3.17 ASN1BitStr32 Struct Reference

```
#include <asn1CppTypes.h>
```

Public Member Functions

- [ASN1BitStr32 \(\)](#)
- [ASN1BitStr32 \(OSUINT32 _numbits, const OSOCTET * _data\)](#)
- [ASN1BitStr32 \(ASN1BitStr32 &_bs\)](#)

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 <= 32 bits.

3.17.2 Constructor & Destructor Documentation

3.17.2.1 ASN1BitStr32::ASN1BitStr32 () [inline]

The default constructor creates an empty bit string.

3.17.2.2 ASN1BitStr32::ASN1BitStr32 (OSUINT32 *_numbits*, const OSOCTET * *_data*) [inline]

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

Parameters

_numbits - Number of bits in the bit string.

_data - The binary bit data values.

3.17.2.3 ASN1BitStr32::ASN1BitStr32 (ASN1BitStr32 & *_bs*) [inline]

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

Parameters

_bs - C bit string structure.

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

- [asn1CppTypes.h](#)

3.18 ASN1TBMPString Struct Reference

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

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 () [inline]

The default constructor creates an empty BMPString value.

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

- [asn1CppType.h](#)

3.19 ASN1TDynBitStr Struct Reference

```
#include <asn1CppTypes.h>
```

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::ASN1TDynBitStr () [inline]

The default constructor creates an empty bit string.

3.19.2.2 ASN1TDynBitStr::ASN1TDynBitStr (OSUINT32 *_numbits*, const OSOCTET * *_data*) [inline]

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

Parameters

_numbits - Number of bits in the bit string.

_data - The binary bit data values.

3.19.2.3 ASN1TDynBitStr::ASN1TDynBitStr (ASN1DynBitStr & *_bs*) [inline]

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

Parameters

_bs - C bit string structure.

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

- [asn1CppTypes.h](#)

3.20 ASN1TDynOctStr Struct Reference

```
#include <ASN1TOctStr.h>
```

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

3.20.1 Detailed Description

Dynamic octet string. This is the base class for generated C++ data type classes for unsized OCTET string's.

3.20.2 Constructor & Destructor Documentation

3.20.2.1 ASN1TDynOctStr::ASN1TDynOctStr () [inline]

The default constructor creates an empty octet string.

3.20.2.2 ASN1TDynOctStr::ASN1TDynOctStr (OSUINT32 _numocts, const OSOCTET * _data) [inline]

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

Parameters

_numocts - Number of octet in the octet string.

_data - The binary octet data values.

3.20.2.3 ASN1TDynOctStr::ASN1TDynOctStr (ASN1TDynOctStr &_os) [inline]

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

Parameters

_os - C octet string structure.

3.20.2.4 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

cstring - C null-terminated string.

3.20.3 Member Function Documentation

3.20.3.1 EXTRTMETHOD 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

n - Number of octets to compare

o - 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.

3.20.3.2 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

octet - Octet string object reference

3.20.3.3 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

cstring - C null-terminated string.

3.20.3.4 EXTRTMETHOD const char* ASN1TDynOctStr::toHexString (OSCTXT * *pctxt*) const

This method converts the binary octet string to a hexadecimal string representation.

Parameters

pctxt - Pointer to a context structure.

3.20.3.5 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

pctxt - Pointer to a context structure.

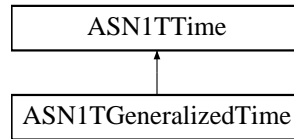
The documentation for this struct was generated from the following file:

- [ASN1TOctStr.h](#)

3.21 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

3.21.1 Detailed Description

ASN.1 GeneralizedTime utility class. The [ASN1TGeneralizedTime](#) class is derived from the [ASN1TTime](#) base class.

3.21.2 Constructor & Destructor Documentation

3.21.2.1 [ASN1TGeneralizedTime::ASN1TGeneralizedTime](#) () [inline]

A default constructor.

3.21.2.2 EXTRTMETHOD [ASN1TGeneralizedTime::ASN1TGeneralizedTime](#) (const char * buf, OSBOOL useDerRules = FALSE)

This constructor creates a time object using the specified time string.

Parameters

buf A pointer to the time string to be parsed.

useDerRules An OSBOOL value.

3.21.2.3 ASNITGeneralizedTime::ASNITGeneralizedTime (OSBOOL *useDerRules*) [inline]

This constructor creates an empty time object.

Parameters

useDerRules An OSBOOL value.

3.21.2.4 ASNITGeneralizedTime::ASNITGeneralizedTime (const ASNITGeneralizedTime & *original*) [inline]

A copy constructor.

3.21.3 Member Function Documentation

3.21.3.1 EXTRTMETHOD int ASNITGeneralizedTime::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

pbuf A pointer to destination buffer.

bufsize A size of destination buffer.

Returns

Completion status of operation:

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

Implements [ASNITTime](#).

3.21.3.2 EXTRTMETHOD int ASNITGeneralizedTime::getCentury () const

This method returns the centry 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 sucessful. If the operation fails, one of the negative status codes is returned.

3.21.3.3 EXTRTMETHOD int ASNITGeneralizedTime::parseString (const char * *string*) [virtual]

Parses sting.

Returns

Completion status of operation:

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

Implements [ASNITTime](#).

3.21.3.4 EXTRTMETHOD int ASNITGeneralizedTime::setCentury (short *century*)

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

Parameters

century Century part (first two digits) of the year component.

Returns

Completion status of operation:

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

3.21.3.5 EXTRTMETHOD int ASNITGeneralizedTime::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

time The time value, expressed as a number of seconds from January 1, 1970.

diffTime 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 [ASNITTime](#).

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

- [ASNITTime.h](#)

3.22 Asn1TObject Struct Reference

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

Public Member Functions

- [Asn1TObject \(\)](#)

3.22.1 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.

3.22.2 Constructor & Destructor Documentation

3.22.2.1 `Asn1TObject::Asn1TObject ()` [`inline`]

The default constructor creates an empty object value.

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

- [asn1CppType.h](#)

3.23 ASN1ObjId Struct Reference

```
#include <ASN1ObjId.h>
```

Public Member Functions

- [ASN1ObjId \(\)](#)
- virtual EXTRTMETHOD [~ASN1ObjId \(\)](#)
- EXTRTMETHOD [ASN1ObjId \(OSOCKET _numids, const OSUINT32 *_subids\)](#)
- EXTRTMETHOD [ASN1ObjId \(const ASN1OBJID &oid\)](#)
- EXTRTMETHOD [ASN1ObjId \(const ASN1ObjId &oid\)](#)
- EXTRTMETHOD [ASN1ObjId \(const char *dotted_oid_string\)](#)
- EXTRTMETHOD [ASN1ObjId & operator= \(const char *dotted_oid_string\)](#)
- EXTRTMETHOD void [operator= \(const ASN1OBJID &rhs\)](#)
- EXTRTMETHOD void [operator= \(const ASN1ObjId &rhs\)](#)
- EXTRTMETHOD [ASN1ObjId & operator+= \(const char *dotted_oid_string\)](#)
- EXTRTMETHOD [ASN1ObjId & operator+= \(const OSUINT32 i\)](#)
- EXTRTMETHOD [ASN1ObjId & operator+= \(const ASN1ObjId &o\)](#)
- EXTRTMETHOD const char * [toString \(OSCTXT *pctx\) const](#)
- EXTRTMETHOD void [set_data \(const OSUINT32 *raw_oid, OSUINT32 oid_len\)](#)
- EXTRTMETHOD int [nCompare \(const OSUINT32 n, const ASN1ObjId &o\) const](#)
- EXTRTMETHOD int [RnCompare \(const OSUINT32 n, const ASN1ObjId &o\) const](#)
- EXTRTMETHOD void [trim \(const OSUINT32 n\)](#)

3.23.1 Detailed Description

Object identifier. This is the base class for generated C++ data type classes for object identifier values.

3.23.2 Constructor & Destructor Documentation

3.23.2.1 ASN1ObjId::ASN1ObjId () [inline]

The default constructor creates an empty object identifier value.

3.23.2.2 virtual EXTRTMETHOD ASN1ObjId::~ASN1ObjId () [virtual]

The Virtual Destructor

3.23.2.3 EXTRTMETHOD ASN1ObjId::ASN1ObjId (OSOCKET _numids, const OSUINT32 *_subids)

This constructor initializes the object identifier to contain the given data values.

Parameters

_numids - Number of subidentifiers in the OID.

_subids - Array of subidentifier values.

3.23.2.4 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

oid - C object identifier value.

3.23.2.5 EXTRTMETHOD ASN1ObjId::ASN1ObjId (const ASN1ObjId & oid)

The copy constructor.

Parameters

oid - C++ object identifier value.

3.23.2.6 EXTRTMETHOD ASN1ObjId::ASN1ObjId (const char * dotted_oid_string)

Construct an OID from a dotted string.

Parameters

dotted_oid_string - for example "1.3.1.6.1.10"

3.23.3 Member Function Documentation

3.23.3.1 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

n - Number of subid values to compare.

o - 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.

3.23.3.2 EXTRTMETHOD ASN1ObjId& ASN1ObjId::operator+= (const ASN1ObjId & o)

Overloaded += operator. This operator allows one object identifier to be appended to another object identifier.

Parameters

o - C++ object identifier value.

Returns

- True if values are equal.

3.23.3.3 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 - Subidentifier to append.

Returns

- True if values are equal.

3.23.3.4 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

dotted_oid_string - C++ object identifier value.

Returns

- True if values are equal.

3.23.3.5 EXTRTMETHOD void ASN1ObjId::operator= (const ASN1ObjId & *rhs*)

This assignment operator sets the object identifier to contain the OID in the given C++ structure.

Parameters

rhs - C++ object identifier value.

3.23.3.6 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

rhs - C object identifier value.

3.23.3.7 EXTRTMETHOD ASN1ObjId& ASN1ObjId::operator= (const char * *dotted_oid_string*)

Assignment from a string.

Parameters

dotted_oid_string - New value (for example "1.3.6.1.6.0");

3.23.3.8 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

- n* - Number of subid values to compare.
- o* - 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.

3.23.3.9 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

- raw_oid* - Pointer to an array of subidentifier values.
- oid_len* - Number of subids in the array,

3.23.3.10 EXTRTMETHOD const char* ASN1ObjId::toString (OSCTXT * *pctxt*) const

Get a printable ASCII string of a part of the value.

Parameters

- pctxt* - Pointer to a context structure.

Returns

- Dotted OID string (for example "3.6.1.6")

3.23.3.11 EXTRTMETHOD void ASN1ObjId::trim (const OSUINT32 *n*)

Trim the given number of rightmost sub elements from this OID.

Parameters

- n* - number of subids to trim from OID

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

- [ASN1ObjId.h](#)

3.24 ASN1ObjId64 Struct Reference

```
#include <asn1CppTypes.h>
```

Public Member Functions

- EXTRTMETHOD **ASN1ObjId64** (OSOCKETET _numids, const OSINT64 *_subids)
- EXTRTMETHOD **ASN1ObjId64** (const ASN1OID64 &oid)
- EXTRTMETHOD **ASN1ObjId64** (const [ASN1ObjId64](#) &oid)
- EXTRTMETHOD void **operator=** (const ASN1OID64 &rhs)
- EXTRTMETHOD void **operator=** (const [ASN1ObjId64](#) &rhs)

3.24.1 Detailed Description

Object identifier with 64-bit arcs. This class is identical to the [ASN1ObjId](#) class except it allows 64-bit integers to be used for the arc (i.e. subidentifier) values.

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

- [asn1CppTypes.h](#)

3.25 ASN1OpenType Struct Reference

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

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 () [inline]

The default constructor creates an empty open type value.

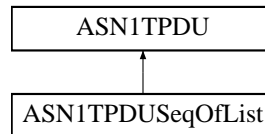
The documentation for this struct was generated from the following file:

- [asn1CppType.h](#)

3.26 ASN1TPDU Struct Reference

```
#include <asn1CppTypes.h>
```

Inheritance diagram for ASN1TPDU:



Public Member Functions

- void [setContext](#) ([OSRTContext](#) *ctxt)

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 Member Function Documentation

3.26.2.1 void ASN1TPDU::setContext (OSRTContext * 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

ctxt A pointer to reference counted ASN.1 context class instance.

3.26.3 Member Data Documentation

3.26.3.1 OSRTCtxtPtr 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.

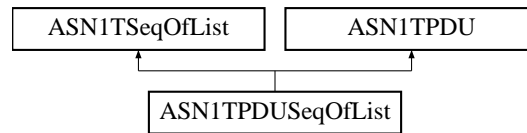
The documentation for this struct was generated from the following file:

- [asn1CppTypes.h](#)

3.27 ASN1TPDUSeqOfList Struct Reference

```
#include <asn1CppTypes.h>
```

Inheritance diagram for ASN1TPDUSeqOfList:



Public Member Functions

- [ASN1TPDUSeqOfList \(\)](#)

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 () [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>
```

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 () [inline]

The default constructor creates an empty open extension element.

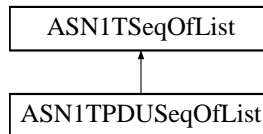
The documentation for this struct was generated from the following file:

- [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 () [inline]

The default constructor creates an empty list.

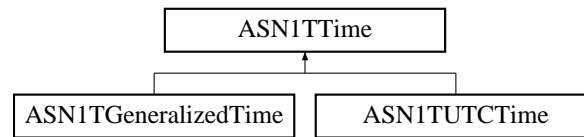
The documentation for this struct was generated from the following file:

- [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](#) (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

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 EXTRTMETHOD ASN1TTime::ASN1TTime (OSBOOL *useDerRules*)

This constructor creates an empty time class.

Parameters

useDerRules Use the Distinguished Encoding Rules (DER) to operate on this time value.

3.30.2.2 EXTRTMETHOD ASN1TTime::ASN1TTime (const ASN1TTime & *original*)

The copy constructor.

Parameters

original The original time string object value.

3.30.2.3 virtual EXTRTMETHOD ASN1TTime::~~ASN1TTime () [virtual]

The destructor.

3.30.3 Member Function Documentation

3.30.3.1 virtual EXTRTMETHOD 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](#).

3.30.3.2 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

pbuf A pointer to destination buffer.

bufsize A size of destination buffer.

Returns

Completion status of operation:

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

Implemented in [ASNITGeneralizedTime](#), and [ASNITUTCTime](#).

3.30.3.3 virtual EXTRTMETHOD int ASNITTime::equals (const ASNITTime &) const [virtual]

This method compares times.

3.30.3.4 virtual EXTRTMETHOD int ASNITTime::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 be differ for different inherited [ASNITTime](#) 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 virtual EXTRTMETHOD int ASNITTime::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 of negative time difference. Note that the return value may differ for different inherited [ASNITTime](#) 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 virtual EXTRTMETHOD int ASNITTime::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 of negative time difference. Note that the return value may differ for different inherited [ASNITTime](#) 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 virtual EXTRTMETHOD int ASNITTime::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 [ASNITTime](#) 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 virtual EXTRTMETHOD int ASNITTime::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 [ASNITUTCTime](#).

3.30.3.9 virtual EXTRTMETHOD double ASNITTime::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 virtual EXTRTMETHOD int ASNITTime::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 virtual EXTRTMETHOD int ASNITTime::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 virtual EXTRTMETHOD int ASN1Time::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 [ASN1Time](#) 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 virtual EXTRTMETHOD int ASN1Time::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 [ASN1Time](#) 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 virtual EXTRTMETHOD int ASN1Time::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 values for decoded months: [ASN1Time::January](#), [ASN1Time::February](#), etc. Also short aliases for months can be used: [ASN1Time::Jan](#), [ASN1Time::Feb](#), etc. Note that the return value may differ for different inherited [ASN1Time](#) 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 virtual EXTRTMETHOD 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 virtual EXTRTMETHOD 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 virtual EXTRTMETHOD 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 virtual EXTRTMETHOD 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 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

string The time string value to be parsed.

Returns

Completion status of operation:

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

Implemented in [ASN1GeneralizedTime](#), and [ASN1TUTCTime](#).

3.30.3.20 virtual EXTRTMETHOD int ASN1TTime::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 [ASN1TTime](#) classes.

Parameters

day_ Day of month component (1 - 31).

Returns

Completion status of operation:

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

3.30.3.21 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.

3.30.3.22 virtual EXTRTMETHOD 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

inMinutes 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.

3.30.3.23 virtual EXTRTMETHOD 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

dhour The negative or positive hour component of the difference between the time zone of the object and the UTC time (-12 - +12).

dminute 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.

3.30.3.24 virtual EXTRTMETHOD 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

dhour 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 virtual EXTRTMETHOD int ASN1TTime::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

frac Second's decimal fraction component.

Returns

Completion status of operation:

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

3.30.3.26 virtual EXTRTMETHOD 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

frac Second's decimal fraction component.

fracLen 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 virtual EXTRTMETHOD 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

fraction Second's decimal fraction component (0 - 9).

fracLen 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](#).

3.30.3.28 virtual EXTRTMETHOD int ASN1TTime::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 [ASN1TTime](#) classes.

Parameters

hour_ Hour component (0 - 23).

Returns

Completion status of operation:

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

3.30.3.29 virtual EXTRTMETHOD 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

minute_ Minute component (0 - 59).

Returns

Completion status of operation:

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

3.30.3.30 virtual EXTRTMETHOD 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

month_ Month component (1 - 12).

Returns

Completion status of operation:

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

3.30.3.31 virtual EXTRTMETHOD 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

second_ Second component (0 - 59).

Returns

Completion status of operation:

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

3.30.3.32 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

time The time value, expressed as a number of seconds from January 1, 1970.

diffTime 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 [ASNITGeneralizedTime](#), and [ASNITUTCTime](#).

3.30.3.33 virtual EXTRTMETHOD int ASNITTime::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

utc UTC flag state.

Returns

Completion status of operation:

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

Reimplemented in [ASNITUTCTime](#).

3.30.3.34 virtual EXTRTMETHOD int ASNITTime::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 [ASNITTime](#) classes.

Parameters

year_ Year component (full 4 digits).

Returns

Completion status of operation:

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

Reimplemented in [ASNITUTCTime](#).

3.30.3.35 EXTRTMETHOD const char* ASN1TTime::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.

3.30.3.36 EXTRTMETHOD const char* ASN1TTime::toString (OSCTXT * *pctxt*) const

Get a printable ASCII string of the time value.

Parameters

pctxt Pointer to a context structure.

Returns

Compiled time string. NULL, if error occurs.

3.30.3.37 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

pbuf Pointer to a destination buffer.

bufsize Size of destination buffer.

Returns

Compiled time string. NULL, if error occurs.

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

- [ASN1TTime.h](#)

3.31 ASN1TUniversalString Struct Reference

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

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 () [inline]

The default constructor creates an empty UniversalString value.

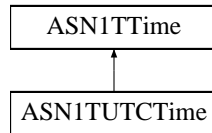
The documentation for this struct was generated from the following file:

- [asn1CppType.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)

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 EXTRTMETHOD ASN1TUTCTime::ASN1TUTCTime ()

A default constructor.

3.32.2.2 EXTRTMETHOD ASN1TUTCTime::ASN1TUTCTime (const char * *timeStr*, OSBOOL *useDerRules* = FALSE)

This constructor creates a time object using the specified time string.

Parameters

timeStr A pointer to the time string to be parsed.

useDerRules Create object using DER rules.

3.32.2.3 EXTRTMETHOD ASN1TUTCTime::ASN1TUTCTime (OSBOOL *useDerRules*)

This constructor creates an empty time object.

Parameters

useDerRules An OSBOOL value.

3.32.2.4 ASN1TUTCTime::ASN1TUTCTime (const ASN1TUTCTime & *original*) [inline]

A copy constructor.

3.32.3 Member Function Documentation

3.32.3.1 EXTRTMETHOD void ASN1TUTCTime::clear () [virtual]

Clears out the time object.

Reimplemented from [ASN1TTime](#).

3.32.3.2 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

pbuf A pointer to destination buffer.

bufsize A size of destination buffer.

Returns

Completion status of operation:

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

Implements [ASN1TTime](#).

3.32.3.3 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 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

string 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 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

fraction Second's decimal fraction component (0 - 9).

fracLen 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 EXTRTMETHOD int ASN1TUTCTime::setTime (time_t *time*, OSBOOL *diffTime*) [virtual]

Converts *time_t* to time string.

Parameters

time time to convert,

diffTime 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 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

utc UTC flag state.

Returns

Completion status of operation:

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

Reimplemented from [ASN1TTime](#).

3.32.3.8 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

year_ 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](#).

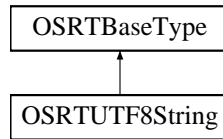
The documentation for this class was generated from the following file:

- [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.

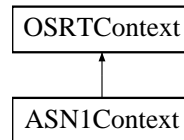
The documentation for this class was generated from the following file:

- [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 [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 [mbInitialized](#)
- 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 EXTRTMETHOD OSRTContext::OSRTContext ()

The default constructor initializes the mCtxt member variable and sets the reference count variable (mCount) to zero.

3.34.2.2 virtual EXTRTMETHOD OSRTContext::~~OSRTContext () [virtual]

The destructor frees all memory held by the context.

3.34.3 Member Function Documentation

3.34.3.1 EXTRTMETHOD void OSRTContext::_ref ()

The _ref method increases the reference count by one.

Referenced by OSRTCtxtPtr::operator=().

3.34.3.2 EXTRTMETHOD void OSRTContext::_unref ()

The _unref method decreases the reference count by one.

3.34.3.3 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

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

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

Returns

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

3.34.3.4 EXTRTMETHOD char* OSRTContext::getErrorInfo (size_t * pBufSize)

Returns error text in a dynamic memory buffer. Buffer will be allocated using 'operator new []'. The calling routine is responsible for freeing the memory by using 'operator delete []'.

Parameters

pBufSize A pointer to buffer size. It will receive the size of allocated dynamic buffer, or (size_t)-1 if an error occurred.

Returns

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

3.34.3.5 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.

3.34.3.6 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.

3.34.3.7 EXTRTMETHOD OSUINT32 OSRTContext::getRefCount ()

The getRefCount method returns the current reference count.

3.34.3.8 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.

3.34.3.9 OSBOOL OSRTContext::isInitialized () [inline]

Returns TRUE, if initialized correctly, FALSE otherwise.

Returns

TRUE, if initialized correctly, FALSE otherwise.

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

The memAlloc method allocates memory using the C runtime memory management functions. The memory is tracked in the underlying context structure. When both this OSXSDGlobalElement derived control class object and the message buffer object are destroyed, this memory will be freed.

Parameters

numocts - Number of bytes of memory to allocate

3.34.3.11 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.

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

The memFreePtr method frees the memory at a specific location. This memory must have been allocated using the memAlloc method described earlier.

Parameters

ptr - Pointer to a block of memory allocated with memAlloc

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

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

Parameters

ptr - Original pointer containing dynamic memory to be resized.

numocts - Number of bytes of memory to allocate

Returns

Reallocated memory pointer

3.34.3.14 void OSRTContext::memReset () [inline]

The memReset method resets dynamic memory using the C runtime memory management functions.

3.34.3.15 void OSRTContext::printErrorInfo () [inline]

The printErrorInfo method prints information on errors contained within the context.

3.34.3.16 void OSRTContext::resetErrorInfo () [inline]

The resetErrorInfo method resets information on errors contained within the context.

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

The setDiag method will turn diagnostic tracing on or off.

Parameters

value - Boolean value (default = TRUE = on)

Returns

- Previous state of the diagnostics enabled boolean

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

This method sets run-time key to the context. This method does nothing for unlimited redistribution libraries.

Parameters

key - array of octets with the key

keylen - number of octets in key array.

Returns

Completion status of operation:

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

Reimplemented in [ASN1Context](#).

3.34.3.19 **int OSRTContext::setStatus (int *stat*) [inline]**

This method sets error status in the context.

Parameters

stat Status value.

Returns

Error status value being set.

3.34.4 Member Data Documentation

3.34.4.1 **OSBOOL OSRTContext::mbInitialized [protected]**

TRUE, if initialized correctly, FALSE otherwise

3.34.4.2 **OSUINT32 OSRTContext::mCount [protected]**

The mCount member variable holds the reference count of this context.

3.34.4.3 **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 **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 <OSRTCtxtPtr.h>
```

Public Member Functions

- [OSRTCtxtPtr](#) ([OSRTCtxtPtr](#) *rf=0)
- [OSRTCtxtPtr](#) (const [OSRTCtxtPtr](#) &o)
- virtual [~OSRTCtxtPtr](#) ()
- [OSRTCtxtPtr](#) & [operator=](#) (const [OSRTCtxtPtr](#) &rf)
- [OSRTCtxtPtr](#) & [operator=](#) ([OSRTCtxtPtr](#) *rf)
- [operator OSRTCtxtPtr](#) * ()
- [operator const OSRTCtxtPtr](#) * () const
- [OSRTCtxtPtr](#) * [operator->](#) ()
- const [OSRTCtxtPtr](#) * [operator->](#) () const
- OSBOOL [operator==](#) (const [OSRTCtxtPtr](#) *o) const
- OSBOOL [isNull](#) () const
- OSCTXT * [getCtxtPtr](#) ()

Protected Attributes

- [OSRTCtxtPtr](#) * 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 [OSRTCtxtPtr](#) object.

3.35.2 Constructor & Destructor Documentation

3.35.2.1 [OSRTCtxtPtr::OSRTCtxtPtr](#) ([OSRTCtxtPtr](#) *rf = 0) [[inline](#)]

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

Parameters

rf - Pointer to [OSRTCtxtPtr](#) object

3.35.2.2 [OSRTCtxtPtr::OSRTCtxtPtr](#) (const [OSRTCtxtPtr](#) &o) [[inline](#)]

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

Parameters

o - Reference to [OSRTCtxtPtr](#) object to be copied

3.35.2.3 virtual OSRTCtxtPtr::~~OSRTCtxtPtr () [inline, virtual]

The destructor decrements the reference counter to the internal context pointer object. The context object will delete itself if its reference count goes to zero.

3.35.3 Member Function Documentation

3.35.3.1 OSCTXT* OSRTCtxtPtr::getCtxtPtr () [inline]

This method returns the standard context pointer used in C function calls.

3.35.3.2 OSBOOL OSRTCtxtPtr::isNull () const [inline]

The isNull method returns TRUE if the underlying context pointer is NULL.

3.35.3.3 OSRTCtxtPtr::operator OSRTContext * () [inline]

The 'OSRTContext*' operator returns the context object pointer.

3.35.3.4 OSRTContext* OSRTCtxtPtr::operator-> () [inline]

The '->' operator returns the context object pointer.

3.35.3.5 OSRTCtxtPtr& OSRTCtxtPtr::operator= (OSRTContext * rf) [inline]

This assignment operator assigns does a direct assignment of an [OSRTContext](#) object to this [OSRTCtxtPtr](#) object. References [OSRTContext::_ref\(\)](#).

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

This assignment operator assigns this [OSRTCtxtPtr](#) to another. The reference count of the context object managed by this object is first decremented. Then the new pointer is assigned and that object's reference count is incremented.

Parameters

rf - Pointer to [OSRTCtxtPtr](#) smart-pointer object

References [OSRTContext::_ref\(\)](#), and [mPointer](#).

3.35.3.7 OSBOOL OSRTCtxtPtr::operator== (const OSRTContext * o) const [inline]

The '==' operator compares two [OSRTContext](#) pointer values.

3.35.4 Member Data Documentation

3.35.4.1 OSRTContext* OSRTCtxtPtr::mPointer [protected]

The [mPointer](#) member variable is a pointer to a reference-counted ASN.1 context wrapper class object.

Referenced by operator=().

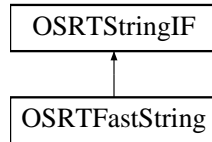
The documentation for this class was generated from the following file:

- [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`

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::OSRTFastString ()

The default constructor sets the internal string member variable pointer to null.

3.36.2.2 OSRTFastString::OSRTFastString (const char * strval)

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

Parameters

strval - Null-terminated C string value

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

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

Parameters

strval - Null-terminated C string value

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

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

Parameters

str - C++ string object to be copied.

3.36.2.5 virtual OSRTFastString::~OSRTFastString () [virtual]

The destructor does nothing.

3.36.3 Member Function Documentation

3.36.3.1 virtual OSRTStringIF* OSRTFastString::clone () [inline, virtual]

This method creates a copy of the given string object.

Implements [OSRTStringIF](#).

3.36.3.2 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 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 OSRTFastString& OSRTFastString::operator= (const OSRTFastString & *original*)

Assignment operator.

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

This method prints the string value to standard output.

Parameters

name - Name of generated string variable.

Implements [OSRTStringIF](#).

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

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

Parameters

str - C null-terminated UTF-8 string.

Implements [OSRTStringIF](#).

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

This method sets the string value to the given string.

Parameters

str - C null-terminated string.

Implements [OSRTStringIF](#).

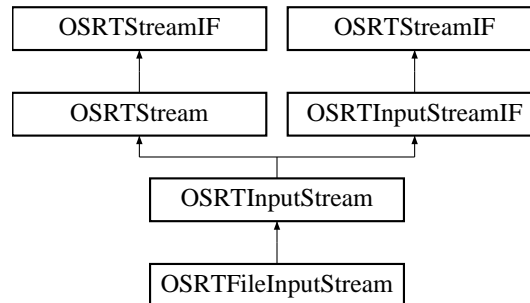
The documentation for this class was generated from the following file:

- [OSRTFastString.h](#)

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

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 EXTRTMETHOD OSRTFileInputStream::OSRTFileInputStream (const char * pFilename)

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

Parameters

pFilename Name of file.

See also

rtxStreamFileOpen

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

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

Parameters

pContext Pointer to a context to use.

pFilename Name of file.

See also

rtxStreamFileOpen

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

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

Parameters

file Pointer to FILE structure.

See also

rtxStreamFileAttach

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

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

Parameters

pContext Pointer to a context to use.

file Pointer to FILE structure.

See also

rtxStreamFileAttach

3.37.3 Member Function Documentation

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

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

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented from [OSRTInputStream](#).

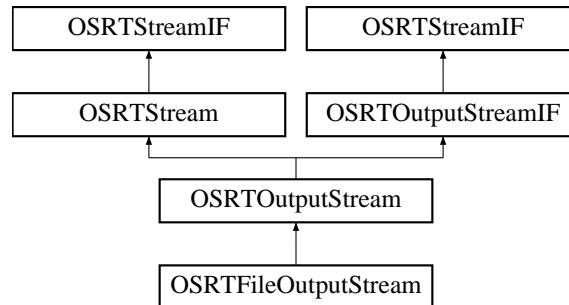
The documentation for this class was generated from the following file:

- [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

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 EXTRTMETHOD OSRTFileOutputStream::OSRTFileOutputStream (const char * pFilename)

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

Parameters

pFilename Name of file.

Exceptions

OSStreamException Stream create or initialize failed.

See also

rtxStreamFileOpen

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

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

Parameters

pContext Pointer to a context to use.

pFilename Name of file.

Exceptions

OSStreamException Stream create or initialize failed.

See also

rtxStreamFileOpen

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

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

Parameters

file Pointer to FILE structure.

Exceptions

OSStreamException Stream create or initialize failed.

See also

rtxStreamFileAttach

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

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

Parameters

pContext Pointer to a context to use.

file Pointer to FILE structure.

Exceptions

OSStreamException Stream create or initialize failed.

See also

rtxStreamFileAttach

3.38.3 Member Function Documentation

3.38.3.1 virtual OSBOOL OSRTFileOutputStream::isA (StreamID *id*) const [inline, virtual]

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

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented from [OSRTOutputStream](#).

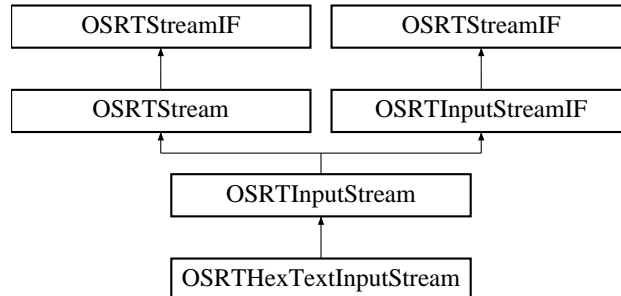
The documentation for this class was generated from the following file:

- [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](#)

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 EXTRTMETHOD OSRTHexTextInputStream::OSRTHexTextInputStream (OSRTInputStream * *pstream*)

Initializes the input stream using the existing standard input stream. Only file and memory underlying stream types are supported.

Parameters

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

See also

[rtxStreamHexTextAttach](#)

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

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

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented from [OSRTInputStream](#).

3.39.3.2 void OSRTHexTextInputStream::setOwnUnderStream (OSBOOL *value* = TRUE) [inline]

This method transfers ownership of the underlying stream to the class.

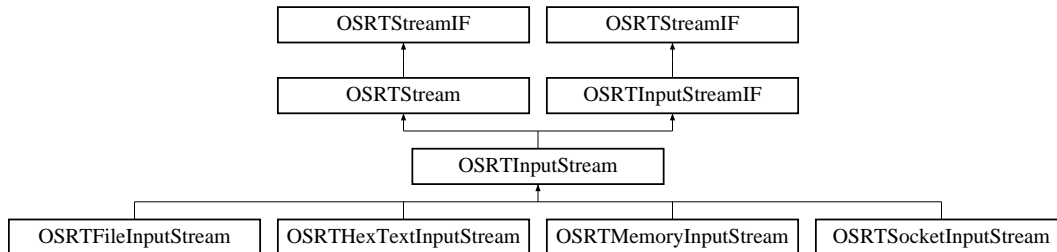
The documentation for this class was generated from the following file:

- [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](#) (OSOCKETET *pDestBuf, size_t maxToRead)
- virtual EXTRTMETHOD long [readBlocking](#) (OSOCKETET *pDestBuf, size_t toReadBytes)
- virtual EXTRTMETHOD int [reset](#) ()
- virtual int [setPosition](#) (size_t pos)
- virtual EXTRTMETHOD int [skip](#) (size_t n)

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 EXTRTMETHOD OSRTInputStream::OSRTInputStream ()

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

Exceptions

OSRTStreamException Stream create or initialize failed.

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

Virtual destructor. Closes the stream if it was opened.

3.40.3 Member Function Documentation

3.40.3.1 virtual EXTRTMETHOD int OSRTInputStream::close () [virtual]

Closes the input or output stream and releases any system resources associated with the stream. For output streams this function also flushes all internal buffers to the stream.

Returns

Completion status of operation:

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

See also

`rtxStreamClose`, `rtxStreamBufClose`

Reimplemented from [OSRTStream](#).

3.40.3.2 virtual EXTRTMETHOD size_t OSRTInputStream::currentPos () [virtual]

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

Returns

The number of octets already read from the stream.

Implements [OSRTInputStreamIF](#).

3.40.3.3 virtual EXTRTMETHOD int OSRTInputStream::flush () [virtual]

Flushes the buffered data to the stream.

Returns

Completion status of operation:

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

See also

rtxStreamFlush, rtxStreamBufFlush

Reimplemented from [OSRTStream](#).

3.40.3.4 virtual OSRTCtxPtr OSRTInputStream::getContext () [inline, virtual]

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

Returns

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

Reimplemented from [OSRTStream](#).

References OSRTStream::getContext().

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

This method returns a pointer to the underlying OSCTXT object. This is the structure used in calls to low-level C encode/decode functions.

Returns

Pointer to a context (OSCTXT) structure.

Reimplemented from [OSRTStream](#).

References OSRTStream::getCtxtPtr().

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

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

Parameters

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

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

Returns

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

Reimplemented from [OSRTStream](#).

References OSRTStream::getErrorInfo().

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

Returns error text in a dynamic memory buffer. Buffer will be allocated by 'operator new []'. The calling routine is responsible to free the memory by using 'operator delete []'.

Returns

A pointer to a newly allocated buffer with error text.

Reimplemented from [OSRTStream](#).

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

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

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

Parameters

ppos Pointer to a variable to receive position.

Returns

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

Implements [OSRTInputStreamIF](#).

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

This method returns the completion status of previous operation. It can be used to check completion status of constructors or methods, which do not return completion status.

Returns

Runtime status code:

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

Reimplemented from [OSRTStream](#).

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

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

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

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Implements [OSRTInputStreamIF](#).

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

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

Checks, is the stream opened or not.

Returns

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

See also

[rtxStreamIsOpened](#)

Reimplemented from [OSRTStream](#).

3.40.3.12 virtual EXTRTMETHOD int OSRTInputStream::mark (size_t readAheadLimit) [virtual]

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

Parameters

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

Returns

Completion status of operation:

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

See also

[rtxStreamMark](#), [rtxStreamReset](#)

Implements [OSRTInputStreamIF](#).

3.40.3.13 virtual EXTRTMETHOD OSBOOL OSRTInputStream::markSupported () [virtual]

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

Returns

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

See also

[rtxStreamIsMarkSupported](#)

Implements [OSRTInputStreamIF](#).

3.40.3.14 void OSRTInputStream::printErrorInfo () [inline]

The printErrorInfo method prints information on errors contained within the context.

Reimplemented from [OSRTStream](#).

References OSRTStream::printErrorInfo().

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

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

Parameters

pDestBuf Pointer to a buffer to receive a data.

maxToRead Size of the buffer.

See also

rtxStreamRead

Implements [OSRTInputStreamIF](#).

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

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

Parameters

pDestBuf Pointer to a buffer to receive a data.

toReadBytes Number of bytes to be read.

See also

rtxStreamRead

Implements [OSRTInputStreamIF](#).

3.40.3.17 virtual EXTRTMETHOD int OSRTInputStream::reset () [virtual]

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

Returns

Completion status of operation:

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

See also

rtxStreamMark, rtxStreamReset

Implements [OSRTInputStreamIF](#).

3.40.3.18 void OSRTInputStream::resetErrorInfo () [inline]

The resetErrorInfo method resets information on errors contained within the context.

Reimplemented from [OSRTStream](#).

References OSRTStream::resetErrorInfo().

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

Sets the current stream position to the given offset.

Parameters

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

Returns

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

Implements [OSRTInputStreamIF](#).

3.40.3.20 virtual EXTRMETHOD int OSRTInputStream::skip (size_t n) [virtual]

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

Parameters

n The number of octets to be skipped.

Returns

Completion status of operation:

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

See also

`rtxStreamSkip`

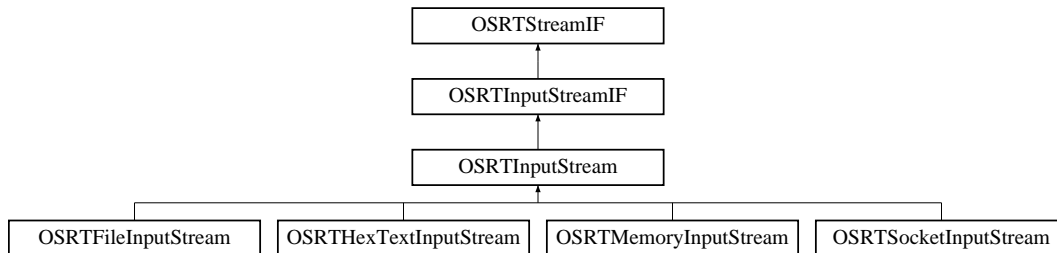
Implements [OSRTInputStreamIF](#).

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` (OSOCKET *pDestBuf, `size_t` maxToRead)=0
- virtual `long` `readBlocking` (OSOCKET *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 virtual EXTRTMETHOD OSRTInputStreamIF::~~OSRTInputStreamIF () [virtual]

Virtual destructor. Closes the stream if it was opened.

3.41.2 Member Function Documentation

3.41.2.1 virtual `size_t` OSRTInputStreamIF::currentPos () [pure virtual]

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

Returns

The number of octets already read from the stream.

Implemented in [OSRTInputStream](#).

3.41.2.2 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

ppos Pointer to a variable to receive position.

Returns

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

Implemented in [OSRTInputStream](#).

3.41.2.3 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

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Implemented in [OSRTInputStream](#), [OSRTFileInputStream](#), [OSRTHexTextInputStream](#), [OSRTMemoryInputStream](#), and [OSRTSocketInputStream](#).

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

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

Parameters

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

Returns

Completion status of operation:

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

See also

`rtxStreamMark`, `rtxStreamReset`

Implemented in [OSRTInputStream](#).

3.41.2.5 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](#).

3.41.2.6 virtual long OSRTInputStreamIF::read(OSOCTET *pDestBuf, size_t maxToRead) [pure virtual]

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

Parameters

pDestBuf Pointer to a buffer to receive a data.

maxToRead Size of the buffer.

Returns

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

See also

rtxStreamRead

Implemented in [OSRTInputStream](#).

3.41.2.7 virtual long OSRTInputStreamIF::readBlocking(OSOCTET *pDestBuf, size_t toReadBytes) [pure virtual]

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

Parameters

pDestBuf Pointer to a buffer to receive a data.

toReadBytes Number of bytes to be read.

Returns

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

See also

rtxStreamRead

Implemented in [OSRTInputStream](#).

3.41.2.8 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](#).

3.41.2.9 virtual int OSRTInputStreamIF::setPosition(size_t pos) [pure virtual]

Sets the current stream position to the given offset.

Parameters

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

Returns

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

Implemented in [OSRTInputStream](#).

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

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

Parameters

n The number of octets to be skipped.

Returns

Completion status of operation:

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

See also

`rtxStreamSkip`

Implemented in [OSRTInputStream](#).

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

- [OSRTInputStreamIF.h](#)

3.42 OSRTInputStreamPtr Class Reference

Public Member Functions

- [OSRTInputStreamPtr](#) ([OSRTInputStreamIF](#) *ptr)
- **operator OSRTInputStreamIF** * ()
- [OSRTInputStreamIF](#) * **operator->** ()

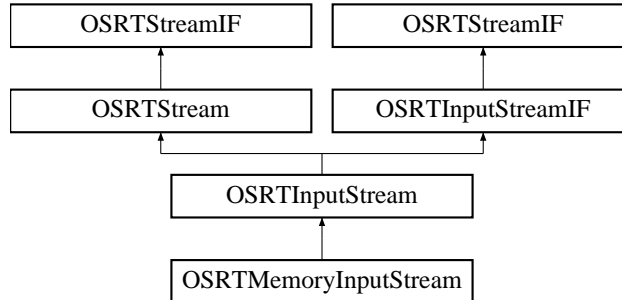
The documentation for this class was generated from the following file:

- [OSRTInputStreamIF.h](#)

3.43 OSRTMemoryInputStream Class Reference

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

Inheritance diagram for OSRTMemoryInputStream:



Public Member Functions

- EXTRTMETHOD `OSRTMemoryInputStream` (const OSOCTET *pMemBuf, size_t bufSize)
- EXTRTMETHOD `OSRTMemoryInputStream` (OSRTContext *pContext, const OSOCTET *pMemBuf, size_t bufSize)
- virtual OSBOOL `isA` (StreamID id) const

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

Initializes the memory input stream using the specified memory buffer.

Parameters

pMemBuf The pointer to the buffer.

bufSize The size of the buffer.

See also

`rtxStreamMemoryAttach`

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

Initializes the memory input stream using the specified memory buffer.

Parameters

pContext Pointer to a context to use.

pMemBuf The pointer to the buffer.

bufSize The size of the buffer.

See also

`rtxStreamMemoryAttach`

3.43.3 Member Function Documentation

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

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

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented from [OSRTInputStream](#).

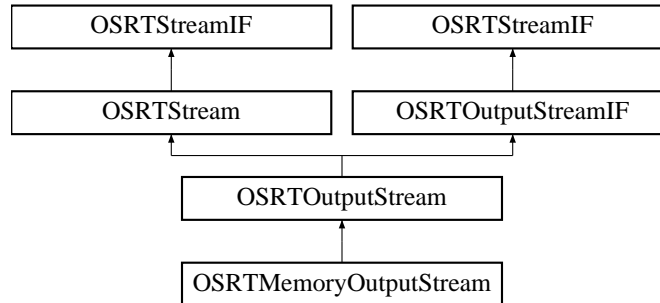
The documentation for this class was generated from the following file:

- [OSRTMemoryInputStream.h](#)

3.44 OSRTMemoryOutputStream Class Reference

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

Inheritance diagram for OSRTMemoryOutputStream:



Public Member Functions

- EXTRTMETHOD [OSRTMemoryOutputStream](#) ()
- EXTRTMETHOD [OSRTMemoryOutputStream](#) (OSOCKET *pMemBuf, size_t bufSize)
- EXTRTMETHOD [OSRTMemoryOutputStream](#) (OSRTContext *pContext, OSOCKET *pMemBuf, size_t bufSize)
- EXTRTMETHOD OSOCKET * [getBuffer](#) (size_t *pSize=0)
- virtual OSBOOL [isA](#) (StreamID id) const
- int [reset](#) ()

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

Initializes the memory output stream using the specified memory buffer. The status of the construction can be obtained by calling the `getStatus` method.

Parameters

pMemBuf The pointer to the buffer.

bufSize The size of the buffer.

See also

rtxStreamMemoryAttach

3.44.2.3 EXTRTMETHOD OSRTMemoryOutputStream::OSRTMemoryOutputStream (OSRTContext * *pContext*, 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

pContext Pointer to a context to use.

pMemBuf The pointer to the buffer.

bufSize The size of the buffer.

See also

rtxStreamMemoryAttach

3.44.3 Member Function Documentation

3.44.3.1 EXTRTMETHOD OSOCTET* OSRTMemoryOutputStream::getBuffer (size_t * *pSize* = 0)

This method returns the address of the memory buffer to which data was written. If the buffer memory is dynamic, it may be freed using the `rtxMemFreePtr` function or it will be freed when the stream object is destroyed.

Parameters

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

Returns

Pointer to memory buffer.

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

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

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented from [OSRTOutputStream](#).

3.44.3.3 int OSRTMemoryOutputStream::reset ()

This method resets the output memory stream internal buffer to allow it to be overwritten with new data. Memory for the buffer is not freed.

Returns

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

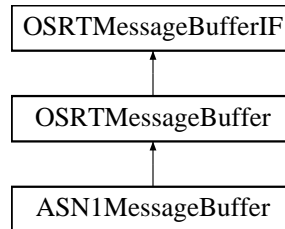
The documentation for this class was generated from the following file:

- [OSRTMemoryOutputStream.h](#)

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 `OSOCTET * getMsgCopy ()`
- virtual `const OSOCTET * getMsgPtr ()`
- `int getStatus () const`
- virtual `int init ()`
- virtual `EXTRTMETHOD int initBuffer (OSOCTET *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`

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

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

Parameters

bufferType Type of message buffer that is being created (for example, XMLEncode).

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

3.45.2.2 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 virtual void* OSRTMessageBuffer::getAppInfo () [inline, virtual]

Returns a pointer to application-specific information block

Implements [OSRTMessageBufferIF](#).

Reimplemented in [ASN1MessageBuffer](#).

3.45.3.2 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 virtual OSRTCtxPtr OSRTMessageBuffer::getContext () [inline, virtual]

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

Implements [OSRTMessageBufferIF](#).

Referenced by [ASN1MessageBuffer::setStatus\(\)](#).

3.45.3.4 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::removeEventHandler\(\)](#), and [ASN1MessageBuffer::setErrorHandler\(\)](#).

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

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

Parameters

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

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

Returns

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

3.45.3.6 `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.

3.45.3.7 `virtual OSOCTET* OSRTMessageBuffer::getMsgCopy () [inline, virtual]`

The getMsgCopy method will return a copy of the encoded message managed by the object.

Implements [OSRTMessageBufferIF](#).

3.45.3.8 `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](#).

3.45.3.9 `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 **virtual int OSRTMessageBuffer::init () [inline, virtual]**

Initializes message buffer.

Returns

Completion status of operation:

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

Implements [OSRTMessageBufferIF](#).

3.45.3.11 **virtual EXTRMETHOD int OSRTMessageBuffer::initBuffer (OSOCTET * *pMsgBuf*, size_t *msgBufLen*) [virtual]**

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

Parameters

pMsgBuf Pointer to message buffer.

msgBufLen Length of message buffer in bytes.

Returns

Completion status of operation:

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

Implements [OSRTMessageBufferIF](#).

3.45.3.12 **virtual void OSRTMessageBuffer::printErrorInfo () [inline, virtual]**

The printErrorInfo method prints information on errors contained within the context.

3.45.3.13 **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 **virtual void OSRTMessageBuffer::setAppInfo (void *) [inline, virtual]**

Sets the application-specific information block.

Implements [OSRTMessageBufferIF](#).

Reimplemented in [ASN1MessageBuffer](#).

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

The setDiag method will turn diagnostic tracing on or off.

Parameters

value - Boolean value (default = TRUE = on)

Implements [OSRTMessageBufferIF](#).

3.45.4 **Member Data Documentation**

3.45.4.1 **Type OSRTMessageBuffer::mBufferType** [**protected**]

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

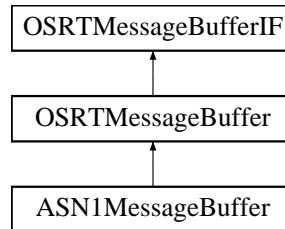
The documentation for this class was generated from the following file:

- [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**,
 Stream }

Public Member Functions

- virtual void * [getAppInfo](#) ()=0
- virtual size_t [getByteIndex](#) ()=0
- virtual [OSRCTxtPtr](#) [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](#) (int 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 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 virtual void* OSRTMessageBufferIF::getAppInfo () [pure virtual]

Returns a pointer to application-specific information block

Implemented in [ASN1MessageBuffer](#), and [OSRTMessageBuffer](#).

3.46.3.2 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 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 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 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 virtual int OSRTMessageBufferIF::initBuffer (OSOCKET * *pMsgBuf*, size_t *msgBufLen*) [pure virtual]

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

Parameters

pMsgBuf Pointer to message buffer.

msgBufLen Length of message buffer in bytes. string.

Returns

Completion status of operation:

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

Implemented in [OSRTMessageBuffer](#).

3.46.3.7 virtual OSBOOL OSRTMessageBufferIF::isA (int *bufferType*) [pure virtual]

This method checks the type of the message buffer.

Parameters

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

Returns

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

3.46.3.8 virtual void OSRTMessageBufferIF::setAppInfo (void * *pAppInfo*) [pure virtual]

Sets the application-specific information block.

Implemented in [ASN1MessageBuffer](#), and [OSRTMessageBuffer](#).

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

The setDiag method will turn diagnostic tracing on or off.

Parameters

value - Boolean value (default = TRUE = on)

Implemented in [OSRTMessageBuffer](#).

3.46.3.10 `virtual void OSRTMessageBufferIF::setNamespace (const OSUTF8CHAR *, const OSUTF8CHAR *, OSRTDList * = 0) [inline, virtual]`

Sets the namespace information.

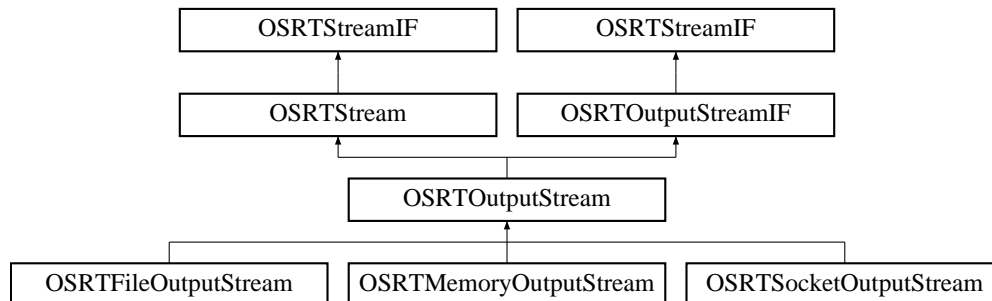
The documentation for this class was generated from the following file:

- [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 \(const OSOCTET *pdata, size_t size\)](#)
- virtual EXTRTMETHOD [long write \(const char *pdata\)](#)

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 EXTRTMETHOD OSRTOutputStream::OSRTOutputStream ()

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

3.47.2.2 **virtual EXTRTMETHOD OSRTOutputStream::~~OSRTOutputStream () [virtual]**

Virtual destructor. Closes the stream if it was opened.

3.47.3 **Member Function Documentation**

3.47.3.1 **virtual EXTRTMETHOD int OSRTOutputStream::close () [virtual]**

Closes the output or output stream and releases any system resources associated with the stream. For output streams this function also flushes all internal buffers to the stream.

Returns

Completion status of operation:

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

See also

`rtxStreamClose`, `rtxStreamBufClose`

Reimplemented from [OSRTStream](#).

3.47.3.2 **virtual EXTRTMETHOD int OSRTOutputStream::flush () [virtual]**

Flushes the buffered data to the stream.

Returns

Completion status of operation:

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

See also

`rtxStreamFlush`, `rtxStreamBufFlush`

Reimplemented from [OSRTStream](#).

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

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

Parameters

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

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

Returns

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

Reimplemented from [OSRTStream](#).

References OSRTStream::getErrorInfo().

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

Returns error text in a dynamic memory buffer. Buffer will be allocated by 'operator new []'. The calling routine is responsible to free the memory by using 'operator delete []'.

Returns

A pointer to a newly allocated buffer with error text.

Reimplemented from [OSRTStream](#).

References OSRTStream::getErrorInfo().

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

This method returns the completion status of previous operation. It can be used to check completion status of constructors or methods, which do not return completion status.

Returns

Runtime status code:

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

Reimplemented from [OSRTStream](#).

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

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

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

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Implements [OSRTOutputStreamIF](#).

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

3.47.3.9 virtual EXTRTMETHOD OSBOOL OSRTOutputStream::isOpened () [virtual]

Checks if the stream open or not.

Returns

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

See also

[rtxStreamIsOpened](#)

Reimplemented from [OSRTStream](#).

3.47.3.10 void OSRTOutputStream::printErrorInfo () [inline]

The `printErrorInfo` method prints information on errors contained within the context.

Reimplemented from [OSRTStream](#).

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

3.47.3.11 void OSRTOutputStream::resetErrorInfo () [inline]

The `resetErrorInfo` method resets information on errors contained within the context.

Reimplemented from [OSRTStream](#).

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

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

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

Parameters

pdata The pointer to the data to be written.

Returns

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

See also

`rtxStreamWrite`

3.47.3.13 **virtual EXTRTMETHOD long OSRTOutputStream::write (const OSOCKET * *pdata*, *size_t size*) [virtual]**

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

Parameters

pdata The pointer to the data to be written.

size The number of octets to write.

Returns

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

See also

`rtxStreamWrite`

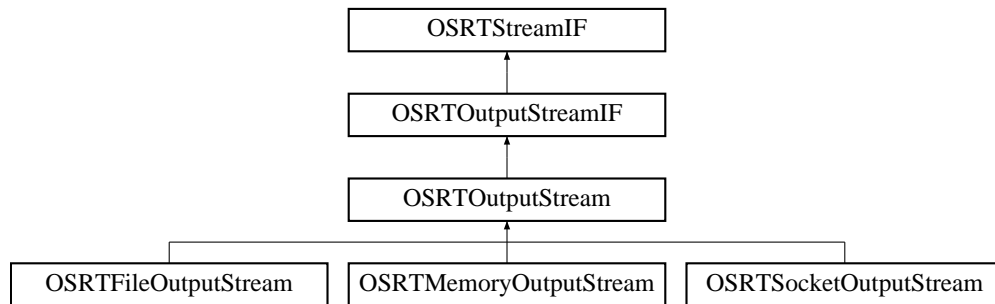
Implements [OSRTOutputStreamIF](#).

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

- [OSRTOutputStream.h](#)

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 OSOCTET *pdata, size_t size)=0

3.48.1 Constructor & Destructor Documentation

3.48.1.1 virtual EXTRTMETHOD OSRTOutputStreamIF::~~OSRTOutputStreamIF () [virtual]

Virtual destructor. Closes the stream if it was opened.

3.48.2 Member Function Documentation

3.48.2.1 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

id Enumerated stream identifier

Returns

True if the stream matches the identifier

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

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

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

Parameters

pdata Pointer to the data to be written.

size The number of octets to write.

Returns

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

See also

[rtxStreamWrite](#)

Implemented in [OSRTOutputStream](#).

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

- [OSRTOutputStreamIF.h](#)

3.49 OSRTOutputStreamPtr Class Reference

Public Member Functions

- **OSRTOutStreamPtr** ([OSRTOutStreamIF](#) *ptr)
- **operator OSRTOutStreamIF** * ()
- [OSRTOutStreamIF](#) * **operator->** ()

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

- [OSRTOutStreamIF.h](#)

3.50 OSRTSocket Class Reference

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

Public Member Functions

- EXTRTMETHOD [OSRTSocket](#) ()
- EXTRTMETHOD [OSRTSocket](#) (OSRTSOCKET socket, OSBOOL ownership=FALSE)
- 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 [connectUrl](#) (const char *url)
- OSBOOL [getOwnership](#) ()
- OSRTSOCKET [getSocket](#) () const
- int [getStatus](#) ()
- EXTRTMETHOD int [listen](#) (int maxConnections)
- EXTRTMETHOD int [recv](#) (OSOCKET *pbuf, size_t bufsize)
- void [setOwnership](#) (OSBOOL ownership)
- EXTRTMETHOD int [send](#) (const OSOCKET *pdata, size_t size)

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](#)
- 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 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 EXTRTMETHOD OSRTSocket::OSRTSocket (OSRTSOCKET *socket*, OSBOOL *ownership* = FALSE)

This constructor initializes an instance by using an existing socket.

Parameters

socket An existing socket handle.

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

3.50.2.3 EXTRTMETHOD OSRTSocket::OSRTSocket (const OSRTSocket & *socket*)

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

3.50.2.4 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 EXTRTMETHOD OSRTSocket* OSRTSocket::accept (OSIPADDR * *destIP* = 0, int * *port* = 0)

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

Parameters

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

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

Returns

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

See also

`rtSocketAccept`

3.50.3.2 `static EXTRTMETHOD const char* OSRTSocket::addrToString (OSIPADDR ipAddr, char * pAddrStr, size_t bufsize) [static]`

This method converts an IP address to its string representation.

Parameters

ipAddr The IP address to be converted.

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

bufsize Size of the buffer.

Returns

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

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

This method associates only a local port with a socket. It is used on an unconnected socket before subsequent calls to the [OSRTSocket::connect](#) or [OSRTSocket::listen](#) methods.

Parameters

port The local port number to assign to the socket.

Returns

Completion status of operation:

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

See also

[rtSocketBind](#)

[bind \(\)](#)

3.50.3.4 `EXTRTMETHOD int OSRTSocket::bind (const char * pAddrStr, int port)`

This method associates a local address with a socket. It is used on an unconnected socket before subsequent calls to the connect or listen methods.

Parameters

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

port The local port number to assign to the socket.

Returns

Completion status of operation:

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

See also

[rtSocketBind](#)

3.50.3.5 EXTRTMETHOD int OSRTSocket::bind (OSIPADDR *addr*, int *port*)

This method associates a local address with a socket. It is used on an unconnected socket before subsequent calls to the connect or listen methods.

Parameters

addr The local IP address to assign to the socket.

port The local port number to assign to the socket.

Returns

Completion status of operation:

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

See also

rtSocketBind

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

This method associates a local address with a socket. It is used on an unconnected socket before subsequent calls to the connect or listen methods. This version of the method allows a URL to be used instead of address and port number.

Parameters

Univeral resource locator (URL) string.

Returns

Completion status of operation:

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

See also

rtSocketBind

3.50.3.7 EXTRTMETHOD int OSRTSocket::blockingRead (OSOCKET * *pbuf*, size_t *readBytes*)

This method receives data from the connected socket. In this case, the connection is blocked until either the requested number of bytes is received or the socket is closed or an error occurs.

Parameters

pbuf Pointer to the buffer for the incoming data.

readBytes Number of bytes to receive.

Returns

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

3.50.3.8 EXTRTMETHOD int OSRTSocket::close ()

This method closes this socket.

Returns

Completion status of operation:

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

See also

rtSocketClose

3.50.3.9 EXTRTMETHOD int OSRTSocket::connect (const char * *host*, int *port*)

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

Parameters

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

port The destination port to connect.

Returns

Completion status of operation:

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

See also

rtSocketConnect

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

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

Parameters

Univeral resource locator (URL) string.

Returns

Completion status of operation:

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

See also

rtSocketConnect

3.50.3.11 OSBOOL OSRTSocket::getOwnership () [inline]

Returns the ownership of underlying O/S socket.

Returns

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

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

This method returns the handle of the socket.

Returns

The handle of the socket.

3.50.3.13 int OSRTSocket::getStatus () [inline]

Returns a completion status of last operation.

Returns

Completion status of last operation:

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

3.50.3.14 EXTRTMETHOD int OSRTSocket::listen (int *maxConnections*)

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

Parameters

maxConnections Maximum length of the queue of pending connections.

Returns

Completion status of operation:

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

See also

rtSocketListen

3.50.3.15 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

pbuf Pointer to the buffer for the incoming data.

bufsize Length of the buffer.

Returns

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

See also

rtSocketRecv

3.50.3.16 EXTRTMETHOD int OSRTSocket::send (const OSOCTET * *pdata*, size_t *size*)

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

Parameters

pdata Buffer containing the data to be transmitted.

size Length of the data in *pdata*.

Returns

Completion status of operation:

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

See also

rtSocketSend

3.50.3.17 void OSRTSocket::setOwnership (OSBOOL *ownership*) [inline]

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

Parameters

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

3.50.3.18 static EXTRTMETHOD OSIPADDR OSRTSocket::stringToAddr (const char * *pAddrStr*) [static]

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

Parameters

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

Returns

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

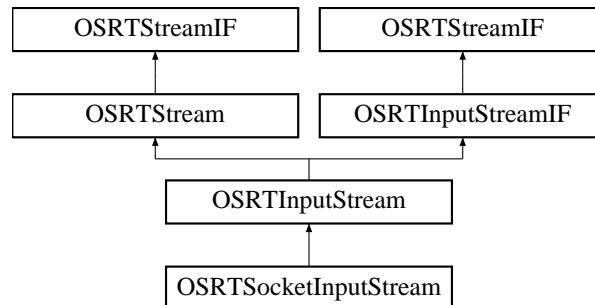
The documentation for this class was generated from the following file:

- [OSRTSocket.h](#)

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](#)

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 EXTRTMETHOD [OSRTSocketInputStream::OSRTSocketInputStream \(OSRTSocket & socket\)](#)

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

Parameters

socket Reference to [OSRTSocket](#) instance.

See also

[rtxStreamSocketOpen](#)

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

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

Parameters

- pContext* Pointer to a context to use.
- socket* Reference to [OSRTSocket](#) instance.

See also

[rtxStreamSocketOpen](#)

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

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

Parameters

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

See also

[rtxStreamSocketAttach](#)

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

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

Parameters

- pContext* Pointer to a context to use.
- socket* Handle of the socket.
- ownership* Indicates ownership of the socket. Set to TRUE to pass ownership to this object instance. The socket will be closed when this object instance is deleted or goes out of scope.

See also

[rtxStreamSocketAttach](#)

3.51.3 Member Function Documentation

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

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

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented from [OSRTInputStream](#).

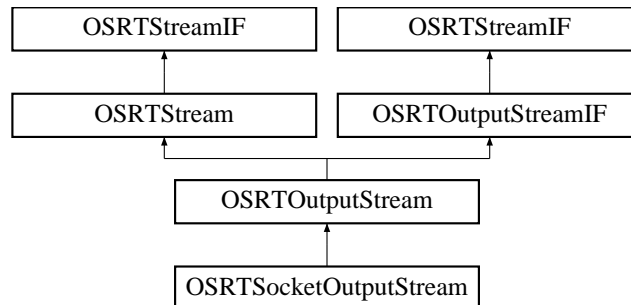
The documentation for this class was generated from the following file:

- [OSRTSocketInputStream.h](#)

3.52 OSRTSocketOutputStream Class Reference

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

Inheritance diagram for OSRTSocketOutputStream:



Public Member Functions

- EXTRTMETHOD [OSRTSocketOutputStream](#) ([OSRTSocket](#) &socket)
- EXTRTMETHOD [OSRTSocketOutputStream](#) ([OSRTContext](#) *pContext, [OSRTSocket](#) &socket)
- EXTRTMETHOD [OSRTSocketOutputStream](#) ([OSRTSOCKET](#) socket, [OSBOOL](#) ownership=FALSE)
- [OSRTSocketOutputStream](#) ([OSRTContext](#) *pContext, [OSRTSOCKET](#) socket, [OSBOOL](#) ownership=FALSE)
- virtual [OSBOOL](#) [isA](#) ([StreamID](#) id) const

Protected Attributes

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

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 EXTRTMETHOD [OSRTSocketOutputStream::OSRTSocketOutputStream](#) ([OSRTSocket](#) & *socket*)

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

Parameters

socket Reference to [OSRTSocket](#) instance.

See also

[rtxStreamSocketOpen](#)

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

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

Parameters

- pContext* Pointer to a context to use.
- socket* Reference to [OSRTSocket](#) instance.

See also

[rtxStreamSocketOpen](#)

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

Initializes the socket output stream using the socket handle.

Parameters

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

See also

[rtxStreamSocketAttach](#)

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

Initializes the socket output stream using the socket handle.

Parameters

- pContext* Pointer to a context to use.
- socket* Handle of the socket.
- ownership* Indicates ownership of the socket. Set to TRUE to pass ownership to this object instance. The socket will be closed when this object instance is deleted or goes out of scope.

See also

[rtxStreamSocketAttach](#)

3.52.3 Member Function Documentation

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

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

Parameters

id Enumerated stream identifier

Returns

True if the stream matches the identifier

Reimplemented from [OSRTOutputStream](#).

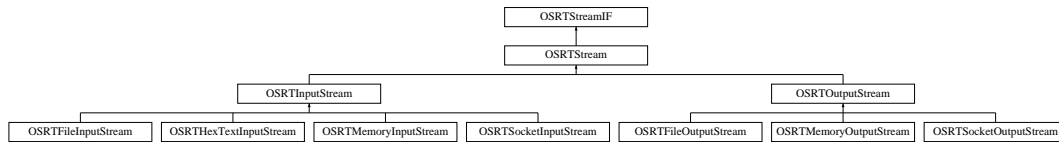
The documentation for this class was generated from the following file:

- [OSRSocketOutputStream.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 * getCtxtPtr ()`
- 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 (OSRTCtxt *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 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 virtual EXTRTMETHOD OSRTStream::~OSRTStream () [virtual]

Virtual destructor. Closes the stream if it was opened.

3.53.3 Member Function Documentation

3.53.3.1 virtual EXTRTMETHOD int OSRTStream::close () [virtual]

Closes the input or output stream and releases any system resources associated with the stream. For output streams this function also flushes all internal buffers to the stream.

Returns

Completion status of operation:

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

See also

[rtxStreamClose](#)

Implements [OSRTStreamIF](#).

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

3.53.3.2 virtual EXTRTMETHOD int OSRTStream::flush () [virtual]

Flushes the buffered data to the stream.

Returns

Completion status of operation:

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

See also

[rtxStreamFlush](#)

Implements [OSRTStreamIF](#).

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

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

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

Returns

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

Implements [OSRTStreamIF](#).

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

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

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

This method returns a pointer to the underlying OSCTXT object. This is the structure used in calls to low-level C encode/decode functions.

Returns

Pointer to a context (OSCTXT) structure.

Implements [OSRTStreamIF](#).

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

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

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

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

Parameters

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

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

Returns

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

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

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

Returns error text in a dynamic memory buffer. Buffer will be allocated by 'operator new []'. The calling routine is responsible to free the memory by using 'operator delete []'.

Returns

A pointer to a newly allocated buffer with error text.

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

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

3.53.3.7 int OSRTStream::getStatus () const [inline]

This method returns the completion status of previous operation. It can be used to check completion status of constructors or methods, which do not return completion status.

Returns

Runtime status code:

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

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

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

3.53.3.8 virtual EXTRTMETHOD OSBOOL OSRTStream::isOpen () [virtual]

Checks, is the stream opened or not.

Returns

TRUE, if the stream is opened, FALSE otherwise.

See also

[rtxStreamIsOpened](#)

Implements [OSRTStreamIF](#).

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

3.53.3.9 void OSRTStream::printErrorInfo () [inline]

The `printErrorInfo` method prints information on errors contained within the context.

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

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

3.53.3.10 void OSRTStream::resetErrorInfo () [inline]

The `resetErrorInfo` method resets information on errors contained within the context.

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

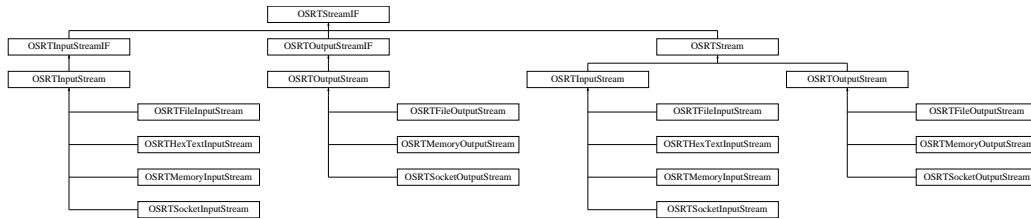
Referenced by [OSRTOutputStream::resetErrorInfo\(\)](#), and [OSRTInputStream::resetErrorInfo\(\)](#).

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 * getCtxPtr ()=0`
- virtual `OSBOOL isOpened ()=0`

3.54.1 Member Function Documentation

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

Closes the input or output stream and releases any system resources associated with the stream. For output streams this function also flushes all internal buffers to the stream.

Returns

Completion status of operation:

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

See also

`rtxStreamClose`

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

3.54.1.2 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 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 [OSRTStream](#), [OSRTInputStream](#), and [OSRTOutputStream](#).

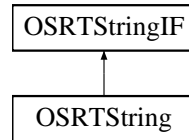
The documentation for this class was generated from the following file:

- [OSRTStreamIF.h](#)

3.55 OSRTString Class Reference

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

Inheritance diagram for OSRTString:



Public Member Functions

- EXTRTMETHOD `OSRTString ()`
- EXTRTMETHOD `OSRTString (const char *strval)`
- EXTRTMETHOD `OSRTString (const OSUTF8CHAR *strval)`
- EXTRTMETHOD `OSRTString (const OSRTString &str)`
- virtual EXTRTMETHOD `~OSRTString ()`
- virtual `OSRTStringIF * clone ()`
- virtual const char * `getValue () const`
- virtual const OSUTF8CHAR * `getUTF8Value () const`
- virtual void `print (const char *name)`
- virtual EXTRTMETHOD void `setValue (const char *str)`
- virtual EXTRTMETHOD void `setValue (const OSUTF8CHAR *str)`
- EXTRTMETHOD `OSRTString & operator= (const OSRTString &original)`

Protected Attributes

- OSUTF8CHAR * `mValue`

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 EXTRTMETHOD OSRTString::OSRTString ()

The default constructor creates an empty string.

3.55.2.2 EXTRTMETHOD OSRTString::OSRTString (const char * *strval*)

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

Parameters

strval - Null-terminated C string value

3.55.2.3 EXTRTMETHOD OSRTString::OSRTString (const OSUTF8CHAR * *strval*)

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

Parameters

strval - Null-terminated C string value

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

Copy constructor.

Parameters

str - C++ string object to be copied.

3.55.2.5 virtual EXTRTMETHOD OSRTString::~OSRTString () [virtual]

The destructor frees string memory using the standard 'delete' operator.

3.55.3 Member Function Documentation

3.55.3.1 virtual OSRTStringIF* OSRTString::clone () [inline, virtual]

This method creates a copy of the given string object.

Implements [OSRTStringIF](#).

3.55.3.2 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](#).

3.55.3.3 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](#).

3.55.3.4 EXTRTMETHOD OSRTString& OSRTString::operator= (const OSRTString & *original*)

Assignment operator.

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

This method prints the string value to standard output.

Parameters

name - Name of generated string variable.

Implements [OSRTStringIF](#).

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

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

Parameters

str - C null-terminated UTF-8 string.

Implements [OSRTStringIF](#).

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

This method sets the string value to the given string.

Parameters

str - C null-terminated string.

Implements [OSRTStringIF](#).

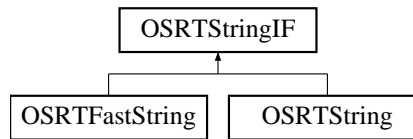
The documentation for this class was generated from the following file:

- [OSRTString.h](#)

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::OSRTStringIF () [`inline`, `protected`]

The default constructor creates an empty string.

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

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

Parameters

- Null-terminated C string value

3.56.2.3 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 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 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 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 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](#).

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

This method prints the string value to standard output.

Parameters

- name* - Name of generated string variable.

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

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

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

Parameters

- utf8str* - C null-terminated UTF-8 string.

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

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

This method sets the string value to the given string.

Parameters

str - C null-terminated string.

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

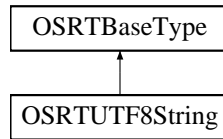
The documentation for this class was generated from the following file:

- [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::OSRTUTF8String ()

The default constructor creates an empty string.

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

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

Parameters

strval - String value

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

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

Parameters

strval - String value

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

Copy constructor.

Parameters

str - C++ XML string class.

3.57.2.5 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 const char* OSRTUTF8String::c_str () const [inline]

This method returns the pointer to C null terminated string.

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

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

Reimplemented from [OSRTBaseType](#).

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

This method copies the given string value to the internal string storage variable. A deep-copy of the given value is done; the class will delete this memory when the object is deleted.

Parameters

str - C null-terminated string.

3.57.3.4 const char* OSRTUTF8String::getValue () const [inline]

This method returns the pointer to UTF-8 null terminated string.

3.57.3.5 OSRTUTF8String& OSRTUTF8String::operator= (const OSRTUTF8String & *original*)

Assignment operator.

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

This method prints the string value to standard output.

Parameters

name - Name of generated string variable.

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

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

Parameters

str - C null-terminated string.

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

- [OSRTUTF8String.h](#)

Chapter 4

File Documentation

4.1 ASN1CBitStr.h File Reference

```
#include "rtsrc/asn1CppTypes.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](#)

Defines

- #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 `asn1CppTypes.h` File Reference

```
#include <new>
#include "rtxsrc/rtxMemory.h"
#include "rtxsrc/rtxDiag.h"
#include "rtxsrc/rtxError.h"
#include "rtxsrc/rtxMemBuf.h"
#include "rtsrc/asn1CppEvtHndlr.h"
#include "rtsrc/ASN1Context.h"
#include "rtxsrc/OSRTMsgBuf.h"
#include "rtsrc/ASN1TOctStr.h"
#include "rtsrc/ASN1TObjId.h"
```

Classes

- class [ASN1MessageBuffer](#)
- class [ASN1CType](#)
- struct [ASN1TDynBitStr](#)
- struct [ASN1TBitStr32](#)
- struct [ASN1TBMPString](#)
- struct [ASN1TUniversalString](#)
- struct [ASN1TOpenType](#)
- struct [Asn1TObject](#)
- struct [ASN1TObjId64](#)
- struct [ASN1TSeqExt](#)
- struct [ASN1TPDU](#)
- struct [ASN1TSeqOfList](#)
- struct [ASN1TPDUSeqOfList](#)

Defines

- `#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.

4.6 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/asn1CppType.h"
#include "rtsrc/ASN1TTime.h"
```

Classes

- class [ASN1CTime](#)

Defines

- #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

Defines

- #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`)

4.9.1 Detailed Description

List of numeric status codes that can be returned by ASN1C run-time functions and generated code.

4.10 ASN1ObjId.h File Reference

```
#include "rtsrc/asn1type.h"
```

Classes

- struct [ASN1ObjId](#)

Functions

- int [operator==](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator==](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator!=](#) (const [ASN1ObjId](#) &lhs, const [ASN1ObjId](#) &rhs)
- int [operator!=](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator!=](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator!=](#) (const [ASN1ObjId](#) &lhs, const char *dotted_oid_string)
- int [operator<](#) (const [ASN1ObjId](#) &lhs, const [ASN1ObjId](#) &rhs)
- int [operator<](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator<](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator<](#) (const [ASN1ObjId](#) &lhs, const char *dotted_oid_string)
- int [operator<=](#) (const [ASN1ObjId](#) &lhs, const [ASN1ObjId](#) &rhs)
- int [operator<=](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator<=](#) (const [ASN1ObjId](#) &lhs, const char *dotted_oid_string)
- int [operator<=](#) (const ASN1OBJID &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 ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator>](#) (const ASN1OBJID &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 ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator>=](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- [ASN1ObjId operator+](#) (const [ASN1ObjId](#) &lhs, const [ASN1ObjId](#) &rhs)

4.10.1 Detailed Description

ASN.1 object identifier class definition.

4.11 ASN1OctStr.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](#)

Defines

- #define `MAX_TIMESTR_SIZE` 64
- #define `LOG_TTMERR(stat)` (mStatus = stat, stat)

4.12.1 Detailed Description

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 *pctxt)
- void [operator delete](#) (void *pmem, OSCTXT *pctxt)

4.14.1 Detailed Description

C++ run-time context class definition.

4.14.2 Function Documentation

4.14.2.1 void operator delete (void * *pmem*, OSCTXT * *pctxt*)

Custom placement delete function to free memory using context memory-management functions.

4.14.2.2 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/OSRCTxtHolder.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/rtxSocket.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/OSRCTxtHolder.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

- ~ASN1Type
 - ASN1Type, 71
- ~ASN1MessageBuffer
 - ASN1MessageBuffer, 82
- ~ASN1ObjId
 - ASN1ObjId, 105
- ~ASN1Time
 - ASN1Time, 118
- ~OSRTContext
 - OSRTContext, 136
- ~OSRTCtxtPtr
 - OSRTCtxtPtr, 140
- ~OSRTFastString
 - OSRTFastString, 144
- ~OSRTHexTextInputStream
 - OSRTHexTextInputStream, 151
- ~OSRTInputStream
 - OSRTInputStream, 154
- ~OSRTInputStreamIF
 - OSRTInputStreamIF, 160
- ~OSRTMessageBuffer
 - OSRTMessageBuffer, 171
- ~OSRTMessageBufferIF
 - OSRTMessageBufferIF, 176
- ~OSRTOutputStream
 - OSRTOutputStream, 179
- ~OSRTOutputStreamIF
 - OSRTOutputStreamIF, 184
- ~OSRTSocket
 - OSRTSocket, 188
- ~OSRTStream
 - OSRTStream, 202
- ~OSRTString
 - OSRTString, 208
- ~OSRTStringIF
 - OSRTStringIF, 211
- ~OSRTUTF8String
 - OSRTUTF8String, 214
- _ref
 - OSRTContext, 136
- _unref
 - OSRTContext, 136
- accept
 - OSRTSocket, 188
- addEventHandler
 - ASN1MessageBuffer, 82
- addrToString
 - OSRTSocket, 188
- append
 - ASN1CSeqOfList, 47
 - ASN1Type, 71
- appendArray
 - ASN1CSeqOfList, 47
- appendArrayCopy
 - ASN1CSeqOfList, 48
- ASN.1 Stream Classes, 15
- ASN.1 Type (ASN1T_) Base Classes, 5
- ASN1CBitStr, 23
 - ASN1CBitStr, 25
 - cardinality, 25
 - change, 25
 - clear, 26
 - doAnd, 27
 - doAndNot, 28
 - doOr, 29
 - doXor, 30
 - get, 30, 31
 - getBytes, 31
 - invert, 32
 - isEmpty, 32
 - isSet, 32
 - length, 33
 - operator ASN1TDynBitStr, 33
 - operator ASN1TDynBitStr *, 33
 - set, 33, 34
 - shiftLeft, 34
 - shiftRight, 34
 - size, 35
 - unusedBitsInLastUnit, 35
- ASN1CBitStr.h, 216
- ASN1CBitStrSizeHolder, 36
- ASN1CBitStrSizeHolder16, 37
- ASN1CBitStrSizeHolder32, 38
- ASN1CBitStrSizeHolder8, 39
- ASN1CGeneralizedTime, 40
 - ASN1CGeneralizedTime, 41
 - getCentury, 41
 - setCentury, 41
 - setTime, 42

- ASN1CGeneralizedTime.h, 217
- ASN1Context, 43
 - ASN1Context, 43
 - setRunTimeKey, 43
- ASN1Context.h, 218
- asn1CppEvtHndlr.h, 219
- asn1CppTypes.h, 220
- ASN1CSeqOfList, 45
 - append, 47
 - appendArray, 47
 - appendArrayCopy, 48
 - ASN1CSeqOfList, 46, 47
 - clear, 48
 - contains, 48
 - free, 48
 - get, 49
 - getFirst, 49
 - getLast, 49
 - indexOf, 49
 - init, 49
 - insert, 49
 - isEmpty, 50
 - iterator, 50
 - iteratorFrom, 50
 - iteratorFromLast, 50
 - remove, 51
 - removeFirst, 51
 - removeLast, 51
 - set, 52
 - size, 52
 - toArray, 52
- ASN1CSeqOfList.h, 221
- ASN1CSeqOfListIterator, 54
 - hasNext, 54
 - hasPrev, 55
 - insert, 55
 - next, 55
 - prev, 55
 - remove, 55
 - set, 56
- ASN1CTime, 57
 - ASN1CTime, 59
 - clear, 60
 - equals, 60
 - getDay, 60
 - getDiff, 60
 - getDiffHour, 60
 - getDiffMinute, 61
 - getFraction, 61
 - getFractionAsDouble, 61
 - getFractionLen, 62
 - getFractionStr, 62
 - getHour, 62
 - getMinute, 62
 - getMonth, 62
 - getSecond, 63
 - getTime, 63
 - getUTC, 63
 - getYear, 64
 - parseString, 64
 - setDay, 64
 - setDER, 64
 - setDiff, 65
 - setDiffHour, 65
 - setFraction, 66
 - setHour, 67
 - setMinute, 67
 - setMonth, 67
 - setSecond, 68
 - setTime, 68
 - setUTC, 68
 - setYear, 69
- ASN1CTime.h, 222
- ASN1CType, 70
 - ~ASN1CType, 71
 - append, 71
 - ASN1CType, 71
 - Decode, 72
 - DecodeFrom, 72
 - Encode, 72
 - EncodeTo, 72
 - getContext, 73
 - getCtxtPtr, 73
 - getStatus, 73
 - memAlloc, 73
 - memFreeAll, 73
 - memFreePtr, 74
 - memRealloc, 74
 - memReset, 74
 - mpContext, 75
 - mpMsgBuf, 75
 - printErrorInfo, 74
 - resetError, 74
 - setDiag, 74
 - setRunTimeKey, 74
- ASN1CUTCCTime, 76
 - ASN1CUTCCTime, 77
 - getFraction, 77
 - setTime, 77
- ASN1CUTCCTime.h, 223
- asn1data
 - operator<, 7, 8
 - operator<=, 8, 9
 - operator>, 10, 11
 - operator>=, 11, 12
 - operator+, 7
 - operator==, 9, 10
- asn1ErrCodes

- ASN_E_BADTAG, 20
- ASN_E_BASE, 20
- ASN_E_CONCMODF, 20
- ASN_E_ILLSTATE, 20
- ASN_E_INVBINS, 20
- ASN_E_INVINDEX, 21
- ASN_E_INVLEN, 21
- ASN_E_INVOBJID, 21
- ASN_E_INVPERENC, 21
- ASN_E_INVTCVAL, 21
- ASN_E_NOTINSEQ, 21
- ASN_E_NOTPDU, 21
- ASN_E_UNDEFTYP, 21
- ASN_OK_FRAG, 21
- asn1ErrCodes.h, 224
- Asn1ErrorHandler, 79
 - error, 79
 - setErrorHandler, 79
- ASN1MessageBuffer, 81
 - ~ASN1MessageBuffer, 82
 - addEventHandler, 82
 - ASN1MessageBuffer, 82
 - CStringToBMPString, 82
 - getAppInfo, 82
 - initBuffer, 83
 - isA, 83
 - removeEventHandler, 84
 - resetErrorInfo, 84
 - setAppInfo, 84
 - setErrorHandler, 84
 - setRunTimeKey, 84
 - setStatus, 85
- Asn1NamedEventHandler, 86
 - bitStringValue, 87
 - boolValue, 87
 - charStringValue, 87, 88
 - endElement, 89
 - enumValue, 89
 - int64Value, 89
 - intValue, 89
 - nullValue, 90
 - octStringValue, 90
 - oidValue, 90
 - openTypeValue, 90
 - realValue, 91
 - startElement, 91
 - uint64Value, 91
 - uintValue, 91
- Asn1NullEventHandler, 93
 - endElement, 93
 - startElement, 93
- ASN1TBitStr32, 95
 - ASN1TBitStr32, 95
- ASN1TBMPString, 96
 - ASN1TBMPString, 96
- ASN1TDynBitStr, 97
 - ASN1TDynBitStr, 97
- ASN1TDynOctStr, 98
 - ASN1TDynOctStr, 98
 - nCompare, 99
 - operator=, 99
 - toHexString, 99
 - toString, 99
- ASN1TGeneralizedTime, 101
 - ASN1TGeneralizedTime, 101, 102
 - compileString, 102
 - getCentury, 102
 - parseString, 102
 - setCentury, 102
 - setTime, 103
- Asn1TObject, 104
 - Asn1TObject, 104
- ASN1TObjId, 105
 - ~ASN1TObjId, 105
 - ASN1TObjId, 105, 106
 - nCompare, 106
 - operator+=", 106, 107
 - operator=, 107
 - RnCompare, 107
 - set_data, 108
 - toString, 108
 - trim, 108
- ASN1TObjId.h, 225
- ASN1TObjId64, 109
- ASN1TOctStr.h, 226
- ASN1TOpenType, 110
 - ASN1TOpenType, 110
- ASN1TPDU, 111
 - mpContext, 111
 - setContext, 111
- ASN1TPDUSeqOfList, 113
 - ASN1TPDUSeqOfList, 113
- ASN1TSeqExt, 114
 - ASN1TSeqExt, 114
- ASN1TSeqOfList, 115
 - ASN1TSeqOfList, 115
- ASN1TTime, 116
 - ~ASN1TTime, 118
 - ASN1TTime, 118
 - clear, 118
 - compileString, 118
 - equals, 119
 - getDay, 119
 - getDiff, 119
 - getDiffHour, 119
 - getDiffMinute, 119
 - getFraction, 120
 - getFractionAsDouble, 120

- getFractionLen, [120](#)
- getFractionStr, [120](#)
- getHour, [121](#)
- getMinute, [121](#)
- getMonth, [121](#)
- getSecond, [121](#)
- getTime, [121](#)
- getUTC, [122](#)
- getYear, [122](#)
- parseString, [122](#)
- setDay, [122](#)
- setDER, [123](#)
- setDiff, [123](#)
- setDiffHour, [124](#)
- setFraction, [124](#), [125](#)
- setHour, [125](#)
- setMinute, [125](#)
- setMonth, [126](#)
- setSecond, [126](#)
- setTime, [126](#)
- setUTC, [127](#)
- setYear, [127](#)
- toString, [127](#), [128](#)
- ASN1TTime.h, [227](#)
- ASN1TUniversalString, [129](#)
 - ASN1TUniversalString, [129](#)
- ASN1TUTCTime, [130](#)
 - ASN1TUTCTime, [130](#), [131](#)
 - clear, [131](#)
 - compileString, [131](#)
 - getFraction, [131](#)
 - parseString, [131](#)
 - setFraction, [132](#)
 - setTime, [132](#)
 - setUTC, [132](#)
 - setYear, [133](#)
- ASN_E_BADTAG
 - asn1ErrCodes, [20](#)
- ASN_E_BASE
 - asn1ErrCodes, [20](#)
- ASN_E_CONCMODF
 - asn1ErrCodes, [20](#)
- ASN_E_ILLSTATE
 - asn1ErrCodes, [20](#)
- ASN_E_INVBINS
 - asn1ErrCodes, [20](#)
- ASN_E_INVINDEX
 - asn1ErrCodes, [21](#)
- ASN_E_INVLEN
 - asn1ErrCodes, [21](#)
- ASN_E_INVOBJID
 - asn1ErrCodes, [21](#)
- ASN_E_INVPERENC
 - asn1ErrCodes, [21](#)
- ASN_E_INVTCVAL
 - asn1ErrCodes, [21](#)
- ASN_E_NOTINSEQ
 - asn1ErrCodes, [21](#)
- ASN_E_NOTPDU
 - asn1ErrCodes, [21](#)
- ASN_E_UNDEFTYP
 - asn1ErrCodes, [21](#)
- ASN_OK_FRAG
 - asn1ErrCodes, [21](#)
- bind
 - OSRSocket, [189](#)
- bindUrl
 - OSRSocket, [190](#)
- bitStringValue
 - Asn1NamedEventHandler, [87](#)
- blockingRead
 - OSRSocket, [190](#)
- boolValue
 - Asn1NamedEventHandler, [87](#)
- C++ Run-Time Classes, [2](#)
- c_str
 - OSRTUTF8String, [214](#)
- cardinality
 - ASN1CBitStr, [25](#)
- change
 - ASN1CBitStr, [25](#)
- charStringValue
 - Asn1NamedEventHandler, [87](#), [88](#)
- clear
 - ASN1CBitStr, [26](#)
 - ASN1CSeqOfList, [48](#)
 - ASN1CTime, [60](#)
 - ASN1TTime, [118](#)
 - ASN1TUTCTime, [131](#)
- clone
 - OSRTFastString, [144](#)
 - OSRTString, [208](#)
 - OSRTStringIF, [211](#)
 - OSRTUTF8String, [214](#)
- close
 - OSRTInputStream, [154](#)
 - OSRTOutputStream, [180](#)
 - OSRSocket, [190](#)
 - OSRTStream, [202](#)
 - OSRTStreamIF, [205](#)
- compileString
 - ASN1TGeneralizedTime, [102](#)
 - ASN1TTime, [118](#)
 - ASN1TUTCTime, [131](#)
- connect
 - OSRSocket, [191](#)

- connectUrl
 - OSRSocket, 191
- contains
 - ASN1CSeqOfList, 48
- Context Management Classes, 13
- Control (ASN1C_) Base Classes, 4
- copyValue
 - OSRTUTF8String, 214
- CStringToBMPString
 - ASN1MessageBuffer, 82
- currentPos
 - OSRTInputStream, 154
 - OSRTInputStreamIF, 160
- Date and Time Runtime Classes, 14
- Decode
 - ASN1CType, 72
- DecodeFrom
 - ASN1CType, 72
- doAnd
 - ASN1CBitStr, 27
- doAndNot
 - ASN1CBitStr, 28
- doOr
 - ASN1CBitStr, 29
- doXor
 - ASN1CBitStr, 30
- Encode
 - ASN1CType, 72
- EncodeTo
 - ASN1CType, 72
- endElement
 - Asn1NamedEventHandler, 89
 - Asn1NullEventHandler, 93
- enumValue
 - Asn1NamedEventHandler, 89
- equals
 - ASN1CTime, 60
 - ASN1TTime, 119
- error
 - Asn1ErrorHandler, 79
- flush
 - OSRTInputStream, 154
 - OSRTOutputStream, 180
 - OSRTStream, 202
 - OSRTStreamIF, 205
- free
 - ASN1CSeqOfList, 48
- Generic Input Stream Classes, 16
- Generic Output Stream Classes, 17
- get
 - ASN1CBitStr, 30, 31
 - ASN1CSeqOfList, 49
- getAppInfo
 - ASN1MessageBuffer, 82
 - OSRTMessageBuffer, 171
 - OSRTMessageBufferIF, 176
- getBuffer
 - OSRTMemoryOutputStream, 168
- getByteIndex
 - OSRTMessageBuffer, 171
 - OSRTMessageBufferIF, 176
- getBytes
 - ASN1CBitStr, 31
- getCentury
 - ASN1CGeneralizedTime, 41
 - ASN1TGeneralizedTime, 102
- getContext
 - ASN1CType, 73
 - OSRTInputStream, 155
 - OSRTMessageBuffer, 171
 - OSRTOutputStream, 180
 - OSRTStream, 202
- getCtxtPtr
 - ASN1CType, 73
 - OSRTCtxtPtr, 141
 - OSRTInputStream, 155
 - OSRTMessageBuffer, 171
 - OSRTOutputStream, 180
 - OSRTStream, 203
- getDay
 - ASN1CTime, 60
 - ASN1TTime, 119
- getDiff
 - ASN1CTime, 60
 - ASN1TTime, 119
- getDiffHour
 - ASN1CTime, 60
 - ASN1TTime, 119
- getDiffMinute
 - ASN1CTime, 61
 - ASN1TTime, 119
- getErrorInfo
 - OSRTContext, 136
 - OSRTInputStream, 155
 - OSRTMessageBuffer, 171, 172
 - OSRTOutputStream, 181
 - OSRTStream, 203
- getFirst
 - ASN1CSeqOfList, 49
- getFraction
 - ASN1CTime, 61
 - ASN1CUTCTime, 77
 - ASN1TTime, 120
 - ASN1TUTCTime, 131
- getFractionAsDouble

- ASN1CTime, 61
- ASN1TTime, 120
- getFractionLen
 - ASN1CTime, 62
 - ASN1TTime, 120
- getFractionStr
 - ASN1CTime, 62
 - ASN1TTime, 120
- getHour
 - ASN1CTime, 62
 - ASN1TTime, 121
- getLast
 - ASN1CSeqOfList, 49
- getMinute
 - ASN1CTime, 62
 - ASN1TTime, 121
- getMonth
 - ASN1CTime, 62
 - ASN1TTime, 121
- getMsgCopy
 - OSRTMessageBuffer, 172
 - OSRTMessageBufferIF, 176
- getMsgPtr
 - OSRTMessageBuffer, 172
 - OSRTMessageBufferIF, 176
- getOwnership
 - OSRTSocket, 191
- getPosition
 - OSRTInputStream, 156
 - OSRTInputStreamIF, 160
- getPtr
 - OSRTContext, 137
- getRefCount
 - OSRTContext, 137
- getSecond
 - ASN1CTime, 63
 - ASN1TTime, 121
- getSocket
 - OSRTSocket, 192
- getStatus
 - ASN1CType, 73
 - OSRTContext, 137
 - OSRTInputStream, 156
 - OSRTMessageBuffer, 172
 - OSRTOutputStream, 181
 - OSRTSocket, 192
 - OSRTStream, 204
- getTime
 - ASN1CTime, 63
 - ASN1TTime, 121
- getUTC
 - ASN1CTime, 63
 - ASN1TTime, 122
- getUTF8Value
 - OSRTFastString, 144
 - OSRTString, 208
 - OSRTStringIF, 211
- getValue
 - OSRTFastString, 144
 - OSRTString, 208
 - OSRTStringIF, 211
 - OSRTUTF8String, 214
- getYear
 - ASN1CTime, 64
 - ASN1TTime, 122
- hasNext
 - ASN1CSeqOfListIterator, 54
- hasPrev
 - ASN1CSeqOfListIterator, 55
- indexOf
 - ASN1CSeqOfList, 49
- init
 - ASN1CSeqOfList, 49
 - OSRTMessageBuffer, 172
 - OSRTMessageBufferIF, 176
- initBuffer
 - ASN1MessageBuffer, 83
 - OSRTMessageBuffer, 173
 - OSRTMessageBufferIF, 176
- insert
 - ASN1CSeqOfList, 49
 - ASN1CSeqOfListIterator, 55
- int64Value
 - Asn1NamedEventHandler, 89
- intValue
 - Asn1NamedEventHandler, 89
- invert
 - ASN1CBitStr, 32
- isA
 - ASN1MessageBuffer, 83
 - OSRTFileInputStream, 147
 - OSRTFileOutputStream, 149
 - OSRTHexTextInputStream, 152
 - OSRTInputStream, 156
 - OSRTInputStreamIF, 161
 - OSRTMemoryInputStream, 166
 - OSRTMemoryOutputStream, 168
 - OSRTMessageBufferIF, 177
 - OSRTOutputStream, 182
 - OSRTOutputStreamIF, 184
 - OSRTSocketInputStream, 196
 - OSRTSocketOutputStream, 199
- isEmpty
 - ASN1CBitStr, 32
 - ASN1CSeqOfList, 50
- isInitialized

- OSRTContext, [137](#)
- isNull
 - OSRTCtxtPtr, [141](#)
- isOpened
 - OSRTInputStream, [157](#)
 - OSRTOutputStream, [182](#)
 - OSRTStream, [204](#)
 - OSRTStreamIF, [205](#)
- isSet
 - ASN1CBitStr, [32](#)
- iterator
 - ASN1CSeqOfList, [50](#)
- iteratorFrom
 - ASN1CSeqOfList, [50](#)
- iteratorFromLast
 - ASN1CSeqOfList, [50](#)
- length
 - ASN1CBitStr, [33](#)
- listen
 - OSRTSocket, [192](#)
- mark
 - OSRTInputStream, [157](#)
 - OSRTInputStreamIF, [161](#)
- markSupported
 - OSRTInputStream, [157](#)
 - OSRTInputStreamIF, [161](#)
- mbInitialized
 - OSRTContext, [139](#)
- mBufferType
 - OSRTMessageBuffer, [174](#)
- mCount
 - OSRTContext, [139](#)
- mCtxt
 - OSRTContext, [139](#)
- memAlloc
 - ASN1CType, [73](#)
 - OSRTContext, [137](#)
- memFreeAll
 - ASN1CType, [73](#)
 - OSRTContext, [137](#)
- memFreePtr
 - ASN1CType, [74](#)
 - OSRTContext, [137](#)
- memRealloc
 - ASN1CType, [74](#)
 - OSRTContext, [138](#)
- memReset
 - ASN1CType, [74](#)
 - OSRTContext, [138](#)
- mpContext
 - ASN1CType, [75](#)
 - ASN1TPDU, [111](#)
- mpMsgBuf
 - ASN1CType, [75](#)
- mPointer
 - OSRTCtxtPtr, [141](#)
- mStatus
 - OSRTContext, [139](#)
- Named Event Handlers, [19](#)
- nCompare
 - ASN1TDynOctStr, [99](#)
 - ASN1TObjId, [106](#)
- next
 - ASN1CSeqOfListIterator, [55](#)
- nullValue
 - Asn1NamedEventHandler, [90](#)
- octStrValue
 - Asn1NamedEventHandler, [90](#)
- oidValue
 - Asn1NamedEventHandler, [90](#)
- openTypeValue
 - Asn1NamedEventHandler, [90](#)
- operator ASN1TDynBitStr
 - ASN1CBitStr, [33](#)
- operator ASN1TDynBitStr *
 - ASN1CBitStr, [33](#)
- operator delete
 - OSRTContext.h, [229](#)
- operator new
 - OSRTContext.h, [229](#)
- operator OSRTContext *
 - OSRTCtxtPtr, [141](#)
- operator<
 - asn1data, [7, 8](#)
- operator<=
 - asn1data, [8, 9](#)
- operator>
 - asn1data, [10, 11](#)
- operator>=
 - asn1data, [11, 12](#)
- operator+
 - asn1data, [7](#)
- operator+=
 - ASN1TObjId, [106, 107](#)
- operator->
 - OSRTCtxtPtr, [141](#)
- operator=
 - ASN1TDynOctStr, [99](#)
 - ASN1TObjId, [107](#)
 - OSRTCtxtPtr, [141](#)
 - OSRTFastString, [144](#)
 - OSRTString, [208](#)
 - OSRTUTF8String, [214](#)
- operator==

- asn1data, 9, 10
- OSRTCtxtPtr, 141
- OSRT Message Buffer Classes, 3
- OSRTBaseType, 134
- OSRTBaseType.h, 228
- OSRTContext, 135
 - ~OSRTContext, 136
 - _ref, 136
 - _unref, 136
 - getErrorInfo, 136
 - getPtr, 137
 - getRefCount, 137
 - getStatus, 137
 - isInitialized, 137
 - mbInitialized, 139
 - mCount, 139
 - mCtxt, 139
 - memAlloc, 137
 - memFreeAll, 137
 - memFreePtr, 137
 - memRealloc, 138
 - memReset, 138
 - mStatus, 139
 - OSRTContext, 136
 - printErrorInfo, 138
 - resetErrorInfo, 138
 - setDiag, 138
 - setRunTimeKey, 138
 - setStatus, 139
- OSRTContext.h, 229
 - operator delete, 229
 - operator new, 229
- OSRTCtxtPtr, 140
 - ~OSRTCtxtPtr, 140
 - getCtxtPtr, 141
 - isNull, 141
 - mPointer, 141
 - operator OSRTContext *, 141
 - operator->, 141
 - operator=, 141
 - operator==, 141
 - OSRTCtxtPtr, 140
- OSRTFastString, 143
 - ~OSRTFastString, 144
 - clone, 144
 - getUTF8Value, 144
 - getValue, 144
 - operator=, 144
 - OSRTFastString, 143, 144
 - print, 144
 - setValue, 145
- OSRTFastString.h, 230
- OSRTFileInputStream, 146
 - isA, 147
 - OSRTFileInputStream, 146, 147
 - OSRTFileInputStream.h, 231
 - OSRTFileOutputStream, 148
 - isA, 149
 - OSRTFileOutputStream, 148, 149
 - OSRTFileOutputStream.h, 232
 - OSRTHexTextInputStream, 151
 - ~OSRTHexTextInputStream, 151
 - isA, 152
 - OSRTHexTextInputStream, 151
 - setOwnUnderStream, 152
 - OSRTHexTextInputStream.h, 233
 - OSRTInputStream, 153
 - ~OSRTInputStream, 154
 - close, 154
 - currentPos, 154
 - flush, 154
 - getContext, 155
 - getCtxtPtr, 155
 - getErrorInfo, 155
 - getPosition, 156
 - getStatus, 156
 - isA, 156
 - isOpened, 157
 - mark, 157
 - markSupported, 157
 - OSRTInputStream, 154
 - printErrorInfo, 157
 - read, 158
 - readBlocking, 158
 - reset, 158
 - resetErrorInfo, 158
 - setPosition, 159
 - skip, 159
 - OSRTInputStream.h, 234
 - OSRTInputStreamIF, 160
 - ~OSRTInputStreamIF, 160
 - currentPos, 160
 - getPosition, 160
 - isA, 161
 - mark, 161
 - markSupported, 161
 - read, 162
 - readBlocking, 162
 - reset, 162
 - setPosition, 163
 - skip, 163
 - OSRTInputStreamIF.h, 235
 - OSRTInputStreamPtr, 164
 - OSRTMemoryInputStream, 165
 - isA, 166
 - OSRTMemoryInputStream, 165
 - OSRTMemoryInputStream.h, 236
 - OSRTMemoryOutputStream, 167

- getBuffer, 168
- isA, 168
- OSRTMemoryOutputStream, 167, 168
- reset, 168
- OSRTMemoryOutputStream.h, 237
- OSRTMessageBuffer, 170
 - ~OSRTMessageBuffer, 171
 - getAppInfo, 171
 - getByteIndex, 171
 - getContext, 171
 - getCtxtPtr, 171
 - getErrorInfo, 171, 172
 - getMsgCopy, 172
 - getMsgPtr, 172
 - getStatus, 172
 - init, 172
 - initBuffer, 173
 - mBufferType, 174
 - OSRTMessageBuffer, 171
 - printErrorInfo, 173
 - resetErrorInfo, 173
 - setAppInfo, 173
 - setDiag, 173
- OSRTMessageBufferIF, 175
 - ~OSRTMessageBufferIF, 176
 - getAppInfo, 176
 - getByteIndex, 176
 - getMsgCopy, 176
 - getMsgPtr, 176
 - init, 176
 - initBuffer, 176
 - isA, 177
 - setAppInfo, 177
 - setDiag, 177
 - setNamespace, 177
- OSRTMsgBuf.h, 238
- OSRTMsgBufIF.h, 239
- OSRTOutputStream, 179
 - ~OSRTOutputStream, 179
 - close, 180
 - flush, 180
 - getContext, 180
 - getCtxtPtr, 180
 - getErrorInfo, 181
 - getStatus, 181
 - isA, 182
 - isOpened, 182
 - OSRTOutputStream, 179
 - printErrorInfo, 182
 - resetErrorInfo, 182
 - write, 182, 183
- OSRTOutputStream.h, 240
- OSRTOutputStreamIF, 184
 - ~OSRTOutputStreamIF, 184
 - isA, 184
 - write, 184
- OSRTOutputStreamIF.h, 241
- OSRTOutputStreamPtr, 186
- OSRTSocket, 187
 - ~OSRTSocket, 188
 - accept, 188
 - addrToString, 188
 - bind, 189
 - bindUrl, 190
 - blockingRead, 190
 - close, 190
 - connect, 191
 - connectUrl, 191
 - getOwnership, 191
 - getSocket, 192
 - getStatus, 192
 - listen, 192
 - OSRTSocket, 188
 - recv, 192
 - send, 193
 - setOwnership, 193
 - stringToAddr, 193
- OSRTSocket.h, 242
- OSRTSocketInputStream, 195
 - isA, 196
 - OSRTSocketInputStream, 195, 196
- OSRTSocketInputStream.h, 243
- OSRTSocketOutputStream, 198
 - isA, 199
 - OSRTSocketOutputStream, 198, 199
- OSRTSocketOutputStream.h, 244
- OSRTStream, 201
 - ~OSRTStream, 202
 - close, 202
 - flush, 202
 - getContext, 202
 - getCtxtPtr, 203
 - getErrorInfo, 203
 - getStatus, 204
 - isOpened, 204
 - OSRTStream, 202
 - printErrorInfo, 204
 - resetErrorInfo, 204
- OSRTStream.h, 245
- OSRTStreamIF, 205
 - close, 205
 - flush, 205
 - isOpened, 205
- OSRTStreamIF.h, 246
- OSRTString, 207
 - ~OSRTString, 208
 - clone, 208
 - getUTF8Value, 208

- getValue, 208
- operator=, 208
- OSRTString, 207, 208
- print, 208
- setValue, 209
- OSRTString.h, 247
- OSRTStringIF, 210
 - ~OSRTStringIF, 211
 - clone, 211
 - getUTF8Value, 211
 - getValue, 211
 - OSRTStringIF, 210
 - print, 211
 - setValue, 211
- OSRTStringIF.h, 248
- OSRTUTF8String, 213
 - ~OSRTUTF8String, 214
 - c_str, 214
 - clone, 214
 - copyValue, 214
 - getValue, 214
 - operator=, 214
 - OSRTUTF8String, 213, 214
 - print, 214
 - setValue, 215
- OSRTUTF8String.h, 249
- parseString
 - ASN1CTime, 64
 - ASN1TGeneralizedTime, 102
 - ASN1TTime, 122
 - ASN1TUTCTime, 131
- prev
 - ASN1CSeqOfListIterator, 55
- print
 - OSRTFastString, 144
 - OSRTString, 208
 - OSRTStringIF, 211
 - OSRTUTF8String, 214
- printErrorInfo
 - ASN1CType, 74
 - OSRTContext, 138
 - OSRTInputStream, 157
 - OSRTMessageBuffer, 173
 - OSRTOutputStream, 182
 - OSRTStream, 204
- read
 - OSRTInputStream, 158
 - OSRTInputStreamIF, 162
- readBlocking
 - OSRTInputStream, 158
 - OSRTInputStreamIF, 162
- realValue
 - Asn1NamedEventHandler, 91
- recv
 - OSRTSocket, 192
- remove
 - ASN1CSeqOfList, 51
 - ASN1CSeqOfListIterator, 55
- removeEventHandler
 - ASN1MessageBuffer, 84
- removeFirst
 - ASN1CSeqOfList, 51
- removeLast
 - ASN1CSeqOfList, 51
- reset
 - OSRTInputStream, 158
 - OSRTInputStreamIF, 162
 - OSRTMemoryOutputStream, 168
- resetError
 - ASN1CType, 74
- resetErrorInfo
 - ASN1MessageBuffer, 84
 - OSRTContext, 138
 - OSRTInputStream, 158
 - OSRTMessageBuffer, 173
 - OSRTOutputStream, 182
 - OSRTStream, 204
- RnCompare
 - ASN1TObjId, 107
- Run-time error status codes., 20
- send
 - OSRTSocket, 193
- set
 - ASN1CBitStr, 33, 34
 - ASN1CSeqOfList, 52
 - ASN1CSeqOfListIterator, 56
- set_data
 - ASN1TObjId, 108
- setAppInfo
 - ASN1MessageBuffer, 84
 - OSRTMessageBuffer, 173
 - OSRTMessageBufferIF, 177
- setCentury
 - ASN1CGeneralizedTime, 41
 - ASN1TGeneralizedTime, 102
- setContext
 - ASN1TPDU, 111
- setDay
 - ASN1CTime, 64
 - ASN1TTime, 122
- setDER
 - ASN1CTime, 64
 - ASN1TTime, 123
- setDiag
 - ASN1CType, 74

- OSRTContext, 138
- OSRTMessageBuffer, 173
- OSRTMessageBufferIF, 177
- setDiff
 - ASN1CTime, 65
 - ASN1TTime, 123
- setDiffHour
 - ASN1CTime, 65
 - ASN1TTime, 124
- setErrorHandler
 - Asn1ErrorHandler, 79
 - ASN1MessageBuffer, 84
- setFraction
 - ASN1CTime, 66
 - ASN1TTime, 124, 125
 - ASN1TUTCTime, 132
- setHour
 - ASN1CTime, 67
 - ASN1TTime, 125
- setMinute
 - ASN1CTime, 67
 - ASN1TTime, 125
- setMonth
 - ASN1CTime, 67
 - ASN1TTime, 126
- setNamespace
 - OSRTMessageBufferIF, 177
- setOwnership
 - OSRTSocket, 193
- setOwnUnderStream
 - OSRTHexTextInputStream, 152
- setPosition
 - OSRTInputStream, 159
 - OSRTInputStreamIF, 163
- setRunTimeKey
 - ASN1Context, 43
 - ASN1CType, 74
 - ASN1MessageBuffer, 84
 - OSRTContext, 138
- setSecond
 - ASN1CTime, 68
 - ASN1TTime, 126
- setStatus
 - ASN1MessageBuffer, 85
 - OSRTContext, 139
- setTime
 - ASN1CGeneralizedTime, 42
 - ASN1CTime, 68
 - ASN1CUTCTime, 77
 - ASN1TGeneralizedTime, 103
 - ASN1TTime, 126
 - ASN1TUTCTime, 132
- setUTC
 - ASN1CTime, 68
 - ASN1TTime, 127
 - ASN1TUTCTime, 132
- setValue
 - OSRTFastString, 145
 - OSRTString, 209
 - OSRTStringIF, 211
 - OSRTUTF8String, 215
- setYear
 - ASN1CTime, 69
 - ASN1TTime, 127
 - ASN1TUTCTime, 133
- shiftLeft
 - ASN1CBitStr, 34
- shiftRight
 - ASN1CBitStr, 34
- size
 - ASN1CBitStr, 35
 - ASN1CSeqOfList, 52
- skip
 - OSRTInputStream, 159
 - OSRTInputStreamIF, 163
- startElement
 - Asn1NamedEventHandler, 91
 - Asn1NullEventHandler, 93
- stringToAddr
 - OSRTSocket, 193
- TCP/IP or UDP Socket Classes, 18
- toArray
 - ASN1CSeqOfList, 52
- toHexString
 - ASN1TDynOctStr, 99
- toString
 - ASN1TDynOctStr, 99
 - ASN1TObjId, 108
 - ASN1TTime, 127, 128
- trim
 - ASN1TObjId, 108
- uInt64Value
 - Asn1NamedEventHandler, 91
- uIntValue
 - Asn1NamedEventHandler, 91
- unusedBitsInLastUnit
 - ASN1CBitStr, 35
- write
 - OSRTOutputStream, 182, 183
 - OSRTOutputStreamIF, 184