

# ASN1C

---

ASN.1 Compiler  
Version 7.2  
BER Runtime  
Reference Manual



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

### **Copyright Notice**

Copyright ©1997–2018 Objective Systems, Inc. All rights reserved.

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

### **Author's Contact Information**

Comments, suggestions, and inquiries regarding ASN1C may be submitted via electronic mail to [info@obj-sys.com](mailto:info@obj-sys.com).



# Contents

<b>1</b>	<b>ASN1C BER/DER Runtime Classes and Library Functions</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>3</b>
2.1	Modules . . . . .	3
<b>3</b>	<b>Hierarchical Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List . . . . .	7
<b>5</b>	<b>File Index</b>	<b>9</b>
5.1	File List . . . . .	9
<b>6</b>	<b>Module Documentation</b>	<b>11</b>
6.1	BER/DER/CER C++ Run-Time Classes. . . . .	11
6.1.1	Detailed Description . . . . .	11
6.2	BER Message Buffer Classes . . . . .	12
6.2.1	Detailed Description . . . . .	12
6.3	BER Runtime Library Functions. . . . .	13
6.3.1	Detailed Description . . . . .	13
6.4	BER/DER C Decode Functions. . . . .	14
6.4.1	Detailed Description . . . . .	15
6.4.2	Macro Definition Documentation . . . . .	16

6.4.2.1	<code>xd_utf8str</code>	16
6.4.3	Function Documentation	16
6.4.3.1	<code>berDecCharArray()</code>	16
6.4.3.2	<code>xd_16BitCharStr()</code>	17
6.4.3.3	<code>xd_16BitCharStr64()</code>	18
6.4.3.4	<code>xd_32BitCharStr()</code>	18
6.4.3.5	<code>xd_32BitCharStr64()</code>	19
6.4.3.6	<code>xd_bigint()</code>	20
6.4.3.7	<code>xd_bitstr()</code>	20
6.4.3.8	<code>xd_bitstr64()</code>	21
6.4.3.9	<code>xd_bitstr64_s()</code>	22
6.4.3.10	<code>xd_bitstr64Ext_s()</code>	22
6.4.3.11	<code>xd_bitstr_s()</code>	23
6.4.3.12	<code>xd_bitstrExt_s()</code>	24
6.4.3.13	<code>xd_boolean()</code>	25
6.4.3.14	<code>xd_charstr()</code>	25
6.4.3.15	<code>xd_charstr64()</code>	26
6.4.3.16	<code>xd_chkend()</code>	27
6.4.3.17	<code>xd_chkend64()</code>	27
6.4.3.18	<code>xd_count()</code>	28
6.4.3.19	<code>xd_count64()</code>	28
6.4.3.20	<code>xd_datestr()</code>	29
6.4.3.21	<code>xd_datetimestr()</code>	30
6.4.3.22	<code>xd_durationstr()</code>	30
6.4.3.23	<code>xd_enum()</code>	31
6.4.3.24	<code>xd_enumUnsigned()</code>	32
6.4.3.25	<code>xd_indeflen64()</code>	32
6.4.3.26	<code>xd_indeflen_ex()</code>	33

6.4.3.27	<code>xd_int16()</code>	33
6.4.3.28	<code>xd_int64()</code>	34
6.4.3.29	<code>xd_int8()</code>	35
6.4.3.30	<code>xd_integer()</code>	35
6.4.3.31	<code>xd_len()</code>	36
6.4.3.32	<code>xd_len64()</code>	36
6.4.3.33	<code>xd_match()</code>	37
6.4.3.34	<code>xd_match1()</code>	38
6.4.3.35	<code>xd_match64()</code>	38
6.4.3.36	<code>xd_memcpy()</code>	39
6.4.3.37	<code>xd_NextElement()</code>	40
6.4.3.38	<code>xd_null()</code>	40
6.4.3.39	<code>xd_objid()</code>	40
6.4.3.40	<code>xd_octstr()</code>	41
6.4.3.41	<code>xd_octstr64()</code>	42
6.4.3.42	<code>xd_octstr64_s()</code>	42
6.4.3.43	<code>xd_octstr_s()</code>	43
6.4.3.44	<code>xd_oid64()</code>	44
6.4.3.45	<code>xd_OpenType()</code>	44
6.4.3.46	<code>xd_OpenTypeAppend()</code>	45
6.4.3.47	<code>xd_OpenTypeExt()</code>	45
6.4.3.48	<code>xd_OpenTypeExt64()</code>	46
6.4.3.49	<code>xd_real()</code>	47
6.4.3.50	<code>xd_real10()</code>	47
6.4.3.51	<code>xd_reloid()</code>	48
6.4.3.52	<code>xd_setp()</code>	48
6.4.3.53	<code>xd_setp64()</code>	49
6.4.3.54	<code>xd_tag()</code>	50

6.4.3.55	<code>xd_Tag1AndLen()</code>	51
6.4.3.56	<code>xd_tag_len()</code>	51
6.4.3.57	<code>xd_tag_len_64()</code>	52
6.4.3.58	<code>xd_timeofdaystr()</code>	52
6.4.3.59	<code>xd_timestr()</code>	53
6.4.3.60	<code>xd_uint16()</code>	54
6.4.3.61	<code>xd_uint64()</code>	54
6.4.3.62	<code>xd_uint8()</code>	55
6.4.3.63	<code>xd_unsigned()</code>	55
6.5	BER/DER C File Functions.	57
6.5.1	Detailed Description	57
6.5.2	Function Documentation	57
6.5.2.1	<code>xdf_len()</code>	57
6.5.2.2	<code>xdf_ReadContents()</code>	58
6.5.2.3	<code>xdf_ReadPastEOC()</code>	58
6.5.2.4	<code>xdf_tag()</code>	60
6.5.2.5	<code>xdf_TagAndLen()</code>	61
6.6	BER/DER C Encode Functions.	62
6.6.1	Detailed Description	63
6.6.2	Macro Definition Documentation	63
6.6.2.1	<code>xe_utf8str</code>	63
6.6.3	Function Documentation	64
6.6.3.1	<code>derEncBitString()</code>	64
6.6.3.2	<code>xe_16BitCharStr()</code>	64
6.6.3.3	<code>xe_32BitCharStr()</code>	65
6.6.3.4	<code>xe_bigint()</code>	66
6.6.3.5	<code>xe_bitstr()</code>	66
6.6.3.6	<code>xe_bitstrExt()</code>	67



6.6.3.7	<code>xe_boolean()</code>	68
6.6.3.8	<code>xe_charstr()</code>	68
6.6.3.9	<code>xe_datestr()</code>	69
6.6.3.10	<code>xe_datetimestr()</code>	69
6.6.3.11	<code>xe_derCanonicalSort()</code>	70
6.6.3.12	<code>xe_derCanSortSet()</code>	71
6.6.3.13	<code>xe_derReal10()</code>	71
6.6.3.14	<code>xe_durationstr()</code>	72
6.6.3.15	<code>xe_enum()</code>	72
6.6.3.16	<code>xe_enumUnsigned()</code>	73
6.6.3.17	<code>xe_expandBuffer()</code>	73
6.6.3.18	<code>xe_free()</code>	74
6.6.3.19	<code>xe_getBufLocDescr()</code>	74
6.6.3.20	<code>xe_getp()</code>	75
6.6.3.21	<code>xe_identifier()</code>	75
6.6.3.22	<code>xe_int16()</code>	76
6.6.3.23	<code>xe_int64()</code>	76
6.6.3.24	<code>xe_int8()</code>	77
6.6.3.25	<code>xe_integer()</code>	77
6.6.3.26	<code>xe_len()</code>	78
6.6.3.27	<code>xe_len64()</code>	78
6.6.3.28	<code>xe_memcpy()</code>	79
6.6.3.29	<code>xe_null()</code>	79
6.6.3.30	<code>xe_objid()</code>	80
6.6.3.31	<code>xe_octstr()</code>	80
6.6.3.32	<code>xe_oid64()</code>	81
6.6.3.33	<code>xe_OpenType()</code>	81
6.6.3.34	<code>xe_OpenTypeExt()</code>	82

6.6.3.35	xe_OpenTypeExtDer()	82
6.6.3.36	xe_real()	82
6.6.3.37	xe_real10()	83
6.6.3.38	xe_reloid()	84
6.6.3.39	xe_setp()	84
6.6.3.40	xe_tag()	85
6.6.3.41	xe_tag_len()	85
6.6.3.42	xe_TagAndIndefLen()	86
6.6.3.43	xe_timeofdaystr()	86
6.6.3.44	xe_timestr()	87
6.6.3.45	xe_uint16()	87
6.6.3.46	xe_uint64()	88
6.6.3.47	xe_uint8()	88
6.6.3.48	xe_unsigned()	89
6.7	BER/PER C Utility Functions	90
6.7.1	Detailed Description	91
6.7.2	Macro Definition Documentation	91
6.7.2.1	xu_hex_dump	91
6.7.3	Function Documentation	91
6.7.3.1	berDefToIndefLen()	91
6.7.3.2	berErrAddTagParm()	92
6.7.3.3	berErrUnexpTag()	92
6.7.3.4	berGetLibInfo()	93
6.7.3.5	berGetLibVersion()	93
6.7.3.6	berIndefToDefLen()	93
6.7.3.7	berParseTagLen()	93
6.7.3.8	berReadMsgFromSocket()	94
6.7.3.9	berTagToDynStr()	95

6.7.3.10	berTagToString()	95
6.7.3.11	berValidateIso8601DateStr()	96
6.7.3.12	berValidateIso8601DurationStr()	96
6.7.3.13	berValidateIso8601TimeStr()	97
6.7.3.14	xu_alloc_array()	97
6.7.3.15	xu_dump()	98
6.7.3.16	xu_dump2()	98
6.7.3.17	xu_fdump()	99
6.7.3.18	xu_RestoreBufferState()	99
6.7.3.19	xu_SaveBufferState()	100
6.8	Streaming BER Runtime Library Functions.	101
6.8.1	Detailed Description	101
6.8.2	Macro Definition Documentation	101
6.8.2.1	BS_CHKEND	101
6.8.2.2	BS_CHKEOB	102
6.8.2.3	cerAddBufLocDescr	102
6.8.2.4	cerEncCanonicalSort	102
6.8.2.5	cerGetBufLocDescr	102
6.8.3	Function Documentation	102
6.8.3.1	berStrmFreeContext()	102
6.8.3.2	berStrmInitContext()	103
6.9	C Streaming BER Encode Functions.	104
6.9.1	Detailed Description	105
6.9.2	Function Documentation	105
6.9.2.1	berEncStrmBigInt()	105
6.9.2.2	berEncStrmBigIntNchars()	105
6.9.2.3	berEncStrmBitStr()	106
6.9.2.4	berEncStrmBMPStr()	107

6.9.2.5	berEncStrmBool()	107
6.9.2.6	berEncStrmCharStr()	108
6.9.2.7	berEncStrmDateStr()	108
6.9.2.8	berEncStrmDateTimeStr()	109
6.9.2.9	berEncStrmDefLength()	110
6.9.2.10	berEncStrmDurationStr()	110
6.9.2.11	berEncStrmEnum()	111
6.9.2.12	berEncStrmEOC()	111
6.9.2.13	berEncStrmInt()	112
6.9.2.14	berEncStrmInt16()	112
6.9.2.15	berEncStrmInt64()	113
6.9.2.16	berEncStrmInt8()	113
6.9.2.17	berEncStrmLength()	114
6.9.2.18	berEncStrmNull()	114
6.9.2.19	berEncStrmObjId()	115
6.9.2.20	berEncStrmObjId64()	115
6.9.2.21	berEncStrmOctStr()	116
6.9.2.22	berEncStrmOpenTypeExt()	116
6.9.2.23	berEncStrmReal()	117
6.9.2.24	berEncStrmReal10()	117
6.9.2.25	berEncStrmRelativeOID()	118
6.9.2.26	berEncStrmTag()	118
6.9.2.27	berEncStrmTagAndDefLen()	119
6.9.2.28	berEncStrmTagAndIndefLen()	119
6.9.2.29	berEncStrmTagAndLen()	120
6.9.2.30	berEncStrmTimeOfDayStr()	120
6.9.2.31	berEncStrmTimeStr()	121
6.9.2.32	berEncStrmUInt()	121

6.9.2.33	berEncStrmUInt16()	122
6.9.2.34	berEncStrmUInt64()	123
6.9.2.35	berEncStrmUInt8()	123
6.9.2.36	berEncStrmUnivStr()	124
6.9.2.37	berEncStrmWriteOctet()	124
6.9.2.38	berEncStrmWriteOctets()	125
6.9.2.39	berEncStrmXSDAny()	125
6.9.2.40	cerEncStrmBitStr()	126
6.9.2.41	cerEncStrmBMPStr()	126
6.9.2.42	cerEncStrmCharStr()	127
6.9.2.43	cerEncStrmOctStr()	127
6.9.2.44	cerEncStrmReal10()	128
6.9.2.45	cerEncStrmUnivStr()	129
6.10	C Streaming BER Decode Functions.	130
6.10.1	Detailed Description	131
6.10.2	Function Documentation	131
6.10.2.1	berDecStrmBigInt()	131
6.10.2.2	berDecStrmBitStr()	132
6.10.2.3	berDecStrmBMPStr()	133
6.10.2.4	berDecStrmBool()	133
6.10.2.5	berDecStrmCharStr()	134
6.10.2.6	berDecStrmCheckEnd()	135
6.10.2.7	berDecStrmDateStr()	135
6.10.2.8	berDecStrmDateTimeStr()	136
6.10.2.9	berDecStrmDurationStr()	137
6.10.2.10	berDecStrmDynBitStr()	137
6.10.2.11	berDecStrmDynBitStr64()	138
6.10.2.12	berDecStrmDynOctStr()	139

6.10.2.13	<code>berDecStrmDynOctStr64()</code>	139
6.10.2.14	<code>berDecStrmEnum()</code>	140
6.10.2.15	<code>berDecStrmFindTag()</code>	141
6.10.2.16	<code>berDecStrmFindTag2()</code>	141
6.10.2.17	<code>berDecStrmGetTLVLength()</code>	142
6.10.2.18	<code>berDecStrmInt()</code>	142
6.10.2.19	<code>berDecStrmInt16()</code>	143
6.10.2.20	<code>berDecStrmInt64()</code>	144
6.10.2.21	<code>berDecStrmInt8()</code>	144
6.10.2.22	<code>berDecStrmLength()</code>	145
6.10.2.23	<code>berDecStrmLength2()</code>	145
6.10.2.24	<code>berDecStrmMatchEOC()</code>	146
6.10.2.25	<code>berDecStrmMatchTag()</code>	146
6.10.2.26	<code>berDecStrmMatchTag2()</code>	147
6.10.2.27	<code>berDecStrmNextElement()</code>	147
6.10.2.28	<code>berDecStrmNull()</code>	148
6.10.2.29	<code>berDecStrmObjId()</code>	148
6.10.2.30	<code>berDecStrmObjId64()</code>	149
6.10.2.31	<code>berDecStrmOctStr()</code>	150
6.10.2.32	<code>berDecStrmOpenType()</code>	150
6.10.2.33	<code>berDecStrmOpenTypeAppend()</code>	151
6.10.2.34	<code>berDecStrmOpenTypeExt()</code>	151
6.10.2.35	<code>berDecStrmPeekTagAndLen()</code>	152
6.10.2.36	<code>berDecStrmReadDynTLV()</code>	152
6.10.2.37	<code>berDecStrmReadTLV()</code>	153
6.10.2.38	<code>berDecStrmReal()</code>	153
6.10.2.39	<code>berDecStrmReal10()</code>	154
6.10.2.40	<code>berDecStrmRelativeOID()</code>	155

6.10.2.41	berDecStrmTag()	155
6.10.2.42	berDecStrmTagAndLen()	156
6.10.2.43	berDecStrmTagAndLen2()	156
6.10.2.44	berDecStrmTestEOC()	157
6.10.2.45	berDecStrmTestTag()	157
6.10.2.46	berDecStrmTimeOfDayStr()	158
6.10.2.47	berDecStrmTimeStr()	158
6.10.2.48	berDecStrmUInt()	160
6.10.2.49	berDecStrmUInt16()	161
6.10.2.50	berDecStrmUInt64()	161
6.10.2.51	berDecStrmUInt8()	162
6.10.2.52	berDecStrmUnivStr()	162
6.11	C++ classes for streaming BER encoding.	164
6.11.1	Detailed Description	164
6.12	C++ classes for streaming BER decoding.	165
6.12.1	Detailed Description	165
<b>7</b>	<b>Class Documentation</b>	<b>167</b>
7.1	ASN1BERDecodeBuffer Class Reference	167
7.1.1	Detailed Description	168
7.1.2	Constructor & Destructor Documentation	168
7.1.2.1	ASN1BERDecodeBuffer() [1/3]	168
7.1.2.2	ASN1BERDecodeBuffer() [2/3]	168
7.1.2.3	ASN1BERDecodeBuffer() [3/3]	169
7.1.3	Member Function Documentation	169
7.1.3.1	findElement()	169
7.1.3.2	getMsgPtr()	170
7.1.3.3	init()	170

7.1.3.4	isA()	170
7.1.3.5	operator>>()	171
7.1.3.6	parseTagLen() [1/3]	171
7.1.3.7	parseTagLen() [2/3]	171
7.1.3.8	parseTagLen() [3/3]	172
7.1.3.9	readBinaryFile()	173
7.1.3.10	setBuffer()	173
7.2	ASN1BERDecodeStream Class Reference	174
7.2.1	Detailed Description	175
7.2.2	Constructor & Destructor Documentation	176
7.2.2.1	ASN1BERDecodeStream()	176
7.2.3	Member Function Documentation	176
7.2.3.1	byteIndex()	176
7.2.3.2	chkend()	176
7.2.3.3	close()	177
7.2.3.4	currentPos()	177
7.2.3.5	decodeBigInt()	177
7.2.3.6	decodeBitStr() [1/2]	178
7.2.3.7	decodeBitStr() [2/2]	179
7.2.3.8	decodeBMPStr()	180
7.2.3.9	decodeBool()	180
7.2.3.10	decodeCharStr()	181
7.2.3.11	decodeEnum()	182
7.2.3.12	decodeEoc()	182
7.2.3.13	decodeInt()	183
7.2.3.14	decodeInt16()	183
7.2.3.15	decodeInt64()	184
7.2.3.16	decodeInt8()	185



7.2.3.17	<code>decodeLength()</code>	186
7.2.3.18	<code>decodeNull()</code>	187
7.2.3.19	<code>decodeObj()</code>	187
7.2.3.20	<code>decodeObjId()</code>	188
7.2.3.21	<code>decodeObjId64()</code>	188
7.2.3.22	<code>decodeOctStr()</code> [1/3]	189
7.2.3.23	<code>decodeOctStr()</code> [2/3]	190
7.2.3.24	<code>decodeOctStr()</code> [3/3]	190
7.2.3.25	<code>decodeOpenType()</code>	191
7.2.3.26	<code>decodeReal()</code>	191
7.2.3.27	<code>decodeRelativeOID()</code>	192
7.2.3.28	<code>decodeTag()</code>	193
7.2.3.29	<code>decodeTagAndLen()</code> [1/2]	193
7.2.3.30	<code>decodeTagAndLen()</code> [2/2]	194
7.2.3.31	<code>decodeUInt()</code>	194
7.2.3.32	<code>decodeUInt16()</code>	195
7.2.3.33	<code>decodeUInt64()</code>	196
7.2.3.34	<code>decodeUInt8()</code>	196
7.2.3.35	<code>decodeUnivStr()</code>	197
7.2.3.36	<code>flush()</code>	198
7.2.3.37	<code>getAppInfo()</code>	198
7.2.3.38	<code>getContext()</code>	198
7.2.3.39	<code>getCtxtPtr()</code>	199
7.2.3.40	<code>getErrorInfo()</code> [1/2]	199
7.2.3.41	<code>getErrorInfo()</code> [2/2]	199
7.2.3.42	<code>getPosition()</code>	200
7.2.3.43	<code>getStatus()</code>	200
7.2.3.44	<code>getTLVLength()</code>	200

7.2.3.45	isA()	201
7.2.3.46	isOpened()	201
7.2.3.47	mark()	201
7.2.3.48	markSupported()	202
7.2.3.49	operator>>()	202
7.2.3.50	peekTagAndLen()	203
7.2.3.51	printErrorInfo()	203
7.2.3.52	read()	203
7.2.3.53	readBlocking()	204
7.2.3.54	readTLV()	204
7.2.3.55	reset()	205
7.2.3.56	resetErrorInfo()	205
7.2.3.57	setAppInfo()	205
7.2.3.58	setDiag()	205
7.2.3.59	setPosition()	206
7.2.3.60	skip()	206
7.3	ASN1BEREncodeBuffer Class Reference	207
7.3.1	Detailed Description	207
7.3.2	Constructor & Destructor Documentation	208
7.3.2.1	ASN1BEREncodeBuffer() [1/3]	208
7.3.2.2	ASN1BEREncodeBuffer() [2/3]	208
7.3.2.3	ASN1BEREncodeBuffer() [3/3]	208
7.3.3	Member Function Documentation	209
7.3.3.1	encodeBigInt()	209
7.3.3.2	encodeBigIntNchars()	209
7.3.3.3	encodeBool()	210
7.3.3.4	encodeObjId()	210
7.3.3.5	freeBuffer()	211

7.3.3.6	<code>getMsgCopy()</code>	211
7.3.3.7	<code>getMsgLen()</code>	212
7.3.3.8	<code>getMsgPtr()</code>	212
7.3.3.9	<code>init()</code>	212
7.3.3.10	<code>isA()</code>	212
7.3.3.11	<code>operator&lt;&lt;()</code>	213
7.3.3.12	<code>setBuffer()</code>	213
7.4	<b>ASN1BEREncodeStream Class Reference</b>	214
7.4.1	Detailed Description	215
7.4.2	Constructor & Destructor Documentation	215
7.4.2.1	<code>ASN1BEREncodeStream()</code>	215
7.4.3	Member Function Documentation	215
7.4.3.1	<code>close()</code>	216
7.4.3.2	<code>encodeBigInt()</code>	216
7.4.3.3	<code>encodeBigIntNchars()</code>	217
7.4.3.4	<code>encodeBitStr()</code> [1/2]	217
7.4.3.5	<code>encodeBitStr()</code> [2/2]	218
7.4.3.6	<code>encodeBMPStr()</code>	218
7.4.3.7	<code>encodeBool()</code>	219
7.4.3.8	<code>encodeCharStr()</code>	220
7.4.3.9	<code>encodeEnum()</code>	220
7.4.3.10	<code>encodeEoc()</code>	221
7.4.3.11	<code>encodeIndefLen()</code>	221
7.4.3.12	<code>encodeInt()</code>	221
7.4.3.13	<code>encodeInt16()</code>	222
7.4.3.14	<code>encodeInt64()</code>	223
7.4.3.15	<code>encodeInt8()</code>	223
7.4.3.16	<code>encodeLen()</code>	224

7.4.3.17	<code>encodeNull()</code>	224
7.4.3.18	<code>encodeObj()</code>	225
7.4.3.19	<code>encodeObjId()</code>	225
7.4.3.20	<code>encodeObjId64()</code>	226
7.4.3.21	<code>encodeOctStr()</code> [1/2]	226
7.4.3.22	<code>encodeOctStr()</code> [2/2]	227
7.4.3.23	<code>encodeReal()</code>	228
7.4.3.24	<code>encodeRelativeOID()</code>	228
7.4.3.25	<code>encodeTag()</code>	229
7.4.3.26	<code>encodeTagAndIndefLen()</code>	229
7.4.3.27	<code>encodeTagAndLen()</code>	230
7.4.3.28	<code>encodeUInt()</code>	231
7.4.3.29	<code>encodeUInt16()</code>	231
7.4.3.30	<code>encodeUInt64()</code>	232
7.4.3.31	<code>encodeUInt8()</code>	232
7.4.3.32	<code>encodeUnivStr()</code>	233
7.4.3.33	<code>flush()</code>	234
7.4.3.34	<code>getAppInfo()</code>	234
7.4.3.35	<code>getContext()</code>	234
7.4.3.36	<code>getCtxtPtr()</code>	234
7.4.3.37	<code>getErrorInfo()</code> [1/2]	235
7.4.3.38	<code>getErrorInfo()</code> [2/2]	235
7.4.3.39	<code>getStatus()</code>	235
7.4.3.40	<code>isA()</code>	236
7.4.3.41	<code>isOpened()</code>	236
7.4.3.42	<code>operator&lt;&lt;()</code>	236
7.4.3.43	<code>printErrorInfo()</code>	237
7.4.3.44	<code>resetErrorInfo()</code>	237

7.4.3.45	setAppInfo()	237
7.4.3.46	setDiag()	237
7.4.3.47	write()	238
7.5	ASN1BERLength Class Reference	238
7.6	ASN1BERMessageBuffer Class Reference	239
7.6.1	Detailed Description	239
7.6.2	Constructor & Destructor Documentation	239
7.6.2.1	ASN1BERMessageBuffer() [1/2]	239
7.6.2.2	ASN1BERMessageBuffer() [2/2]	240
7.6.3	Member Function Documentation	240
7.6.3.1	binDump()	240
7.6.3.2	calcIndefLen() [1/2]	241
7.6.3.3	calcIndefLen() [2/2]	241
7.6.3.4	hexDump()	242
<b>8</b>	<b>File Documentation</b>	<b>243</b>
8.1	asn1ber.h File Reference	243
8.1.1	Detailed Description	247
8.2	asn1BerCppType.h File Reference	247
8.2.1	Detailed Description	247
8.3	ASN1BERDecodeStream.h File Reference	247
8.3.1	Detailed Description	248
8.4	ASN1BEREncodeStream.h File Reference	248
8.4.1	Detailed Description	248
8.5	asn1berSocket.h File Reference	248
8.6	asn1berStream.h File Reference	248
8.6.1	Detailed Description	251
<b>Index</b>		<b>253</b>



## Chapter 1

# ASN1C BER/DER Runtime Classes and Library Functions

The **ASN.1 C++ Runtime Classes** are wrapper classes that provide an object-oriented interface to the ASN.1 C runtime library functions. The classes described in this manual are derived from the common classes documented in the ASN1C C/C++ Common Runtime manual. They are specific to the Basic Encoding Rules (BER) and Distinguished Encoding Rules (DER) as defined in the ITU-T X.690 standard.

These BER/DER specific C++ runtime classes include:

- classes for streaming BER/DER decoding
- classes for streaming BER/DER encoding.

The **ASN.1 BER Runtime Library** contains all of the low-level constants, types, and functions that are assembled by the compiler to encode/decode more complex structures.

This library consists of the following two items:

- A global include file ("asn1ber.h") that is compiled into all generated source files.
- An object library of functions that are linked with the C functions after compilation with a C compiler.

In general, programmers will not need to be too concerned with the details of these functions. The ASN.1 compiler generates calls to them in the C or C++ source files that it creates. However, the functions in the library may also be called on their own in applications requiring their specific functionality.





# Chapter 2

## Module Index

### 2.1 Modules

Here is a list of all modules:

BER/DER/CER C++ Run-Time Classes. . . . .	11
BER Message Buffer Classes . . . . .	12
C++ classes for streaming BER encoding. . . . .	164
C++ classes for streaming BER decoding. . . . .	165
BER Runtime Library Functions. . . . .	13
BER/DER C Decode Functions. . . . .	14
BER/DER C File Functions. . . . .	57
BER/DER C Encode Functions. . . . .	62
BER/PER C Utility Functions . . . . .	90
Streaming BER Runtime Library Functions. . . . .	101
C Streaming BER Encode Functions. . . . .	104
C Streaming BER Decode Functions. . . . .	130



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

ASN1BERLength . . . . .	238
ASN1MessageBuffer	
ASN1BERDecodeStream . . . . .	174
ASN1BEREncodeStream . . . . .	214
ASN1BERMessageBuffer . . . . .	239
ASN1BERDecodeBuffer . . . . .	167
ASN1BEREncodeBuffer . . . . .	207



# Chapter 4

## Class Index

### 4.1 Class List

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

<a href="#">ASN1BERDecodeBuffer</a>	167
<a href="#">ASN1BERDecodeStream</a>	174
<a href="#">ASN1BEREncodeBuffer</a>	207
<a href="#">ASN1BEREncodeStream</a>	214
<a href="#">ASN1BERLength</a>	238
<a href="#">ASN1BERMessageBuffer</a>	239



# Chapter 5

## File Index

### 5.1 File List

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

<a href="#">asn1ber.h</a>	243
<a href="#">asn1BerCppType.h</a>	247
<a href="#">ASN1BERDecodeStream.h</a>	247
<a href="#">ASN1BEREncodeStream.h</a>	248
<a href="#">asn1berSocket.h</a>	248
<a href="#">asn1berStream.h</a>	248





## Chapter 6

# Module Documentation

### 6.1 BER/DER/CER C++ Run-Time Classes.

#### Modules

- [BER Message Buffer Classes](#)
- [C++ classes for streaming BER encoding.](#)
- [C++ classes for streaming BER decoding.](#)

#### 6.1.1 Detailed Description

## 6.2 BER Message Buffer Classes

### Classes

- class [ASN1BERLength](#)
- class [ASN1BERMessageBuffer](#)
- class [ASN1BEREncodeBuffer](#)
- class [ASN1BERDecodeBuffer](#)

### 6.2.1 Detailed Description

These classes manage the buffers for encoding and decoding ASN.1 BER/DER messages.

## 6.3 BER Runtime Library Functions.

### Modules

- [BER/DER C Decode Functions.](#)
- [BER/DER C File Functions.](#)
- [BER/DER C Encode Functions.](#)
- [BER/PER C Utility Functions](#)
- [Streaming BER Runtime Library Functions.](#)

### Typedefs

- typedef OSRTBufLocDescr **Asn1BufLocDescr**

### Functions

- int **xd\_MovePastEOC** (OSCTXT \*pctxt)
- int **xd\_consStrInDefLenAndSize** (OSCTXT \*pctxt, ASN1TAG expectedTag, OSSIZE \*length, OSSIZE \*size)

#### 6.3.1 Detailed Description

The ASN.1 Basic Encoding Rules (BER) Runtime Library contains the low-level constants, types, and functions that are assembled by the compiler to encode/decode more complex structures.

## 6.4 BER/DER C Decode Functions.

### Macros

- #define `xd_utf8str`(pctxt, object\_p, tagging, length) `xd_charstr` (pctxt, (const char\*\*)object\_p, tagging, ASN\_ID↔\_UTF8String, length)
- #define `xd_indeflen`(m) `xd_indeflen_ex`(m, INT\_MAX)

### Functions

- int `xd_tag` (OSCTXT \*pctxt, ASN1TAG \*tag\_p)
- int `xd_tag_len` (OSCTXT \*pctxt, ASN1TAG \*tag\_p, int \*len\_p, OSOCTET flags)
- int `xd_tag_len_64` (OSCTXT \*pctxt, ASN1TAG \*tag\_p, OSSIZE \*len\_p, OSBOOL \*pIndefLen, OSOCTET flags)
- int `xd_match` (OSCTXT \*pctxt, ASN1TAG tag, int \*len\_p, OSOCTET flags)
- int `xd_match64` (OSCTXT \*pctxt, ASN1TAG tag, OSSIZE \*len\_p, OSBOOL \*pindef, OSOCTET flags)
- int `xd_boolean` (OSCTXT \*pctxt, OSBOOL \*object\_p, ASN1TagType tagging, int length)
- int `xd_integer` (OSCTXT \*pctxt, OSINT32 \*object\_p, ASN1TagType tagging, int length)
- int `xd_int8` (OSCTXT \*pctxt, OSINT8 \*object\_p, ASN1TagType tagging, int length)
- int `xd_int16` (OSCTXT \*pctxt, OSINT16 \*object\_p, ASN1TagType tagging, int length)
- int `xd_unsigned` (OSCTXT \*pctxt, OSUINT32 \*object\_p, ASN1TagType tagging, int length)
- int `xd_uint8` (OSCTXT \*pctxt, OSUINT8 \*object\_p, ASN1TagType tagging, int length)
- int `xd_uint16` (OSCTXT \*pctxt, OSUINT16 \*object\_p, ASN1TagType tagging, int length)
- int `xd_int64` (OSCTXT \*pctxt, OSINT64 \*object\_p, ASN1TagType tagging, int length)
- int `xd_uint64` (OSCTXT \*pctxt, OSUINT64 \*object\_p, ASN1TagType tagging, int length)
- int `xd_bigint` (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, int length)
- int `xd_bitstr_s` (OSCTXT \*pctxt, OSOCTET \*object\_p, OSUINT32 \*numbits\_p, ASN1TagType tagging, int length)
- int `xd_bitstrExt_s` (OSCTXT \*pctxt, OSOCTET \*object\_p, OSUINT32 \*numbits\_p, OSOCTET \*\*extdata, AS↔N1TagType tagging, int length)
- int `xd_bitstr64_s` (OSCTXT \*pctxt, OSOCTET \*object\_p, OSSIZE \*numbits\_p, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int `xd_bitstr64Ext_s` (OSCTXT \*pctxt, OSOCTET \*object\_p, OSSIZE \*numbits\_p, OSOCTET \*\*extdata, AS↔N1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int `xd_bitstr` (OSCTXT \*pctxt, const OSOCTET \*\*object\_p2, OSUINT32 \*numbits\_p, ASN1TagType tagging, int length)
- int `xd_bitstr64` (OSCTXT \*pctxt, const OSOCTET \*\*object\_p2, OSSIZE \*numbits\_p, ASN1TagType tagging, O↔SSIZE length, OSBOOL indefLen)
- int `xd_octstr_s` (OSCTXT \*pctxt, OSOCTET \*object\_p, OSUINT32 \*pnumocts, ASN1TagType tagging, int length)
- int `xd_octstr64_s` (OSCTXT \*pctxt, OSOCTET \*object\_p, OSSIZE \*pnumocts, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int `xd_octstr` (OSCTXT \*pctxt, const OSOCTET \*\*object\_p2, OSUINT32 \*pnumocts, ASN1TagType tagging, int length)
- int `xd_octstr64` (OSCTXT \*pctxt, OSOCTET \*\*object\_p2, OSSIZE \*pnumocts, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int `xd_charstr` (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int `xd_charstr64` (OSCTXT \*pctxt, char \*\*object\_p, ASN1TagType tagging, ASN1TAG tag, OSSIZE length, O↔SBOOL indefLen)
- int `xd_datestr` (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int `xd_timestr` (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int `xd_datetimestr` (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int `xd_timeofdaystr` (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, ASN1TAG tag, int length)

- int `xd_durationstr` (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int `berDecCharArray` (OSCTXT \*pctxt, char \*charArray, OSSIZE arraySize, ASN1TagType tagging, ASN1TAG tag, int length)
- int `xd_16BitCharStr` (OSCTXT \*pctxt, Asn116BitCharString \*object\_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int `xd_16BitCharStr64` (OSCTXT \*pctxt, Asn116BitCharString \*object\_p, ASN1TagType tagging, ASN1TAG tag, OSSIZE length, OSBOOL indefLen)
- int `xd_32BitCharStr` (OSCTXT \*pctxt, Asn132BitCharString \*object\_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int `xd_32BitCharStr64` (OSCTXT \*pctxt, Asn132BitCharString \*object\_p, ASN1TagType tagging, ASN1TAG tag, OSSIZE length, OSBOOL indefLen)
- int `xd_null` (OSCTXT \*pctxt, ASN1TagType tagging)
- int `xd_objid` (OSCTXT \*pctxt, ASN1OBJID \*object\_p, ASN1TagType tagging, int length)
- int `xd_oid64` (OSCTXT \*pctxt, ASN1OID64 \*object\_p, ASN1TagType tagging, int length)
- int `xd_reloid` (OSCTXT \*pctxt, ASN1OBJID \*object\_p, ASN1TagType tagging, int length)
- int `xd_real` (OSCTXT \*pctxt, OSREAL \*object\_p, ASN1TagType tagging, int length)
- int `xd_enum` (OSCTXT \*pctxt, OSINT32 \*object\_p, ASN1TagType tagging, int length)
- int `xd_enumUnsigned` (OSCTXT \*pctxt, OSUINT32 \*object\_p, ASN1TagType tagging, int length)
- int `xd_OpenType` (OSCTXT \*pctxt, const OSOCTET \*\*object\_p2, OSSIZE \*pnumocts)
- int `xd_OpenTypeExt` (OSCTXT \*pctxt, ASN1CCB \*ccb\_p, ASN1TAG \*tags, int tagCount, OSRTDList \*pElemList)
- int `xd_OpenTypeExt64` (OSCTXT \*pctxt, const OSOCTET \*conspr, OSSIZE conslen, OSBOOL indefLen, ASN1TAG \*tags, OSSIZE tagCount, OSRTDList \*pElemList)
- int `xd_OpenTypeAppend` (OSCTXT \*pctxt, OSRTDList \*pElemList)
- int `xd_real10` (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, int length)
- int `xd_setp` (OSCTXT \*pctxt, const OSOCTET \*msg\_p, int msglen, ASN1TAG \*tag\_p, int \*len\_p)
- int `xd_setp64` (OSCTXT \*pctxt, const OSOCTET \*msg\_p, OSSIZE msglen, ASN1TAG \*tag\_p, OSSIZE \*len\_p, OSBOOL \*pIndefLen)
- int `xd_indeflen_ex` (const OSOCTET \*msg\_p, int bufSize)
- int `xd_indeflen64` (const OSOCTET \*msg\_p, OSSIZE bufSize, OSSIZE \*plength)
- int `xd_len` (OSCTXT \*pctxt, int \*len\_p)
- int `xd_len64` (OSCTXT \*pctxt, OSSIZE \*len\_p, OSBOOL \*p indef)
- OSBOOL `xd_chkend` (OSCTXT \*pctxt, const ASN1CCB \*ccb\_p)
- OSBOOL `xd_chkend64` (OSCTXT \*pctxt, const OSOCTET \*conspr, OSSIZE conslen, OSBOOL indef)
- int `xd_count` (OSCTXT \*pctxt, int length, int \*count\_p)
- int `xd_count64` (OSCTXT \*pctxt, OSSIZE length, OSBOOL indefLen, OSSIZE \*count\_p)
- int `xd_NextElement` (OSCTXT \*pctxt)
- int `xd_Tag1AndLen` (OSCTXT \*pctxt, OSINT32 \*len\_p)
- int `xd_memcpy` (OSCTXT \*pctxt, OSOCTET \*object\_p, int length)
- int `xd_match1` (OSCTXT \*pctxt, OSOCTET tag, int \*len\_p)
- int `xd_match1_64` (OSCTXT \*pctxt, OSOCTET tag, OSSIZE \*len\_p, OSBOOL \*p indef)

### 6.4.1 Detailed Description

BER/DER C decode functions handle the decoding of the primitive ASN.1 data types and ASN.1 length and tag fields within a message. Calls to these functions are assembled in the C source code generated by the ASN1C compiler to decode complex ASN.1 structures. These functions are also directly callable from within a user's application program if the need to decode a primitive data item exists.

The procedure to decode a primitive data item is as follows:

1. Call the `xd_setp` low-level decode function to specify the address of the buffer containing the encoded ASN.1 data to be decoded.
2. Call the specific decode function to decode the value. The tag value obtained in the first step can be used to determine which decode function to call for decoding the variable.

## 6.4.2 Macro Definition Documentation

### 6.4.2.1 `xd_utf8str`

```
#define xd_utf8str(
    pctxt,
    object_p,
    tagging,
    length ) xd_charstr (pctxt, (const char**)object_p, tagging, ASN_ID_UTF8String,
length)
```

This function is used to decode a variable of the ASN.1 UTF-8 string type. This function allocates memory for the decoded string and returns a pointer to the data.

#### Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>object_p</i>	Pointer to a pointer to receive the address of the allocated memory into which the decoded character string data will be stored. The string is assumed to be a normal C string containing non-null characters. A null terminator is automatically added to the end of the string by this function.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

#### Returns

Completion status of operation:

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

## 6.4.3 Function Documentation

### 6.4.3.1 `berDecCharArray()`

```
int berDecCharArray (
    OSCTXT * pctxt,
```

```

char * charArray,
OSSIZE arraySize,
ASN1TagType tagging,
ASN1TAG tag,
int length )

```

This function is the base function for decoding any of the 8-bit character string useful types such as IA5String, VisibleString, etc. This function decodes the character string into a static array variable.

#### Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>charArray</i>	Static character array variable (char[]) large enough to hold decoded data plus a null-termination byte. If the array is not large enough, an RTERR_TOOBIG error status will be returned.
<i>arraySize</i>	Size (in bytes) of the charArray variable.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>tag</i>	Tag variable to match
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

#### Returns

Completion status of operation:

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

#### 6.4.3.2 xd\_16BitCharStr()

```

int xd_16BitCharStr (
    OSCTXT * pctxt,
    Asn116BitCharString * object_p,
    ASN1TagType tagging,
    ASN1TAG tag,
    int length )

```

This function is the base function for decoding a 16-bit character string useful type. In the current version of ASN.1, the only string type based on 16-bit Unicode characters is the UniCharString type. This function allocates memory for the decoded string and returns a pointer to the data.

#### Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>object_p</i>	Pointer to a pointer to receive the address of the allocated memory into which the decoded character string data will be stored. The string is assumed to contain all non-null 16-bit unicode characters. A null terminator is automatically added to the end of the string by this function.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>tag</i>	Tag variable to match
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

## Returns

Completion status of operation:

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

### 6.4.3.3 xd\_16BitCharStr64()

```
int xd_16BitCharStr64 (  
    OSCTXT * pctxt,  
    Asn116BitCharString * object_p,  
    ASN1TagType tagging,  
    ASN1TAG tag,  
    OSSIZE length,  
    OSBOOL indefLen )
```

This function is the base function for decoding a 16-bit character string useful type. In the current version of ASN.1, the only string type based on 16-bit Unicode characters is the UniCharString type. This function allocates memory for the decoded string and returns a pointer to the data.

This version of the function supports full 64-bit lengths.

## Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>object_p</i>	Pointer to a pointer to receive the address of the allocated memory into which the decoded character string data will be stored. The string is assumed to contain all non-null 16-bit unicode characters. A null terminator is automatically added to the end of the string by this function.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>tag</i>	Tag variable to match
<i>length</i>	Length of data to retrieve. Valid for implicit case only.
<i>indefLen</i>	Flag indicating length of string is indefinite.

## Returns

Completion status of operation:

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

### 6.4.3.4 xd\_32BitCharStr()

```
int xd_32BitCharStr (  
    OSCTXT * pctxt,
```



```

Asn132BitCharString * object_p,
ASN1TagType tagging,
ASN1TAG tag,
int length )

```

This function is the base function for decoding a 32-bit character string useful type. In the current version of ASN.1, the only string type based on 32-bit characters is the 32BitCharString type. This function allocates memory for the decoded string and returns a pointer to the data.

#### Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>object_p</i>	Pointer to a pointer to receive the address of the allocated memory into which the decoded character string data will be stored. The string is assumed to contain all non-null 32-bit unicode characters. A null terminator is automatically added to the end of the string by this function.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>tag</i>	Tag variable to match.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

#### Returns

Completion status of operation:

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

#### 6.4.3.5 xd\_32BitCharStr64()

```

int xd_32BitCharStr64 (
    OSCTXT * pctxt,
    Asn132BitCharString * object_p,
    ASN1TagType tagging,
    ASN1TAG tag,
    OSSIZE length,
    OSBOOL indefLen )

```

64-bit version of xd\_32BitCharStr.

#### Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>object_p</i>	Pointer to a pointer to receive the address of the allocated memory into which the decoded character string data will be stored. The string is assumed to contain all non-null 32-bit unicode characters. A null terminator is automatically added to the end of the string by this function.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>tag</i>	Tag variable to match.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.
<i>indefLen</i>	Indefinite length indicator.

## Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error. `xd_32BitCharStr`

### 6.4.3.6 `xd_bigint()`

```
int xd_bigint (
    OSCTXT * pctxt,
    const char ** object_p,
    ASN1TagType tagging,
    int length )
```

This function parses an ASN.1 INTEGER tag/length/value at the current message pointer location and advances the pointer to the next field.

This function will decode a variable of the ASN.1 INTEGER type. In this case, the integer is assumed to be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits). For example, parameters used to calculate security values are typically larger than these sizes. These variables are stored in character string constant variables. They are represented as hexadecimal strings with prefix '0x'. A leading zero is also added to positive values where the leading digit would otherwise be greater than 8 (i.e. representing a negative value in two's complement). If it is necessary to convert a hexadecimal string to another radix, then use `::rtxBigIntSetStr` / `::rtxBigIntToString` functions.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.
<i>object_↔ _p</i>	Pointer to a character pointer variable to receive the decoded hexadecimal value. Dynamic memory is allocated for the variable using the <code>rtxMemAlloc</code> function. The decoded variable is represented as a hexadecimal string with prefix '0x'.

## Returns

Completion status of operation:

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

### 6.4.3.7 `xd_bitstr()`

```
int xd_bitstr (
    OSCTXT * pctxt,
```

```

const OSOCTET ** object_p2,
OSUINT32 * numbits_p,
ASN1TagType tagging,
int length )

```

This function decodes a variable of the ASN.1 BIT STRING. This function will allocate dynamic memory to store the decoded result.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	The length, in OCTETs, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.
<i>numbits↔ _p</i>	Pointer to an integer value to receive the decoded number of bits.
<i>object_p2</i>	Pointer to a pointer variable to receive the decoded bit string. Dynamic memory is allocated to hold the string. Pointer to a variable to receive the decoded bit string.

#### Returns

Completion status of operation:

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

#### 6.4.3.8 xd\_bitstr64()

```

int xd_bitstr64 (
    OSCTXT * pctxt,
    const OSOCTET ** object_p2,
    OSSIZE * numbits_p,
    ASN1TagType tagging,
    OSSIZE length,
    OSBOOL indefLen )

```

This function decodes a variable of the ASN.1 BIT STRING. This function will allocate dynamic memory to store the decoded result. It support 64-bit lengths.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	The length, in OCTETs, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.
<i>numbits↔ _p</i>	Pointer to an integer value to receive the decoded number of bits.
<i>object_p2</i>	Pointer to a pointer variable to receive the decoded bit string. Dynamic memory is allocated to hold the string. Pointer to a variable to receive the decoded bit string.
<i>indefLen</i>	Boolean flag indicating length is indefinite.

## Returns

Completion status of operation:

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

### 6.4.3.9 xd\_bitstr64\_s()

```
int xd_bitstr64_s (
    OSCTXT * pctxt,
    OSOCTET * object_p,
    OSSIZE * numbits_p,
    ASN1TagType tagging,
    OSSIZE length,
    OSBOOL indefLen )
```

This function decodes a variable of the ASN.1 BIT STRING type into a static memory structure. This function call is generated by ASN1C to decode a sized bit string production. This version supports 64-bit lengths.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>object_p</i>	Pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of bits specified in the *numbits_p input parameter.
<i>numbits_p</i>	Pointer to an integer variable containing the size (in bits) of the sized ASN.1 bit string. An error will occur if the number of bits in the decoded string is larger than this value. Note that this is also used as an output variable - the actual number of decoded bits will be returned in this variable.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	The length, in OCTETs, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.
<i>indefLen</i>	Boolean flag indicating length is indefinite.

## Returns

Completion status of operation:

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

### 6.4.3.10 xd\_bitstr64Ext\_s()

```
int xd_bitstr64Ext_s (
    OSCTXT * pctxt,
```

```

OSOCKET * object_p,
OSSIZE * numbits_p,
OSOCKET ** extdata,
ASN1TagType tagging,
OSSIZE length,
OSBOOL indefLen )

```

This function decodes a variable of the ASN.1 BIT STRING type into a static memory structure. This function call is generated by ASN1C to decode a sized bit string production. It includes support for BIT STRING's with extension data. This version supports 64-bit lengths.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>object_p</i>	Pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of bits specified in the *numbits_p input parameter.
<i>numbits_p</i>	Pointer to an integer variable containing the size (in bits) of the sized ASN.1 bit string. An error will occur if the number of bits in the decoded string is larger than this value. Note that this is also used as an output variable - the actual number of decoded bits will be returned in this variable.
<i>extdata</i>	Pointer to byte array containing extension data.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	The length, in OCTETs, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.
<i>indefLen</i>	Boolean flag indicating length is indefinite.

#### Returns

Completion status of operation:

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

#### 6.4.3.11 xd\_bitstr\_s()

```

int xd_bitstr_s (
    OSCTXT * pctxt,
    OSOCKET * object_p,
    OSUINT32 * numbits_p,
    ASN1TagType tagging,
    int length )

```

This function decodes a variable of the ASN.1 BIT STRING type into a static memory structure. This function call is generated by ASN1C to decode a sized bit string production.

#### Parameters

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

## Parameters

<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	The length, in OCTETs, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.
<i>numbits</i> ↔ <i>_p</i>	Pointer to an integer variable containing the size (in bits) of the sized ASN.1 bit string. An error will occur if the number of bits in the decoded string is larger than this value. Note that this is also used as an output variable - the actual number of decoded bits will be returned in this variable.
<i>object_p</i>	Pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of bits specified in the *numbits_p input parameter.

## Returns

Completion status of operation:

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

### 6.4.3.12 xd\_bitstrExt\_s()

```
int xd_bitstrExt_s (
    OSCTXT * pctxt,
    OSOCTET * object_p,
    OSUINT32 * numbits_p,
    OSOCTET ** extdata,
    ASN1TagType tagging,
    int length )
```

This function decodes a variable of the ASN.1 BIT STRING type into a static memory structure. This function call is generated by ASN1C to decode a sized bit string production. It includes support for BIT STRING's with extension data.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>object_p</i>	Pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of bits specified in the *numbits_p input parameter.
<i>numbits</i> ↔ <i>_p</i>	Pointer to an integer variable containing the size (in bits) of the sized ASN.1 bit string. An error will occur if the number of bits in the decoded string is larger than this value. Note that this is also used as an output variable - the actual number of decoded bits will be returned in this variable.
<i>extdata</i>	Pointer to byte array containing extension data.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	The length, in OCTETs, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

### 6.4.3.13 xd\_boolean()

```
int xd_boolean (
    OSCTXT * pctxt,
    OSBOOL * object_p,
    ASN1TagType tagging,
    int length )
```

This function parses an ASN.1 BOOLEAN tag/length/value at the current message pointer location and advances the pointer to the next field.

The function first checks to see if explicit tagging is specified. If yes, the universal tag for this message type is checked to make sure it is of the expected value. If the match is not successful, a negative value is returned to indicate the parse was not successful. Otherwise, the pointer is advanced to the length field and the length parsed.

The length value is then check to see if it is equal to one which is the only valid length for boolean. If it is equal, the boolean data value is parsed; otherwise, and error is returned.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.
<i>object_↔_p</i>	Decoded boolean data value.

## Returns

Completion status of operation:

- 0 (0) = success,
- RTERR\_INVLEN invalid length (!= 1)
- RTERR\_IDNOTFOU unexpected tag value (not UNIV 1)

### 6.4.3.14 xd\_charstr()

```
int xd_charstr (
    OSCTXT * pctxt,
```

```

const char ** object_p,
ASN1TagType tagging,
ASN1TAG tag,
int length )

```

This function is the base function for decoding any of the 8-bit character string useful types such as IA5String, VisibleString, etc. This function allocates memory for the decoded string and returns a pointer to the data.

#### Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>object_p</i>	Pointer to a pointer to receive the address of the allocated memory into which the decoded character string data will be stored. The string is assumed to be a normal C string containing non-null characters. A null terminator is automatically added to the end of the string by this function.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>tag</i>	Tag variable to match
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

#### Returns

Completion status of operation:

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

#### 6.4.3.15 xd\_charstr64()

```

int xd_charstr64 (
    OSCTXT * pctxt,
    char ** object_p,
    ASN1TagType tagging,
    ASN1TAG tag,
    OSSIZE length,
    OSBOOL indefLen )

```

This function is the base function for decoding any of the 8-bit character string useful types such as IA5String, VisibleString, etc. This function allocates memory for the decoded string and returns a pointer to the data. It supports a character string lengths up to 64-bits on 64-bit systems.

#### Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>object_p</i>	Pointer to a pointer to receive the address of the allocated memory into which the decoded character string data will be stored. The string is assumed to be a normal C string containing non-null characters. A null terminator is automatically added to the end of the string by this function.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>tag</i>	Tag variable to match
<i>length</i>	Length of data to retrieve. Valid for implicit case only.
<i>indefLen</i>	Indicates length of data passed in is indefinite. Valid for implicit case only.



## Returns

Completion status of operation:

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

### 6.4.3.16 `xd_chkend()`

```
OSBOOL xd_chkend (
    OSCTXT * pctxt,
    const ASN1CCB * ccb_p )
```

This function checks for the end of a constructed element. It is typically used by the ASN1C compiler to set up a loop for decoding a constructed type such as a SEQUENCE or SET.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>ccb</i> ↔ <i>_p</i>	Pointer to a context control block (ccb). This is a structure added to compiler generated code to keep track of the current position within nested structures.

## Returns

Completion status of operation:

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

### 6.4.3.17 `xd_chkend64()`

```
OSBOOL xd_chkend64 (
    OSCTXT * pctxt,
    const OSOCTET * consptr,
    OSSIZE conslen,
    OSBOOL indef )
```

64-bit version of `xd_chkend`.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>consptr</i>	Pointer to start of encoded constructed element.
<i>conslen</i>	Length of encoded constructed element.
<i>indef</i>	True if constructor is indefinite length.

## Returns

Completion status of operation:

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

## See also

[xd\\_chkend](#)

### 6.4.3.18 xd\_count()

```
int xd_count (
    OSCTXT * pctxt,
    int length,
    int * count_p )
```

This function determines the count of elements within a constructed type.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>length</i>	Length of the constructed type.
<i>count_↔</i> <i>_p</i>	Pointer to an integer variable to receive the element count.

## Returns

Completion status of operation:

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

### 6.4.3.19 xd\_count64()

```
int xd_count64 (
    OSCTXT * pctxt,
    OSSIZE length,
    OSBOOL indefLen,
    OSSIZE * count_p )
```

64-bit version of `xd_count` function

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>length</i>	Length of the constructed type.
<i>indefLen</i>	Flag indicating length is indefinite.
<i>count</i> ↔ <i>_p</i>	Pointer to a size-typed variable to receive the element count.

## Returns

Completion status of operation:

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

## See also

[xd\\_count](#)

### 6.4.3.20 `xd_datestr()`

```
int xd_datestr (
    OSCTXT * pctxt,
    const char ** object_p,
    ASN1TagType tagging,
    ASN1TAG tag,
    int length )
```

This function is the base function for decoding ISO 8601 Date character string types. This function allocates memory for the decoded string and returns a pointer to the data.

## Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>object</i> ↔ <i>_p</i>	Pointer to a pointer to receive the address of the allocated memory into which the decoded character string data will be stored. The string is assumed to be a normal C string containing non-null characters. A null terminator is automatically added to the end of the string by this function.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>tag</i>	Tag variable to match
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

## Returns

Completion status of operation:

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

#### 6.4.3.21 xd\_datetimestr()

```
int xd_datetimestr (
    OSCTXT * pctxt,
    const char ** object_p,
    ASN1TagType tagging,
    ASN1TAG tag,
    int length )
```

This function is the base function for decoding ISO 8601 Date/Time character string types. This function allocates memory for the decoded string and returns a pointer to the data.

#### Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>object_p</i>	Pointer to a pointer to receive the address of the allocated memory into which the decoded character string data will be stored. The string is assumed to be a normal C string containing non-null characters. A null terminator is automatically added to the end of the string by this function.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>tag</i>	Tag variable to match
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

#### Returns

Completion status of operation:

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

#### 6.4.3.22 xd\_durationstr()

```
int xd_durationstr (
    OSCTXT * pctxt,
    const char ** object_p,
    ASN1TagType tagging,
    ASN1TAG tag,
    int length )
```

This function is the base function for decoding ISO 8601 Duration character string types. This function allocates memory for the decoded string and returns a pointer to the data.

## Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>object_↔_p</i>	Pointer to a pointer to receive the address of the allocated memory into which the decoded character string data will be stored. The string is assumed to be a normal C string containing non-null characters. A null terminator is automatically added to the end of the string by this function.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>tag</i>	Tag variable to match
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

## Returns

Completion status of operation:

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

### 6.4.3.23 `xd_enum()`

```
int xd_enum (
    OSCTXT * pctxt,
    OSINT32 * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a value of the ASN.1 ENUMERATED type. This function is identical to the integer decode function (`xd_integer`) except that the enumerated universal tag value is validated.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>object_↔_p</i>	Pointer to value to receive decoded result.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

## Returns

Completion status of operation:

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

#### 6.4.3.24 xd\_enumUnsigned()

```
int xd_enumUnsigned (
    OSCTXT * pctxt,
    OSUINT32 * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a value of the ASN.1 ENUMERATED type. This function is identical to the integer decode function (xd\_integer) except that the enumerated universal tag value is validated.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>object_p</i>	Pointer to value to receive decoded result.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

##### Returns

Completion status of operation:

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

#### 6.4.3.25 xd\_indeflen64()

```
int xd_indeflen64 (
    const OSOCTET * msg_p,
    OSSIZE bufSize,
    OSSIZE * plength )
```

This function calculates the actual length of an indefinite length message component. This version can support lengths up to 64 bits in size.

##### Parameters

<i>msg_p</i>	Pointer to an indefinite length message component.
<i>bufSize</i>	Size of the message buffer
<i>plength</i>	Pointer to a size-typed variable to receive decoded length;

## Returns

Completion status of operation:

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

Referenced by ASN1BERMessageBuffer::calcIndefLen().

### 6.4.3.26 xd\_indeflen\_ex()

```
int xd_indeflen_ex (
    const OSOCTET * msg_p,
    int bufSize )
```

This function calculates the actual length of an indefinite length message component.

#### Parameters

<i>msg</i> ↔ <i>_p</i>	Pointer to an indefinite length message component.
<i>bufSize</i>	Size of the message buffer

## Returns

Zero or positive = decoded length, negative = error status

Referenced by ASN1BERMessageBuffer::calcIndefLen().

### 6.4.3.27 xd\_int16()

```
int xd_int16 (
    OSCTXT * pctxt,
    OSINT16 * object_p,
    ASN1TagType tagging,
    int length )
```

This function parses an ASN.1 INTEGER tag/length/value at the current message pointer location and advances the pointer to the next field.

This function is similar to [xd\\_integer](#) but it is used to parse 16-bit integer values.

### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.
<i>object_↔ _p</i>	Pointer to decoded 16-bit integer value.

### Returns

Completion status of operation:

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

#### 6.4.3.28 `xd_int64()`

```
int xd_int64 (
    OSCTXT * pctxt,
    OSINT64 * object_p,
    ASN1TagType tagging,
    int length )
```

This function parses an ASN.1 INTEGER tag/length/value at the current message pointer location and advances the pointer to the next field.

The function is similar to [xd\\_integer](#) but it can be used to parse integer values with sizes up to 64 bits (if platform supports such integer's size).

If the match is successful or implicit tagging is specified, the integer data value is parsed.

### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.
<i>object_↔ _p</i>	Pointer to decoded 64-bit integer value.

### Returns

Completion status of operation:

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



#### 6.4.3.29 xd\_int8()

```
int xd_int8 (
    OSCTXT * pctxt,
    OSINT8 * object_p,
    ASN1TagType tagging,
    int length )
```

This function parses an ASN.1 INTEGER tag/length/value at the current message pointer location and advances the pointer to the next field.

This function is similar to [xd\\_integer](#) but it is used to parse 8-bit integer values.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.
<i>object_p</i>	Pointer to decoded 8-bit integer value.

##### Returns

Completion status of operation:

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

#### 6.4.3.30 xd\_integer()

```
int xd_integer (
    OSCTXT * pctxt,
    OSINT32 * object_p,
    ASN1TagType tagging,
    int length )
```

This function parses an ASN.1 INTEGER tag/length/value at the current message pointer location and advances the pointer to the next field.

The function first checks to see if explicit tagging is specified. If yes, the universal tag value is parsed and checked to make sure it matches the expected tag for this message type. If not, a negative value is returned to indicate the parse was not successful. Otherwise, the length value is parsed.

If the match is successful or implicit tagging is specified, the integer data value is parsed.

### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.
<i>object</i> <i>_p</i>	Pointer to decoded integer value.

### Returns

Completion status of operation:

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

#### 6.4.3.31 `xd_len()`

```
int xd_len (
    OSCTXT * pctxt,
    int * len_p )
```

This function decodes an ASN.1 length value.

### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>len</i> <i>_p</i>	Pointer to integer variable to receive the the decoded length value. If the length is indefinite, the constant ASN_K_INDEFLEN is returned.

### Returns

Completion status of operation:

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

#### 6.4.3.32 `xd_len64()`

```
int xd_len64 (
    OSCTXT * pctxt,
    OSSIZE * len_p,
    OSBOOL * p indef )
```

This function decodes an ASN.1 length value. It supports lengths up to 64-bits in size on 64-bit platforms.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>len</i> ↔ <i>_p</i>	Length of message component. The value returned is undefined if length is indefinite.
<i>pindef</i>	Pointer to boolean indicating if length is indefinite.

## Returns

Completion status of operation:

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

### 6.4.3.33 xd\_match()

```
int xd_match (
    OSCTXT * pctxt,
    ASN1TAG tag,
    int * len_p,
    OSOCTET flags )
```

This function compares the tag at the current message pointer position with the given tag for a match. If a match occurs, the length field is decoded and the length is returned to the caller. If the input parameter 'advance' is set to TRUE, the message pointer is advanced to the beginning of the contents field.

If a match does not occur, the routine will skip to subsequent fields in search of a match. If a match is eventually found, the processing described above is done; otherwise, a not found status is returned to the caller.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tag</i>	Tag variable to match.
<i>len</i> ↔ <i>_p</i>	Length of message component. Returned as follows: >= 0 component is fixed length ASN_K_INDEFLEN component is indefinite length
<i>flags</i>	Bit flags used to set the following options: <ul style="list-style-type: none"><li>• XM_ADVANCE: Advance decode pointer on match.</li><li>• XM_SEEK : Seek until match found or EOM.</li><li>• XM_SKIP : Skip to next tag before search</li></ul>

## Returns

Completion status of operation:

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

#### 6.4.3.34 `xd_match1()`

```
int xd_match1 (
    OSCTXT * pctxt,
    OSOCTET tag,
    int * len_p )
```

The `xd_match1` function does a comparison between the given tag and the tag at the current decode pointer position to determine if they match. It then returns the result of the match operation. In contrast to `xd_match` function `xd_match1` is an optimized version and can be used only to compare primitive tags (with number less than 31). It is always advance the decode pointer to the contents field if matching is successful. Note, that the tag should be specified as an octet, like it is used in BER encoding.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tag</i>	Tag variable to match in octet format.
<i>len</i> ↔ <i>_p</i>	Length of message component. Returned as follows: $\geq 0$ component is fixed length ASN_K_INDEFLEN component is indefinite length

#### Returns

Completion status of operation:

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

#### 6.4.3.35 `xd_match64()`

```
int xd_match64 (
    OSCTXT * pctxt,
    ASN1TAG tag,
    OSSIZE * len_p,
    OSBOOL * pindef,
    OSOCTET flags )
```

This function compares the tag at the current message pointer position with the given tag for a match. If a match occurs, the length field is decoded and the length is returned to the caller. If the input parameter 'advance' is set to TRUE, the message pointer is advanced to the beginning of the contents field.

If a match does not occur, the routine will skip to subsequent fields in search of a match. If a match is eventually found, the processing described above is done; otherwise, a not found status is returned to the caller.

This version of the function works with lengths up to 64-bits in size.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tag</i>	Tag variable to match.
<i>len</i> ↔ <i>_p</i>	Pointer to variable to receive length of message component.
<i>pindef</i>	Pointer to boolean variable to indicate if parsed length value is indefinite.
<i>flags</i>	Bit flags used to set the following options: <ul style="list-style-type: none"><li>• XM_ADVANCE: Advance decode pointer on match.</li><li>• XM_SEEK : Seek until match found or EOM.</li><li>• XM_SKIP : Skip to next tag before search</li></ul>

## Returns

Completion status of operation:

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

### 6.4.3.36 xd\_memcpy()

```
int xd_memcpy (
    OSCTXT * pctxt,
    OSOCTET * object_p,
    int length )
```

This function copies data from the contents field of a message component into the target object.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
* <i>object</i> ↔ <i>_p</i>	A pointer to a memory structure to receive the copied data.
<i>length</i>	The number of bytes to copy from the contents field.

## Returns

Completion status of operation:

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

#### 6.4.3.37 `xd_NextElement()`

```
int xd_NextElement (
    OSCTXT * pctxt )
```

This function skips to the next element in the decode buffer.

##### Parameters

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

##### Returns

Completion status of operation:

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

#### 6.4.3.38 `xd_null()`

```
int xd_null (
    OSCTXT * pctxt,
    ASN1TagType tagging )
```

This function decoded the ASN.1 NULL placeholder type. The null data type contains no data; however, if explicit tagging is specified, it will contain a universal tag value (5) and zero length. This function will parse those values.

##### Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged. The function will do nothing if implicit tagging is specified.

##### Returns

Completion status of operation:

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

#### 6.4.3.39 `xd_objid()`

```
int xd_objid (
    OSCTXT * pctxt,
```

```
ASN1OBJID * object_p,
ASN1TagType tagging,
int length )
```

This function decodes a value of the ASN.1 object identifier type.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>object_p</i>	Pointer to value to receive decoded result. The ASN1OBJID structure contains an integer to hold the number of subidentifiers and an array to hold the subidentifier values.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

#### Returns

Completion status of operation:

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

#### 6.4.3.40 xd\_octstr()

```
int xd_octstr (
    OSCTXT * pctxt,
    const OSOCTET ** object_p2,
    OSUINT32 * pnumocts,
    ASN1TagType tagging,
    int length )
```

This function decodes the octet string at the current message pointer location and returns its value. This version of the function allocates memory for the decoded string and returns a pointer to the data.

#### Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>object_p2</i>	Pointer to a pointer to receive the address of the allocated memory into which the decoded data will be stored.
<i>pnumocts</i>	Pointer to an integer variable to receive length of the decoded octet string.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

#### Returns

Completion status of operation:

- 0 (0) = success,

- negative return value is error.

#### 6.4.3.41 `xd_octstr64()`

```
int xd_octstr64 (
    OSCTXT * pctxt,
    OSOCTET ** object_p2,
    OSSIZE * pnumocts,
    ASN1TagType tagging,
    OSSIZE length,
    OSBOOL indefLen )
```

This function decodes the octet string at the current message pointer location and returns its value. This version of the function allocates memory for the decoded string and returns a pointer to the data. It supports OCTET STRING lengths up to 64-bits on 64-bit systems.

#### Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>object_p2</i>	Pointer to a pointer to receive the address of the allocated memory into which the decoded data will be stored.
<i>pnumocts</i>	Pointer to an integer variable to receive length of the decoded octet string.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.
<i>indefLen</i>	Indicates length of data passed in is indefinite. Valid for implicit case only.

#### Returns

Completion status of operation:

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

#### 6.4.3.42 `xd_octstr64_s()`

```
int xd_octstr64_s (
    OSCTXT * pctxt,
    OSOCTET * object_p,
    OSSIZE * pnumocts,
    ASN1TagType tagging,
    OSSIZE length,
    OSBOOL indefLen )
```

This function decodes the octet string at the current message pointer location and returns its value. The value is returned in the buffer pointed to by the given character buffer pointer. This is a static buffer that must be large enough to hold the decoded data. It supports OCTET STRING lengths up to 64-bits on 64-bit systems.



#### Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>object_p</i>	Pointer to static octet array to receive decoded data
<i>pnumocts</i>	Pointer to an integer variable to receive the length of the decoded octet string. On input, this parameter is used to specify the size of the octet array.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

#### Returns

Completion status of operation:

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

#### 6.4.3.43 `xd_octstr_s()`

```
int xd_octstr_s (  
    OSCTXT * pctxt,  
    OSOCTET * object_p,  
    OSUINT32 * pnumocts,  
    ASN1TagType tagging,  
    int length )
```

This function decodes the octet string at the current message pointer location and returns its value. The value is returned in the buffer pointed to by the given character buffer pointer. This is a static buffer that must be large enough to hold the decoded data.

#### Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>object_p</i>	Pointer to static octet array to receive decoded data
<i>pnumocts</i>	Pointer to an integer variable to receive the length of the decoded octet string. On input, this parameter is used to specify the size of the octet array.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

#### Returns

Completion status of operation:

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

#### 6.4.3.44 xd\_oid64()

```
int xd_oid64 (
    OSCTXT * pctxt,
    ASN1OID64 * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a value of the ASN.1 object identifier type using 64-bit subidentifiers.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>object_p</i>	Pointer to value to receive decoded result. The ASN1OID64 structure contains an integer to hold the number of subidentifiers and an array of 64-bit unsigned integers to hold the subidentifier values.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

##### Returns

Completion status of operation:

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

#### 6.4.3.45 xd\_OpenType()

```
int xd_OpenType (
    OSCTXT * pctxt,
    const OSOCTET ** object_p2,
    OSSIZE * pnumocts )
```

This function decodes a value of an ASN.1 open type. An open type is used to model the old ASN.1 ANY and ANY DEFINED BY types. It is also used to model variable type references within information objects (for example, TYPE-IDENTIFIER.&Type). Dynamic memory is allocated to hold the decoded result.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>object_p2</i>	Pointer to value to receive decoded result.
<i>pnumocts</i>	Pointer to an integer variable to receive length of the decoded octet string.

##### Returns

Completion status of operation:

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

#### 6.4.3.46 xd\_OpenTypeAppend()

```
int xd_OpenTypeAppend (
    OSCTXT * pctxt,
    OSRTDList * pElemList )
```

This function appends a decoded open type element onto a list of elements. It is used by the ASN1C compiler to decode messages with multiple extension fields following an extension marker (...).

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pElemList</i>	Pointer to element list onto which the decoded element should be appended.

##### Returns

Completion status of operation:

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

#### 6.4.3.47 xd\_OpenTypeExt()

```
int xd_OpenTypeExt (
    OSCTXT * pctxt,
    ASN1CCB * ccb_p,
    ASN1TAG * tags,
    int tagCount,
    OSRTDList * pElemList )
```

This function decodes an ASN.1 open type extension. This is the optional data that follows the ... in multi-version messages. Dynamic memory is allocated to hold the decoded result.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>ccb_p</i>	Pointer to a 'context control block' structure. This is basically a loop control mechanism to keep the variable associated with parsing a nested constructed element straight.
<i>tags</i>	Array of next expected tag values (null if last field). The routine will loop through elements until a matching tag is found or some other error occurs.
<i>tagCount</i>	The number of tags in the tags array.
<i>pElemList</i>	The pointer to linked list structure. The list will contain elements of ASN1OpenType type.

## Returns

Completion status of operation:

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

### 6.4.3.48 xd\_OpenTypeExt64()

```
int xd_OpenTypeExt64 (
    OSCTXT * pctxt,
    const OSOCTET * consptr,
    OSSIZE conslen,
    OSBOOL indefLen,
    ASN1TAG * tags,
    OSSIZE tagCount,
    OSRTDList * pElemList )
```

64-bit version of xd\_OpenTypeExt.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>consptr</i>	Pointer to start of encoded constructed element.
<i>conslen</i>	Length of encoded constructed element.
<i>indefLen</i>	True if constructor is indefinite length.
<i>tags</i>	Array of next expected tag values (null if last field). The routine will loop through elements until a matching tag is found or some other error occurs.
<i>tagCount</i>	The number of tags in the tags array.
<i>pElemList</i>	The pointer to linked list structure. The list will contain elements of ASN1OpenType type.

## Returns

Completion status of operation:

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

## See also

[xd\\_OpenTypeExt](#)

#### 6.4.3.49 xd\_real()

```
int xd_real (
    OSCTXT * pctxt,
    OSREAL * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a value of the binary encoded ASN.1 REAL type.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>object_↔_p</i>	Pointer to value to receive decoded result. The OSREAL data type is a double-precision floating point number type (C double type).
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

##### Returns

Completion status of operation:

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

#### 6.4.3.50 xd\_real10()

```
int xd_real10 (
    OSCTXT * pctxt,
    const char ** object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a value of the decimal encoded ASN.1 REAL type.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>object_↔_p</i>	Pointer to a character pointer variable to receive the decoded result. Dynamic memory is allocated for the variable using the ::rtxMemAlloc function.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

## Returns

Completion status of operation:

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

### 6.4.3.51 xd\_reloid()

```
int xd_reloid (
    OSCTXT * pctxt,
    ASN1OBJID * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a value of the ASN.1 RELATIVE-OID type.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>object_p</i>	Pointer to value to receive decoded result. The ASN1OBJID structure contains an integer to hold the number of subidentifiers and an array to hold the subidentifier values.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

## Returns

Completion status of operation:

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

### 6.4.3.52 xd\_setp()

```
int xd_setp (
    OSCTXT * pctxt,
    const OSOCTET * msg_p,
    int msglen,
    ASN1TAG * tag_p,
    int * len_p )
```

This function sets the decode pointer (cursor) to point at the beginning of the encoded ASN.1 message that is to be decoded. This function must be called prior to calling any of the other decode functions.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>msg_p</i>	Pointer to message buffer containing data to be decoded.
<i>msglen</i>	Size of the message data buffer. This is an optional parameter. It is used to verify that the length of the data encoded in the message is less than or equal to the given size of the message buffer. If the message size is unknown at the time of decoding, this argument can be set to zero and the size check will be bypassed.
<i>tag_p</i>	Pointer to an ASN.1 tag variable to receive the value of the initial tag parsed from a message. This is an optional parameter. It can be set to NULL if the user does not require the initial tag value.
<i>len_p</i>	Pointer to an integer variable to receive the decoded message length. Note that this is the total length of the message from the start of message, NOT the actual length decoded after the initial tag. In other words the length of the initial tag and length are added on to the parsed length. This is an optional parameter. It can be set to NULL if the user does not require the message length value.

## Returns

Completion status of operation:

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

Referenced by ASN1BERDecodeBuffer::parseTagLen().

### 6.4.3.53 `xd_setp64()`

```
int xd_setp64 (
    OSCTXT * pctxt,
    const OSOCTET * msg_p,
    OSSIZE msglen,
    ASN1TAG * tag_p,
    OSSIZE * len_p,
    OSBOOL * pIndefLen )
```

This function sets the decode pointer (cursor) to point at the beginning of the encoded ASN.1 message that is to be decoded. This function must be called prior to calling any of the other decode functions. This version of the function supports 64-bit lengths.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>msg_p</i>	Pointer to message buffer containing data to be decoded.
<i>msglen</i>	Size of the message data buffer. This is an optional parameter. It is used to verify that the length of the data encoded in the message is less than or equal to the given size of the message buffer. If the message size is unknown at the time of decoding, this argument can be set to zero and the size check will be bypassed.

## Parameters

<i>tag_p</i>	Pointer to an ASN.1 tag variable to receive the value of the initial tag parsed from a message. This is an optional parameter. It can be set to NULL if the user does not require the initial tag value.
<i>len_p</i>	Pointer to an integer variable to receive the decoded message length. Note that this is the total length of the message from the start of message, NOT the actual length decoded after the initial tag. In other words the length of the initial tag and length are added on to the parsed length. This is an optional parameter. It can be set to NULL if the user does not require the message length value.
<i>pIndefLen</i>	Boolean flag indicating parsed length is indefinite.

## Returns

Completion status of operation:

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

Referenced by ASN1BERDecodeBuffer::parseTagLen().

### 6.4.3.54 xd\_tag()

```
int xd_tag (
    OSCTXT * pctxt,
    ASN1TAG * tag_p )
```

This function decodes an ASN.1 tag into a standard 32-bit unsigned integer type. The bits used to represent the components of a tag are as follows:

Bit Fields:

- 31-30 Class (00 = UNIV, 01 = APPL, 10 = CTXT, 11 = PRIV)
- 29 Form (0 = primitive, 1 = constructed)
- 28-0 ID code value

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tag</i> <sub>↔</sub> <i>_p</i>	Pointer to variable to receive decoded tag info.

## Returns

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



#### 6.4.3.55 xd\_Tag1AndLen()

```
int xd_Tag1AndLen (
    OSCTXT * pctxt,
    OSINT32 * len_p )
```

This function is an optimized version of the `xd_tag_len` function. If the ASN1C compiler determines the tag at a given location to be parsed is only one byte long (a typical case) then it will use this function. It reads a single tag byte and then immediately parses the length field.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>len</i> ↔ <i>_p</i>	Length of message component. Returned as follows: $\geq 0$ component is fixed length ASN_K_INDEFLEN component is indefinite length

##### Returns

Completion status of operation:

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

#### 6.4.3.56 xd\_tag\_len()

```
int xd_tag_len (
    OSCTXT * pctxt,
    ASN1TAG * tag_p,
    int * len_p,
    OSOCTET flags )
```

This function parses the ASN.1 tag and length fields located at the current message pointer position.

`xd_tag_len` monitors indefinite length messages as follows: Each time a length is parsed, it is checked to see if it is an indefinite length value. If it is, an indefinite length section counter is incremented. Each time an end-of-contents (EOC) identifier is parsed, this counter is decremented. When the counter goes to zero, end of message is signaled.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tag</i> ↔ <i>_p</i>	Pointer to variable to receive decoded tag info.
<i>len</i> ↔ <i>_p</i>	Length of message component. Returned as follows: $\geq 0$ component is fixed length ASN_K_INDEFLEN component is indefinite length
<i>flags</i>	Bit flags used to set the following options: XM_ADVANCE: Advance decode pointer on match.

## Returns

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

### 6.4.3.57 `xd_tag_len_64()`

```
int xd_tag_len_64 (
    OSCTXT * pctxt,
    ASN1TAG * tag_p,
    OSSIZE * len_p,
    OSBOOL * pIndefLen,
    OSOCTET flags )
```

This version of the `xd_tag_len` function supports lengths up to 64 bits in size on 64-bit systems.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tag_p</i>	Pointer to variable to receive decoded tag info.
<i>len_p</i>	Length of message component. The value returned is undefined if length is indefinite.
<i>pIndefLen</i>	Boolean flag indicating parsed length is indefinite.
<i>flags</i>	Bit flags used to set the following options: <code>XM_ADVANCE</code> : Advance decode pointer on match.

## Returns

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

### 6.4.3.58 `xd_timeofdaystr()`

```
int xd_timeofdaystr (
    OSCTXT * pctxt,
    const char ** object_p,
    ASN1TagType tagging,
    ASN1TAG tag,
    int length )
```

This function is the base function for decoding ISO 8601 Time-of-day character string types. This function allocates memory for the decoded string and returns a pointer to the data.

## Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
--------------	--

## Parameters

<i>object↔ _p</i>	Pointer to a pointer to receive the address of the allocated memory into which the decoded character string data will be stored. The string is assumed to be a normal C string containing non-null characters. A null terminator is automatically added to the end of the string by this function.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>tag</i>	Tag variable to match
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

## Returns

Completion status of operation:

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

### 6.4.3.59 xd\_timestr()

```
int xd_timestr (
    OSCTXT * pctxt,
    const char ** object_p,
    ASN1TagType tagging,
    ASN1TAG tag,
    int length )
```

This function is the base function for decoding ISO 8601 Time character string types. This function allocates memory for the decoded string and returns a pointer to the data.

## Parameters

<i>pctxt</i>	Pointer to ASN.1 context block structure
<i>object↔ _p</i>	Pointer to a pointer to receive the address of the allocated memory into which the decoded character string data will be stored. The string is assumed to be a normal C string containing non-null characters. A null terminator is automatically added to the end of the string by this function.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>tag</i>	Tag variable to match
<i>length</i>	Length of data to retrieve. Valid for implicit case only.

## Returns

Completion status of operation:

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

### 6.4.3.60 xd\_uint16()

```
int xd_uint16 (
    OSCTXT * pctxt,
    OSUINT16 * object_p,
    ASN1TagType tagging,
    int length )
```

This function parses an unsigned variant of ASN.1 INTEGER tag/length/value at the current message pointer location and advances the pointer to the next field.

This function is similar to [xd\\_unsigned](#) but it is used to parse 16-bit unsigned integer values.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.
<i>object_↔ _p</i>	Pointer to decoded 16-bit unsigned integer value.

#### Returns

Completion status of operation:

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

### 6.4.3.61 xd\_uint64()

```
int xd_uint64 (
    OSCTXT * pctxt,
    OSUINT64 * object_p,
    ASN1TagType tagging,
    int length )
```

This function parses an unsigned variant of ASN.1 INTEGER tag/length/value at the current message pointer location and advances the pointer to the next field.

The function is similar to [xd\\_unsigned](#) but it can be used to parse integer values with sizes up to 64 bits (if platform supports such integer's size).

If the match is successful or implicit tagging is specified, the integer data value is parsed.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.
<i>object_↔ _p</i>	Pointer to decoded 64-bit unsigned integer value.

## Returns

Completion status of operation:

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

### 6.4.3.62 `xd_uint8()`

```
int xd_uint8 (
    OSCTXT * pctxt,
    OSUINT8 * object_p,
    ASN1TagType tagging,
    int length )
```

This function parses an unsigned variant of ASN.1 INTEGER tag/length/value at the current message pointer location and advances the pointer to the next field.

This function is similar to [xd\\_unsigned](#) but it is used to parse 8-bit unsigned integer values.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.
<i>object_p</i>	Pointer to decoded 8-bit unsigned integer value.

## Returns

Completion status of operation:

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

### 6.4.3.63 `xd_unsigned()`

```
int xd_unsigned (
    OSCTXT * pctxt,
    OSUINT32 * object_p,
    ASN1TagType tagging,
    int length )
```

This function parses an unsigned variant of ASN.1 INTEGER tag/length/value at the current message pointer location and advances the pointer to the next field.

The function first checks to see if explicit tagging is specified. If yes, the universal tag value is parsed and checked to make sure it matches the expected tag for this message type. If not, a negative value is returned to indicate the parse was not successful. Otherwise, the length value is parsed.

If the match is successful or implicit tagging is specified, the integer data value is parsed.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	Specifies whether element is implicitly or explicitly tagged.
<i>length</i>	Length of data to retrieve. Valid for implicit case only.
<i>object</i> <i>_p</i>	Pointer to decoded unsigned integer value.

#### Returns

Completion status of operation:

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

## 6.5 BER/DER C File Functions.

### Functions

- `int xdf_tag` (FILE \*fp, ASN1TAG \*ptag, OSOCTET \*buffer, int \*pbufidx)
- `int xdf_len` (FILE \*fp, OSINT32 \*plen, OSOCTET \*buffer, int \*pbufidx)
- `int xdf_TagAndLen` (FILE \*fp, ASN1TAG \*ptag, OSINT32 \*plen, OSOCTET \*buffer, int \*pbufidx)
- `int xdf_ReadPastEOC` (FILE \*fp, OSOCTET \*buffer, int bufsiz, int \*pbufidx)
- `int xdf_ReadContents` (FILE \*fp, int len, OSOCTET \*buffer, int bufsiz, int \*pbufidx)

### 6.5.1 Detailed Description

The BER/DER file decode functions allow decode operations to be performed directly on encoded entities within a binary file as opposed to in memory. This makes it possible to parse tag and length variables to determine when pieces of a message can be read into memory. The "tap3batch" sample program provides a good illustration of how these functions are used. They can be applied to a TAP3 batch file to get at the call-detail records for sequential processing without having to read the entire file into memory.

These functions all begin with the prefix "xdf\_" to distinguish them from the other decode functions. The following is a description of the various functions that make up this package.

### 6.5.2 Function Documentation

#### 6.5.2.1 xdf\_len()

```
int xdf_len (  
    FILE * fp,  
    OSINT32 * plen,  
    OSOCTET * buffer,  
    int * pbufidx )
```

This function decodes the ASN.1 length from a file stream.

#### Parameters

<i>fp</i>	The file pointer of the binary file to be decoded. It is expected that the current file position is at the first byte of the tag to be decoded.
<i>plen</i>	A pointer to an integer to receive the decoded length value.
<i>buffer</i>	The buffer to receive the parsed data.
<i>pbufidx</i>	The pointer to the current buffer index containing offset to the location where the decoded bytes should be copied in the output buffer. The updated buffer index set to point at the first free byte after the tag is parsed and copied into the buffer.

## Returns

Completion status of operation:

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

### 6.5.2.2 xdf\_ReadContents()

```
int xdf_ReadContents (
    FILE * fp,
    int len,
    OSOCTET * buffer,
    int bufsiz,
    int * pbufidx )
```

This routine reads the contents of a BER tag-length-value (TLV) into the given buffer. The TLV can be of indefinite length.

#### Parameters

<i>fp</i>	A file pointer of a binary file to be decoded. It is expected that the current file position is the first byte following an indefinite length marker (0x80 byte).
<i>len</i>	The length of data to be read from the file. This can be the indefinite constant (ASN_K_INDEFLEN) indicating all data up to the corresponding end-of-context (EOC) marker should be read.
<i>buffer</i>	The buffer to receive the parsed data.
<i>bufsiz</i>	The size of the buffer to receive the parsed data.
<i>pbufidx</i>	The pointer to the current buffer index containing offset to the location where the decoded bytes should be copied in the output buffer. The updated buffer index set to point at the first free byte index after the parsed data is copied to the buffer.

## Returns

Completion status of operation:

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

### 6.5.2.3 xdf\_ReadPastEOC()

```
int xdf_ReadPastEOC (
    FILE * fp,
    OSOCTET * buffer,
    int bufsiz,
    int * pbufidx )
```



This function consumes bytes from the file stream until a matching end-of-context (EOC) marker is found. The bytes read from the file are stored in the given buffer for later processing. An indefinite marker is assumed to have been parsed prior to calling this function.

## Parameters

<i>fp</i>	A file pointer of a binary file to be decoded. It is expected that the current file position is the first byte following an indefinite length marker (0x80 byte).
<i>buffer</i>	The buffer to receive the parsed data.
<i>bufsiz</i>	The size of the buffer to receive the parsed data.
<i>pbufidx</i>	The pointer to the current buffer index containing offset to the location where the decoded bytes should be copied in the output buffer. The updated buffer index set to point at the first free byte index after the parsed data is copied to the buffer.

## Returns

Completion status of operation:

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

### 6.5.2.4 xdf\_tag()

```
int xdf_tag (
    FILE * fp,
    ASN1TAG * ptag,
    OSOCTET * buffer,
    int * pbufidx )
```

This function decodes ASN.1 tag from a file stream into a standard 32-bit ASN.1 tag structure.

## Parameters

<i>fp</i>	The file pointer of the binary file to be decoded. It is expected that the current file position is at the first byte of the tag to be decoded.
<i>ptag</i>	A pointer to an ASN.1 tag structure to receive the decoded tag.
<i>buffer</i>	The buffer to receive the parsed data.
<i>pbufidx</i>	The pointer to the current buffer index containing offset to the location where the decoded bytes should be copied in the output buffer. The updated buffer index set to point at the first free byte after the tag is parsed and copied into the buffer.

## Returns

Completion status of operation:

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

### 6.5.2.5 xdf\_TagAndLen()

```
int xdf_TagAndLen (
    FILE * fp,
    ASN1TAG * ptag,
    OSINT32 * plen,
    OSOCTET * buffer,
    int * pbufidx )
```

This function decodes an ASN.1 tag and length pair from a file stream.

#### Parameters

<i>fp</i>	The file pointer of the binary file to be decoded. It is expected that the current file position is at the first byte of the tag to be decoded.
<i>ptag</i>	A pointer to an ASN1TAG variable to receive parsed the tag value.
<i>plen</i>	A pointer to an integer to receive the decoded length value.
<i>buffer</i>	The buffer to receive the parsed data.
<i>pbufidx</i>	The pointer to the current buffer index containing offset to the location where the decoded bytes should be copied in the output buffer. The updated buffer index set to point at the first free byte after the tag is parsed and copied into the buffer.

#### Returns

Completion status of operation:

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

## 6.6 BER/DER C Encode Functions.

### Macros

- #define `xe_utf8str`(pctxt, object\_p, tagging) `xe_charstr` (pctxt, (const char\*)object\_p, tagging, ASN\_ID\_UTF8↵String)

### Functions

- int `xe_identifier` (OSCTXT \*pctxt, OSUINT32 ident)
- int `xe_tag` (OSCTXT \*pctxt, ASN1TAG tag)
- int `xe_tag_len` (OSCTXT \*pctxt, ASN1TAG tag, int length)
- int `xe_boolean` (OSCTXT \*pctxt, OSBOOL \*object\_p, ASN1TagType tagging)
- int `xe_integer` (OSCTXT \*pctxt, int \*object\_p, ASN1TagType tagging)
- int `xe_unsigned` (OSCTXT \*pctxt, OSUINT32 \*object\_p, ASN1TagType tagging)
- int `xe_int8` (OSCTXT \*pctxt, OSINT8 \*object\_p, ASN1TagType tagging)
- int `xe_int16` (OSCTXT \*pctxt, OSINT16 \*object\_p, ASN1TagType tagging)
- int `xe_int64` (OSCTXT \*pctxt, OSINT64 \*object\_p, ASN1TagType tagging)
- int `xe_uint64` (OSCTXT \*pctxt, OSUINT64 \*object\_p, ASN1TagType tagging)
- int `xe_uint8` (OSCTXT \*pctxt, OSUINT8 \*object\_p, ASN1TagType tagging)
- int `xe_uint16` (OSCTXT \*pctxt, OSUINT16 \*object\_p, ASN1TagType tagging)
- int `xe_bigint` (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging)
- int `xe_bigintn` (OSCTXT \*pctxt, const char \*object\_p, size\_t nchars, ASN1TagType tagging)
- int `xe_bitstr` (OSCTXT \*pctxt, const OSOCTET \*object\_p, OSSIZE numbits, ASN1TagType tagging)
- int `xe_bitstrExt` (OSCTXT \*pctxt, const OSOCTET \*object\_p, OSSIZE numbits, OSSIZE dataSize, const OSO↵CTET \*extdata, ASN1TagType tagging)
- int `xe_octstr` (OSCTXT \*pctxt, const OSOCTET \*object\_p, OSSIZE numocts, ASN1TagType tagging)
- int `xe_charstr` (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int `xe_16BitCharStr` (OSCTXT \*pctxt, Asn116BitCharString \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int `xe_32BitCharStr` (OSCTXT \*pctxt, Asn132BitCharString \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int `xe_datestr` (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int `xe_timestr` (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int `xe_datetimestr` (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int `xe_timeofdaystr` (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int `xe_durationstr` (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int `xe_null` (OSCTXT \*pctxt, ASN1TagType tagging)
- int `xe_objid` (OSCTXT \*pctxt, ASN1OBJID \*object\_p, ASN1TagType tagging)
- int `xe_oid64` (OSCTXT \*pctxt, ASN1OID64 \*object\_p, ASN1TagType tagging)
- int `xe_reloid` (OSCTXT \*pctxt, ASN1OBJID \*object\_p, ASN1TagType tagging)
- int `xe_enum` (OSCTXT \*pctxt, OSINT32 \*object\_p, ASN1TagType tagging)
- int `xe_enumUnsigned` (OSCTXT \*pctxt, OSUINT32 \*object\_p, ASN1TagType tagging)
- int `xe_real` (OSCTXT \*pctxt, OSREAL \*object\_p, ASN1TagType tagging)
- int `xe_OpenType` (OSCTXT \*pctxt, const OSOCTET \*object\_p, OSSIZE numocts)
- int `xe_OpenTypeExt` (OSCTXT \*pctxt, OSRTDList \*pElemList)
- int `xe_OpenTypeExtDer` (OSCTXT \*pctxt, OSRTDList \*pElemList, OSRTSList \*pBufList)
- int `xe_real10` (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging)
- int `xe_derReal10` (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging)
- int `xe_setp` (OSCTXT \*pctxt, OSOCTET \*buf\_p, OSSIZE bufsiz)
- OSOCTET \* `xe_getp` (OSCTXT \*pctxt)

- void `xe_free` (OSCTXT \*pctxt)
- int `xe_expandBuffer` (OSCTXT \*pctxt, size\_t length)
- int `xe_memcpy` (OSCTXT \*pctxt, const OSOCTET \*object\_p, size\_t length)
- int `xe_len` (OSCTXT \*pctxt, int length)
- int `xe_len64` (OSCTXT \*pctxt, OSSIZE length, OSBOOL indef)
- int `xe_derCanonicalSort` (OSCTXT \*pctxt, OSRTSList \*pList)
- int `xe_derCanSortSet` (OSCTXT \*pctxt, OSRTSList \*pList)
- int `xe_TagAndIndefLen` (OSCTXT \*pctxt, ASN1TAG tag, int length)
- void `xe_getBufLocDescr` (OSCTXT \*pctxt, OSSIZE length, Asn1BufLocDescr \*pDescr)
- int `derEncBitString` (OSCTXT \*pctxt, const OSOCTET \*pvalue, OSSIZE numbits, ASN1TagType tagging)

## 6.6.1 Detailed Description

BER/DER C encode functions handle the BER encoding of the primitive ASN.1 data types and ASN.1 length and tag fields within a message. Calls to these functions are assembled in the C source code generated by the ASN1C compiler to accomplish the encoding of complex ASN.1 structures. These functions are also directly callable from within a user's application program if the need to accomplish a low level encoding function exists.

The procedure to call the encode function that encodes a primitive type is the same as the procedure to call a compiler generated encode function described above. The `rtxInitContext` and `xe_setp` functions must first be called to initialize a context variable and set a pointer to the buffer into which the variable is to be encoded. A static encode buffer is specified by specifying a pointer to a buffer and buffer size. Setting the buffer address to NULL and buffer size to 0 specifies a dynamic buffer. The primitive encode function is invoked. Finally, `xe_getp` is called to retrieve a pointer to the encoded message component.

## 6.6.2 Macro Definition Documentation

### 6.6.2.1 `xe_utf8str`

```
#define xe_utf8str(
    pctxt,
    object_p,
    tagging ) xe_charstr (pctxt, (const char*)object_p, tagging, ASN_ID_UTF8String)
```

This function will encode a variable one of ASN.1 UTF-8 character string type.

#### Parameters

<i>pctxt</i>	Pointer to a context block structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to a null-terminated UTF-8 C character string to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

## 6.6.3 Function Documentation

### 6.6.3.1 derEncBitString()

```
int derEncBitString (
    OSCTXT * pctx,
    const OSOCTET * pvalue,
    OSSIZE numbits,
    ASN1TagType tagging )
```

This function encodes a BIT STRING value in accordance with the Distinguished Encoding Rules (DER). It adjusts the bit count to remove trailing zero bits and then invokes the BER encode function.

#### Parameters

<i>pctx</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pvalue</i>	Bit string to be encoded.
<i>numbits</i>	Number of bits in the bit string.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.2 xe\_16BitCharStr()

```
int xe_16BitCharStr (
    OSCTXT * pctx,
    Asn116BitCharString * object_p,
    ASN1TagType tagging,
    ASN1TAG tag )
```

This function will encode a variable one of the ASN.1 character string types that are based on a 16-bit character set. This includes the ASN.1 BMP character string type.

## Parameters

<i>pctxt</i>	Pointer to a context block structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object↔ _p</i>	A pointer to a structure representing a 16-bit character string to be encoded. This structure contains a character count element and a pointer to an array of 16-bit character elements represented as 16-bit short integers.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The Universal ASN.1 tag to be encoded in the message. This parameter is passed using the internal representation discussed in Section 4.1.2. It is passed as an unsigned 16-bit integer. The tag value must represent one of the 16-bit character string types documented in the X.680 standard.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.3 xe\_32BitCharStr()

```
int xe_32BitCharStr (
    OSCTXT * pctxt,
    Asn132BitCharString * object_p,
    ASN1TagType tagging,
    ASN1TAG tag )
```

This function will encode a variable one of the ASN.1 character string types that are based on a 32-bit character set. This includes the ASN.1 Universal character string type.

## Parameters

<i>pctxt</i>	Pointer to a context block structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object↔ _p</i>	A pointer to a structure representing a 32-bit character string to be encoded. This structure contains a character count element and a pointer to an array of 32-bit character elements represented as 16-bit short integers.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The Universal ASN.1 tag to be encoded in the message. This parameter is passed using the internal representation discussed in Section 4.1.2. It is passed as an unsigned 32-bit integer. The tag value must represent one of the 32-bit character string types documented in the X.680 standard.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

#### 6.6.3.4 xe\_bigint()

```
int xe_bigint (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 INTEGER type. In this case, the integer is assumed to be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits). For example, parameters used to calculate security values are typically larger than these sizes.

Items of this type are stored in character string constant variables. They can be represented as decimal strings (with no prefixes), as hexadecimal strings starting with a "0x" prefix, as octal strings starting with a "0o" prefix or as binary strings starting with a "0b" prefix. Other radices currently are not supported. Non-decimal values are assumed to be in two's complement form, unless a leading zero is included. It is highly recommended to use the hexadecimal or binary strings for better performance.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>object_p</i>	A pointer to a character string containing the value to be encoded.

##### Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

#### 6.6.3.5 xe\_bitstr()

```
int xe_bitstr (
    OSCTXT * pctxt,
    const OSOCTET * object_p,
    OSSIZE numbits,
    ASN1TagType tagging )
```

This function will encode a variable of the ASN.1 BIT STRING type.

##### Parameters

<i>pctxt</i>	Pointer to a context block structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
--------------	--



## Parameters

<i>object↔ _p</i>	A pointer to an OCTET string containing the bit data to be encoded. The string contains bytes having the actual bit settings as they are to be encoded in the message.
<i>numbits</i>	The number of bits within the bit string to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.6 xe\_bitstrExt()

```
int xe_bitstrExt (
    OSCTXT * pctxt,
    const OSOCTET * object_p,
    OSSIZE numbits,
    OSSIZE dataSize,
    const OSOCTET * extdata,
    ASN1TagType tagging )
```

This function will encode a variable of the ASN.1 BIT STRING type. It includes support for BIT STRING's with extension data.

## Parameters

<i>pctxt</i>	Pointer to a context block structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object↔ _p</i>	A pointer to an OCTET string containing the bit data to be encoded. The string contains bytes having the actual bit settings as they are to be encoded in the message.
<i>numbits</i>	The number of bits within the bit string to be encoded.
<i>dataSize</i>	Size, in octets, of the root bit string data.
<i>extdata</i>	Pointer to byte array containing extension data.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.7 xe\_boolean()

```
int xe_boolean (
    OSCTXT * pctxt,
    OSBOOL * object_p,
    ASN1TagType tagging )
```

This function will encode a variable of the ASN.1 BOOLEAN type.

#### Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
* <i>object_p</i>	A pointer to the BOOLEAN value to be encoded (note that pointer to the BOOLEAN is passed, not the BOOLEAN itself. This may seem awkward, but to keep the calling sequence of all encode functions the same, pointers were used in all cases). A BOOLEAN is defined as a single OCTET whose value is 0 for FALSE and any other value for TRUE.
<i>tagging</i>	An enumerated type whose value is set to either ASN1EXPL (for explicit tagging) or ASN1IMPL (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to ASN1EXPL.

#### Returns

The length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.8 xe\_charstr()

```
int xe_charstr (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging,
    ASN1TAG tag )
```

This function will encode a variable one of the ASN.1 character string types that are based 8-bit character sets. This includes IA5String, VisibleString, PrintableString, and NumericString

#### Parameters

<i>pctxt</i>	Pointer to a context block structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to a null-terminated C character string to be encoded
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The Universal ASN.1 tag to be encoded in the message. This parameter is passed using the internal representation discussed in Section 4.1.2. It is passed as an unsigned 16-bit integer. The tag value must represent one of the 8-bit character string types documented in the X.680 standard.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.9 xe\_datestr()

```
int xe_datestr (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging,
    ASN1TAG tag )
```

This function will encode a variable one of the ASN.1 ISO 8601 Date character string types.

## Parameters

<i>pctxt</i>	Pointer to a context block structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to a null-terminated C character string to be encoded
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The Universal ASN.1 tag to be encoded in the message. This parameter is passed using the internal representation discussed in Section 4.1.2. It is passed as an unsigned 16-bit integer. The tag value must represent one of the 8-bit character string types documented in the X.680 standard.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.10 xe\_datetimestr()

```
int xe_datetimestr (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging,
    ASN1TAG tag )
```

This function will encode a variable one of the ASN.1 ISO 8601 Date/Time character string types.

## Parameters

<i>pctxt</i>	Pointer to a context block structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object← _p</i>	A pointer to a null-terminated C character string to be encoded
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The Universal ASN.1 tag to be encoded in the message. This parameter is passed using the internal representation discussed in Section 4.1.2. It is passed as an unsigned 16-bit integer. The tag value must represent one of the 8-bit character string types documented in the X.680 standard.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.11 xe\_derCanonicalSort()

```
int xe_derCanonicalSort (
    OSCTXT * pctxt,
    OSRTSList * pList )
```

This function is added to the generated code for SEQUENCE OF/SET OF constructs to ensure the elements are in the required canonical order for DER.

If the elements are not in the right order, they are sorted to be in the correct order after encoding.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pList</i>	Linked List of message components to be sorted. The elements of this list are offsets to encoded components within the encode buffer.

## Returns

Completion status of operation:

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

### 6.6.3.12 xe\_derCanSortSet()

```
int xe_derCanSortSet (
    OSCTXT * pctxt,
    OSRTSList * pList )
```

This function is added to the generated code for SET constructs the contain embedded, untagged CHOICE elements to ensure the elements are in the required canonical order for DER. It is also used to sort SET elements in ASN2TXT.

If the elements are not in the right order, they are sorted to be in the correct order after encoding.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pList</i>	Linked List of message components to be sorted. The elements of this list are offsets to encoded components within the encode buffer.

#### Returns

Completion status of operation:

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

### 6.6.3.13 xe\_derReal10()

```
int xe_derReal10 (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging )
```

This function will encode a number from character string to ASN.1 real type with using CER/DER decimal encoding. Number may be represented in integer, decimal, and exponent formats.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	Value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.14 xe\_durationstr()

```
int xe_durationstr (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging,
    ASN1TAG tag )
```

This function will encode a variable one of the ASN.1 ISO 8601 Duration character string types.

#### Parameters

<i>pctxt</i>	Pointer to a context block structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to a null-terminated C character string to be encoded
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The Universal ASN.1 tag to be encoded in the message. This parameter is passed using the internal representation discussed in Section 4.1.2. It is passed as an unsigned 16-bit integer. The tag value must represent one of the 8-bit character string types documented in the X.680 standard.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.15 xe\_enum()

```
int xe_enum (
    OSCTXT * pctxt,
    OSINT32 * object_p,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 ENUMERATED type.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
--------------	--

## Parameters

<i>object↔ _p</i>	A pointer to an integer containing the enumerated value to be encoded (Note that a pointer to the value is passed not the value itself. This may seem awkward, but to keep the calling sequence of all the encode function the same, pointers were used in all cases.)
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.16 xe\_enumUnsigned()

```
int xe_enumUnsigned (
    OSCTXT * pctxt,
    OSUUINT32 * object_p,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 ENUMERATED type.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object↔ _p</i>	A pointer to an integer containing the enumerated value to be encoded (Note that a pointer to the value is passed not the value itself. This may seem awkward, but to keep the calling sequence of all the encode function the same, pointers were used in all cases.)
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.17 xe\_expandBuffer()

```
int xe_expandBuffer (
    OSCTXT * pctxt,
    size_t length )
```

This function will expand a dynamic encode buffer.

The dynamic encode buffer is the buffer that is allocated if dynamic encoding of a message is enabled (passing NULL as the buffer pointer argument to `xe_setp` enables dynamic encoding.)

The size of the new buffer is determined by the length argument. If the length is less than a configurable buffer expansion increment size (the constant `ASN_K_ENCBUFSIZ`), the buffer is expanded by the increment size; otherwise it is expanded by the actual length value.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls. The dynamic encode buffer pointer is contained within the structure.
<i>length</i>	The number of bytes required. This may not be size the size the buffer is actually expanded by. The buffer will be expanded by a fixed-size increment defined by <code>ASN_K_ENCBUFSIZ</code> for small requests to limit the required number of expansions.

#### 6.6.3.18 `xe_free()`

```
void xe_free (
    OSCTXT * pctxt )
```

This function will free a dynamic encode buffer.

The dynamic encode buffer is the buffer that is allocated if dynamic encoding of a message is enabled (passing NULL as the buffer pointer argument to `xe_setp` enables dynamic encoding.)

Note that this is different than the `xu_freeall` function associated with freeing decoder memory. This function only releases the memory associated with a dynamic encoded buffer. The `xu_freeall` will not release this memory.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls. The dynamic encode buffer pointer is contained within the structure.
--------------	---

Referenced by `ASN1BEREncodeBuffer::freeBuffer()`.

#### 6.6.3.19 `xe_getBufLocDescr()`

```
void xe_getBufLocDescr (
    OSCTXT * pctxt,
    OSSIZE length,
    Asn1BufLocDescr * pDescr )
```



#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>length</i>	The actual length of existing message components.
<i>pDescr</i>	A pointer to a buffer location descriptor.

#### 6.6.3.20 xe\_getp()

```
OSOCKET* xe_getp (
    OSCTXT * pctxt )
```

This function is used to obtain a pointer to the start of an encoded message after calls to the encode function(s) are complete. ASN.1 BER messages are encoded from the end of a given buffer toward the beginning. Therefore, in practically all cases, the start of the message will not be at the beginning of the buffer.

#### Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
--------------	--

#### Returns

A pointer to the beginning of the encoded message. If nothing has been encoded, NULL is returned.

#### 6.6.3.21 xe\_identifier()

```
int xe_identifier (
    OSCTXT * pctxt,
    OSUINT32 ident )
```

This function is used to encode an ASN.1 identifier value.

#### Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ident</i>	Identifier value to be encoded.

## Returns

Length of the encoded item.

### 6.6.3.22 xe\_int16()

```
int xe_int16 (
    OSCTXT * pctxt,
    OSINT16 * object_p,
    ASN1TagType tagging )
```

This function encodes a 16-bit variable of the ASN.1 INTEGER type.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>object_p</i>	A pointer to the 16-bit INTEGER value to be encoded (note that a pointer is passed, not the INTEGER value itself. This may seem awkward, but to keep the calling sequence of all encode functions the same, pointers were used in all cases). The OSINT16 type is set to the C type 'short' in the rtxsrc/rtxCommon.h file. This is assumed to represent a 16-bit integer value.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.23 xe\_int64()

```
int xe_int64 (
    OSCTXT * pctxt,
    OSINT64 * object_p,
    ASN1TagType tagging )
```

This function encodes a 64-bit variable of the ASN.1 INTEGER type.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>object_p</i>	A pointer to the 64-bit INTEGER value to be encoded (note that a pointer to the INTEGER is passed, not the INTEGER value itself. This may seem awkward, but to keep the calling sequence of all encode functions the same, pointers were used in all cases). The OSINT64 type is set to the C type '__int64', 'long long' or 'long' in the rtxsrc/rtxCommon.h file (depends on the used platform and the compiler). This is assumed to represent a 64-bit integer value.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.24 xe\_int8()

```
int xe_int8 (
    OSCTXT * pctxt,
    OSINT8 * object_p,
    ASN1TagType tagging )
```

This function encodes an 8-bit variable of the ASN.1 INTEGER type.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>object_p</i>	A pointer to the 8-bit INTEGER value to be encoded (note that a pointer is passed, not the INTEGER value itself. This may seem awkward, but to keep the calling sequence of all encode functions the same, pointers were used in all cases). The OSINT8 type is set to the C type 'signed char' in the rtxsrc/rtxCommon.h file. This is assumed to represent an 8-bit integer value.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.25 xe\_integer()

```
int xe_integer (
    OSCTXT * pctxt,
    int * object_p,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 INTEGER type.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>object_p</i>	A pointer to the INTEGER value to be encoded (note that a pointer to the INTEGER is passed, not the INTEGER value itself. This may seem awkward, but to keep the calling sequence of all encode functions the same, pointers were used in all cases). The OSINT32 type is set to the C type 'int' in the rtxsrc/rtxCommon.h file. This is assumed to represent a 32-bit integer value.

### Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

#### 6.6.3.26 xe\_len()

```
int xe_len (
    OSCTXT * pctxt,
    int length )
```

This function is used to encode BER or DER length determinant values.

### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>length</i>	The length variable to encode. For indefinite length, use ASN_K_INDEFLEN.

### Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

#### 6.6.3.27 xe\_len64()

```
int xe_len64 (
    OSCTXT * pctxt,
    OSSIZE length,
    OSBOOL indef )
```

This function is used to encode BER or DER length determinant values. This variant of the function can be used to encode lengths up to 64-bits in size.

### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>length</i>	The length variable to encode.
<i>indef</i>	Boolean flag indicating indefinite length marker should be encoded.

### Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

**See also**

[xe\\_len](#)

**6.6.3.28 xe\_memcpy()**

```
int xe_memcpy (
    OSCTXT * pctxt,
    const OSOCTET * object_p,
    size_t length )
```

This function is used to copy bytes into the encode buffer. BER and DER messages are encoded from back-to-front and this function will take this into account when copying bytes. It will also check to ensure that enough space is available in the buffer for the bytes to be copied. If the encode buffer is dynamic, it will be expanded to hold the number of bytes to be copied if not enough space is available. If the buffer is static and enough space is not available, an error (RTERR\_BUFOVFLW) will be returned.

**Parameters**

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls. The dynamic encode buffer pointer is contained within the structure.
<i>object_p</i>	A pointer to a buffer containing the bytes to be copied.
<i>length</i>	The number of bytes to copy.

**Returns**

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

**6.6.3.29 xe\_null()**

```
int xe_null (
    OSCTXT * pctxt,
    ASN1TagType tagging )
```

This function will encode an ASN.1 NULL placeholder.

**Parameters**

<i>pctxt</i>	Pointer to a context block structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.30 xe\_objid()

```
int xe_objid (
    OSCTXT * pctxt,
    ASN1OBJID * object_p,
    ASN1TagType tagging )
```

This function encodes a value of the ASN.1 object identifier type.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>object_p</i>	Pointer to value to be encoded. The ASN1OBJID structure contains a numids fields to hold the number of subidentifiers and an array to hold the subidentifier values.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.31 xe\_octstr()

```
int xe_octstr (
    OSCTXT * pctxt,
    const OSOCTET * object_p,
    OSSIZE numocts,
    ASN1TagType tagging )
```

This function will encode a variable of the ASN.1 OCTET STRING type.

#### Parameters

<i>pctxt</i>	Pointer to a context block structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to an OCTET string containing the bit data to be encoded. The string contains bytes having the actual bit settings as they are to be encoded in the message.
<i>numocts</i>	The number of octets (bytes) within the OCTET STRING to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.32 xe\_oid64()

```
int xe_oid64 (
    OSCTXT * pctxt,
    ASN1OID64 * object_p,
    ASN1TagType tagging )
```

This function encodes a value of the ASN.1 object identifier type, using 64-bit subidentifiers.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>object_p</i>	Pointer to value to be encoded. The ASN1OID64 structure contains a numids fields to hold the number of subidentifiers and an array of unsigned 64-bit integers to hold the subidentifier values.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.33 xe\_OpenType()

```
int xe_OpenType (
    OSCTXT * pctxt,
    const OSOCTET * object_p,
    OSSIZE numocts )
```

This function will encode a variable of the old (pre- 1994) ASN.1 ANY type or other elements defined in the later standards to be Open Types (for example, a variable type declaration in a CLASS construct as defined in X.681).

A variable of this type is considered to be previously encoded ASN.1 message component.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to a buffer containing an encoded ASN.1 message component.
<i>numocts</i>	The number of octets to be encoded.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.34 xe\_OpenTypeExt()

```
int xe_OpenTypeExt (
    OSCTXT * pctxt,
    OSRTDList * pElemList )
```

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pElemList</i>	A pointer to open type to be encoded.

### 6.6.3.35 xe\_OpenTypeExtDer()

```
int xe_OpenTypeExtDer (
    OSCTXT * pctxt,
    OSRTDList * pElemList,
    OSRTSList * pBufList )
```

Encode list of unknown extension elements. Add an entry to the buffer list for each element so that `xe_derCanSortSet` can be used later to sort the elements.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pElemList</i>	The unknown extension elements, a list of ASN1OpenType.
<i>pBufList</i>	The list of buffer locations to add to, a list of Asn1BufLocDescr.

### 6.6.3.36 xe\_real()

```
int xe_real (
    OSCTXT * pctxt,
    OSREAL * object_p,
    ASN1TagType tagging )
```



This function will encode a variable of the REAL data type.

This function provides support for the plus-infinity and minus-infinity special real values. Use the `rtxGetPlusInfinity` or `rtxGetMinusInfinity` functions to get these special values.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object↔ _p</i>	A pointer to a variable of the OSREAL data type. This is defined to be the C double type. Special real values plus and minus infinity are encoded by using the <code>rtxGetPlusInfinity</code> and <code>rtxGetMinusInfinity</code> functions to set the real value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

#### 6.6.3.37 `xe_real10()`

```
int xe_real10 (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging )
```

This function will encode a number from character string to ASN.1 real type using decimal encoding. Number may be represented in integer, decimal, and exponent formats.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object↔ _p</i>	Value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.38 xe\_reloid()

```
int xe_reloid (
    OSCTXT * pctxt,
    ASN1OBJID * object_p,
    ASN1TagType tagging )
```

This function encodes a value of the ASN.1 RELATIVE-OID type.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>object_p</i>	Pointer to value to be encoded. The ASN1OBJID structure contains a numids fields to hold the number of subidentifiers and an array to hold the subidentifier values.

#### Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.39 xe\_setp()

```
int xe_setp (
    OSCTXT * pctxt,
    OSOCTET * buf_p,
    OSSIZE bufsiz )
```

This function is used to set the internal buffer within the runtime library encoding context. It must be called after the context variable is initialized by the `rtxInitContext` function and before any other compiler generated or runtime library encode function.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>buf_p</i>	A pointer to a memory buffer to use to encode a message. The buffer should be declared as an array of unsigned characters (OSOCTETs). This parameter can be set to NULL to specify dynamic encoding (i.e., the encode functions will dynamically allocate a buffer for the message).
<i>bufsiz</i>	The length of the memory buffer in bytes.

#### 6.6.3.40 xe\_tag()

```
int xe_tag (
    OSCTXT * pctxt,
    ASN1TAG tag )
```

This function is used to encode an ASN.1 tag value given its parts (class, form, and ID code).

##### Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.

##### Returns

Length of the encoded tag component.

#### 6.6.3.41 xe\_tag\_len()

```
int xe_tag_len (
    OSCTXT * pctxt,
    ASN1TAG tag,
    int length )
```

This function is used to encode ASN.1 tag and length fields that preface each block of message data. The ASN1C compiler generates calls to this function to handle the encoding of user-defined tags within an ASN.1 specification. The function is also called from within the runtime library functions to handle the addition of the universal tags defined for each of the ASN.1 primitive data types.

##### Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.
<i>length</i>	The length of the contents field previously encoded. This parameter can be used to specify the actual length, or the special constant ASN_K_INDFLEN can be used to specify that an indefinite length specification should be encoded.

##### Returns

Length of the encoded message component. This is equal to the given length plus the additional bytes that are added for the tag and length fields. A negative status will be returned if the encoding is not successful.

### 6.6.3.42 xe\_TagAndIndefLen()

```
int xe_TagAndIndefLen (
    OSCTXT * pctxt,
    ASN1TAG tag,
    int length )
```

This function is used to encode a tag value and an indefinite length.

This can be used to manually create an indefinite length wrapper around long records.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tag</i>	ASN.1 tag value to be encoded.
<i>length</i>	The actual length of existing message components.

#### Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.43 xe\_timeofdaystr()

```
int xe_timeofdaystr (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging,
    ASN1TAG tag )
```

This function will encode a variable one of the ASN.1 ISO 8601 Time-of-day character string types.

#### Parameters

<i>pctxt</i>	Pointer to a context block structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to a null-terminated C character string to be encoded
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The Universal ASN.1 tag to be encoded in the message. This parameter is passed using the internal representation discussed in Section 4.1.2. It is passed as an unsigned 16-bit integer. The tag value must represent one of the 8-bit character string types documented in the X.680 standard.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.44 xe\_timestr()

```
int xe_timestr (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging,
    ASN1TAG tag )
```

This function will encode a variable one of the ASN.1 ISO 8601 Time character string types.

#### Parameters

<i>pctxt</i>	Pointer to a context block structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to a null-terminated C character string to be encoded
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The Universal ASN.1 tag to be encoded in the message. This parameter is passed using the internal representation discussed in Section 4.1.2. It is passed as an unsigned 16-bit integer. The tag value must represent one of the 8-bit character string types documented in the X.680 standard.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.45 xe\_uint16()

```
int xe_uint16 (
    OSCTXT * pctxt,
    OSUINT16 * object_p,
    ASN1TagType tagging )
```

This function encodes an unsigned 16-bit variable of the ASN.1 INTEGER type.

#### Parameters

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

## Parameters

<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>object↔ _p</i>	A pointer to the unsigned 16-bit INTEGER value to be encoded (note that a pointer is passed, not the INTEGER value itself. This may seem awkward, but to keep the calling sequence of all encode functions the same, pointers were used in all cases). The OSUINT16 type is set to the C type 'unsigned short' in the rtxsrc/rtxCommon.h file. This is assumed to represent a 16-bit integer value.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.46 xe\_uint64()

```
int xe_uint64 (
    OSCTXT * pctxt,
    OSUINT64 * object_p,
    ASN1TagType tagging )
```

This function encodes an unsigned 64-bit variable of the ASN.1 INTEGER type.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>object↔ _p</i>	A pointer to the unsigned 64-bit INTEGER value to be encoded (note that a pointer to the unsigned INTEGER is passed, not the INTEGER value itself. This may seem awkward, but to keep the calling sequence of all encode functions the same, pointers were used in all cases). The OSUINT64 type is set to the C type 'unsigned __int64', 'unsigned long long' or 'unsigned long' in the rtxsrc/rtxCommon.h file (depends on the used platform and the compiler). This is assumed to represent a 64-bit integer value.

## Returns

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

### 6.6.3.47 xe\_uint8()

```
int xe_uint8 (
    OSCTXT * pctxt,
```

```

OSUINT8 * object_p,
ASN1TagType tagging )

```

This function encodes an unsigned 8-bit variable of the ASN.1 INTEGER type.

**Parameters**

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>object_p</i>	A pointer to the unsigned 8-bit INTEGER value to be encoded (note that a pointer is passed, not the INTEGER value itself. This may seem awkward, but to keep the calling sequence of all encode functions the same, pointers were used in all cases). The OSOCTET type is set to the C type 'unsigned char' in the rtxsrc/rtxCommon.h file. This is assumed to represent an 8-bit integer value.

**Returns**

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

**6.6.3.48 xe\_unsigned()**

```

int xe_unsigned (
    OSCTXT * pctxt,
    OSUINT32 * object_p,
    ASN1TagType tagging )

```

This function encodes an unsigned variable of the ASN.1 INTEGER type.

**Parameters**

<i>pctxt</i>	Pointer to context block structure.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>object_p</i>	A pointer to the unsigned INTEGER value to be encoded (note that a pointer to the unsigned INTEGER is passed, not the INTEGER value itself. This may seem awkward, but to keep the calling sequence of all encode functions the same, pointers were used in all cases). The OSUINT32 type is set to the C type 'unsigned int' in the rtxsrc/rtxCommon.h file. This is assumed to represent a 32-bit integer value.

**Returns**

Length of the encoded message component. A negative status value will be returned if encoding is not successful.

## 6.7 BER/PER C Utility Functions

### Macros

- #define `xu_addTagErrParm` `berErrAddTagParm`
- #define `xu_hex_dump`(msg, numoct, hdr) `rtxHexDump(msg,numoct)`

### Functions

- int `berDefToIndefLen` (OSCTXT \*pSrcCtxt, OSCTXT \*pDstCtxt)
- int `berIndefToDefLen` (OSCTXT \*pSrcCtxt, OSCTXT \*pDstCtxt)
- OSBOOL `berErrAddTagParm` (OSCTXT \*pctxt, ASN1TAG tag)
- int `berErrUnexpTag` (OSCTXT \*pctxt, ASN1TAG expTag)
- int `berGetLibVersion` (OSVOIDARG)
- const char \* `berGetLibInfo` (OSVOIDARG)
- int `berParseTagLen` (const OSOCTET \*buffer, size\_t bufix, size\_t bufsiz, ASN1TAG \*pTag, size\_t \*plen)
- const char \* `berTagToString` (ASN1TAG tag, char \*buffer, size\_t bufsiz)
- const char \* `berTagToDynStr` (OSCTXT \*pctxt, ASN1TAG tag)
- int `berValidateIso8601DateStr` (OSCTXT \*pctxt, const char \*\*ppvalue)
- int `berValidateIso8601DurationStr` (OSCTXT \*pctxt, const char \*\*ppvalue)
- int `berValidateIso8601TimeStr` (OSCTXT \*pctxt, const char \*\*ppvalue)
- int `xu_verify_len` (OSOCTET \*msg\_p)
- void \* `xu_parse_mmbuf` (OSOCTET \*\*buf\_p2, int \*buflen\_p, OSOCTET \*start\_p, int bufsiz)
- void `xu_alloc_array` (OSCTXT \*pctxt, ASN1SeqOf \*seqOf\_p, int recSize, int recCount)
- void `xu_octscopy_s` (OSUINT32 \*nocts\_p, OSOCTET \*data\_p, char \*cstr, char zterm)
- void `xu_octscopy_ss` (ASN1OctStr \*octStr\_p, char \*cstring, char zterm)
- void `xu_octscopy_d` (OSCTXT \*pctxt, OSUINT32 \*nocts\_p, const OSOCTET \*\*data\_p2, char \*cstring, char zterm)
- void `xu_octscopy_ds` (OSCTXT \*pctxt, ASN1DynOctStr \*octStr\_p, char \*cstring, char zterm)
- void `xu_octmcpy_s` (ASN1OctStr \*octStr\_p, void \*data\_p, int datalen)
- void `xu_octmcpy_d` (OSCTXT \*pctxt, ASN1DynOctStr \*octStr\_p, void \*data\_p, int datalen)
- char \* `xu_fetchstr` (int numocts, char \*data)
- int `xu_hexstrcpy` (char \*data, char \*hstring)
- int `xu_binstrcpy` (char \*data, char \*bstring)
- int `xu_dump` (const OSOCTET \*msgptr, ASN1DumpCbFunc cb, void \*cbArg\_p)
- int `xu_fdump` (FILE \*file\_p, const OSOCTET \*msgptr)
- int `xu_dump2` (OSCTXT \*pctxt, const OSOCTET \*msgptr)
- void `xu_fmt_tag` (ASN1TAG \*tag\_p, char \*class\_p, char \*form\_p, char \*id\_code)
- char \* `xu_fmt_tag2` (ASN1TAG \*tag\_p, char \*bufp)
- char \* `xu_fmt_contents` (OSCTXT \*pctxt, int len, int \*count)
- int `xu_fread` (FILE \*fp, OSOCTET \*bufp, int bufsiz)
- void `xu_SaveBufferState` (OSCTXT \*pCtxt, OSRTBufSave \*pSavedInfo)
- void `xu_RestoreBufferState` (OSCTXT \*pCtxt, OSRTBufSave \*pSavedInfo)
- int `berReadMsgFromSocket` (OSCTXT \*pctxt, OSRTSOCKET socket, OSOCTET \*buffer, int bufsiz, OSOCTET \*\*ppDestBuffer, int \*pMessageSize)



## 6.7.1 Detailed Description

BER/DER utility functions are provided for memory management, formatting output of ASN.1 messages, and error reporting. In many cases, these functions are closely coupled with the *rt* (runtime) series of common functions. The common functions provide common functionality shared between the BER/DER, PER, and XER runtime libraries. In many cases, these functions simply provide wrappers to the *rt* functions to maintain compatibility with existing versions of the ASN1C compiler.

## 6.7.2 Macro Definition Documentation

### 6.7.2.1 `xu_hex_dump`

```
#define xu_hex_dump(  
    msg,  
    numoct,  
    hdr ) rtHexDump(msg, numoct)
```

This function dumps binary data in raw hexadecimal and character text formats. It can be used only to examine runtime library encode/decode functions input or output data. This function outputs data only to the standard output device.

This function is considered deprecated and is only being maintained to provide backward compatibility with older versions of ASN1C.

#### Parameters

<i>msg</i>	A pointer to the start of the block of memory to be dumped.
<i>numoct</i>	The number of octets (bytes) to be dumped.
<i>hdr</i>	A Boolean variable indicating whether or not a head line should be dumped as the first line of the display.

## 6.7.3 Function Documentation

### 6.7.3.1 `berDefToIndefLen()`

```
int berDefToIndefLen (  
    OSCTXT * pSrcCtxt,  
    OSCTXT * pDstCtxt )
```

This function converts BER encoded message with definite form of length to BER encoded message with indefinite form of length.

#### Parameters

<i>pSrcCtxt</i>	A pointer to a source OSCTXT structure.
<i>pDstCtxt</i>	A pointer to a destination OSCTXT structure.

#### Returns

Status of the conversion operation. Possible values are 0 if conversion is successful or one of the negative status codes.

#### 6.7.3.2 berErrAddTagParm()

```
OSBOOL berErrAddTagParm (
    OSCTXT * pctxt,
    ASN1TAG tag )
```

This function adds a tag parameter to the context error structure.

#### Parameters

<i>pctxt</i>	Pointer to a context structure.
<i>tag</i>	The tag to add.

#### Returns

Pointer to the text string (buffer).

#### 6.7.3.3 berErrUnexpTag()

```
int berErrUnexpTag (
    OSCTXT * pctxt,
    ASN1TAG exptag )
```

This function logs a BER unexpected tag error (IDNOTFOU) by adding the expected and parsed tag to the error context and returning the error status.

#### Parameters

<i>pctxt</i>	Pointer to a context structure.
<i>exptag</i>	Expected tag.

## Returns

RTERR\_IDNOTFOU status code.

### 6.7.3.4 berGetLibInfo()

```
const char* berGetLibInfo (
    OSVOIDARG )
```

This function returns a string that describes the features of the BER runtime library. Different release kits will return different strings. An example would be "ASN1BER v6.1.1, opt, compact, limited."

### 6.7.3.5 berGetLibVersion()

```
int berGetLibVersion (
    OSVOIDARG )
```

This function returns an integer representation of the BER library version.

### 6.7.3.6 berIndefToDefLen()

```
int berIndefToDefLen (
    OSCTXT * pSrcCtxt,
    OSCTXT * pDstCtxt )
```

This function converts BER encoded message with indefinite form of length to BER encoded message with definite form of length.

#### Parameters

<i>pSrcCtxt</i>	A pointer to a source OSCTXT structure.
<i>pDstCtxt</i>	A pointer to a destination OSCTXT structure.

## Returns

Status of the conversion operation. Possible values are 0 if conversion is successful or one of the negative status codes.

### 6.7.3.7 berParseTagLen()

```
int berParseTagLen (
    const OSOCTET * buffer,
```

```

size_t bufix,
size_t bufsiz,
ASN1TAG * ptag,
size_t * plen )

```

This function parses a tag and length value starting at the given buffer pointer location. This is assumed to be pointing at the beginning of a TLV. It is similar to `xd_tag_len`, except it does not require an initialized context to work. This makes it a better alternative in terms of performance when it is only necessary to get tag and length info without having to do full decoding.

#### Parameters

<i>buffer</i>	Pointer to buffer containing BER-encoded data to parse.
<i>bufidx</i>	Index within buffer that parsing is to begin. It is expected that <code>buffer[bufidx]</code> is pointing at the start of a TLV (tag-length-value).
<i>bufsiz</i>	Size of the BER-encoded message.
<i>ptag</i>	Pointer to tag variable to receive parsed tag value.
<i>plen</i>	Pointer to length variable to receive parsed length.

#### Returns

Pointer to the text string (`buffer`).

#### 6.7.3.8 berReadMsgFromSocket()

```

int berReadMsgFromSocket (
    OSCTXT * pctxt,
    OSRTSOCKET socket,
    OSOCTET * buffer,
    int bufsiz,
    OSOCTET ** ppDestBuffer,
    int * pMessageSize )

```

This routine reads the BER message into the given buffer. The TLV can be of indefinite length. If `buffer` is NULL, dynamic buffer will be allocated and returned as `ppDestBuffer`. Length of the message will be returned in `pMessageSize`.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>socket</i>	The socket to read from. It should be already connected TCP socket.
<i>buffer</i>	The static buffer to receive the parsed data. Can be NULL.
<i>bufsiz</i>	The size of the buffer to receive the parsed data. Should be 0, if <code>buffer</code> is NULL.
<i>ppDestBuffer</i>	The pointer to receive the destination buffer pointer. If <code>buffer</code> is static, this parameter can be NULL.
<i>pMessageSize</i>	The pointer to integer to receive the size of read message.

## Returns

Completion status of operation:

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

### 6.7.3.9 berTagToDynStr()

```
const char* berTagToDynStr (
    OSCTXT * pctxt,
    ASN1TAG tag )
```

This function converts an internal binary ASN.1 tag to a string in standard ASN.1 syntax form. This version creates a dynamic string by allocating memory using the rxMemAlloc function. This memory will be release when context memory is freed.

#### Parameters

<i>pctxt</i>	Pointer to a context structure.
<i>tag</i>	The tag to convert.

## Returns

Pointer to dynamic text string.

### 6.7.3.10 berTagToString()

```
const char* berTagToString (
    ASN1TAG tag,
    char * buffer,
    size_t bufsiz )
```

This function converts an internal binary ASN.1 tag to a string in standard ASN.1 syntax form.

#### Parameters

<i>tag</i>	The tag to convert.
<i>buffer</i>	Text buffer into which the string will be written.
<i>bufsiz</i>	Size of the text buffer.

## Returns

Pointer to the text string (buffer).

### 6.7.3.11 berValidateIso8601DateStr()

```
int berValidateIso8601DateStr (
    OSCTXT * pctxt,
    const char ** ppvalue )
```

This function will validate that an ASN.1 ISO 8601 Date character string is in a proper format. This method is used for both buffer and stream encoding.

#### Parameters

<i>pctxt</i>	Pointer to a context block structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ppvalue</i>	A pointer to a null-terminated C character string to be validated. Note the string will be modified.

## Returns

Status of the validation attempt. A negative status value will be returned if validation is not successful.

### 6.7.3.12 berValidateIso8601DurationStr()

```
int berValidateIso8601DurationStr (
    OSCTXT * pctxt,
    const char ** ppvalue )
```

This function will validate that an ASN.1 ISO 8601 Duration character string is in a proper format. This method is used for both buffer and stream encoding.

#### Parameters

<i>pctxt</i>	Pointer to a context block structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ppvalue</i>	A pointer to a null-terminated C character string to be validated. Note the string will be modified.

## Returns

Status of the validation attempt. A negative status value will be returned if validation is not successful.

### 6.7.3.13 berValidateIso8601TimeStr()

```
int berValidateIso8601TimeStr (
    OSCTXT * pctxt,
    const char ** ppvalue )
```

This function will validate that an ASN.1 ISO 8601 Time character string is in a proper format. This method is used for both buffer and stream encoding.

#### Parameters

<i>pctxt</i>	Pointer to a context block structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ppvalue</i>	A pointer to a null-terminated C character string to be validated. Note the string will be modified.

#### Returns

Status of the validation attempt. A negative status value will be returned if validation is not successful.

### 6.7.3.14 xu\_alloc\_array()

```
void xu_alloc_array (
    OSCTXT * pctxt,
    ASN1SeqOf * seqOf_p,
    int recSize,
    int recCount )
```

This function will allocate space for a given count of fixed size elements.

#### Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>seqOf_p</i>	A pointer to a generic sequence of structure variables to receive the returned memory. This structure contains a record count and a data pointer element. The record count is populated with the <i>recCount</i> passed into the function. The data pointer is set to the value that is returned from the memory allocation function.
<i>recSize</i>	The number of bytes in one record in the array.
<i>recCount</i>	The number of records to allocate. A pointer to a generic sequence of structure variable to receive the returned memory. This structure contains a record count and data pointer element. The record count is populated with the <i>recCount</i> passed into the function. The data pointer is set to the value that is returned from the memory allocation function.

### 6.7.3.15 xu\_dump()

```
int xu_dump (
    const OSOCTET * msgptr,
    ASN1DumpCbFunc cb,
    void * cbArg_p )
```

This function dumps an encoded ASN.1 message to the standard output device or to another interface in a formatted display. The display includes, for each message component, message tag (class, form, and ID code) and data (in hexadecimal and character text formats).

This function is invoked for each line of text formatted from the given message. The formatted line is passed on the `text_p` argument. The `cbArg_p` argument allows a user to defined callback argument to be passed to the callback function. This argument is specified in the call to `xu_dump`.

Use of the callback function is optional. If dump to standard output (`stdout`) is desired, the argument should be specified as `NULL` (note: the macro `XU_DUMP` in `berMacros.h` can be used for this purpose).

#### Parameters

<i>*msgptr</i>	A pointer to an encoded ASN.1 message.
<i>cb</i>	Callback function that gets invoked for each line of formatted output. For a dump to standard output ( <code>stdout</code> ), this parameter can be specified as <code>NULL</code> .
<i>cbArg↔ _p</i>	Callback function argument will be passed to the callback function.

#### Returns

Status of the dump operation. Possible values are 0 if decoding is successful or one of the negative status codes.

Referenced by `ASN1BERMessageBuffer::binDump()`.

### 6.7.3.16 xu\_dump2()

```
int xu_dump2 (
    OSCTXT * pctxt,
    const OSOCTET * msgptr )
```

This function dumps an encoded ASN.1 message to the standard output device or to another interface in a formatted display. The display includes, for each message component, message tag (class, form, and ID code) and data (in hexadecimal and character text formats).

This function is invoked for each line of text formatted from the given message. If a print stream has been registered using the `::rtxSetPrintStream` or `::rtxSetGlobalPrintStream` functions, the output of `xu_dump2` will be sent to the stream instead of standard output.



#### Parameters

<i>*pctxt</i>	A pointer to an OSCTXT structure.
<i>*msgptr</i>	A pointer to an encoded ASN.1 message.

#### Returns

Status of the dump operation. Possible values are 0 if decoding is successful or one of the negative status codes.

#### See also

::rtxSetPrintStream  
::rtxSetGlobalPrintStream

#### 6.7.3.17 xu\_fdump()

```
int xu_fdump (
    FILE * file_p,
    const OSOCTET * msgptr )
```

This function dumps an encoded ASN.1 message to a text file. The display includes, for each message component, message tag (class, form, and ID code), length, and data (in hexadecimal and character text formats).

#### Parameters

<i>*file_p</i>	A text file pointer.
<i>*msgptr</i>	A pointer to an encoded ASN.1 message buffer.

#### Returns

Status of the dump operation. Possible values are 0 if decoding is successful or one of the negative status codes.

#### 6.7.3.18 xu\_RestoreBufferState()

```
void xu_RestoreBufferState (
    OSCTXT * pCtxt,
    OSRTBufSave * pSavedInfo )
```

This function is used to restore the current buffer state from previously saved info.

#### Parameters

<i>pCtxt</i>	- A pointer to a context structure.
<i>pSavedInfo</i>	- A pointer to a structure holding the saved info.

#### 6.7.3.19 xu\_SaveBufferState()

```
void xu_SaveBufferState (
    OSCTXT * pCtxt,
    OSRTBufSave * pSavedInfo )
```

This function is used to save the current buffer state.

#### Parameters

<i>pCtxt</i>	- A pointer to a context structure.
<i>pSavedInfo</i>	- A pointer to a structure to hold the saved info.

## 6.8 Streaming BER Runtime Library Functions.

### Modules

- [C Streaming BER Encode Functions.](#)
- [C Streaming BER Decode Functions.](#)

### Macros

- #define **cerEncCanonicalSort**
- #define **cerGetBufLocDescr**
- #define **cerAddBufLocDescr**
- #define **BS\_CHKEOB**(pctxt)
- #define **BS\_CHKEND**(pctxt, ccb\_p)

### Functions

- int [berStrmInitContext](#) (OSCTXT \*pctxt)
- int [berStrmInitContextUsingKey](#) (OSCTXT \*pctxt, const OSOCTET \*key, size\_t keylen)
- int [berStrmFreeContext](#) (OSCTXT \*pctxt)

#### 6.8.1 Detailed Description

The streaming BER functions handle the BER encode of primitive ASN.1 data types. Calls to these functions are assembled in the C source code generated by the ASN1C compiler (with -stream option) to accomplish the encoding of complex ASN.1 structures. These functions are also directly callable from within a user's application program if the need to accomplish a low level encoding function exists. In contrast to the non-streaming C BER functions, these operate with streams, rather than the memory buffer, the sources or destination. Thus, the data may be encoded directly to the file stream or socket output stream and may be decoded directly from the file or socket input stream.

#### 6.8.2 Macro Definition Documentation

##### 6.8.2.1 BS\_CHKEND

```
#define BS_CHKEND(  
    pctxt,  
    ccb_p )
```

#### Value:

```
((ccb_p)->stat = 0, \  
((ccb_p)->len == ASN_K_INDEFLEN) ? berDecStrmTestEOC (pctxt,ccb_p) : \  
((int) (OSRTSTREAM_BYTEINDEX(pctxt) - (ccb_p)->bytes) >= (ccb_p)->len)))
```

### 6.8.2.2 BS\_CHKEOB

```
#define BS_CHKEOB(  
    pctxt )
```

#### Value:

```
((pctxt)->buffer.byteIndex + 2 > (pctxt)->buffer.size) ? TRUE : \  
((pctxt)->buffer.data[(pctxt)->buffer.byteIndex] == 0 && \  
(pctxt)->buffer.data[(pctxt)->buffer.byteIndex + 1] == 0) ? \  
TRUE : FALSE)
```

### 6.8.2.3 cerAddBufLocDescr

```
#define cerAddBufLocDescr
```

#### Value:

```
(pctxt, pElemList, pDescr) \  
    rtxAddBufLocDescr(pctxt, pElemList, pDescr)
```

### 6.8.2.4 cerEncCanonicalSort

```
#define cerEncCanonicalSort
```

#### Value:

```
(pctxt, pMemCtxt, pList) \  
    rtxEncCanonicalSort(pctxt, pMemCtxt, pList)
```

### 6.8.2.5 cerGetBufLocDescr

```
#define cerGetBufLocDescr
```

#### Value:

```
(pctxt, pDescr) \  
    rtxGetBufLocDescr(pctxt, pDescr)
```

## 6.8.3 Function Documentation

### 6.8.3.1 berStrmFreeContext()

```
int berStrmFreeContext (  
    OSCTXT * pctxt )
```

This function closes the stream and frees all dynamic memory associated with a context.

#### Parameters

<i>pctxt</i>	A pointer to a context structure.
--------------	-----------------------------------

#### Returns

Completion status of operation:

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

#### 6.8.3.2 berStrmInitContext()

```
int berStrmInitContext (  
    OSCTXT * pctxt )
```

This function initializes an OSCTXT block for further streaming operations. It makes sure that if the block was not previously initialized, that all key working parameters are set to their correct initial state values (i.e. declared within a function as a normal working variable). It also initialize stream block.

#### Parameters

<i>pctxt</i>	Pointer to context structure variable to be initialized.
--------------	--

#### Returns

Completion status of operation:

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

## 6.9 C Streaming BER Encode Functions.

### Functions

- int `berEncStrmBigInt` (OSCTXT \*pctx, const char \*pvalue, ASN1TagType tagging)
- int `berEncStrmBigIntNchars` (OSCTXT \*pctx, const char \*pvalue, size\_t nchars, ASN1TagType tagging)
- int `berEncStrmBitStr` (OSCTXT \*pctx, const OSOCTET \*object\_p, OSUINT32 numbits, ASN1TagType tagging)
- int `berEncStrmBMPStr` (OSCTXT \*pctx, const Asn116BitCharString \*object\_p, ASN1TagType tagging)
- int `berEncStrmBool` (OSCTXT \*pctx, OSBOOL value, ASN1TagType tagging)
- int `berEncStrmCharStr` (OSCTXT \*pctx, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int `berEncStrmDateStr` (OSCTXT \*pctx, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int `berEncStrmDateTimeStr` (OSCTXT \*pctx, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int `berEncStrmDurationStr` (OSCTXT \*pctx, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int `berEncStrmTimeStr` (OSCTXT \*pctx, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int `berEncStrmTimeOfDayStr` (OSCTXT \*pctx, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int `berEncStrmDefLength` (OSCTXT \*pctx, size\_t length)
- int `berEncStrmEOC` (OSCTXT \*pctx)
- int `berEncStrmEnum` (OSCTXT \*pctx, OSINT32 value, ASN1TagType tagging)
- int `berEncStrmInt` (OSCTXT \*pctx, OSINT32 value, ASN1TagType tagging)
- int `berEncStrmInt8` (OSCTXT \*pctx, OSINT8 value, ASN1TagType tagging)
- int `berEncStrmInt16` (OSCTXT \*pctx, OSINT16 value, ASN1TagType tagging)
- int `berEncStrmInt64` (OSCTXT \*pctx, OSINT64 value, ASN1TagType tagging)
- int `berEncStrmLength` (OSCTXT \*pctx, int length)
- int `berEncStrmNull` (OSCTXT \*pctx, ASN1TagType tagging)
- int `berEncStrmObjId` (OSCTXT \*pctx, const ASN1OBJID \*object\_p, ASN1TagType tagging)
- int `berEncStrmObjId64` (OSCTXT \*pctx, const ASN1OID64 \*object\_p, ASN1TagType tagging)
- int `berEncStrmOctStr` (OSCTXT \*pctx, const OSOCTET \*object\_p, OSSIZE numocts, ASN1TagType tagging)
- int `berEncStrmOpenTypeExt` (OSCTXT \*pctx, OSRTDList \*pElemList)
- int `berEncStrmReal` (OSCTXT \*pctx, OSREAL value, ASN1TagType tagging)
- int `berEncStrmReal10` (OSCTXT \*pctx, const char \*object\_p, ASN1TagType tagging)
- int `cerEncStrmReal10` (OSCTXT \*pctx, const char \*object\_p, ASN1TagType tagging)
- int `berEncStrmRelativeOID` (OSCTXT \*pctx, const ASN1OBJID \*object\_p, ASN1TagType tagging)
- int `berEncStrmTag` (OSCTXT \*pctx, ASN1TAG tag)
- int `berEncStrmTagAndLen` (OSCTXT \*pctx, ASN1TAG tag, int length)
- int `berEncStrmTagAndDefLen` (OSCTXT \*pctx, ASN1TAG tag, OSSIZE length)
- int `berEncStrmTagAndIndefLen` (OSCTXT \*pctx, ASN1TAG tag)
- int `berEncStrmUInt` (OSCTXT \*pctx, OSUINT32 value, ASN1TagType tagging)
- int `berEncStrmUInt8` (OSCTXT \*pctx, OSUINT8 value, ASN1TagType tagging)
- int `berEncStrmUInt16` (OSCTXT \*pctx, OSUINT16 value, ASN1TagType tagging)
- int `berEncStrmUInt64` (OSCTXT \*pctx, OSUINT64 value, ASN1TagType tagging)
- int `berEncStrmUnivStr` (OSCTXT \*pctx, const Asn132BitCharString \*object\_p, ASN1TagType tagging)
- int `berEncStrmXSDAny` (OSCTXT \*pctx, OSXSDAny \*pvalue, ASN1TagType tagging)
- int `berEncStrmWriteOctet` (OSCTXT \*pctx, OSOCTET octet)
- int `berEncStrmWriteOctets` (OSCTXT \*pctx, const OSOCTET \*pOctets, size\_t numocts)
- int `cerEncStrmBMPStr` (OSCTXT \*pctx, const Asn116BitCharString \*object\_p, ASN1TagType tagging)
- int `cerEncStrmBitStr` (OSCTXT \*pctx, const OSOCTET \*object\_p, OSUINT32 numbits, ASN1TagType tagging)
- int `cerEncStrmCharStr` (OSCTXT \*pctx, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int `cerEncStrmOctStr` (OSCTXT \*pctx, const OSOCTET \*object\_p, OSUINT32 numocts, ASN1TagType tagging)
- int `cerEncStrmUnivStr` (OSCTXT \*pctx, const Asn132BitCharString \*object\_p, ASN1TagType tagging)

## 6.9.1 Detailed Description

The C streaming BER encode functions encode ASN.1 primitive data types directly to the output stream. It must be already opened by using any of the following functions: `::rtxStreamFileOpen`, `::rtxStreamFileAttach`, `::rtxStreamSocketAttach`, `::rtxStreamMemoryCreate`, `::rtxStreamMemoryAttach`.

The streaming BER encoding uses indefinite length form for encode complex ASN.1 types.

## 6.9.2 Function Documentation

### 6.9.2.1 `berEncStrmBigInt()`

```
int berEncStrmBigInt (
    OSCTXT * pctxt,
    const char * pvalue,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 INTEGER type. In this case, the integer is assumed to be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits). For example, parameters used to calculate security values are typically larger than these sizes.

Items of this type are stored in character string constant variables. They can be represented as decimal strings (with no prefixes), as hexadecimal strings starting with a "0x" prefix, as octal strings starting with a "0o" prefix or as binary strings starting with a "0b" prefix. Other radices currently are not supported. It is highly recommended to use the hexadecimal or binary strings for better performance.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pvalue</i>	A pointer to a character string containing the value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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

### 6.9.2.2 `berEncStrmBigIntNchars()`

```
int berEncStrmBigIntNchars (
    OSCTXT * pctxt,
```

```

const char * pvalue,
size_t nchars,
ASN1TagType tagging )

```

This function is similar to the `berEncStrmBigInt` function except that only the given number of characters from the character string are used as the value to be encoded.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pvalue</i>	A pointer to a character string containing the value to be encoded.
<i>nchars</i>	Number of characters from <i>pvalue</i> to encode.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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

#### 6.9.2.3 berEncStrmBitStr()

```

int berEncStrmBitStr (
    OSCTXT * pctxt,
    const OSOCTET * object_p,
    OSUINTEGER numbits,
    ASN1TagType tagging )

```

This function encodes a variable of the ASN.1 BIT STRING type.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to an OCTET string containing the bit data to be encoded. This string contains bytes having the actual bit settings as they are to be encoded in the message.
<i>numbits</i>	The number of bits within the bit string to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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



#### 6.9.2.4 berEncStrmBMPStr()

```
int berEncStrmBMPStr (
    OSCTXT * pctxt,
    const Asn116BitCharString * object_p,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 BMPString type that is based on a 16-bit character sets.

##### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to a structure representing a 16-bit character string to be encoded. This structure contains a character count element and a pointer to an array of 16-bit character elements represented as 16-bit short integers.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

##### Returns

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

#### 6.9.2.5 berEncStrmBool()

```
int berEncStrmBool (
    OSCTXT * pctxt,
    OSBOOL value,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 BOOLEAN type.

##### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	A BOOLEAN value to be encoded. A BOOLEAN is defined as a single OCTET whose value is 0 for FALSE and any other value for TRUE.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

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

### 6.9.2.6 berEncStrmCharStr()

```
int berEncStrmCharStr (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging,
    ASN1TAG tag )
```

This function encodes a variable one of the ASN.1 character string types that are based on 8-bit character sets. This includes IA5String, VisibleString, PrintableString, and NumericString.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to a null-terminated C character string to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.

## Returns

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

### 6.9.2.7 berEncStrmDateStr()

```
int berEncStrmDateStr (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging,
    ASN1TAG tag )
```

This function encodes a variable one of the ASN.1 ISO 8601 Date character string types.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object↔ _p</i>	A pointer to a null-terminated C character string to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.

## Returns

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

### 6.9.2.8 berEncStrmDateTimeStr()

```
int berEncStrmDateTimeStr (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging,
    ASN1TAG tag )
```

This function encodes a variable one of the ASN.1 ISO 8601 Date/Time character string types.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object↔ _p</i>	A pointer to a null-terminated C character string to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.

## Returns

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

### 6.9.2.9 berEncStrmDefLength()

```
int berEncStrmDefLength (
    OSCTXT * pctxt,
    size_t length )
```

This function encodes a definite length field to the stream.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>length</i>	Length to be encoded.

#### Returns

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

### 6.9.2.10 berEncStrmDurationStr()

```
int berEncStrmDurationStr (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging,
    ASN1TAG tag )
```

This function encodes a variable one of the ASN.1 ISO 8601 Duration character string types.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to a null-terminated C character string to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.

#### Returns

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

### 6.9.2.11 berEncStrmEnum()

```
int berEncStrmEnum (
    OSCTXT * pctxt,
    OSINT32 value,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 ENUMERATED type. The enumerated encoding is identical to that of an integer. The compiler adds additional checks to the generated code to ensure the value is within the given set.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	An integer containing the enumerated value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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

### 6.9.2.12 berEncStrmEOC()

```
int berEncStrmEOC (
    OSCTXT * pctxt )
```

This function encodes end-of-contents octets (EOC) into the stream. EOC is two zero octets (it is documented in the X.690 standard). This function must be called when the encoding of the complex type with indefinite length is finishing (see [berEncStrmTagAndIndefLen](#)).

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
--------------	--

#### Returns

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

### 6.9.2.13 berEncStrmInt()

```
int berEncStrmInt (
    OSCTXT * pctxt,
    OSINT32 value,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 INTEGER type.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	An INTEGER value to be encoded. The OSINT32 type is set to the C type 'int' in the asn1type.h file. This is assumed to represent a 32-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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

### 6.9.2.14 berEncStrmInt16()

```
int berEncStrmInt16 (
    OSCTXT * pctxt,
    OSINT16 value,
    ASN1TagType tagging )
```

This function encodes a 16-bit variable of the ASN.1 INTEGER type.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	A 16-bit INTEGER value to be encoded. The OSINT16 type is set to the C type 'signed short' in the asn1type.h file. This is assumed to represent a 16-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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

### 6.9.2.15 berEncStrmInt64()

```
int berEncStrmInt64 (
    OSCTXT * pctxt,
    OSINT64 value,
    ASN1TagType tagging )
```

This function encodes a 64-bit variable of the ASN.1 INTEGER type.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	A 64-bit INTEGER value to be encoded. The OSINT64 type is set to the C type '__int64', 'long long' or 'long' in the asn1type.h file (depends on the used platform and the compiler). This is assumed to represent a 64-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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

### 6.9.2.16 berEncStrmInt8()

```
int berEncStrmInt8 (
    OSCTXT * pctxt,
    OSINT8 value,
    ASN1TagType tagging )
```

This function encodes an 8-bit variable of the ASN.1 INTEGER type.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	An 8-bit INTEGER value to be encoded. The OSINT8 type is set to the C type 'signed char' in the asn1type.h file. This is assumed to represent an 8-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

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

### 6.9.2.17 berEncStrmLength()

```
int berEncStrmLength (
    OSCTXT * pctxt,
    int length )
```

This function is used to encode a BER length determinant value.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>length</i>	The length variable to encode. An ASN_K_INDEFLEN constant is interpreted that an indefinite length identifier should be encoded.

## Returns

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

### 6.9.2.18 berEncStrmNull()

```
int berEncStrmNull (
    OSCTXT * pctxt,
    ASN1TagType tagging )
```

This function encodes an ASN.1 NULL placeholder.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

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



### 6.9.2.19 berEncStrmObjId()

```
int berEncStrmObjId (
    OSCTXT * pctxt,
    const ASN1OBJID * object_p,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 OBJECT IDENTIFIER type.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to an object identifier structure. This structure contains an integer to hold the number of subidentifiers in the object and an array to hold the subidentifier values.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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

### 6.9.2.20 berEncStrmObjId64()

```
int berEncStrmObjId64 (
    OSCTXT * pctxt,
    const ASN1OID64 * object_p,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 OBJECT IDENTIFIER type using 64-bit subidentifiers.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to a 64-bit object identifier structure. This structure contains an integer to hold the number of subidentifiers in the object and an array of 64-bit unsigned integers to hold the subidentifier values.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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

### 6.9.2.21 berEncStrmOctStr()

```
int berEncStrmOctStr (
    OSCTXT * pctxt,
    const OSOCTET * object_p,
    OSSIZE numocts,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 OCTET STRING type.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to an OCTET STRING containing the octet data to be encoded.
<i>numocts</i>	The number of octets (bytes) within the OCTET STRING to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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

### 6.9.2.22 berEncStrmOpenTypeExt()

```
int berEncStrmOpenTypeExt (
    OSCTXT * pctxt,
    OSRTDList * pElemList )
```

This function encodes an ASN.1 open type extension. An open type extension is defined as an extensibility marker on a constructed type without any extension elements defined (for example, SEQUENCE { a INTEGER, : }). The difference is that this is an implicit field that can span one or more elements whereas the standard Open Type is assumed to be a single tagged field.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pElemList</i>	The pointer to linked list structure. The list will contain elements of ASN1OpenType type.

#### Returns

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

### 6.9.2.23 berEncStrmReal()

```
int berEncStrmReal (
    OSCTXT * pctxt,
    OSREAL value,
    ASN1TagType tagging )
```

This function encodes a variable of the REAL data type. This function provides support for the plus-infinity and minus-infinity special real values. Use the ::rtxGetPlusInfinity or ::rtxGetMinusInfinity functions to get these special values.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	An OSREAL data type. This is defined to be the C double type. Special real values plus and minus infinity are encoded by using the ::rtxGetPlusInfinity and ::rtxGetMinusInfinity functions to set the real value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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

### 6.9.2.24 berEncStrmReal10()

```
int berEncStrmReal10 (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging )
```

This function will encode a number from character string to ASN.1 real type using decimal encoding. Number may be represented in integer, decimal, and exponent formats.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	Value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

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

### 6.9.2.25 berEncStrmRelativeOID()

```
int berEncStrmRelativeOID (
    OSCTXT * pctxt,
    const ASN1OBJID * object_p,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 RELATIVE-OID type.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to an object identifier structure. This structure contains an integer to hold the number of subidentifiers in the object and an array to hold the subidentifier values.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

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

### 6.9.2.26 berEncStrmTag()

```
int berEncStrmTag (
    OSCTXT * pctxt,
    ASN1TAG tag )
```

This function is used to encode the ASN.1 tag field that preface each block of message data. The ASN1C compiler generates calls to this function to handle the encoding of user-defined tags within an ASN.1 specification.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.

## Returns

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

### 6.9.2.27 berEncStrmTagAndDefLen()

```
int berEncStrmTagAndDefLen (
    OSCTXT * pctxt,
    ASN1TAG tag,
    OSSIZE length )
```

This function is used to encode the ASN.1 tag and length fields that preface each block of message data. The ASN1C compiler generates calls to this function to handle the encoding of user-defined tags within an ASN.1 specification. This function is also called from within the run-time library functions to handle the addition of the universal tags defined for each of the ASN.1 primitive data types. This version will always encode the length value in definite length form.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.
<i>length</i>	The length of the contents field.

## Returns

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

### 6.9.2.28 berEncStrmTagAndIndefLen()

```
int berEncStrmTagAndIndefLen (
    OSCTXT * pctxt,
    ASN1TAG tag )
```

This function is used to encode a tag value and an indefinite length. This can be used to manually create an indefinite length wrapper around long or constructed records.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.

## Returns

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

### 6.9.2.29 berEncStrmTagAndLen()

```
int berEncStrmTagAndLen (
    OSCTXT * pctxt,
    ASN1TAG tag,
    int length )
```

This function is used to encode the ASN.1 tag and length fields that preface each block of message data. The ASN1C compiler generates calls to this function to handle the encoding of user-defined tags within an ASN.1 specification. This function is also called from within the run-time library functions to handle the addition of the universal tags defined for each of the ASN.1 primitive data types.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.
<i>length</i>	The length of the contents field. This parameter can be used to specify the actual length, or the special constant 'ASN_K_INDFLEN' can be used to specify that an indefinite length specification should be encoded.

## Returns

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

### 6.9.2.30 berEncStrmTimeOfDayStr()

```
int berEncStrmTimeOfDayStr (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging,
    ASN1TAG tag )
```

This function encodes a variable one of the ASN.1 ISO 8601 Time-Of-Day character string types.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
--------------	--

## Parameters

<i>object↔ _p</i>	A pointer to a null-terminated C character string to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.

## Returns

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

### 6.9.2.31 berEncStrmTimeStr()

```
int berEncStrmTimeStr (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging,
    ASN1TAG tag )
```

This function encodes a variable one of the ASN.1 ISO 8601 Time character string types.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object↔ _p</i>	A pointer to a null-terminated C character string to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.

## Returns

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

### 6.9.2.32 berEncStrmUInt()

```
int berEncStrmUInt (
    OSCTXT * pctxt,
```

```

    OSUINT32 value,
    ASN1TagType tagging )

```

This function encodes an unsigned variable of the ASN.1 INTEGER type.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	An unsigned INTEGER value to be encoded. The OSUINT32 type is set to the C type 'unsigned int' in the <code>asn1type.h</code> file. This is assumed to represent a 32-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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

#### 6.9.2.33 berEncStrmUInt16()

```

int berEncStrmUInt16 (
    OSCTXT * pctxt,
    OSUINT16 value,
    ASN1TagType tagging )

```

This function encodes an unsigned 16-bit variable of the ASN.1 INTEGER type.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	An unsigned 16-bit INTEGER value to be encoded. The OSUINT16 type is set to the C type 'unsigned short' in the <code>asn1type.h</code> file. This is assumed to represent a 16-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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



### 6.9.2.34 berEncStrmUInt64()

```
int berEncStrmUInt64 (
    OSCTXT * pctxt,
    OSUINT64 value,
    ASN1TagType tagging )
```

This function encodes an unsigned 64-bit variable of the ASN.1 INTEGER type.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	An unsigned 64-bit INTEGER value to be encoded. The OSUINT64 type is set to the C type 'unsigned __int64', 'unsigned long long' or 'unsigned long' in the asn1type.h file (depends on the used platform and the compiler). This is assumed to represent a 64-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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

### 6.9.2.35 berEncStrmUInt8()

```
int berEncStrmUInt8 (
    OSCTXT * pctxt,
    OSUINT8 value,
    ASN1TagType tagging )
```

This function encodes an unsigned 8-bit variable of the ASN.1 INTEGER type.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	An unsigned 8-bit INTEGER value to be encoded. The OSOCTET type is set to the C type 'unsigned char' in the asn1type.h file. This is assumed to represent an 8-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

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

### 6.9.2.36 berEncStrmUnivStr()

```
int berEncStrmUnivStr (
    OSCTXT * pctxt,
    const Asn132BitCharString * object_p,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 UniversalString type that is based on a 32-bit character sets.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to a structure representing a 32-bit character string to be encoded. This structure contains a character count element and a pointer to an array of 32-bit character elements represented as 32-bit unsigned integers.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

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

### 6.9.2.37 berEncStrmWriteOctet()

```
int berEncStrmWriteOctet (
    OSCTXT * pctxt,
    OSOCTET octet )
```

This function puts one octet into the output stream. It is used inside the run-time library or may be used by user to encode indicator of indefinite length (0x80, as defined in ITU-T X.690 standard).

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>octet</i>	The octet to be encoded.

## Returns

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

### 6.9.2.38 berEncStrmWriteOctets()

```
int berEncStrmWriteOctets (
    OSCTXT * pctxt,
    const OSOCTET * poctets,
    size_t numocts )
```

This function puts an array of octets into the output stream. It is used inside the run-time library or may be used by user to encode end-of-contents octets (EOC) or open type's content.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>poctets</i>	The array of octets to be encoded.
<i>numocts</i>	The number of octets in the array.

## Returns

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

### 6.9.2.39 berEncStrmXSDAny()

```
int berEncStrmXSDAny (
    OSCTXT * pctxt,
    OSXSDAny * pvalue,
    ASN1TagType tagging )
```

This function encodes a variable of the XSD any element wildcard type. It is only used in generated code when and XSD file is compiled. It provides the option to encode the wildcard element as XML text or in binary form if ASN.1 binary encoding rules are being used.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pvalue</i>	A pointer to a structure representing an XSD Any (xsd:any) type to be encoded. The structure contains a union of XML text or binary data.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

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

### 6.9.2.40 cerEncStrmBitStr()

```
int cerEncStrmBitStr (
    OSCTXT * pctxt,
    const OSOCTET * object_p,
    OSUINT32 numbits,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 BIT STRING type with using Canonical Encoding Rules (CER).

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to an OCTET string containing the bit data to be encoded. This string contains bytes having the actual bit settings as they are to be encoded in the message.
<i>numbits</i>	The number of bits within the bit string to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

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

### 6.9.2.41 cerEncStrmBMPStr()

```
int cerEncStrmBMPStr (
    OSCTXT * pctxt,
    const Asn116BitCharString * object_p,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 BMPString type with using Canonical Encoding Rules (CER). BMPString type is based on a 16-bit character sets.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
--------------	--

## Parameters

<i>object</i> ↔ <i>_p</i>	A pointer to a structure representing a 16-bit character string to be encoded. This structure contains a character count element and a pointer to an array of 16-bit character elements represented as 16-bit short integers.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

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

### 6.9.2.42 cerEncStrmCharStr()

```
int cerEncStrmCharStr (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging,
    ASN1TAG tag )
```

This function encodes a variable one of the ASN.1 character string types that are based on 8-bit character sets with using Canonical Encoding Rules (CER). This includes IA5String, VisibleString, PrintableString, and NumericString.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object</i> ↔ <i>_p</i>	A pointer to a null-terminated C character string to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.

## Returns

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

### 6.9.2.43 cerEncStrmOctStr()

```
int cerEncStrmOctStr (
    OSCTXT * pctxt,
```

```

const OSOCTET * object_p,
OSUINT32 numocts,
ASN1TagType tagging )

```

This function encodes a variable of the ASN.1 OCTET STRING type with using Canonical Encoding Rules (CER).

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to an OCTET STRING containing the octet data to be encoded.
<i>numocts</i>	The number of octets (bytes) within the OCTET STRING to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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

#### 6.9.2.44 cerEncStrmReal10()

```

int cerEncStrmReal10 (
    OSCTXT * pctxt,
    const char * object_p,
    ASN1TagType tagging )

```

This function will encode a number from character string to ASN.1 real type with using CER/DER decimal encoding. Number may be represented in integer, decimal, and exponent formats.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	Value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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

#### 6.9.2.45 cerEncStrmUnivStr()

```
int cerEncStrmUnivStr (
    OSCTXT * pctxt,
    const Asn132BitCharString * object_p,
    ASN1TagType tagging )
```

This function encodes a variable of the ASN.1 UniversalString type with using Canonical Encoding Rules (CER). UniversalString type is based on a 32-bit character sets.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	A pointer to a structure representing a 32-bit character string to be encoded. This structure contains a character count element and a pointer to an array of 32-bit character elements represented as 32-bit unsigned integers.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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

## 6.10 C Streaming BER Decode Functions.

### Functions

- int [berDecStrmBMPStr](#) (OSCTXT \*pctxt, Asn116BitCharString \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmBigInt](#) (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmBigEnum](#) (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmBitStr](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnbits, ASN1TagType tagging, int length)
- int [berDecStrmBool](#) (OSCTXT \*pctxt, OSBOOL \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmCharStr](#) (OSCTXT \*pctxt, const char \*\*ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int [berDecStrmDateStr](#) (OSCTXT \*pctxt, const char \*\*ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int [berDecStrmDateTimeStr](#) (OSCTXT \*pctxt, const char \*\*ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int [berDecStrmDurationStr](#) (OSCTXT \*pctxt, const char \*\*ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int [berDecStrmTimeStr](#) (OSCTXT \*pctxt, const char \*\*ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int [berDecStrmTimeOfDayStr](#) (OSCTXT \*pctxt, const char \*\*ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- OSBOOL [berDecStrmCheckEnd](#) (OSCTXT \*pctxt, ASN1CCB \*pccb)
- int [berDecStrmDynBitStr](#) (OSCTXT \*pctxt, const OSOCTET \*\*ppvalue, OSUINT32 \*pnbits, ASN1TagType tagging, int length)
- int [berDecStrmDynBitStr64](#) (OSCTXT \*pctxt, const OSOCTET \*\*ppvalue, OSSIZE \*pnbits, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int [berDecStrmDynOctStr](#) (OSCTXT \*pctxt, const OSOCTET \*\*ppvalue, OSUINT32 \*pnocts, ASN1TagType tagging, int length)
- int [berDecStrmDynOctStr64](#) (OSCTXT \*pctxt, OSOCTET \*\*ppvalue, OSSIZE \*pnocts, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int [berDecStrmEnum](#) (OSCTXT \*pctxt, OSINT32 \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmFindTag](#) (OSCTXT \*pctxt, ASN1TAG tag, int \*len\_p, OSBOOL advance)
- int [berDecStrmFindTag2](#) (OSCTXT \*pctxt, ASN1TAG tag, OSSIZE \*len\_p, OSBOOL \*plndefLen, OSBOOL advance)
- int [berDecStrmGetTLVLength](#) (OSCTXT \*pctxt)
- int [berDecStrmInt](#) (OSCTXT \*pctxt, OSINT32 \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmInt8](#) (OSCTXT \*pctxt, OSINT8 \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmInt16](#) (OSCTXT \*pctxt, OSINT16 \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmInt64](#) (OSCTXT \*pctxt, OSINT64 \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmLength](#) (OSCTXT \*pctxt, int \*len\_p)
- int [berDecStrmLength2](#) (OSCTXT \*pctxt, OSSIZE \*pLength, OSBOOL \*plndefLen)
- int [berDecStrmMatchEOC](#) (OSCTXT \*pctxt)
- int [berDecStrmMatchTag](#) (OSCTXT \*pctxt, ASN1TAG tag, int \*len\_p, OSBOOL advance)
- int [berDecStrmMatchTag2](#) (OSCTXT \*pctxt, ASN1TAG tag, OSSIZE \*len\_p, OSBOOL \*plndefLen, OSBOOL advance)
- int [berDecStrmNextElement](#) (OSCTXT \*pctxt)
- int [berDecStrmNull](#) (OSCTXT \*pctxt, ASN1TagType tagging)
- int [berDecStrmObjId](#) (OSCTXT \*pctxt, ASN1OBJID \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmObjId64](#) (OSCTXT \*pctxt, ASN1OID64 \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmOctStr](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, ASN1TagType tagging, int length)
- int [berDecStrmOpenType](#) (OSCTXT \*pctxt, const OSOCTET \*\*object\_p2, OSSIZE \*pnumocts)
- int [berDecStrmOpenTypeAppend](#) (OSCTXT \*pctxt, OSRTDList \*pElemList)



- int `berDecStrmOpenTypeExt` (OSCTXT \*pctxt, ASN1CCB \*ccb\_p, ASN1TAG \*tags, int tagCount, OSRTDList \*pElemList)
- int `berDecStrmPeekTagAndLen` (OSCTXT \*pctxt, ASN1TAG \*ptag, int \*plen)
- int `berDecStrmReadDynTLV` (OSCTXT \*pctxt, OSOCTET \*\*ppbuf)
- int `berDecStrmReadTLV` (OSCTXT \*pctxt, OSOCTET \*buf, OSSIZE bufsiz)
- int `berDecStrmReal` (OSCTXT \*pctxt, OSREAL \*object\_p, ASN1TagType tagging, int length)
- int `berDecStrmReal10` (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, int length)
- int `berDecStrmRelativeOID` (OSCTXT \*pctxt, ASN1OBJID \*object\_p, ASN1TagType tagging, int length)
- int `berDecStrmTag` (OSCTXT \*pctxt, ASN1TAG \*tag\_p)
- int `berDecStrmTagAndLen` (OSCTXT \*pctxt, ASN1TAG \*tag\_p, int \*len\_p)
- int `berDecStrmTagAndLen2` (OSCTXT \*pctxt, ASN1TAG \*tag\_p, OSSIZE \*len\_p, OSBOOL \*pIndefLen)
- OSBOOL `berDecStrmTestEOC` (OSCTXT \*pctxt, ASN1CCB \*ccb\_p)
- OSBOOL `berDecStrmTestEOC2` (OSCTXT \*pctxt)
- int `berDecStrmTestTag` (OSCTXT \*pctxt, ASN1TAG tag, int \*len\_p, OSBOOL advance)
- int `berDecStrmUInt` (OSCTXT \*pctxt, OSUINT32 \*object\_p, ASN1TagType tagging, int length)
- int `berDecStrmUInt8` (OSCTXT \*pctxt, OSUINT8 \*object\_p, ASN1TagType tagging, int length)
- int `berDecStrmUInt16` (OSCTXT \*pctxt, OSUINT16 \*object\_p, ASN1TagType tagging, int length)
- int `berDecStrmUInt64` (OSCTXT \*pctxt, OSUINT64 \*object\_p, ASN1TagType tagging, int length)
- int `berDecStrmUnivStr` (OSCTXT \*pctxt, Asn132BitCharString \*object\_p, ASN1TagType tagging, int length)

### 6.10.1 Detailed Description

The C streaming BER Decode Functions decode ASN.1 primitive data types directly from the input stream. The input stream should be initialized as buffered stream by using `::rtxStreamInit` function. It also must be already opened by using any of the following functions: `::rtxStreamFileOpen`, `::rtxStreamFileAttach`, `::rtxStreamSocketAttach`, `::rtxStream←MemoryCreate`, `::rtxStreamMemoryAttach`.

### 6.10.2 Function Documentation

#### 6.10.2.1 `berDecStrmBigInt()`

```
int berDecStrmBigInt (
    OSCTXT * pctxt,
    const char ** object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a variable of the ASN.1 INTEGER type. In this case, the integer is assumed to be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits). For example, parameters used to calculate security values are typically larger than these sizes.

These variables are stored in character string constant variables. They are represented as hexadecimal strings starting with a "0x" prefix. If it is necessary to convert a hexadecimal string to another radix, then use the `::rtxBigIntSetStr / ::rtxBigIntToString` functions.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object← _p</i>	Pointer to a character pointer variable to receive the decoded value. Dynamic memory is allocated for the variable using the <code>::rtxMemAlloc</code> function. The decoded variable is represented as a hexadecimal string starting with a "0x" prefix.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

### 6.10.2.2 berDecStrmBitStr()

```
int berDecStrmBitStr (
    OSCTXT * pctxt,
    OSOCTET * pvalue,
    OSUINT32 * pnbits,
    ASN1TagType tagging,
    int length )
```

This function decodes a variable of the ASN.1 BIT STRING type into a static memory structure. This function call is generated by ASN1C to decode a sized bit string production.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pvalue</i>	Pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of bits specified in the <code>*pnbits</code> input parameter.
<i>pnbits</i>	As input parameter it is a pointer to an integer variable containing the size (in bits) of the sized ASN.1 bit string. An error will occur if the number of bits in the decoded string is larger than this value. Note that this is also used as an output variable - the actual number of decoded bits will be returned in this variable.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

### 6.10.2.3 berDecStrmBMPStr()

```
int berDecStrmBMPStr (
    OSCTXT * pctxt,
    Asn116BitCharString * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a variable an ASN.1 16-bit character string type. This includes the BMPString type.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	Pointer to a structure variable to receive the decoded string. The string is stored as an array of short integer characters. Memory is allocated for the string by the ::rtxMemAlloc function.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

### 6.10.2.4 berDecStrmBool()

```
int berDecStrmBool (
    OSCTXT * pctxt,
    OSBOOL * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a variable of the ASN.1 BOOLEAN type.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object</i> ↔ <i>_p</i>	Pointer to a variable to receive the decoded BOOLEAN value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

### 6.10.2.5 berDecStrmCharStr()

```
int berDecStrmCharStr (
    OSCTXT * pctxt,
    const char ** ppvalue,
    ASN1TagType tagging,
    ASN1TAG tag,
    int length )
```

This function decodes a variable of one of the ASN.1 8-bit character string types. These types include IA5String, VisibleString, PrintableString, and NumericString.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ppvalue</i>	Pointer to a character string pointer variable to receive the decoded string. The string is stored as a standard null-terminated C string. Memory is allocated for the string by the ::rtxMemAlloc function.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The ASN.1 tag to be decoded. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer. This parameter only has meaning if the tagging parameter specifies explicit decoding.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

### 6.10.2.6 berDecStrmCheckEnd()

```
OSBOOL berDecStrmCheckEnd (
    OSCTXT * pctxt,
    ASN1CCB * pccb )
```

This function tests for the end of a constructed context. It is used in loop control statements to determine when a block of repeating elements has ended.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pccb</i>	Pointer to a 'context control block' structure. This is basically a loop control mechanism to keep the variable associated with parsing a nested constructed element straight.

## Returns

A result of testing:

- TRUE, if end of current context has been reached;
- FALSE, otherwise.

### 6.10.2.7 berDecStrmDateStr()

```
int berDecStrmDateStr (
    OSCTXT * pctxt,
    const char ** ppvalue,
    ASN1TagType tagging,
    ASN1TAG tag,
    int length )
```

This function decodes a variable of one of the ASN.1 ISO 8601 Date character string types.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ppvalue</i>	Pointer to a character string pointer variable to receive the decoded string. The string is stored as a standard null-terminated C string. Memory is allocated for the string by the ::rtxMemAlloc function.

#### Parameters

<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The ASN.1 tag to be decoded. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer. This parameter only has meaning if the tagging parameter specifies explicit decoding.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

#### Returns

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

#### 6.10.2.8 berDecStrmDateTimeStr()

```
int berDecStrmDateTimeStr (
    OSCTXT * pctxt,
    const char ** ppvalue,
    ASN1TagType tagging,
    ASN1TAG tag,
    int length )
```

This function decodes a variable of one of the ASN.1 ISO 8601 Date/Time character string types.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ppvalue</i>	Pointer to a character string pointer variable to receive the decoded string. The string is stored as a standard null-terminated C string. Memory is allocated for the string by the ::rtxMemAlloc function.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The ASN.1 tag to be decoded. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer. This parameter only has meaning if the tagging parameter specifies explicit decoding.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

#### Returns

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

### 6.10.2.9 berDecStrmDurationStr()

```
int berDecStrmDurationStr (
    OSCTXT * pctxt,
    const char ** ppvalue,
    ASN1TagType tagging,
    ASN1TAG tag,
    int length )
```

This function decodes a variable of one of the ASN.1 ISO 8601 Duration character string types.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ppvalue</i>	Pointer to a character string pointer variable to receive the decoded string. The string is stored as a standard null-terminated C string. Memory is allocated for the string by the <code>::rtxMemAlloc</code> function.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The ASN.1 tag to be decoded. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer. This parameter only has meaning if the tagging parameter specifies explicit decoding.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

#### Returns

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

### 6.10.2.10 berDecStrmDynBitStr()

```
int berDecStrmDynBitStr (
    OSCTXT * pctxt,
    const OSOCTET ** ppvalue,
    OSUINT32 * pnbits,
    ASN1TagType tagging,
    int length )
```

This function decodes a variable of the ASN.1 BIT STRING type. It will allocate dynamic memory to store the decoded result.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ppvalue</i>	Pointer to a pointer variable to receive the decoded bit string. Dynamic memory is allocated to hold the string.

#### Parameters

<i>pnbits</i>	Pointer to an integer value to receive the decoded number of bits.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

#### Returns

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

#### 6.10.2.11 berDecStrmDynBitStr64()

```
int berDecStrmDynBitStr64 (
    OSCTXT * pctxt,
    const OSOCTET ** ppvalue,
    OSSIZE * pnbits,
    ASN1TagType tagging,
    OSSIZE length,
    OSBOOL indefLen )
```

64-bit version of berDecStrmDynBitStr.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ppvalue</i>	Pointer to a pointer variable to receive the decoded bit string. Dynamic memory is allocated to hold the string.
<i>pnbits</i>	Pointer to an integer value to receive the decoded number of bits.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.
<i>indefLen</i>	Boolean indicating component is indefinite length.

#### Returns

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



### 6.10.2.12 berDecStrmDynOctStr()

```
int berDecStrmDynOctStr (
    OSCTXT * pctxt,
    const OSOCTET ** ppvalue,
    OSUINT32 * pnocts,
    ASN1TagType tagging,
    int length )
```

This function decodes a variable of the ASN.1 OCTET STRING type. It will allocate dynamic memory to store the decoded result.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ppvalue</i>	Pointer to a pointer variable to receive the decoded octet string. Dynamic memory is allocated to hold the string.
<i>pnocts</i>	Pointer to an integer value to receive the decoded number of octets.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

#### Returns

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

### 6.10.2.13 berDecStrmDynOctStr64()

```
int berDecStrmDynOctStr64 (
    OSCTXT * pctxt,
    OSOCTET ** ppvalue,
    OSSIZE * pnocts,
    ASN1TagType tagging,
    OSSIZE length,
    OSBOOL indefLen )
```

This version of berDecStrmDynOctStr provides true 64-bit support by using an OSSIZE type (by default, an abstraction for size\_t) to hold the decoded number of octets. On 64-bit systems, this will be a 64-bit number. This also fixes the issue of using a const pointer for the returned data since the variable is mutable.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
--------------	--

## Parameters

<i>ppvalue</i>	A pointer to a pointer (**) to hold the address of a byte buffer to receive decoded OCTET STRING content.
<i>pnocts</i>	Pointer to an integer value to receive the decoded number of octets.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.
<i>indefLen</i>	Boolean indicating component is indefinite length.

## Returns

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

### 6.10.2.14 berDecStrmEnum()

```
int berDecStrmEnum (
    OSCTXT * pctxt,
    OSINT32 * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a variable of the ASN.1 ENUMERATED type.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object← _p</i>	Pointer to a variable to receive the decoded enumerated value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

### 6.10.2.15 berDecStrmFindTag()

```
int berDecStrmFindTag (
    OSCTXT * pctxt,
    ASN1TAG tag,
    int * len_p,
    OSBOOL advance )
```

This function searches the stream being decoded for the first match of the given tag from the current stream position. If the tag is found, the decode stream position will be set to either the start of the tag or to the start of the contents depending on the value of the advance flag. If not found, the stream position will be returned to the position from which the search started.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tag</i>	Tag variable to match.
<i>len_p</i>	Pointer to a variable to receive the decoded length of the tagged component. The returned value will either be the actual length or the special constant 'ASN_K_INDEFLEN', which indicates indefinite length.
<i>advance</i>	The boolean value indicates the behaviour of the decode pointer. If it is set to TRUE and match is successful, then the pointer will be moved behind the tag. Otherwise, it will be left unchanged.

#### Returns

Completion status of operation:

- 0 (0) - match is successful;
- RTERR\_IDNOTFOU - match is not successful;
- negative value - error occurred.

### 6.10.2.16 berDecStrmFindTag2()

```
int berDecStrmFindTag2 (
    OSCTXT * pctxt,
    ASN1TAG tag,
    OSSIZE * len_p,
    OSBOOL * pIndefLen,
    OSBOOL advance )
```

This version of berDecStrmFindTag supports messages with length fields longer than will fit in a signed 32-bit integer value (up to unsigned 64-bit lengths on 64-bit machines).

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tag</i>	Tag variable to match.

#### Parameters

<i>len_p</i>	Pointer to a variable to receive the decoded length of the tagged component.
<i>pIndefLen</i>	Pointer to boolean variable which will be set to true if length is indefinite. Length returned in pLength should be disregarded in this case.
<i>advance</i>	The boolean value indicates the behaviour of the decode pointer. If it is set to TRUE and match is successful, then the pointer will be moved behind the tag. Otherwise, it will be left unchanged.

#### Returns

Completion status of operation:

- 0 (0) - match is successful;
- RTERR\_IDNOTFOU - match is not successful;
- negative value - error occurred.

#### 6.10.2.17 berDecStrmGetTLVLength()

```
int berDecStrmGetTLVLength (
    OSCTXT * pctxt )
```

This function returns the length of the tag-length-value at the current position in the stream.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
--------------	--

#### Returns

Total length (in bytes) of TLV component or negative error code.

#### 6.10.2.18 berDecStrmInt()

```
int berDecStrmInt (
    OSCTXT * pctxt,
    OSINT32 * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a variable of the ASN.1 INTEGER type.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object↔ _p</i>	Pointer to a variable to receive a decoded 32-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

### 6.10.2.19 berDecStrmInt16()

```
int berDecStrmInt16 (
    OSCTXT * pctxt,
    OSINT16 * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a 16-bit variable of the ASN.1 INTEGER type.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object↔ _p</i>	Pointer to a variable to receive a decoded 16-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

### 6.10.2.20 berDecStrmInt64()

```
int berDecStrmInt64 (
    OSCTXT * pctxt,
    OSINT64 * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a 64-bit variable of the ASN.1 INTEGER type.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	Pointer to a variable to receive a decoded 64-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

#### Returns

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

### 6.10.2.21 berDecStrmInt8()

```
int berDecStrmInt8 (
    OSCTXT * pctxt,
    OSINT8 * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes an 8-bit variable of the ASN.1 INTEGER type.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	Pointer to a variable to receive a decoded 8-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

### 6.10.2.22 berDecStrmLength()

```
int berDecStrmLength (
    OSCTXT * pctxt,
    int * len_p )
```

This function decodes a BER length determinant value.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>len_p</i>	Pointer to a variable to receive the decoded length of the tagged component. The returned value will either be the actual length or the special constant 'ASN_K_INDEFLEN', which indicates indefinite length.

## Returns

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

### 6.10.2.23 berDecStrmLength2()

```
int berDecStrmLength2 (
    OSCTXT * pctxt,
    OSSIZE * pLength,
    OSBOOL * pIndefLen )
```

This function decodes a BER length determinant value. This version of the function can support lengths larger than can be held in a signed 32-bit integer value (up to 64-bits unsigned on a 64-bit machine).

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pLength</i>	Pointer to a variable to receive the decoded length of the tagged component.
<i>pIndefLen</i>	Pointer to boolean variable which will be set to true if length is indefinite. Length returned in <i>pLength</i> should be disregarded in this case.

## Returns

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

### 6.10.2.24 berDecStrmMatchEOC()

```
int berDecStrmMatchEOC (
    OSCTXT * pctxt )
```

This function does a comparison between the end-of-contents octets (EOC) and the tag at the current decode pointer position to determine if they match. It then returns the result of the match operation. If match is not successful, the decode pointer will be unchanged; otherwise the pointer will be moved behind the EOC.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
--------------	--

## Returns

Completion status of operation:

- 0 (0) - match is successful;
- RTERR\_IDNOTFOU - match is not successful;
- negative value - error occurred.

### 6.10.2.25 berDecStrmMatchTag()

```
int berDecStrmMatchTag (
    OSCTXT * pctxt,
    ASN1TAG tag,
    int * len_p,
    OSBOOL advance )
```

This function does a comparison between the given tag and the tag at the current decode pointer position to determine if they match. It then returns the result of the match operation.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tag</i>	Tag variable to match.
<i>len_p</i>	Pointer to a variable to receive the decoded length of the tagged component. The returned value will either be the actual length or the special constant 'ASN_K_INDEFLEN', which indicates indefinite length.
<i>advance</i>	The boolean value indicates the behaviour of the decode pointer. If it is set to TRUE and match is successful, then the pointer will be moved behind the tag. Otherwise, it will be left unchanged.



## Returns

Completion status of operation:

- 0 (0) - match is successful;
- RTERR\_IDNOTFOU - match is not successful;
- negative value - error occurred.

### 6.10.2.26 berDecStrmMatchTag2()

```
int berDecStrmMatchTag2 (
    OSCTXT * pctxt,
    ASN1TAG tag,
    OSSIZE * len_p,
    OSBOOL * pIndefLen,
    OSBOOL advance )
```

This function does a comparison between the given tag and the tag at the current decode pointer position to determine if they match. It then returns the result of the match operation.

This version of the function can support lengths larger than can be held in a signed 32-bit integer value (up to 64-bits unsigned on a 64-bit machine).

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tag</i>	Tag variable to match.
<i>len_p</i>	Pointer to a variable to receive the decoded length of the tagged component.
<i>pIndefLen</i>	Pointer to boolean variable which will be set to true if length is indefinite. Length returned in <i>len_p</i> should be disregarded in this case.
<i>advance</i>	The boolean value indicates the behaviour of the decode pointer. If it is set to TRUE and match is successful, then the pointer will be moved behind the tag. Otherwise, it will be left unchanged.

## Returns

Completion status of operation:

- 0 (0) - match is successful;
- RTERR\_IDNOTFOU - match is not successful;
- negative value - error occurred.

### 6.10.2.27 berDecStrmNextElement()

```
int berDecStrmNextElement (
    OSCTXT * pctxt )
```

This function moves the decode pointer to the next tagged element in the decode stream. It is useful for use in an error handling callback function because it allows an unknown or bogus element to be skipped.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
--------------	--

#### Returns

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

#### 6.10.2.28 berDecStrmNull()

```
int berDecStrmNull (
    OSCTXT * pctxt,
    ASN1TagType tagging )
```

This function decodes an ASN.1 NULL placeholder.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

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

#### 6.10.2.29 berDecStrmObjId()

```
int berDecStrmObjId (
    OSCTXT * pctxt,
    ASN1OBJID * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a value of the ASN.1 OBJECT IDENTIFIER type.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object↔ _p</i>	Pointer to value to receive decoded result. The ASN1OBJID structure contains an integer to hold the number of subidentifiers and an array to hold the subidentifier values.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

### 6.10.2.30 berDecStrmObjId64()

```
int berDecStrmObjId64 (
    OSCTXT * pctxt,
    ASN1OID64 * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a value of the ASN.1 OBJECT IDENTIFIER type using 64-bit subidentifiers.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object↔ _p</i>	Pointer to value to receive decoded result. The ASN1OID64 structure contains an integer to hold the number of subidentifiers and an array of 64-bit unsigned integers to hold the subidentifier values.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

### 6.10.2.31 berDecStrmOctStr()

```
int berDecStrmOctStr (
    OSCTXT * pctxt,
    OSOCTET * pvalue,
    OSUIN32 * pnocts,
    ASN1TagType tagging,
    int length )
```

This function decodes a variable of the ASN.1 OCTET STRING type into a static memory structure. This function call is generated by ASN1C to decode a sized octet string production.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pvalue</i>	Pointer to a variable to receive the decoded octet string. This is assumed to be a static array large enough to hold the number of octets specified in the *pnocts input parameter.
<i>pnocts</i>	Pointer to an integer variable containing the size (in octets) of the sized ASN.1 octet string. An error will occur if the number of octets in the decoded string is larger than this value. Note that this is also used as an output variable - the actual number of decoded octets will be returned in this variable.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

#### Returns

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

### 6.10.2.32 berDecStrmOpenType()

```
int berDecStrmOpenType (
    OSCTXT * pctxt,
    const OSOCTET ** object_p2,
    OSSIZE * pnumocts )
```

This function decodes a variable of an ASN.1 open type. This includes the now deprecated ANY and ANY DEFINED BY types from the 1990 standard as well as other types defined to be open in the new standards (for example, a variable type declaration in an X.681 Information Object Class definition).

Decoding is accomplished by returning a pointer to the encoded message component at the current decode pointer location and skipping to the next field. The caller must then call additional decode functions to further decode the component.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p2</i>	A pointer to a pointer (**) to hold the address of a byte buffer. This buffer will contain a copy of the encoded message component located at the current decode pointer location.
<i>pnumocts</i>	Pointer to a size-typed variable to receive the decoded number of octets.

## Returns

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

### 6.10.2.33 berDecStrmOpenTypeAppend()

```
int berDecStrmOpenTypeAppend (
    OSCTXT * pctxt,
    OSRTDList * pElemList )
```

This function is similar to the [berDecStrmOpenType](#). The difference is that after it decodes the open type data into an ASN1OpenType structure, it appends the structure to a doubly-linked list. This function is typically used for decoding extension items in extensible types. The user is provided with a list of each extension item in the message.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pElemList</i>	The pointer to linked list structure. The decoded ASN1OpenType structure will be appended to this list.

## Returns

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

### 6.10.2.34 berDecStrmOpenTypeExt()

```
int berDecStrmOpenTypeExt (
    OSCTXT * pctxt,
    ASN1CCB * ccb_p,
    ASN1TAG * tags,
    int tagCount,
    OSRTDList * pElemList )
```

This function is similar to the [berDecStrmOpenType](#) function except that it is used in places where open type extensions are specified. An open type extension is defined as an extensibility marker on a constructed type without any extension elements defined (for example, SEQUENCE { a INTEGER, : }). The difference is that this is an implicit field that can span one or more elements whereas the standard Open Type is assumed to be a single tagged field.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ccb_p</i>	Pointer to a 'context control block' structure. This is basically a loop control mechanism to keep the variable associated with parsing a nested constructed element straight.
<i>tags</i>	Array of next expected tag values (null if last field). The routine will loop through elements until a matching tag is found or some other error occurs.
<i>tagCount</i>	The number of tags in the tags array.
<i>pElemList</i>	The pointer to linked list structure. The decoded ASN1OpenType structure will be appended to this list.

## Returns

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

### 6.10.2.35 berDecStrmPeekTagAndLen()

```
int berDecStrmPeekTagAndLen (
    OSCTXT * pctxt,
    ASN1TAG * ptag,
    int * plen )
```

This function "peeks" the tag and length at the current decode pointer location and returns the results. The decode pointer location is left as it was before call to this function.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ptag</i>	Pointer to a variable to receive the decoded ASN.1 tag value.
<i>plen</i>	Pointer to a variable to receive the decoded length of the tagged component. The returned value will either be the actual length or the special constant 'ASN_K_INDEFLEN', which indicates indefinite length.

## Returns

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

### 6.10.2.36 berDecStrmReadDynTLV()

```
int berDecStrmReadDynTLV (
    OSCTXT * pctxt,
    OSOCTET ** pbuf )
```

This function reads a complete tag-length-value (TLV) from the decode stream into a dynamic memory buffer.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ppbuf</i>	Pointer to a pointer to a buffer to receive the allocated memory pointer. The memory is allocated using the <code>rtxMem*</code> memory management functions. This may be freed using <code>rtxMemFree</code> or <code>rtxMemFreePtr</code> .

## Returns

The total number of octets read into the buffer, or negative error code.

### 6.10.2.37 `berDecStrmReadTLV()`

```
int berDecStrmReadTLV (
    OSCTXT * pctxt,
    OSOCTET * buf,
    OSSIZE bufsiz )
```

This function reads a complete tag-length-value (TLV) from the decode stream into the given memory buffer.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>buf</i>	Pointer to a buffer to receive a data.
<i>bufsiz</i>	Size of the buffer.

## Returns

The total number of octets read into the buffer, or negative error code.

### 6.10.2.38 `berDecStrmReal()`

```
int berDecStrmReal (
    OSCTXT * pctxt,
    OSREAL * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a variable of the binary encoded ASN.1 REAL type.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object↔ _p</i>	Pointer to a variable to receive the decoded real value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

### 6.10.2.39 berDecStrmReal10()

```
int berDecStrmReal10 (
    OSCTXT * pctxt,
    const char ** object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a value of the decimal encoded ASN.1 REAL type.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object↔ _p</i>	Pointer to a character pointer variable to receive the decoded result. Dynamic memory is allocated for the variable using the ::rtxMemAlloc function.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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



#### 6.10.2.40 berDecStrmRelativeOID()

```
int berDecStrmRelativeOID (
    OSCTXT * pctxt,
    ASN1OBJID * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a value of the ASN.1 RELATIVE-OID type.

##### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	Pointer to value to receive decoded result. The ASN1OBJID structure contains an integer to hold the number of subidentifiers and an array to hold the subidentifier values.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

##### Returns

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

#### 6.10.2.41 berDecStrmTag()

```
int berDecStrmTag (
    OSCTXT * pctxt,
    ASN1TAG * tag_p )
```

This function decodes the tag at the current decode pointer location and returns the results.

##### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tag_p</i>	Pointer to a variable to receive the decoded ASN.1 tag value.

##### Returns

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

#### 6.10.2.42 berDecStrmTagAndLen()

```
int berDecStrmTagAndLen (
    OSCTXT * pctxt,
    ASN1TAG * tag_p,
    int * len_p )
```

This function decodes the tag and length at the current decode pointer location and returns the results.

##### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tag</i> <sub>←</sub> <i>_p</i>	Pointer to a variable to receive the decoded ASN.1 tag value.
<i>len</i> <sub>←</sub> <i>_p</i>	Pointer to a variable to receive the decoded length of the tagged component. The returned value will either be the actual length or the special constant 'ASN_K_INDEFLEN', which indicates indefinite length.

##### Returns

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

#### 6.10.2.43 berDecStrmTagAndLen2()

```
int berDecStrmTagAndLen2 (
    OSCTXT * pctxt,
    ASN1TAG * tag_p,
    OSSIZE * len_p,
    OSBOOL * pIndefLen )
```

This function decodes the tag and length at the current decode pointer location and returns the results.

This version of the function can support lengths larger than can be held in a signed 32-bit integer value (up to 64-bits unsigned on a 64-bit machine).

##### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tag_p</i>	Pointer to a variable to receive the decoded ASN.1 tag value.
<i>len_p</i>	Pointer to a variable to receive the decoded length of the tagged component.
<i>pIndefLen</i>	Pointer to boolean variable which will be set to true if length is indefinite. Length returned in <i>len_p</i> should be disregarded in this case.

## Returns

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

### 6.10.2.44 berDecStrmTestEOC()

```
OSBOOL berDecStrmTestEOC (
    OSCTXT * pctxt,
    ASN1CCB * ccb_p )
```

This function does a quick test on end-of-contents octets at the current decode pointer. In contrast to the [berDecStrmMatchEOC](#) this function never moves the decode pointer and it returns a boolean result of testing.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ccb_p</i>	Pointer to a 'context control block' structure. This is basically a loop control mechanism to keep the variable associated with parsing a nested constructed element straight.

## Returns

A result of testing:

- TRUE, if EOC at the current decode pointer;
- FALSE, otherwise.

### 6.10.2.45 berDecStrmTestTag()

```
int berDecStrmTestTag (
    OSCTXT * pctxt,
    ASN1TAG tag,
    int * len_p,
    OSBOOL advance )
```

This function does a comparison between the given tag and the tag at the current decode pointer position to determine if they match. It then returns the result of the match operation. In contrary to the [berDecStrmMatchTag](#) function this one does NOT log error, if tags are not matched.

#### Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>tag</i>	Tag variable to match.
<i>len_p</i>	Pointer to a variable to receive the decoded length of the tagged component. The returned value will either be the actual length or the special constant 'ASN_K_INDEFLEN', which indicates indefinite length.
<i>advance</i>	The boolean value indicates the behaviour of the decode pointer. If it is set to TRUE and match is successful, then the pointer will be moved behind the tag. Otherwise, it will be left unchanged.

## Returns

Completion status of operation:

- 0 (0) - match is successful;
- RTERR\_IDNOTFOU - match is not successful;
- another negative value - error occurred.

### 6.10.2.46 berDecStrmTimeOfDayStr()

```
int berDecStrmTimeOfDayStr (
    OSCTXT * pctxt,
    const char ** ppvalue,
    ASN1TagType tagging,
    ASN1TAG tag,
    int length )
```

This function decodes a variable of one of the ASN.1 ISO 8601 Time-Of-Day character string types.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ppvalue</i>	Pointer to a character string pointer variable to receive the decoded string. The string is stored as a standard null-terminated C string. Memory is allocated for the string by the <code>::rtxMemAlloc</code> function.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The ASN.1 tag to be decoded. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer. This parameter only has meaning if the tagging parameter specifies explicit decoding.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

### 6.10.2.47 berDecStrmTimeStr()

```
int berDecStrmTimeStr (
    OSCTXT * pctxt,
    const char ** ppvalue,
```

```
ASN1TagType tagging,  
ASN1TAG tag,  
int length )
```

This function decodes a variable of one of the ASN.1 ISO 8601 Time character string types.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ppvalue</i>	Pointer to a character string pointer variable to receive the decoded string. The string is stored as a standard null-terminated C string. Memory is allocated for the string by the <code>::rtxMemAlloc</code> function.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The ASN.1 tag to be decoded. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer. This parameter only has meaning if the tagging parameter specifies explicit decoding.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

### 6.10.2.48 berDecStrmUInt()

```
int berDecStrmUInt (
    OSCTXT * pctxt,
    OSUINT32 * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a variable of the unsigned variant of ASN.1 INTEGER type.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object↔ _p</i>	Pointer to a variable to receive a decoded unsigned 32-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

#### 6.10.2.49 berDecStrmUInt16()

```
int berDecStrmUInt16 (
    OSCTXT * pctx,
    OSUINT16 * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a 16-bit variable of the unsigned variant of ASN.1 INTEGER type.

##### Parameters

<i>pctx</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	Pointer to a variable to receive a decoded unsigned 16-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

##### Returns

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

#### 6.10.2.50 berDecStrmUInt64()

```
int berDecStrmUInt64 (
    OSCTXT * pctx,
    OSUINT64 * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a 64-bit variable of the unsigned variant of ASN.1 INTEGER type.

##### Parameters

<i>pctx</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	Pointer to a variable to receive a decoded unsigned 64-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

### 6.10.2.51 berDecStrmUInt8()

```
int berDecStrmUInt8 (
    OSCTXT * pctxt,
    OSUINT8 * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes an 8-bit variable of the unsigned variant of ASN.1 INTEGER type.

## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p</i>	Pointer to a variable to receive a decoded unsigned 8-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

### 6.10.2.52 berDecStrmUnivStr()

```
int berDecStrmUnivStr (
    OSCTXT * pctxt,
    Asn132BitCharString * object_p,
    ASN1TagType tagging,
    int length )
```

This function decodes a variable an ASN.1 32-bit character UniversalString type.



## Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object</i> <i>_p</i>	Pointer to a structure variable to receive the decoded string. The string is stored as an array of unsigned integer characters. Memory is allocated for the string by the ::rtxMemAlloc function.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

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

## 6.11 C++ classes for streaming BER encoding.

### Classes

- class [ASN1BEREncodeStream](#)

### 6.11.1 Detailed Description

These classes are used to perform BER encoding directly to a stream (file, network, memory).

## 6.12 C++ classes for streaming BER decoding.

### Classes

- class [ASN1BERDecodeStream](#)

### 6.12.1 Detailed Description

These classes are used to perform BER decoding directly from a stream (file, network, memory).



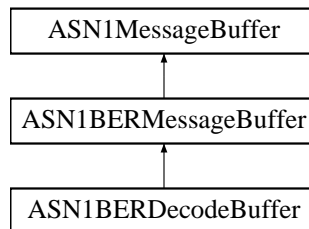
# Chapter 7

## Class Documentation

### 7.1 ASN1BERDecodeBuffer Class Reference

```
#include <asn1BerCppTypes.h>
```

Inheritance diagram for ASN1BERDecodeBuffer:



#### Public Member Functions

- [ASN1BERDecodeBuffer](#) ()
- [ASN1BERDecodeBuffer](#) (const OSOCTET \*pMsgBuf, OSSIZE msgBufLen)
- [ASN1BERDecodeBuffer](#) (const OSOCTET \*pMsgBuf, OSSIZE msgBufLen, OSRTContext \*pContext)
- OSOCTET \* [findElement](#) (ASN1TAG tag, OSINT32 &elemLen, OSBOOL firstFlag=TRUE)
- virtual const OSOCTET \* [getMsgPtr](#) ()
- int [init](#) ()
- virtual OSBOOL [isA](#) (Type bufferType)
- int [parseTagLen](#) (ASN1TAG &tag, int &msglen)
- int [parseTagLen](#) (ASN1TAG &tag, OSSIZE &msglen, OSBOOL \*pIndefLen)
- int [parseTagLen](#) (ASN1TAG &tag, [ASN1BERLength](#) &msglen)
- int [readBinaryFile](#) (const char \*filePath)
- int [setBuffer](#) (const OSOCTET \*pMsgBuf, OSSIZE msgBufLen)
- OSOCTET \* [FindElement](#) (ASN1TAG tag, int &elemLen, int firstFlag=1)
- int [ParseTagLen](#) (ASN1TAG &tag, int &msglen)
- [ASN1BERDecodeBuffer](#) & [operator>>](#) (ASN1CType &val)

## Protected Attributes

- const OSOCTET \* **mpMsgBuf**
- OSSIZE **mMsgBufLen**
- OSBOOL **mBufSetFlag**

## Additional Inherited Members

### 7.1.1 Detailed Description

The [ASN1BERDecodeBuffer](#) class is derived from the [ASN1BERMessageBuffer](#) base class. It contains variables and methods specific to decoding ASN.1 BER/DER messages. It is used to manage the input buffer containing an ASN.1 message to be decoded.

### 7.1.2 Constructor & Destructor Documentation

#### 7.1.2.1 ASN1BERDecodeBuffer() [1/3]

```
ASN1BERDecodeBuffer::ASN1BERDecodeBuffer ( )
```

Default constructor. Use the `getStatus()` method to determine if an error occurred during initialization or not.

#### 7.1.2.2 ASN1BERDecodeBuffer() [2/3]

```
ASN1BERDecodeBuffer::ASN1BERDecodeBuffer (
    const OSOCTET * pMsgBuf,
    OSSIZE msgBufLen )
```

Parameterized constructor. This constructor constructs a buffer describing an encoded ASN.1 message. Parameters describing the message to be decoded are passed as arguments.

#### Parameters

<i>pMsgBuf</i>	A pointer to a buffer containing an encoded ASN.1 message.
<i>msgBufLen</i>	Size of the message buffer. This does not have to be equal to the length of the message. The message length can be determined from the outer tag-length-value in the message. This parameter is used to determine if the length of the message is valid; therefore it must be greater than or equal to the actual length. Typically, the size of the buffer the message was read into is passed.

### 7.1.2.3 ASN1BERDecodeBuffer() [3/3]

```
ASN1BERDecodeBuffer::ASN1BERDecodeBuffer (
    const OSOCTET * pMsgBuf,
    OSSIZE msgBufLen,
    OSRTContext * pContext )
```

Parameterized constructor. This constructor constructs a buffer describing an encoded ASN.1 message. Parameters describing the message to be decoded are passed as arguments.

#### Parameters

<i>pMsgBuf</i>	A pointer to a buffer containing an encoded ASN.1 message.
<i>msgBufLen</i>	Size of the message buffer. This does not have to be equal to the length of the message. The message length can be determined from the outer tag-length-value in the message. This parameter is used to determine if the length of the message is valid; therefore it must be greater than or equal to the actual length. Typically, the size of the buffer the message was read into is passed.
<i>pContext</i>	A pointer to an existing OSRTContext structure.

## 7.1.3 Member Function Documentation

### 7.1.3.1 findElement()

```
OSOCTET* ASN1BERDecodeBuffer::findElement (
    ASN1TAG tag,
    OSINT32 & elemLen,
    OSBOOL firstFlag = TRUE )
```

This method finds a tagged element within a message.

Calling Sequence:

```
ptr = decodeBuffer.findElement (tag, elemLen, firstFlag);
```

where decodeBuffer is an [ASN1BERDecodeBuffer](#) object.

#### Returns

Pointer to tagged component in message or NULL if component not found.

#### Parameters

<i>tag</i>	ASN.1 tag value to search for.
<i>firstFlag</i>	Flag indicating if this the first time this search is being done. If true, internal pointers will be set to start the search from the beginning of the message. If false, the search will be resumed from the point at which the last matching tag was found. This makes it possible to find all instances of a particular tagged element within a message 169
<i>elemLen</i>	Reference to an integer value to receive the length of the found element.

### 7.1.3.2 getMsgPtr()

```
virtual const OSOCTET* ASN1BERDecodeBuffer::getMsgPtr ( ) [inline], [virtual]
```

This method returns the internal pointer to the current message that has been set to be decoded.

#### Returns

Pointer to message.

### 7.1.3.3 init()

```
int ASN1BERDecodeBuffer::init ( )
```

Initializes message buffer.

#### Returns

Completion status of operation:

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

### 7.1.3.4 isA()

```
virtual OSBOOL ASN1BERDecodeBuffer::isA (
    Type bufferType ) [inline], [virtual]
```

This method checks the type of the message buffer.

#### Parameters

<i>bufferType</i>	Enumerated identifier specifying a derived class. The only possible value for this class is BERDecode.
-------------------	--

#### Returns

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



### 7.1.3.5 operator>>()

```
ASN1BERDecodeBuffer& ASN1BERDecodeBuffer::operator>> (
    ASN1CType & val )
```

This operator decodes an instance of an ASN1CType derived class. Use the getStatus() method to determine if an error occurred during the operation or not.

### 7.1.3.6 parseTagLen() [1/3]

```
int ASN1BERDecodeBuffer::parseTagLen (
    ASN1TAG & tag,
    int & msglen ) [inline]
```

This method will parse the initial tag-length pair from the message.

Calling Sequence:

```
stat = decodeBuffer.parseTagLen (tag, msglen);
```

where decodeBuffer is an [ASN1BERDecodeBuffer](#) object.

#### Returns

Status of the operation. Possible values are 0 if successful or one of the negative error status codes defined in Appendix A of the C/C++ Common Functions Reference Manual.

#### Parameters

<i>tag</i>	Reference to a tag structure to receive the outer level tag value parsed from the message.
<i>msglen</i>	Length of the message. This is the total length of the message obtained by adding the number of bytes in initial tag-length to the parsed length value.

References xd\_setp().

### 7.1.3.7 parseTagLen() [2/3]

```
int ASN1BERDecodeBuffer::parseTagLen (
    ASN1TAG & tag,
    OSIZE & msglen,
    OSBOOL * pIndefLen ) [inline]
```

This method overloaded version of parseTagLen will parse the initial tag-length pair from the message. In this case, the length is returned in a size-typed variable which will hold lengths up to 64 bits in size.

Calling Sequence:

```
stat = decodeBuffer.parseTagLen (tag, msglen);
```

where decodeBuffer is an [ASN1BERDecodeBuffer](#) object.

## Returns

Status of the operation. Possible values are 0 if successful or one of the negative error status codes defined in Appendix A of the C/C++ Common Functions Reference Manual.

## Parameters

<i>tag</i>	Reference to a tag structure to receive the outer level tag value parsed from the message.
<i>msglen</i>	Length of the message. This is the total length of the message obtained by adding the number of bytes in initial tag-length to the parsed length value.
<i>plnDefLen</i>	Boolean flag indicating parsed length is indefinite.

## Returns

Completion status of operation:

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

References `xd_setp64()`.

### 7.1.3.8 `parseTagLen()` [3/3]

```
int ASN1BERDecodeBuffer::parseTagLen (  
    ASN1TAG & tag,  
    ASN1BERLength & msglen ) [inline]
```

This method overloaded version of `parseTagLen` will parse the initial tag-length pair from the message. This variant allows a reference to a BER length object to be used for the length. Lengths up to 64 bits in size are supported.

Calling Sequence:

```
stat = decodeBuffer.parseTagLen (tag, msglen);
```

where `decodeBuffer` is an [ASN1BERDecodeBuffer](#) object.

## Returns

Status of the operation. Possible values are 0 if successful or one of the negative error status codes defined in Appendix A of the C/C++ Common Functions Reference Manual.

## Parameters

<i>tag</i>	Reference to a tag structure to receive the outer level tag value parsed from the message.
<i>msglen</i>	Reference to an object used to describe the length of the message.

## Returns

Completion status of operation:

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

### 7.1.3.9 readBinaryFile()

```
int ASN1BERDecodeBuffer::readBinaryFile (
    const char * filePath )
```

This method reads a file containing a single BER/DER/CER encoded data record into the buffer for decoding.

#### Parameters

<i>filePath</i>	The zero-terminated string containing the path to the file.
-----------------	---

## Returns

Completion status of operation:

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

### 7.1.3.10 setBuffer()

```
int ASN1BERDecodeBuffer::setBuffer (
    const OSOCTET * pMsgBuf,
    OSSIZE msgBufLen )
```

This method sets a buffer containing a message to be decoded.

#### Parameters

<i>pMsgBuf</i>	A pointer to a memory buffer containing a message to be decoded. The buffer should be declared as an array of unsigned characters (OSOCTETs).
<i>msgBufLen</i>	The length of the memory buffer in bytes.

## Returns

Completion status of operation:

- 0 (0) = success,

- negative return value is error.

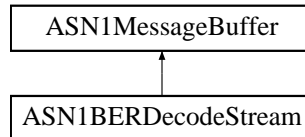
The documentation for this class was generated from the following file:

- [asn1BerCppTypes.h](#)

## 7.2 ASN1BERDecodeStream Class Reference

```
#include <ASN1BERDecodeStream.h>
```

Inheritance diagram for ASN1BERDecodeStream:



### Public Member Functions

- [ASN1BERDecodeStream](#) (OSRTInputStreamIF &is)
- **ASN1BERDecodeStream** (OSRTInputStreamIF \*pis, OSBOOL bOwnStream=TRUE)
- virtual void \* [getAppInfo](#) ()
- virtual OSRTCtxtPtr [getContext](#) ()
- virtual OSCTXT \* [getCtxtPtr](#) ()
- virtual char \* [getErrorInfo](#) ()
- virtual char \* [getErrorInfo](#) (char \*pBuf, size\_t &bufSize)
- virtual int [getStatus](#) () const
- virtual void [printErrorInfo](#) ()
- virtual void [resetErrorInfo](#) ()
- virtual void [setAppInfo](#) (void \*pAppInfo)
- virtual void [setDiag](#) (OSBOOL value=TRUE)
- virtual int [close](#) ()
- virtual int [flush](#) ()
- virtual int [getPosition](#) (size\_t \*ppos)
- virtual OSBOOL [isOpened](#) ()
- virtual size\_t [currentPos](#) ()
- virtual OSBOOL [markSupported](#) ()
- int [mark](#) (size\_t readAheadLimit)
- virtual long [read](#) (OSOCKET \*pDestBuf, size\_t maxToRead)
- virtual long [readBlocking](#) (OSOCKET \*pDestBuf, size\_t toReadBytes)
- int [reset](#) ()
- virtual int [setPosition](#) (size\_t pos)
- virtual int [skip](#) (size\_t n)
- [ASN1BERDecodeStream](#) & [operator>>](#) (ASN1CType &val)
- size\_t [byteIndex](#) ()

- OSBOOL [chkend](#) (ASN1CCB &ccb)
- int [decodeBigInt](#) (const char \*&pval, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeBitStr](#) (OSOCKET \*pbits, OSUINT32 &numbits, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeBitStr](#) (ASN1DynBitStr &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeBMPStr](#) (Asn116BitCharString &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeBool](#) (OSBOOL &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeCharStr](#) (const char \*&pval, ASN1TagType tagging=ASN1EXPL, ASN1TAG tag=0, int length=0)
- int [decodeEnum](#) (OSINT32 &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeEoc](#) ()
- int [decodeInt](#) (OSINT32 &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeInt8](#) (OSINT8 &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeInt16](#) (OSINT16 &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeInt64](#) (OSINT64 &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeLength](#) (OSINT32 &length)
- int [decodeNull](#) (ASN1TagType tagging=ASN1EXPL)
- int [decodeObj](#) (ASN1CType &val)
- int [decodeObjId](#) (ASN1OBJID &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeObjId64](#) (ASN1OID64 &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeOctStr](#) (OSOCKET \*pocets, OSUINT32 &numocets, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeOctStr](#) (ASN1DynOctStr &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeOctStr](#) (OSDynOctStr64 &val, ASN1TagType tagging=ASN1EXPL, OSSIZE length=0, OSBOOL indef←Len=FALSE)
- int [decodeOpenType](#) (ASN1OpenType &val)
- int [decodeReal](#) (OSREAL &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeRelativeOID](#) (ASN1OBJID &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeTag](#) (ASN1TAG &tag)
- int [decodeTagAndLen](#) (ASN1TAG &tag, OSINT32 &len)
- int [decodeTagAndLen](#) (ASN1TAG &tag, [ASN1BERLength](#) &len)
- int [decodeUInt](#) (OSUINT32 &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeUInt8](#) (OSUINT8 &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeUInt16](#) (OSUINT16 &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeUInt64](#) (OSUINT64 &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [decodeUnivStr](#) (Asn132BitCharString &val, ASN1TagType tagging=ASN1EXPL, int length=0)
- int [getTLVLength](#) ()
- OSBOOL [isA](#) (Type bufferType)
- int [peekTagAndLen](#) (ASN1TAG &tag, int &len)
- int [readTLV](#) (OSOCKET \*pDestBuf, size\_t bufsiz)

## Protected Attributes

- OSRTInputStreamIF \* **mpStream**
- OSBOOL **mbOwnStream**

### 7.2.1 Detailed Description

This class is a base class for other ASN.1 BER input stream's classes. It is derived from the ASN1Stream base class. It contains variables and methods specific to streaming decoding of BER messages. It is used to manage the input stream containing the ASN.1 message to be decoded.

## 7.2.2 Constructor & Destructor Documentation

### 7.2.2.1 ASN1BERDecodeStream()

```
ASN1BERDecodeStream::ASN1BERDecodeStream (
    OSRTInputStreamIF & is )
```

A default constructor. Use [getStatus\(\)](#) method to determine if an error occurred during the initialization or not.

## 7.2.3 Member Function Documentation

### 7.2.3.1 byteIndex()

```
size_t ASN1BERDecodeStream::byteIndex ( )
```

This method returns the total number of octets (bytes) already decoded from the stream.

#### Returns

Number of octets (bytes) already decoded from the stream.

### 7.2.3.2 chkend()

```
OSBOOL ASN1BERDecodeStream::chkend (
    ASN1CCB & ccb )
```

This method determines if the decoder has reached the end of a message context block. This method could be called when decoding a SET or SEQUENCE OF/SET OF construct.

#### Parameters

<i>ccb</i>	Reference to a 'context control block' structure. This is basically a loop control mechanism to keep the variable associated with parsing a nested constructed element straight.
------------	--

#### Returns

Boolean value indicating whether or not the end-of-context has been reached.

### 7.2.3.3 close()

```
virtual int ASN1BERDecodeStream::close ( ) [inline], [virtual]
```

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

#### Returns

Completion status of operation:

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

#### See also

`::rxStreamClose`

### 7.2.3.4 currentPos()

```
virtual size_t ASN1BERDecodeStream::currentPos ( ) [inline], [virtual]
```

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

#### Returns

The number of octets already read from the stream.

### 7.2.3.5 decodeBigInt()

```
int ASN1BERDecodeStream::decodeBigInt (
    const char *& pval,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a variable of the ASN.1 INTEGER type. In this case, the integer is assumed to be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits).

## Parameters

<i>pval</i>	Reference to a pointer to a variable to receive a decoded big integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

## See also

[berDecStrmBigInt](#)

### 7.2.3.6 decodeBitStr() [1/2]

```
int ASN1BERDecodeStream::decodeBitStr (
    OSOCTET * pbits,
    OSUINT32 & numbits,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a variable of the ASN.1 BIT STRING type into a static memory structure. It is used to decode a sized bit string production.

## Parameters

<i>pbits</i>	Pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of bits specified in the numbits input parameter.
<i>numbits</i>	As input parameter it is a reference to an integer variable containing the size (in bits) of the sized ASN.1 bit string. An error will occur if the number of bits in the decoded string is larger than this value. Note that this is also used as an output variable - the actual number of decoded bits will be returned in this variable.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.



## Returns

Completion status of operation:

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

## See also

[berDecStrmBitStr](#)

### 7.2.3.7 decodeBitStr() [2/2]

```
int ASN1BERDecodeStream::decodeBitStr (
    ASN1DynBitStr & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a variable of the ASN.1 BIT STRING type. It will allocate dynamic memory to store the decoded result.

## Parameters

<i>val</i>	Reference to an ASN1DynBitStr variable to receive the decoded bit string.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

## See also

[berDecStrmDynBitStr](#)

### 7.2.3.8 decodeBMPStr()

```
int ASN1BERDecodeStream::decodeBMPStr (
    Asn116BitCharString & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a variable of the ASN.1 BMPString type.

#### Parameters

<i>val</i>	Reference to a variable to receive the decoded BMPString value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

#### Returns

Completion status of operation:

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

#### See also

[berDecStrmBMPStr](#)

### 7.2.3.9 decodeBool()

```
int ASN1BERDecodeStream::decodeBool (
    OSBOOL & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a variable of the ASN.1 BOOLEAN type.

#### Parameters

<i>val</i>	Reference to a variable to receive the decoded boolean value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

## See also

[berDecStrmBool](#)

### 7.2.3.10 decodeCharStr()

```
int ASN1BERDecodeStream::decodeCharStr (
    const char *& pval,
    ASN1TagType tagging = ASN1EXPL,
    ASN1TAG tag = 0,
    int length = 0 )
```

This method decodes a variable of one of the ASN.1 8-bit character string types. These types include IA5String, VisibleString, PrintableString, and NumericString.

## Parameters

<i>pval</i>	Reference to a character string pointer variable to receive the decoded string. The string is stored as a standard null-terminated C string. Memory is allocated for the string by the <code>::rtxMemAlloc</code> function.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The ASN.1 tag to be decoded. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer. This parameter only has meaning if the tagging parameter specifies explicit decoding.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

## See also

[berDecStrmCharStr](#)

### 7.2.3.11 decodeEnum()

```
int ASN1BERDecodeStream::decodeEnum (
    OSINT32 & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a variable of the ASN.1 ENUMERATED type.

#### Parameters

<i>val</i>	Reference to a variable to receive the decoded enumerated value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

#### Returns

Completion status of operation:

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

#### See also

[berDecStrmEnum](#)

### 7.2.3.12 decodeEoc()

```
int ASN1BERDecodeStream::decodeEoc ( )
```

This method decodes the end-of-contents octets.

#### Returns

Completion status of operation:

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

#### See also

[berDecStrmMatchEOC](#)

### 7.2.3.13 decodeInt()

```
int ASN1BERDecodeStream::decodeInt (
    OSINT32 & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a variable of the ASN.1 INTEGER type.

#### Parameters

<i>val</i>	Reference to a variable to receive a decoded 32-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

#### Returns

Completion status of operation:

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

#### See also

[berDecStrmlnt](#)

### 7.2.3.14 decodeInt16()

```
int ASN1BERDecodeStream::decodeInt16 (
    OSINT16 & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a 16-bit variable of the ASN.1 INTEGER type.

#### Parameters

<i>val</i>	Reference to a variable to receive a decoded 16-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

## See also

[berDecStrmInt16](#)

### 7.2.3.15 decodeInt64()

```
int ASN1BERDecodeStream::decodeInt64 (
    OSINT64 & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a 64-bit variable of the ASN.1 INTEGER type.

## Parameters

<i>val</i>	Reference to a variable to receive a decoded 64-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

## See also

[berDecStrmInt64](#)

### 7.2.3.16 decodeInt8()

```
int ASN1BERDecodeStream::decodeInt8 (
    OSINT8 & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes an 8-bit variable of the ASN.1 INTEGER type.

## Parameters

<i>val</i>	Reference to a variable to receive a decoded 8-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

## See also

[berDecStrmInt8](#)

### 7.2.3.17 decodeLength()

```
int ASN1BERDecodeStream::decodeLength (
    OSINT32 & length )
```

This method decodes a BER length determinant value.

## Parameters

<i>length</i>	Reference to a variable to receive the decoded length of the tagged component. The returned value will either be the actual length or the special constant 'ASN_K_INDEFLEN', which indicates indefinite length.
---------------	---

## Returns

Completion status of operation:

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

## See also

[berDecStrmLength](#)



### 7.2.3.18 decodeNull()

```
int ASN1BERDecodeStream::decodeNull (
    ASN1TagType tagging = ASN1EXPL )
```

This method decodes a variable of the ASN.1 NULL type.

#### Parameters

<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
----------------	---

#### Returns

Completion status of operation:

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

#### See also

[berDecStrmNull](#)

### 7.2.3.19 decodeObj()

```
int ASN1BERDecodeStream::decodeObj (
    ASN1CType & val )
```

This method decodes an ASN.1 constructed object from the stream.

#### Parameters

<i>val</i>	A reference to an object to be decoded.
------------	---

#### Returns

Completion status of operation:

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

### 7.2.3.20 decodeObjId()

```
int ASN1BERDecodeStream::decodeObjId (
    ASN1OBJID & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a variable of the ASN.1 OBJECT IDENTIFIER type.

#### Parameters

<i>val</i>	Reference to a variable to receive the decoded OBJECT IDENTIFIER value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

#### Returns

Completion status of operation:

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

#### See also

[berDecStrmObjId](#)

### 7.2.3.21 decodeObjId64()

```
int ASN1BERDecodeStream::decodeObjId64 (
    ASN1OID64 & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a variable of the ASN.1 OBJECT IDENTIFIER type using 64-bit subidentifiers..

#### Parameters

<i>val</i>	Reference to a variable to receive the decoded 64-bit OBJECT IDENTIFIER value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

## See also

[berDecStrmObjId64](#)

### 7.2.3.22 decodeOctStr() [1/3]

```
int ASN1BERDecodeStream::decodeOctStr (
    OSOCTET * pocts,
    OSUINT32 & numocts,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a variable of the ASN.1 OCTET STRING type into a static memory structure. It is used to decode a sized octet string production.

## Parameters

<i>pocts</i>	Pointer to a variable to receive the decoded octet string. This is assumed to be a static array large enough to hold the number of octets specified in the numocts input parameter.
<i>numocts</i>	Reference to an integer variable containing the size (in octets) of the sized ASN.1 octet string. An error will occur if the number of octets in the decoded string is larger than this value. Note that this is also used as an output variable - the actual number of decoded octets will be returned in this variable.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

## See also

[berDecStrmOctStr](#)

### 7.2.3.23 decodeOctStr() [2/3]

```
int ASN1BERDecodeStream::decodeOctStr (
    ASN1DynOctStr & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a variable of the ASN.1 OCTET STRING type. It will allocate dynamic memory to store the decoded result.

#### Parameters

<i>val</i>	Reference to an ASN1DynOctStr variable to receive the decoded octet string.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

#### Returns

Completion status of operation:

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

#### See also

[berDecStrmDynOctStr](#)

### 7.2.3.24 decodeOctStr() [3/3]

```
int ASN1BERDecodeStream::decodeOctStr (
    OSDynOctStr64 & val,
    ASN1TagType tagging = ASN1EXPL,
    OSSIZE length = 0,
    OSBOOL indefLen = FALSE )
```

This method decodes a variable of the ASN.1 OCTET STRING type. It will allocate dynamic memory to store the decoded result. It supports 64-bit length variables.

#### Parameters

<i>val</i>	Reference to an ASN1DynOctStr variable to receive the decoded octet string.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.
<i>indefLen</i>	Flag indicating if component to be decoded is indefinite length. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

## See also

[berDecStrmDynOctStr64](#)

### 7.2.3.25 decodeOpenType()

```
int ASN1BERDecodeStream::decodeOpenType (
    ASN1OpenType & val )
```

This method decodes an ASN.1 open type value. This is a value of any ASN.1 data type. It may be constructed and contain multiple elements including elements encoded with indefinite lengths. This method will allocate dynamic memory to store the decoded result.

## Parameters

<i>val</i>	Reference to an ASN1OpenType variable to receive the decoded open type data.
------------	--

## Returns

Completion status of operation:

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

## See also

[berDecStrmDynOctStr](#)

### 7.2.3.26 decodeReal()

```
int ASN1BERDecodeStream::decodeReal (
    OSREAL & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a variable of the ASN.1 REAL type.

## Parameters

<i>val</i>	Reference to a variable to receive the decoded REAL value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

## See also

[berDecStrmReal](#)

### 7.2.3.27 decodeRelativeOID()

```
int ASN1BERDecodeStream::decodeRelativeOID (
    ASN1OBJID & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a variable of the ASN.1 RELATIVE-OID type.

## Parameters

<i>val</i>	Reference to a variable to receive the decoded RELATIVE-OID value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

See also

[berDecStrmRelativeOID](#)

### 7.2.3.28 decodeTag()

```
int ASN1BERDecodeStream::decodeTag (
    ASN1TAG & tag )
```

This method decodes the tag at the current decode pointer location and returns the results.

#### Parameters

<i>tag</i>	Reference to a variable to receive the decoded ASN.1 tag value.
------------	---

#### Returns

Completion status of operation:

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

See also

[berDecStrmTag](#)

### 7.2.3.29 decodeTagAndLen() [1/2]

```
int ASN1BERDecodeStream::decodeTagAndLen (
    ASN1TAG & tag,
    OSINT32 & len )
```

This method decodes the tag and length at the current decode pointer location and returns the results.

#### Parameters

<i>tag</i>	Reference to a variable to receive the decoded ASN.1 tag value.
<i>len</i>	Reference to a variable to receive the decoded length of the tagged component. The returned value will either be the actual length or the special constant 'ASN_K_INDEFLEN', which indicates indefinite length.

## Returns

Completion status of operation:

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

## See also

[berDecStrmTagAndLen](#)

### 7.2.3.30 decodeTagAndLen() [2/2]

```
int ASN1BERDecodeStream::decodeTagAndLen (
    ASN1TAG & tag,
    ASN1BERLength & len )
```

This method decodes the tag and length at the current decode pointer location and returns the results. This variant of the method supports 64-bit length values.

## Parameters

<i>tag</i>	Reference to a variable to receive the decoded ASN.1 tag value.
<i>len</i>	Reference to an object to receive the decoded length of the tagged component.

## Returns

Completion status of operation:

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

## See also

[berDecStrmTagAndLen](#)

### 7.2.3.31 decodeUInt()

```
int ASN1BERDecodeStream::decodeUInt (
    OSUINT32 & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a variable of the unsigned variant of ASN.1 INTEGER type.



## Parameters

<i>val</i>	Reference to a variable to receive a decoded unsigned 32-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

## See also

[berDecStrmUInt](#)

### 7.2.3.32 decodeUInt16()

```
int ASN1BERDecodeStream::decodeUInt16 (
    OSUINT16 & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a 16-bit variable of the unsigned variant of ASN.1 INTEGER type.

## Parameters

<i>val</i>	Reference to a variable to receive a decoded unsigned 16-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

See also

[berDecStrmUInt16](#)

### 7.2.3.33 decodeUInt64()

```
int ASN1BERDecodeStream::decodeUInt64 (
    OSUINT64 & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a 64-bit variable of the unsigned variant of ASN.1 INTEGER type.

#### Parameters

<i>val</i>	Reference to a variable to receive a decoded unsigned 64-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

#### Returns

Completion status of operation:

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

See also

[berDecStrmUInt64](#)

### 7.2.3.34 decodeUInt8()

```
int ASN1BERDecodeStream::decodeUInt8 (
    OSUINT8 & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes an 8-bit variable of the unsigned variant of ASN.1 INTEGER type.

## Parameters

<i>val</i>	Reference to a variable to receive a decoded unsigned 8-bit integer value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

## See also

[berDecStrmUInt8](#)

### 7.2.3.35 decodeUnivStr()

```
int ASN1BERDecodeStream::decodeUnivStr (
    Asn132BitCharString & val,
    ASN1TagType tagging = ASN1EXPL,
    int length = 0 )
```

This method decodes a variable of the ASN.1 UniversalString type.

## Parameters

<i>val</i>	Reference to a variable to receive the decoded UniversalString value.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>length</i>	The length, in octets, of the contents field to be decoded. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

## Returns

Completion status of operation:

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

**See also**

[berDecStrmUnivStr](#)

**7.2.3.36 flush()**

```
virtual int ASN1BERDecodeStream::flush ( ) [inline], [virtual]
```

Flushes the buffered data to the stream.

**Returns**

Completion status of operation:

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

**See also**

[::rxStreamFlush](#)

**7.2.3.37 getAppInfo()**

```
virtual void* ASN1BERDecodeStream::getAppInfo ( ) [inline], [virtual]
```

Returns a pointer to application-specific information block

**7.2.3.38 getContext()**

```
virtual OSRITxtPtr ASN1BERDecodeStream::getContext ( ) [inline], [virtual]
```

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

**Returns**

Context smart pointer object.

### 7.2.3.39 getCtxPtr()

```
virtual OSCTXT* ASN1BERDecodeStream::getCtxPtr ( ) [inline], [virtual]
```

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

#### Returns

The pointer to C runtime context.

### 7.2.3.40 getErrorInfo() [1/2]

```
virtual char* ASN1BERDecodeStream::getErrorInfo ( ) [inline], [virtual]
```

Returns error text in a dynamic memory buffer. The 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.

### 7.2.3.41 getErrorInfo() [2/2]

```
virtual char* ASN1BERDecodeStream::getErrorInfo (
    char * pBuf,
    size_t & bufSize ) [inline], [virtual]
```

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

#### Parameters

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

#### Returns

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

#### 7.2.3.42 getPosition()

```
virtual int ASN1BERDecodeStream::getPosition (
    size_t * ppos ) [inline], [virtual]
```

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

##### Parameters

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

##### Returns

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

#### 7.2.3.43 getStatus()

```
virtual int ASN1BERDecodeStream::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. 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.

#### 7.2.3.44 getTLVLength()

```
int ASN1BERDecodeStream::getTLVLength ( )
```

Get the total length of a tag-length-value (TLV) component. This is not the length stored in the L field. It is the total length of the component which is equal to the parsed length plus the number of bytes in the tag and length fields.

##### Returns

The total number of octets in the TLV or a negative error code.

##### See also

[berDecStrmGetTLVLength](#)

### 7.2.3.45 isA()

```
OSBOOL ASN1BERDecodeStream::isA (
    Type bufferType )
```

This method matches an enumerated identifier defined in the base class. One identifier is declared for each of the derived classes.

#### Parameters

<i>bufferType</i>	Enumerated identifier specifying a derived class. This type is defined as a public access type in the ASN1MessageBufferIF base interface. Possible values are: BEREncode, BERDecode, PEREncode, PERDecode, XEREncode, XERDecode, XMLEncode, XMLDecode, Stream.
-------------------	--

#### Returns

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

### 7.2.3.46 isOpened()

```
virtual OSBOOL ASN1BERDecodeStream::isOpened ( ) [inline], [virtual]
```

Checks, is the stream opened or not.

#### Returns

TRUE, if the stream is opened, FALSE otherwise.

#### See also

`::rxStreamIsOpened`

### 7.2.3.47 mark()

```
int ASN1BERDecodeStream::mark (
    size_t readAheadLimit ) [inline]
```

This method marks the current position in this input stream. A subsequent call to the [ASN1BERDecodeStream::reset](#) method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. The `readAheadLimit` argument tells this input stream to allow that many bytes to be read before the mark position gets invalidated.

#### Parameters

<i>readAheadLimit</i>	the maximum limit of bytes that can be read before the mark position becomes invalid.
-----------------------	---

#### Returns

Completion status of operation:

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

#### See also

`::rtxStreamBufMark`, `::rtxStreamBufReset`

#### 7.2.3.48 markSupported()

```
virtual OSBOOL ASN1BERDecodeStream::markSupported ( ) [inline], [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`

#### 7.2.3.49 operator>>()

```
ASN1BERDecodeStream& ASN1BERDecodeStream::operator>> (
    ASN1CType & val )
```

Decodes an ASN.1 constructed object from the stream. Use `getStatus()` method to determine if an error occurred during the operation or not.

#### Parameters

<i>val</i>	A reference to an object to be decoded.
------------	---



## Returns

reference to this class to perform sequential decoding.

### 7.2.3.50 peekTagAndLen()

```
int ASN1BERDecodeStream::peekTagAndLen (
    ASN1TAG & tag,
    int & len )
```

This method "peeks" the tag and length at the current decode pointer location and returns the results. The decode pointer location is left as it was before call to this function.

#### Parameters

<i>tag</i>	Reference to a variable to receive the decoded ASN.1 tag value.
<i>len</i>	Reference to a variable to receive the decoded length of the tagged component. The returned value will either be the actual length or the special constant 'ASN_K_INDEFLEN', which indicates indefinite length.

## Returns

Completion status of operation:

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

## See also

[berDecStrmTagAndLen](#)

### 7.2.3.51 printErrorInfo()

```
virtual void ASN1BERDecodeStream::printErrorInfo ( ) [inline], [virtual]
```

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

### 7.2.3.52 read()

```
virtual long ASN1BERDecodeStream::read (
    OSOCKET * pDestBuf,
    size_t maxToRead ) [inline], [virtual]
```

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

#### Parameters

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

#### Returns

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

#### See also

::rtxStreamRead

#### 7.2.3.53 readBlocking()

```
virtual long ASN1BERDecodeStream::readBlocking (
    OSOCKET * pDestBuf,
    size_t toReadBytes ) [inline], [virtual]
```

Read data from the stream. This method reads `maxToRead` bytes from the stream. It will block until either the bytes are available or an error occurs.

#### Parameters

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

#### Returns

The total number of octets read into the buffer, or negative error code.

#### See also

::rtxStreamReadBlocking

#### 7.2.3.54 readTLV()

```
int ASN1BERDecodeStream::readTLV (
    OSOCKET * pDestBuf,
    size_t bufsiz )
```

Read a complete tag-length-value (TLV) from the decode stream into the given memory buffer.

## Parameters

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

## Returns

The total number of octets read into the buffer, or negative error code.

## See also

[berDecStrmReadTLV](#)

### 7.2.3.55 reset()

```
int ASN1BERDecodeStream::reset ( ) [inline]
```

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

## See also

`::rtxStreamBufMark`, `::rtxStreamBufReset`

### 7.2.3.56 resetErrorInfo()

```
virtual void ASN1BERDecodeStream::resetErrorInfo ( ) [inline], [virtual]
```

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

### 7.2.3.57 setAppInfo()

```
virtual void ASN1BERDecodeStream::setAppInfo (
    void * pAppInfo ) [inline], [virtual]
```

Sets the application-specific information block.

### 7.2.3.58 setDiag()

```
virtual void ASN1BERDecodeStream::setDiag (
    OSBOOL value = TRUE ) [inline], [virtual]
```

The `setDiag` method will turn diagnostic tracing on or off.

#### Parameters

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

#### 7.2.3.59 setPosition()

```
virtual int ASN1BERDecodeStream::setPosition (
    size_t pos ) [inline], [virtual]
```

Sets the current stream position to the given offset.

#### Parameters

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

#### Returns

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

#### 7.2.3.60 skip()

```
virtual int ASN1BERDecodeStream::skip (
    size_t n ) [inline], [virtual]
```

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

#### Parameters

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

#### Returns

Completion status of operation:

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

See also

`::rxStreamSkip`

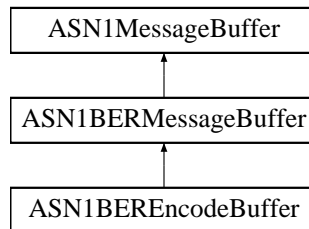
The documentation for this class was generated from the following file:

- [ASN1BERDecodeStream.h](#)

## 7.3 ASN1BEREncodeBuffer Class Reference

```
#include <asn1BerCppTypes.h>
```

Inheritance diagram for ASN1BEREncodeBuffer:



### Public Member Functions

- [ASN1BEREncodeBuffer](#) ()
- [ASN1BEREncodeBuffer](#) (OSOCKET \*pMsgBuf, OSSIZE msgBufLen)
- [ASN1BEREncodeBuffer](#) (OSOCKET \*pMsgBuf, OSSIZE msgBufLen, OSRTContext \*pContext)
- int [encodeBool](#) (OSBOOL val, ASN1TagType tagging=ASN1EXPL)
- int [encodeBigIntNchars](#) (const char \*pval, size\_t nchars, ASN1TagType tagging=ASN1EXPL)
- int [encodeBigInt](#) (const char \*pval, ASN1TagType tagging=ASN1EXPL)
- int [encodeObjId](#) (ASN1OBJID \*pval, ASN1TagType tagging=ASN1EXPL)
- void [freeBuffer](#) ()
- virtual OSOCKET \* [getMsgCopy](#) ()
- virtual const OSOCKET \* [getMsgPtr](#) ()
- virtual size\_t [getMsgLen](#) ()
- int [init](#) ()
- virtual OSBOOL [isA](#) (Type bufferType)
- int [setBuffer](#) (OSOCKET \*pMsgBuf, OSSIZE msgBufLen)
- [ASN1BEREncodeBuffer](#) & [operator<<](#) (ASN1CType &val)

### Additional Inherited Members

#### 7.3.1 Detailed Description

The [ASN1BEREncodeBuffer](#) class is derived from the [ASN1BERMessageBuffer](#) base class. It contains variables and methods specific to encoding ASN.1 messages using the Basic Encoding Rules (BER). It is used to manage the buffer into which an ASN.1 message is to be encoded.

## 7.3.2 Constructor & Destructor Documentation

### 7.3.2.1 ASN1BEREncodeBuffer() [1/3]

```
ASN1BEREncodeBuffer::ASN1BEREncodeBuffer ( )
```

Default constructor. This sets all internal variables to their initial values. Use the `getStatus()` method to determine if an error occurred during initialization or not.

### 7.3.2.2 ASN1BEREncodeBuffer() [2/3]

```
ASN1BEREncodeBuffer::ASN1BEREncodeBuffer (
    OSOCKET * pMsgBuf,
    OSSIZE msgBufLen )
```

Parameterized constructor. This version takes a message buffer and size argument (static encoding version). Use the `getStatus()` method to determine if an error occurred during initialization or not.

#### Parameters

<i>pMsgBuf</i>	A pointer to a fixed size message buffer to receive the encoded message.
<i>msgBufLen</i>	Size of the fixed-size message buffer.

### 7.3.2.3 ASN1BEREncodeBuffer() [3/3]

```
ASN1BEREncodeBuffer::ASN1BEREncodeBuffer (
    OSOCKET * pMsgBuf,
    OSSIZE msgBufLen,
    OSRTContext * pContext )
```

Parameterized constructor. This version takes a message buffer, a size argument (static encoding version), and a context. Use the `getStatus()` method to determine if an error occurred during initialization or not.

#### Parameters

<i>pMsgBuf</i>	A pointer to a fixed size message buffer to receive the encoded message.
<i>msgBufLen</i>	Size of the fixed-size message buffer.
<i>pContext</i>	A pointer to an existing OSRTContext structure.

### 7.3.3 Member Function Documentation

#### 7.3.3.1 encodeBigInt()

```
int ASN1BEREncodeBuffer::encodeBigInt (
    const char * pval,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 INTEGER type. In this case, the integer may be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits).

##### Parameters

<i>*pval</i>	A pointer to a character string containing the value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

##### Returns

Completion status of operation:

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

#### 7.3.3.2 encodeBigIntNchars()

```
int ASN1BEREncodeBuffer::encodeBigIntNchars (
    const char * pval,
    size_t nchars,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 INTEGER type. In this case, the integer may be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits). This variant of the method allows the number of characters to encode to be passed in.

##### Parameters

<i>*pval</i>	A pointer to a character string containing the value to be encoded.
<i>nchars</i>	Number of characters from pval to encode.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Completion status of operation:

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

### 7.3.3.3 encodeBool()

```
int ASN1BEREncodeBuffer::encodeBool (
    OSBOOL val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 BOOLEAN type.

#### Parameters

<i>val</i>	BOOLEAN value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Completion status of operation:

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

### 7.3.3.4 encodeObjId()

```
int ASN1BEREncodeBuffer::encodeObjId (
    ASN1OBJID * pval,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 OBJECT IDENTIFIER type.

#### Parameters

<i>val</i>	Pointer to object identifier structure. This structure contains an integer to hold the number of subidentifiers in the object and an array to hold the subidentifier values.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.



## Returns

Completion status of operation:

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

## See also

[berEncStrmObjId](#)

### 7.3.3.5 freeBuffer()

```
void ASN1BEREncodeBuffer::freeBuffer ( ) [inline]
```

This method releases memory that was allocated for a dynamic encode buffer.

References `xe_free()`.

### 7.3.3.6 getMsgCopy()

```
virtual OSOCTET* ASN1BEREncodeBuffer::getMsgCopy ( ) [virtual]
```

This method returns a copy of the current encoded message. Memory is allocated for the message using the 'new []' operation. It is the users's responsibility to free the memory using 'delete []'.

Calling Sequence:

```
ptr = encodeBuffer.getMsgCopy ( );
```

where `encodeBuffer` is an [ASN1BEREncodeBuffer](#) object.

## Returns

Pointer to copy of encoded message. It is the users's responsibility to free the memory using 'delete []' (i.e., `delete [] ptr;`).

### 7.3.3.7 getMsgLen()

```
virtual size_t ASN1BEREncodeBuffer::getMsgLen ( ) [virtual]
```

This method returns the length of the current encoded message. Calling Sequence:

```
len = encodeBuffer.getMsgLen ( );
```

where encodeBuffer is an [ASN1BEREncodeBuffer](#) object.

#### Returns

Encoded message length;

### 7.3.3.8 getMsgPtr()

```
virtual const OSOCTET* ASN1BEREncodeBuffer::getMsgPtr ( ) [virtual]
```

This method returns the internal pointer to the current encoded message. Calling Sequence:

```
ptr = encodeBuffer.getMsgPtr ( );
```

where encodeBuffer is an [ASN1BEREncodeBuffer](#) object.

#### Returns

Pointer to encoded message.

### 7.3.3.9 init()

```
int ASN1BEREncodeBuffer::init ( )
```

This method reinitializes the encode buffer pointer to allow a new message to be encoded. This makes it possible to reuse one message buffer object in a loop to encode multiple messages. After this method is called, any previously encoded message in the buffer will be overwritten on the next encode call.

Calling Sequence:

```
encodeBuffer.init ( );
```

where encodeBuffer is an [ASN1BEREncodeBuffer](#) object.

#### Returns

Completion status of operation:

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

### 7.3.3.10 isA()

```
virtual OSBOOL ASN1BEREncodeBuffer::isA (
    Type bufferType ) [inline], [virtual]
```

This method checks the type of the message buffer.

## Parameters

<i>bufferType</i>	Enumerated identifier specifying a derived class. The only possible value for this class is BEREncode.
-------------------	--

## Returns

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

### 7.3.3.11 operator<<()

```
ASN1BEREncodeBuffer& ASN1BEREncodeBuffer::operator<< (
    ASN1CType & val )
```

This operator encodes instance of ASN1CType derived class. Use getStatus() method to determine has error occurred during the operation or not.

### 7.3.3.12 setBuffer()

```
int ASN1BEREncodeBuffer::setBuffer (
    OSOCTET * pMsgBuf,
    OSSIZE msgBufLen )
```

This method sets a buffer to receive the encoded message.

## Parameters

<i>pMsgBuf</i>	A pointer to a memory buffer to use to encode a message. The buffer should be declared as an array of unsigned characters (OSOCTETs). This parameter can be set to NULL to specify dynamic encoding (i.e., the encode functions will dynamically allocate a buffer for the message).
<i>msgBufLen</i>	The length of the memory buffer in bytes. If pMsgBuf is NULL, this parameter specifies the initial size of the dynamic buffer; if 0 - the default size will be used.

## Returns

Completion status of operation:

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

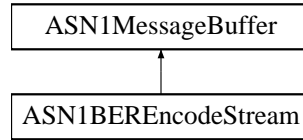
The documentation for this class was generated from the following file:

- [asn1BerCppTypes.h](#)

## 7.4 ASN1BEREncodeStream Class Reference

```
#include <ASN1BEREncodeStream.h>
```

Inheritance diagram for ASN1BEREncodeStream:



### Public Member Functions

- [ASN1BEREncodeStream](#) (OSRTOutputStreamIF &os)
- **ASN1BEREncodeStream** (OSRTOutputStreamIF \*pos, OSBOOL bOwnStream=TRUE)
- virtual void \* [getAppInfo](#) ()
- virtual OSRTCtxtPtr [getContext](#) ()
- virtual OSCTXT \* [getCtxtPtr](#) ()
- virtual char \* [getErrorInfo](#) ()
- virtual char \* [getErrorInfo](#) (char \*pBuf, size\_t &bufSize)
- virtual int [getStatus](#) () const
- virtual void [printErrorInfo](#) ()
- virtual void [resetErrorInfo](#) ()
- virtual void [setAppInfo](#) (void \*pAppInfo)
- virtual void [setDiag](#) (OSBOOL value=TRUE)
- virtual int [close](#) ()
- virtual int [flush](#) ()
- virtual OSBOOL [isOpened](#) ()
- virtual long [write](#) (const OSOCTET \*pdata, size\_t size)
- [ASN1BEREncodeStream](#) & [operator<<](#) (ASN1CType &val)
- int [encodeBMPStr](#) (const Asn116BitCharString &val, ASN1TagType tagging=ASN1EXPL)
- int [encodeBigInt](#) (const char \*pval, ASN1TagType tagging=ASN1EXPL)
- int [encodeBigIntNchars](#) (const char \*pval, size\_t nchars, ASN1TagType tagging=ASN1EXPL)
- int [encodeBitStr](#) (const OSOCTET \*pbits, OSUINT32 numbits, ASN1TagType tagging=ASN1EXPL)
- int [encodeBitStr](#) (const ASN1DynBitStr &val, ASN1TagType tagging=ASN1EXPL)
- int [encodeBool](#) (OSBOOL val, ASN1TagType tagging=ASN1EXPL)
- int [encodeCharStr](#) (const char \*pval, ASN1TagType tagging=ASN1EXPL, ASN1TAG tag=0)
- int [encodeEnum](#) (OSINT32 val, ASN1TagType tagging=ASN1EXPL)
- int [encodeEoc](#) ()
- int [encodeIndefLen](#) ()
- int [encodeInt](#) (OSINT32 val, ASN1TagType tagging=ASN1EXPL)
- int [encodeInt8](#) (OSINT8 val, ASN1TagType tagging=ASN1EXPL)
- int [encodeInt16](#) (OSINT16 val, ASN1TagType tagging=ASN1EXPL)
- int [encodeInt64](#) (OSINT64 val, ASN1TagType tagging=ASN1EXPL)
- int [encodeLen](#) (size\_t len)
- int [encodeNull](#) (ASN1TagType tagging=ASN1EXPL)
- int [encodeObj](#) (ASN1CType &val)
- int [encodeObjId](#) (const ASN1OBJID &val, ASN1TagType tagging=ASN1EXPL)

- int [encodeObjId64](#) (const ASN1OID64 &val, ASN1TagType tagging=ASN1EXPL)
- int [encodeOctStr](#) (const OSOCTET \*pocets, OSUINT32 numocets, ASN1TagType tagging=ASN1EXPL)
- int [encodeOctStr](#) (const ASN1DynOctStr &val, ASN1TagType tagging=ASN1EXPL)
- int [encodeReal](#) (OSREAL val, ASN1TagType tagging=ASN1EXPL)
- int [encodeRelativeOID](#) (const ASN1OBJID &val, ASN1TagType tagging=ASN1EXPL)
- int [encodeTag](#) (ASN1TAG tag)
- int [encodeTagAndIndefLen](#) (ASN1TAG tag)
- int [encodeTagAndLen](#) (ASN1TAG tag, OSINT32 len)
- int [encodeUInt](#) (OSUINT32 val, ASN1TagType tagging=ASN1EXPL)
- int [encodeUInt8](#) (OSUINT8 val, ASN1TagType tagging=ASN1EXPL)
- int [encodeUInt16](#) (OSUINT16 val, ASN1TagType tagging=ASN1EXPL)
- int [encodeUInt64](#) (OSUINT64 val, ASN1TagType tagging=ASN1EXPL)
- int [encodeUnivStr](#) (const Asn132BitCharString &val, ASN1TagType tagging=ASN1EXPL)
- OSBOOL [isA](#) (Type bufferType)

## Protected Attributes

- OSRTOutputStreamIF \* **mpStream**
- OSBOOL **mbOwnStream**

### 7.4.1 Detailed Description

This class is a base class for other ASN.1 BER output stream's classes. It is derived from the ASN1Stream base class. It contains variables and methods specific to streaming encoding of BER messages.

### 7.4.2 Constructor & Destructor Documentation

#### 7.4.2.1 ASN1BEREncodeStream()

```
ASN1BEREncodeStream::ASN1BEREncodeStream (
    OSRTOutputStreamIF & os )
```

A default constructor. Use [getStatus\(\)](#) method to determine has error occurred during the initialization or not.

### 7.4.3 Member Function Documentation

### 7.4.3.1 close()

```
virtual int ASN1BEREncodeStream::close ( ) [inline], [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`

### 7.4.3.2 encodeBigInt()

```
int ASN1BEREncodeStream::encodeBigInt (
    const char * pval,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 INTEGER type. In this case, the integer is assumed to be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits).

#### Parameters

<i>*pval</i>	A pointer to a character string containing the value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

Completion status of operation:

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

#### See also

[berEncStrmBigInt](#)

### 7.4.3.3 encodeBigIntNchars()

```
int ASN1BEREncodeStream::encodeBigIntNchars (
    const char * pval,
    size_t nchars,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 INTEGER type. In this case, the integer is assumed to be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits).

#### Parameters

<i>*pval</i>	A pointer to a character string containing the value to be encoded.
<i>nchars</i>	Number of characters from pval to encode.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

Completion status of operation:

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

#### See also

[berEncStrmBigInt](#)

### 7.4.3.4 encodeBitStr() [1/2]

```
int ASN1BEREncodeStream::encodeBitStr (
    const OSOCTET * pbits,
    OSUINT32 numbits,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 BIT STRING type.

#### Parameters

<i>pbits</i>	A pointer to an OCTET string containing the bit data to be encoded. This string contains bytes having the actual bit settings as they are to be encoded in the message.
<i>numbits</i>	The number of bits within the bit string to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Completion status of operation:

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

## See also

[berEncStrmBitStr](#)

### 7.4.3.5 encodeBitStr() [2/2]

```
int ASN1BEREncodeStream::encodeBitStr (
    const ASN1DynBitStr & val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 BIT STRING type.

## Parameters

<i>val</i>	A reference to the ASN1DynBitStr structure containing a bit data and number of bits to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Completion status of operation:

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

## See also

[berEncStrmBitStr](#)

### 7.4.3.6 encodeBMPStr()

```
int ASN1BEREncodeStream::encodeBMPStr (
    const Asn116BitCharString & val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 BMPString type that is based on a 16-bit character sets.



## Parameters

<i>val</i>	A reference to a structure representing a 16-bit character string to be encoded. This structure contains a character count element and a pointer to an array of 16-bit character elements represented as 16-bit short integers.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Completion status of operation:

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

## See also

[berEncStrmBMPStr](#)

### 7.4.3.7 encodeBool()

```
int ASN1BEREncodeStream::encodeBool (
    OSBOOL val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 BOOLEAN type.

## Parameters

<i>val</i>	A BOOLEAN value to be encoded. A BOOLEAN is defined as a single OCTET whose value is 0 for FALSE and any other value for TRUE.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Completion status of operation:

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

## See also

[berEncStrmBool](#)

### 7.4.3.8 encodeCharStr()

```
int ASN1BEREncodeStream::encodeCharStr (
    const char * pval,
    ASN1TagType tagging = ASN1EXPL,
    ASN1TAG tag = 0 )
```

This method encodes a variable of the ASN.1 character string type.

#### Parameters

<i>pval</i>	A pointer to a null-terminated C character string to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.

#### Returns

Completion status of operation:

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

#### See also

[berEncStrmCharStr](#)

### 7.4.3.9 encodeEnum()

```
int ASN1BEREncodeStream::encodeEnum (
    OSINT32 val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 ENUMERATED type. The enumerated encoding is identical to that of an integer. The compiler adds additional checks to the generated code to ensure the value is within the given set.

#### Parameters

<i>val</i>	An integer containing the enumerated value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Completion status of operation:

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

## See also

[berEncStrmEnum](#)

### 7.4.3.10 encodeEoc()

```
int ASN1BEREncodeStream::encodeEoc ( )
```

This method encodes end-of-contents octets (EOC) into the stream. EOC is two zero octets (it is documented in the X.690 standard). This method must be called when the encoding of the complex type with indefinite length is finishing.

## Returns

Completion status of operation:

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

## See also

[berEncStrmEOC](#), [berEncStrmTagAndIndefLen](#),

### 7.4.3.11 encodeIndefLen()

```
int ASN1BEREncodeStream::encodeIndefLen ( )
```

This method is used to encode the indefinite length indicator. This can be used to manually create an indefinite length wrapper around long or constructed records.

## Returns

Completion status of operation:

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

## See also

[berEncStrmWriteOctet](#)

### 7.4.3.12 encodeInt()

```
int ASN1BEREncodeStream::encodeInt (
    OSINT32 val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 INTEGER type.

#### Parameters

<i>val</i>	A 32-bit INTEGER value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

Completion status of operation:

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

#### See also

[berEncStrmInt](#)

#### 7.4.3.13 encodeInt16()

```
int ASN1BEREncodeStream::encodeInt16 (
    OSINT16 val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a 16-bit variable of the ASN.1 INTEGER type.

#### Parameters

<i>val</i>	A 16-bit INTEGER value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

Completion status of operation:

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

#### See also

[berEncStrmInt16](#)

#### 7.4.3.14 encodeInt64()

```
int ASN1BEREncodeStream::encodeInt64 (
    OSINT64 val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a 64-bit variable of the ASN.1 INTEGER type.

##### Parameters

<i>val</i>	A 64-bit INTEGER value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

##### Returns

Completion status of operation:

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

##### See also

[berEncStrmInt64](#)

#### 7.4.3.15 encodeInt8()

```
int ASN1BEREncodeStream::encodeInt8 (
    OSINT8 val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes an 8-bit variable of the ASN.1 INTEGER type.

##### Parameters

<i>val</i>	An 8-bit INTEGER value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

##### Returns

Completion status of operation:

- 0 (0) = success,

- negative return value is error.

See also

[berEncStrmInt8](#)

#### 7.4.3.16 encodeLen()

```
int ASN1BEREncodeStream::encodeLen (
    size_t len )
```

This method is used to encode a length in BER format.

Parameters

<i>len</i>	The length of the contents field.
------------	-----------------------------------

Returns

Completion status of operation:

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

See also

[berEncStrmLength](#)

#### 7.4.3.17 encodeNull()

```
int ASN1BEREncodeStream::encodeNull (
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 NULL type.

Parameters

<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.
----------------	---

## Returns

Completion status of operation:

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

## See also

[berEncStrmRelativeOID](#)

### 7.4.3.18 encodeObj()

```
int ASN1BEREncodeStream::encodeObj (
    ASN1CType & val )
```

This method encodes an ASN.1 constructed object to the stream.

#### Parameters

<i>val</i>	A reference to an object to be encoded.
------------	---

## Returns

Completion status of operation:

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

### 7.4.3.19 encodeObjId()

```
int ASN1BEREncodeStream::encodeObjId (
    const ASN1OBJID & val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 OBJECT IDENTIFIER type.

#### Parameters

<i>val</i>	A reference to an object identifier structure. This structure contains an integer to hold the number of subidentifiers in the object and an array to hold the subidentifier values.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Completion status of operation:

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

## See also

[berEncStrmObjId](#)

### 7.4.3.20 encodeObjId64()

```
int ASN1BEREncodeStream::encodeObjId64 (
    const ASN1OID64 & val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 OBJECT IDENTIFIER type using 64-bit subidentifiers.

## Parameters

<i>val</i>	A reference to a 64-bit object identifier structure. This structure contains an integer to hold the number of subidentifiers in the object and an array of 64-bit unsigned integers to hold the subidentifier values.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Completion status of operation:

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

## See also

[berEncStrmObjId64](#)

### 7.4.3.21 encodeOctStr() [1/2]

```
int ASN1BEREncodeStream::encodeOctStr (
    const OSOCTET * popts,
    OSUINT32 numopts,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 OCTET STRING type.



#### Parameters

<i>pocts</i>	A pointer to an OCTET STRING containing the octet data to be encoded.
<i>numocts</i>	The number of octets (bytes) within the OCTET STRING to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

Completion status of operation:

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

#### See also

[berEncStrmOctStr](#)

#### 7.4.3.22 encodeOctStr() [2/2]

```
int ASN1BEREncodeStream::encodeOctStr (  
    const ASN1DynOctStr & val,  
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 OCTET STRING type.

#### Parameters

<i>val</i>	A reference to the ASN1DynOctStr structure containing an octet data and number of octets to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

Completion status of operation:

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

#### See also

[berEncStrmOctStr](#)

### 7.4.3.23 encodeReal()

```
int ASN1BEREncodeStream::encodeReal (
    OSREAL val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the REAL data type. It provides support for the plus-infinity and minus-infinity special real values. Use the `::rtxGetPlusInfinity` or `::rtxGetMinusInfinity` functions to get these special values.

#### Parameters

<i>val</i>	An OSREAL data type. This is defined to be the C double type. Special real values plus and minus infinity are encoded by using the <code>::rtxGetPlusInfinity</code> and <code>::rtxGetMinusInfinity</code> functions to set the real value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

Completion status of operation:

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

#### See also

[berEncStrmReal](#)

### 7.4.3.24 encodeRelativeOID()

```
int ASN1BEREncodeStream::encodeRelativeOID (
    const ASN1OBJID & val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 RELATIVE-OID type.

#### Parameters

<i>val</i>	A reference to an object identifier structure. This structure contains an integer to hold the number of subidentifiers in the object and an array to hold the subidentifier values.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Completion status of operation:

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

## See also

[berEncStrmRelativeOID](#)

### 7.4.3.25 encodeTag()

```
int ASN1BEREncodeStream::encodeTag (  
    ASN1TAG tag )
```

This method is used to encode the ASN.1 tag field that preface each block of message data. The ASN1C compiler generates calls to this function to handle the encoding of user-defined tags within an ASN.1 specification.

## Parameters

<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.
------------	---

## Returns

Completion status of operation:

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

## See also

[berEncStrmTag](#)

### 7.4.3.26 encodeTagAndIndefLen()

```
int ASN1BEREncodeStream::encodeTagAndIndefLen (  
    ASN1TAG tag )
```

This method is used to encode a tag value and an indefinite length. This can be used to manually create an indefinite length wrapper around long or constructed records.

#### Parameters

<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.
------------	---

#### Returns

Completion status of operation:

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

#### See also

[berEncStrmTagAndIndefLen](#)

#### 7.4.3.27 encodeTagAndLen()

```
int ASN1BEREncodeStream::encodeTagAndLen (
    ASN1TAG tag,
    OSINT32 len )
```

This method is used to encode the ASN.1 tag and length fields that preface each block of message data.

#### Parameters

<i>tag</i>	The ASN.1 tag to be encoded in the message. This parameter is passed using the ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.
<i>len</i>	The length of the contents field. This parameter can be used to specify the actual length, or the special constant 'ASN_K_INDEFLEN' can be used to specify that an indefinite length specification should be encoded.

#### Returns

Completion status of operation:

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

#### See also

[berEncStrmTagAndLen](#)

### 7.4.3.28 encodeUInt()

```
int ASN1BEREncodeStream::encodeUInt (
    OSUINT32 val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes an unsigned variable of the ASN.1 INTEGER type.

#### Parameters

<i>val</i>	An unsigned INTEGER value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

Completion status of operation:

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

#### See also

[berEncStrmUInt](#)

### 7.4.3.29 encodeUInt16()

```
int ASN1BEREncodeStream::encodeUInt16 (
    OSUINT16 val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a 16-bit unsigned variable of the ASN.1 INTEGER type.

#### Parameters

<i>val</i>	A 16-bit unsigned INTEGER value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

#### Returns

Completion status of operation:

- 0 (0) = success,

- negative return value is error.

See also

[berEncStrmUInt16](#)

#### 7.4.3.30 encodeUInt64()

```
int ASN1BEREncodeStream::encodeUInt64 (
    OSUINT64 val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a 64-bit unsigned variable of the ASN.1 INTEGER type.

Parameters

<i>val</i>	A 64-bit unsigned INTEGER value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

Returns

Completion status of operation:

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

See also

[berEncStrmUInt64](#)

#### 7.4.3.31 encodeUInt8()

```
int ASN1BEREncodeStream::encodeUInt8 (
    OSUINT8 val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes an 8-bit unsigned variable of the ASN.1 INTEGER type.

## Parameters

<i>val</i>	An 8-bit unsigned INTEGER value to be encoded.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Completion status of operation:

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

## See also

[berEncStrmUInt8](#)

### 7.4.3.32 encodeUnivStr()

```
int ASN1BEREncodeStream::encodeUnivStr (
    const Asn132BitCharString & val,
    ASN1TagType tagging = ASN1EXPL )
```

This method encodes a variable of the ASN.1 UniversalString type that is based on a 32-bit character sets.

## Parameters

<i>val</i>	A reference to a structure representing a 32-bit character string to be encoded. This structure contains a character count element and a pointer to an array of 32-bit character elements represented as 32-bit unsigned integers.
<i>tagging</i>	An enumerated type whose value is set to either 'ASN1EXPL' (for explicit tagging) or 'ASN1IMPL' (for implicit). Controls whether the universal tag value for this type is added or not. Users will generally always set this value to 'ASN1EXPL'.

## Returns

Completion status of operation:

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

## See also

[berEncStrmUnivStr](#)

#### 7.4.3.33 flush()

```
virtual int ASN1BEREncodeStream::flush ( ) [inline], [virtual]
```

Flushes the buffered data to the stream.

#### Returns

Completion status of operation:

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

#### See also

::rtxStreamFlush

#### 7.4.3.34 getAppInfo()

```
virtual void* ASN1BEREncodeStream::getAppInfo ( ) [inline], [virtual]
```

Returns a pointer to application-specific information block

#### 7.4.3.35 getContext()

```
virtual OSRTCtxtPtr ASN1BEREncodeStream::getContext ( ) [inline], [virtual]
```

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

#### Returns

Context smart pointer object.

#### 7.4.3.36 getCtxtPtr()

```
virtual OSCTXT* ASN1BEREncodeStream::getCtxtPtr ( ) [inline], [virtual]
```

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

#### Returns

The pointer to C runtime context.



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

```
virtual char* ASN1BEREncodeStream::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.

#### 7.4.3.38 `getErrorInfo()` [2/2]

```
virtual char* ASN1BEREncodeStream::getErrorInfo (
    char * pBuf,
    size_t & bufSize ) [inline], [virtual]
```

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

#### Parameters

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

#### Returns

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

#### 7.4.3.39 `getStatus()`

```
virtual int ASN1BEREncodeStream::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. 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.

#### 7.4.3.40 isA()

```
OSBOOL ASN1BEREncodeStream::isA (
    Type bufferType )
```

This method matches an enumerated identifier defined in the base class. One identifier is declared for each of the derived classes.

##### Parameters

<i>bufferType</i>	Enumerated identifier specifying a derived class. This type is defined as a public access type in the ASN1MessageBufferIF base interface. Possible values are: BEREncode, BERDecode, PEREncode, PERDecode, XEREncode, XERDecode, XMLEncode, XMLDecode, Stream.
-------------------	--

##### Returns

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

#### 7.4.3.41 isOpened()

```
virtual OSBOOL ASN1BEREncodeStream::isOpened ( ) [inline], [virtual]
```

Checks, is the stream opened or not.

##### Returns

TRUE, if the stream is opened, FALSE otherwise.

##### See also

`::rtxStreamIsOpened`

#### 7.4.3.42 operator<<()

```
ASN1BEREncodeStream& ASN1BEREncodeStream::operator<< (
    ASN1Type & val )
```

Encodes an ASN.1 constructed object to the stream. Use [getStatus\(\)](#) method to determine has error occurred during the operation or not.

## Parameters

<i>val</i>	A reference to an object to be encoded.
------------	---

## Returns

reference to this class to perform sequential encoding.

### 7.4.3.43 printErrorInfo()

```
virtual void ASN1BEREncodeStream::printErrorInfo ( ) [inline], [virtual]
```

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

### 7.4.3.44 resetErrorInfo()

```
virtual void ASN1BEREncodeStream::resetErrorInfo ( ) [inline], [virtual]
```

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

### 7.4.3.45 setAppInfo()

```
virtual void ASN1BEREncodeStream::setAppInfo (
    void * pAppInfo ) [inline], [virtual]
```

Sets the application-specific information block.

### 7.4.3.46 setDiag()

```
virtual void ASN1BEREncodeStream::setDiag (
    OSBOOL value = TRUE ) [inline], [virtual]
```

The setDiag method will turn diagnostic tracing on or off.

## Parameters

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

### 7.4.3.47 write()

```
virtual long ASN1BEREncodeStream::write (
    const OSOCTET * pdata,
    size_t size ) [inline], [virtual]
```

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

#### Parameters

<i>pdata</i>	Pointer to the data to be written.
<i>size</i>	The number of octets to write.

#### Returns

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

#### See also

`::rtxStreamWrite`

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

- [ASN1BEREncodeStream.h](#)

## 7.5 ASN1BERLength Class Reference

### Public Member Functions

- **ASN1BERLength** (size\_t length, OSBOOL indef=FALSE)
- size\_t **getLength** () const
- OSBOOL **isIndef** () const
- void **setLength** (size\_t length)
- void **setIndef** (OSBOOL value=TRUE)
- **operator size\_t** () const

### Protected Attributes

- size\_t **mLength**
- OSBOOL **mblIndef**

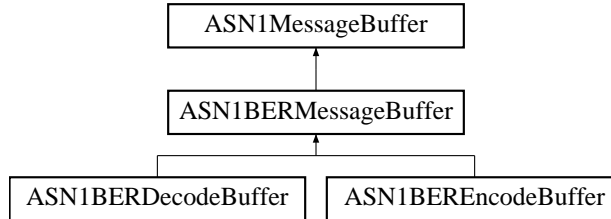
The documentation for this class was generated from the following file:

- [asn1BerCppType.h](#)

## 7.6 ASN1BERMessageBuffer Class Reference

```
#include <asn1BerCppTypes.h>
```

Inheritance diagram for ASN1BERMessageBuffer:



### Public Member Functions

- int `calcIndefLen` (OSOCTET \*buf\_p, OSSIZE bufSize, OSSIZE \*pSize)
- int `calcIndefLen` (OSOCTET \*buf\_p, int bufSize=INT\_MAX)
- void `binDump` ()
- void `hexDump` (OSSIZE numocts)
- int `CalcIndefLen` (OSOCTET \*buf\_p)
- void `BinDump` ()
- void `HexDump` (OSSIZE numocts)

### Protected Member Functions

- `ASN1BERMessageBuffer` (Type bufferType)
- `ASN1BERMessageBuffer` (Type bufferType, OSRTContext \*pContext)

### 7.6.1 Detailed Description

The `ASN1BERMessageBuffer` class is derived from the `ASN1MessageBuffer` base class. It is the base class for the `ASN1BEREncodeBuffer` and `ASN1BERDecodeBuffer` derived classes. It contains variables and methods specific to encoding or decoding ASN.1 messages using the Basic Encoding Rules (BER) and Distinguished Encoding Rules (DER). It is used to manage the buffer into which an ASN.1 message is to be encoded or decoded.

### 7.6.2 Constructor & Destructor Documentation

#### 7.6.2.1 `ASN1BERMessageBuffer()` [1/2]

```
ASN1BERMessageBuffer::ASN1BERMessageBuffer (  
    Type bufferType ) [inline], [protected]
```

The protected constructor creates a new context and sets the buffer class type. Use `getStatus()` method to determine has error occurred during the initialization or not.

## Parameters

<i>bufferType</i>	Type of message buffer that is being created (for example, BEREncode or BERDecode).
-------------------	---

### 7.6.2.2 ASN1BERMessageBuffer() [2/2]

```
ASN1BERMessageBuffer::ASN1BERMessageBuffer (
    Type bufferType,
    OSRContext * pContext ) [inline], [protected]
```

The protected constructor uses an existing context and sets the buffer class type. Use getStatus() method to determine has error occurred during the initialization or not.

## Parameters

<i>bufferType</i>	Type of message buffer that is being created (for example, BEREncode or BERDecode).
<i>pContext</i>	A pointer to the context structure with which this buffer is associated.

## 7.6.3 Member Function Documentation

### 7.6.3.1 binDump()

```
void ASN1BERMessageBuffer::binDump ( ) [inline]
```

This method outputs a formatted binary dump of the current buffer contents to stdout.

Calling Sequence:

```
messageBuffer.binDump ( );
```

where messageBuffer is an [ASN1BERMessageBuffer](#) derived class object.

References xu\_dump().

### 7.6.3.2 calcIndefLen() [1/2]

```
int ASN1BERMessageBuffer::calcIndefLen (
    OSOCTET * buf_p,
    OSSIZE bufSize,
    OSSIZE * pSize ) [inline]
```

This method calculates the actual length of an indefinite length message component. This version of the method will calculate lengths up the full size of a size type (64 bits on 64-bit systems).

Calling Sequence:

```
len=messageBuffer.calcIndefLen (buf_p, bufSize, pSize);
```

where messageBuffer is an [ASN1BERMessageBuffer](#) derived class object.

#### Parameters

<i>buf_p</i>	A pointer to a message component encoded using indefinite length encoding.
<i>bufSize</i>	Size of the buffer <i>buf_p</i> (in octets).
<i>pSize</i>	Pointer to size-typed variable to receive size.

#### Returns

Zero if successful, or a negative status value if error.

References [xd\\_indeflen64\(\)](#).

### 7.6.3.3 calcIndefLen() [2/2]

```
int ASN1BERMessageBuffer::calcIndefLen (
    OSOCTET * buf_p,
    int bufSize = INT_MAX ) [inline]
```

This method calculates the actual length of an indefinite length message component. This version of the method will only calculate lengths up the size of a signed integer (INT\_MAX). It is considered to be deprecated.

Calling Sequence:

```
len=messageBuffer.calcIndefLen (buf_p);
```

where messageBuffer is an [ASN1BERMessageBuffer](#) derived class object.

#### Parameters

<i>buf_p</i>	A pointer to a message component encoded using indefinite length encoding.
<i>bufSize</i>	Size of the buffer <i>buf_p</i> (in octets).

#### Returns

Length, in octets, of message component, as int.

References `xd_indeflen_ex()`.

#### 7.6.3.4 `hexDump()`

```
void ASN1BERMessageBuffer::hexDump (
    OSSIZE numocts ) [inline]
```

This method outputs a hexadecimal dump of the current buffer contents to stdout.

Calling sequence:

```
messageBuffer.hexDump ( );
```

where `messageBuffer` is an [ASN1BERMessageBuffer](#) derived class object.

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

- [asn1BerCppType.h](#)



## Chapter 8

# File Documentation

### 8.1 asn1ber.h File Reference

```
#include "rtsrc/asn1type.h"  
#include "rtbersrc/berMacros.h"  
#include "rtxsrc/rtxBuffer.h"
```

#### Macros

- #define `xd_utf8str`(pctxt, object\_p, tagging, length) `xd_charstr` (pctxt, (const char\*\*)object\_p, tagging, ASN\_ID\_UTF8String, length)
- #define `xd_indeflen`(m) `xd_indeflen_ex`(m, INT\_MAX)
- #define `xe_utf8str`(pctxt, object\_p, tagging) `xe_charstr` (pctxt, (const char\*)object\_p, tagging, ASN\_ID\_UTF8String)
- #define `xu_addTagErrParm` `berErrAddTagParm`
- #define `xu_hex_dump`(msg, numoct, hdr) `rtxHexDump`(msg,numoct)

#### Typedefs

- typedef OSRTBufLocDescr **Asn1BufLocDescr**

#### Functions

- int `xd_tag` (OSCTXT \*pctxt, ASN1TAG \*tag\_p)
- int `xd_tag_len` (OSCTXT \*pctxt, ASN1TAG \*tag\_p, int \*len\_p, OSOCTET flags)
- int `xd_tag_len_64` (OSCTXT \*pctxt, ASN1TAG \*tag\_p, OSSIZE \*len\_p, OSBOOL \*pIndefLen, OSOCTET flags)
- int `xd_match` (OSCTXT \*pctxt, ASN1TAG tag, int \*len\_p, OSOCTET flags)
- int `xd_match64` (OSCTXT \*pctxt, ASN1TAG tag, OSSIZE \*len\_p, OSBOOL \*pindef, OSOCTET flags)
- int `xd_boolean` (OSCTXT \*pctxt, OSBOOL \*object\_p, ASN1TagType tagging, int length)
- int `xd_integer` (OSCTXT \*pctxt, OSINT32 \*object\_p, ASN1TagType tagging, int length)

- int [xd\\_int8](#) (OSCTXT \*pctxt, OSINT8 \*object\_p, ASN1TagType tagging, int length)
- int [xd\\_int16](#) (OSCTXT \*pctxt, OSINT16 \*object\_p, ASN1TagType tagging, int length)
- int [xd\\_unsigned](#) (OSCTXT \*pctxt, OSUINT32 \*object\_p, ASN1TagType tagging, int length)
- int [xd\\_uint8](#) (OSCTXT \*pctxt, OSUINT8 \*object\_p, ASN1TagType tagging, int length)
- int [xd\\_uint16](#) (OSCTXT \*pctxt, OSUINT16 \*object\_p, ASN1TagType tagging, int length)
- int [xd\\_int64](#) (OSCTXT \*pctxt, OSINT64 \*object\_p, ASN1TagType tagging, int length)
- int [xd\\_uint64](#) (OSCTXT \*pctxt, OSUINT64 \*object\_p, ASN1TagType tagging, int length)
- int [xd\\_bigint](#) (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, int length)
- int [xd\\_bitstr\\_s](#) (OSCTXT \*pctxt, OSOCTET \*object\_p, OSUINT32 \*numbits\_p, ASN1TagType tagging, int length)
- int [xd\\_bitstrExt\\_s](#) (OSCTXT \*pctxt, OSOCTET \*object\_p, OSUINT32 \*numbits\_p, OSOCTET \*\*extdata, ASN1TagType tagging, int length)
- int [xd\\_bitstr64\\_s](#) (OSCTXT \*pctxt, OSOCTET \*object\_p, OSSIZE \*numbits\_p, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int [xd\\_bitstr64Ext\\_s](#) (OSCTXT \*pctxt, OSOCTET \*object\_p, OSSIZE \*numbits\_p, OSOCTET \*\*extdata, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int [xd\\_bitstr](#) (OSCTXT \*pctxt, const OSOCTET \*\*object\_p2, OSUINT32 \*numbits\_p, ASN1TagType tagging, int length)
- int [xd\\_bitstr64](#) (OSCTXT \*pctxt, const OSOCTET \*\*object\_p2, OSSIZE \*numbits\_p, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int [xd\\_octstr\\_s](#) (OSCTXT \*pctxt, OSOCTET \*object\_p, OSUINT32 \*pnumocts, ASN1TagType tagging, int length)
- int [xd\\_octstr64\\_s](#) (OSCTXT \*pctxt, OSOCTET \*object\_p, OSSIZE \*pnumocts, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int [xd\\_octstr](#) (OSCTXT \*pctxt, const OSOCTET \*\*object\_p2, OSUINT32 \*pnumocts, ASN1TagType tagging, int length)
- int [xd\\_octstr64](#) (OSCTXT \*pctxt, OSOCTET \*\*object\_p2, OSSIZE \*pnumocts, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int [xd\\_charstr](#) (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int [xd\\_charstr64](#) (OSCTXT \*pctxt, char \*\*object\_p, ASN1TagType tagging, ASN1TAG tag, OSSIZE length, OSBOOL indefLen)
- int [xd\\_datestr](#) (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int [xd\\_timestr](#) (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int [xd\\_datetimestr](#) (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int [xd\\_timeofdaystr](#) (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int [xd\\_durationstr](#) (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int [berDecCharArray](#) (OSCTXT \*pctxt, char \*charArray, OSSIZE arraySize, ASN1TagType tagging, ASN1TAG tag, int length)
- int [xd\\_16BitCharStr](#) (OSCTXT \*pctxt, Asn116BitCharString \*object\_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int [xd\\_16BitCharStr64](#) (OSCTXT \*pctxt, Asn116BitCharString \*object\_p, ASN1TagType tagging, ASN1TAG tag, OSSIZE length, OSBOOL indefLen)
- int [xd\\_32BitCharStr](#) (OSCTXT \*pctxt, Asn132BitCharString \*object\_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int [xd\\_32BitCharStr64](#) (OSCTXT \*pctxt, Asn132BitCharString \*object\_p, ASN1TagType tagging, ASN1TAG tag, OSSIZE length, OSBOOL indefLen)
- int [xd\\_null](#) (OSCTXT \*pctxt, ASN1TagType tagging)
- int [xd\\_objid](#) (OSCTXT \*pctxt, ASN1OBJID \*object\_p, ASN1TagType tagging, int length)
- int [xd\\_oid64](#) (OSCTXT \*pctxt, ASN1OID64 \*object\_p, ASN1TagType tagging, int length)
- int [xd\\_reloid](#) (OSCTXT \*pctxt, ASN1OBJID \*object\_p, ASN1TagType tagging, int length)
- int [xd\\_real](#) (OSCTXT \*pctxt, OSREAL \*object\_p, ASN1TagType tagging, int length)
- int [xd\\_enum](#) (OSCTXT \*pctxt, OSINT32 \*object\_p, ASN1TagType tagging, int length)
- int [xd\\_enumUnsigned](#) (OSCTXT \*pctxt, OSUINT32 \*object\_p, ASN1TagType tagging, int length)
- int [xd\\_OpenType](#) (OSCTXT \*pctxt, const OSOCTET \*\*object\_p2, OSSIZE \*pnumocts)

- int [xd\\_OpenTypeExt](#) (OSCTXT \*pctxt, ASN1CCB \*ccb\_p, ASN1TAG \*tags, int tagCount, OSRTDList \*pElemList)
- int [xd\\_OpenTypeExt64](#) (OSCTXT \*pctxt, const OSOCTET \*conspr, OSSIZE conslen, OSBOOL indefLen, ASN1TAG \*tags, OSSIZE tagCount, OSRTDList \*pElemList)
- int [xd\\_OpenTypeAppend](#) (OSCTXT \*pctxt, OSRTDList \*pElemList)
- int [xd\\_real10](#) (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, int length)
- int [xd\\_setp](#) (OSCTXT \*pctxt, const OSOCTET \*msg\_p, int msglen, ASN1TAG \*tag\_p, int \*len\_p)
- int [xd\\_setp64](#) (OSCTXT \*pctxt, const OSOCTET \*msg\_p, OSSIZE msglen, ASN1TAG \*tag\_p, OSSIZE \*len\_p, OSBOOL \*pIndefLen)
- int [xd\\_indeflen\\_ex](#) (const OSOCTET \*msg\_p, int bufSize)
- int [xd\\_indeflen64](#) (const OSOCTET \*msg\_p, OSSIZE bufSize, OSSIZE \*plength)
- int [xd\\_len](#) (OSCTXT \*pctxt, int \*len\_p)
- int [xd\\_len64](#) (OSCTXT \*pctxt, OSSIZE \*len\_p, OSBOOL \*p indef)
- OSBOOL [xd\\_chkend](#) (OSCTXT \*pctxt, const ASN1CCB \*ccb\_p)
- OSBOOL [xd\\_chkend64](#) (OSCTXT \*pctxt, const OSOCTET \*conspr, OSSIZE conslen, OSBOOL indef)
- int [xd\\_count](#) (OSCTXT \*pctxt, int length, int \*count\_p)
- int [xd\\_count64](#) (OSCTXT \*pctxt, OSSIZE length, OSBOOL indefLen, OSSIZE \*count\_p)
- int [xd\\_NextElement](#) (OSCTXT \*pctxt)
- int [xd\\_Tag1AndLen](#) (OSCTXT \*pctxt, OSINT32 \*len\_p)
- int [xd\\_memcpy](#) (OSCTXT \*pctxt, OSOCTET \*object\_p, int length)
- int [xd\\_match1](#) (OSCTXT \*pctxt, OSOCTET tag, int \*len\_p)
- int [xd\\_match1\\_64](#) (OSCTXT \*pctxt, OSOCTET tag, OSSIZE \*len\_p, OSBOOL \*p indef)
- int [xdf\\_tag](#) (FILE \*fp, ASN1TAG \*ptag, OSOCTET \*buffer, int \*pbufidx)
- int [xdf\\_len](#) (FILE \*fp, OSINT32 \*plen, OSOCTET \*buffer, int \*pbufidx)
- int [xdf\\_TagAndLen](#) (FILE \*fp, ASN1TAG \*ptag, OSINT32 \*plen, OSOCTET \*buffer, int \*pbufidx)
- int [xdf\\_ReadPastEOC](#) (FILE \*fp, OSOCTET \*buffer, int bufsiz, int \*pbufidx)
- int [xdf\\_ReadContents](#) (FILE \*fp, int len, OSOCTET \*buffer, int bufsiz, int \*pbufidx)
- int [xe\\_identifler](#) (OSCTXT \*pctxt, OSUINT32 ident)
- int [xe\\_tag](#) (OSCTXT \*pctxt, ASN1TAG tag)
- int [xe\\_tag\\_len](#) (OSCTXT \*pctxt, ASN1TAG tag, int length)
- int [xe\\_boolean](#) (OSCTXT \*pctxt, OSBOOL \*object\_p, ASN1TagType tagging)
- int [xe\\_integer](#) (OSCTXT \*pctxt, int \*object\_p, ASN1TagType tagging)
- int [xe\\_unsigned](#) (OSCTXT \*pctxt, OSUINT32 \*object\_p, ASN1TagType tagging)
- int [xe\\_int8](#) (OSCTXT \*pctxt, OSINT8 \*object\_p, ASN1TagType tagging)
- int [xe\\_int16](#) (OSCTXT \*pctxt, OSINT16 \*object\_p, ASN1TagType tagging)
- int [xe\\_int64](#) (OSCTXT \*pctxt, OSINT64 \*object\_p, ASN1TagType tagging)
- int [xe\\_uint64](#) (OSCTXT \*pctxt, OSUINT64 \*object\_p, ASN1TagType tagging)
- int [xe\\_uint8](#) (OSCTXT \*pctxt, OSUINT8 \*object\_p, ASN1TagType tagging)
- int [xe\\_uint16](#) (OSCTXT \*pctxt, OSUINT16 \*object\_p, ASN1TagType tagging)
- int [xe\\_bigint](#) (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging)
- int [xe\\_bigintn](#) (OSCTXT \*pctxt, const char \*object\_p, size\_t nchars, ASN1TagType tagging)
- int [xe\\_bitstr](#) (OSCTXT \*pctxt, const OSOCTET \*object\_p, OSSIZE numbits, ASN1TagType tagging)
- int [xe\\_bitstrExt](#) (OSCTXT \*pctxt, const OSOCTET \*object\_p, OSSIZE numbits, OSSIZE dataSize, const OSOCTET \*extdata, ASN1TagType tagging)
- int [xe\\_octstr](#) (OSCTXT \*pctxt, const OSOCTET \*object\_p, OSSIZE numocts, ASN1TagType tagging)
- int [xe\\_charstr](#) (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int [xe\\_16BitCharStr](#) (OSCTXT \*pctxt, Asn116BitCharString \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int [xe\\_32BitCharStr](#) (OSCTXT \*pctxt, Asn132BitCharString \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int [xe\\_datestr](#) (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int [xe\\_timestr](#) (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int [xe\\_datetimestr](#) (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)

- int [xe\\_timeofdaystr](#) (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int [xe\\_durationstr](#) (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int [xe\\_null](#) (OSCTXT \*pctxt, ASN1TagType tagging)
- int [xe\\_objid](#) (OSCTXT \*pctxt, ASN1OBJID \*object\_p, ASN1TagType tagging)
- int [xe\\_oid64](#) (OSCTXT \*pctxt, ASN1OID64 \*object\_p, ASN1TagType tagging)
- int [xe\\_reloid](#) (OSCTXT \*pctxt, ASN1OBJID \*object\_p, ASN1TagType tagging)
- int [xe\\_enum](#) (OSCTXT \*pctxt, OSINT32 \*object\_p, ASN1TagType tagging)
- int [xe\\_enumUnsigned](#) (OSCTXT \*pctxt, OSUINT32 \*object\_p, ASN1TagType tagging)
- int [xe\\_real](#) (OSCTXT \*pctxt, OSREAL \*object\_p, ASN1TagType tagging)
- int [xe\\_OpenType](#) (OSCTXT \*pctxt, const OSOCTET \*object\_p, OSSIZE numocts)
- int [xe\\_OpenTypeExt](#) (OSCTXT \*pctxt, OSRTDList \*pElemList)
- int [xe\\_OpenTypeExtDer](#) (OSCTXT \*pctxt, OSRTDList \*pElemList, OSRTSList \*pBufList)
- int [xe\\_real10](#) (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging)
- int [xe\\_derReal10](#) (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging)
- int [xe\\_setp](#) (OSCTXT \*pctxt, OSOCTET \*buf\_p, OSSIZE bufsiz)
- OSOCTET \* [xe\\_getp](#) (OSCTXT \*pctxt)
- void [xe\\_free](#) (OSCTXT \*pctxt)
- int [xe\\_expandBuffer](#) (OSCTXT \*pctxt, size\_t length)
- int [xe\\_memcpy](#) (OSCTXT \*pctxt, const OSOCTET \*object\_p, size\_t length)
- int [xe\\_len](#) (OSCTXT \*pctxt, int length)
- int [xe\\_len64](#) (OSCTXT \*pctxt, OSSIZE length, OSBOOL indef)
- int [xe\\_derCanonicalSort](#) (OSCTXT \*pctxt, OSRTSList \*pList)
- int [xe\\_derCanSortSet](#) (OSCTXT \*pctxt, OSRTSList \*pList)
- int [xe\\_TagAndIndefLen](#) (OSCTXT \*pctxt, ASN1TAG tag, int length)
- void [xe\\_getBufLocDescr](#) (OSCTXT \*pctxt, OSSIZE length, Asn1BufLocDescr \*pDescr)
- int [derEncBitString](#) (OSCTXT \*pctxt, const OSOCTET \*pvalue, OSSIZE numbits, ASN1TagType tagging)
- int [berDefToIndefLen](#) (OSCTXT \*pSrcCtxt, OSCTXT \*pDstCtxt)
- int [berIndefToDefLen](#) (OSCTXT \*pSrcCtxt, OSCTXT \*pDstCtxt)
- OSBOOL [berErrAddTagParm](#) (OSCTXT \*pctxt, ASN1TAG tag)
- int [berErrUnexpTag](#) (OSCTXT \*pctxt, ASN1TAG exptag)
- int [berGetLibVersion](#) (OSVOIDARG)
- const char \* [berGetLibInfo](#) (OSVOIDARG)
- int [berParseTagLen](#) (const OSOCTET \*buffer, size\_t bufix, size\_t bufsiz, ASN1TAG \*ptag, size\_t \*plen)
- const char \* [berTagToString](#) (ASN1TAG tag, char \*buffer, size\_t bufsiz)
- const char \* [berTagToDynStr](#) (OSCTXT \*pctxt, ASN1TAG tag)
- int [berValidateIso8601DateStr](#) (OSCTXT \*pctxt, const char \*\*ppvalue)
- int [berValidateIso8601DurationStr](#) (OSCTXT \*pctxt, const char \*\*ppvalue)
- int [berValidateIso8601TimeStr](#) (OSCTXT \*pctxt, const char \*\*ppvalue)
- int [xu\\_verify\\_len](#) (OSOCTET \*msg\_p)
- void \* [xu\\_parse\\_mmbuf](#) (OSOCTET \*\*buf\_p2, int \*buflen\_p, OSOCTET \*start\_p, int bufsiz)
- void [xu\\_alloc\\_array](#) (OSCTXT \*pctxt, ASN1SeqOf \*seqOf\_p, int recSize, int recCount)
- void [xu\\_octscopy\\_s](#) (OSUINT32 \*nocts\_p, OSOCTET \*data\_p, char \*cstr, char zterm)
- void [xu\\_octscopy\\_ss](#) (ASN1OctStr \*octStr\_p, char \*cstring, char zterm)
- void [xu\\_octscopy\\_d](#) (OSCTXT \*pctxt, OSUINT32 \*nocts\_p, const OSOCTET \*\*data\_p2, char \*cstring, char zterm)
- void [xu\\_octscopy\\_ds](#) (OSCTXT \*pctxt, ASN1DynOctStr \*octStr\_p, char \*cstring, char zterm)
- void [xu\\_octmcpy\\_s](#) (ASN1OctStr \*octStr\_p, void \*data\_p, int datalen)
- void [xu\\_octmcpy\\_d](#) (OSCTXT \*pctxt, ASN1DynOctStr \*octStr\_p, void \*data\_p, int datalen)
- char \* [xu\\_fetchstr](#) (int numocts, char \*data)
- int [xu\\_hexstrcpy](#) (char \*data, char \*hstring)
- int [xu\\_binstrcpy](#) (char \*data, char \*bstring)

- int `xu_dump` (const OSOCTET \*msgptr, ASN1DumpCbFunc cb, void \*cbArg\_p)
- int `xu_fdump` (FILE \*file\_p, const OSOCTET \*msgptr)
- int `xu_dump2` (OSCTXT \*pctx, const OSOCTET \*msgptr)
- void `xu_fmt_tag` (ASN1TAG \*tag\_p, char \*class\_p, char \*form\_p, char \*id\_code)
- char \* `xu_fmt_tag2` (ASN1TAG \*tag\_p, char \*bufp)
- char \* `xu_fmt_contents` (OSCTXT \*pctx, int len, int \*count)
- int `xu_fread` (FILE \*fp, OSOCTET \*bufp, int bufsiz)
- void `xu_SaveBufferState` (OSCTXT \*pctx, OSRTBufSave \*pSavedInfo)
- void `xu_RestoreBufferState` (OSCTXT \*pctx, OSRTBufSave \*pSavedInfo)
- int `xd_MovePastEOC` (OSCTXT \*pctx)
- int `xd_consStrIndefLenAndSize` (OSCTXT \*pctx, ASN1TAG expectedTag, OSSIZE \*length, OSSIZE \*size)

### 8.1.1 Detailed Description

ASN.1 runtime constants, data structure definitions, and functions to support the Basic Encoding Rules (BER) and Distinguished Encoding Rules (DER) as defined in the ITU-T X.690 standard.

## 8.2 asn1BerCppType.h File Reference

```
#include "rtsrc/asn1CppType.h"
#include "rtbersrc/asn1ber.h"
#include "rtxsrc/rtxPrint.h"
```

### Classes

- class [ASN1BERLength](#)
- class [ASN1BERMessageBuffer](#)
- class [ASN1BEREncodeBuffer](#)
- class [ASN1BERDecodeBuffer](#)

### 8.2.1 Detailed Description

BER/DER/CER C++ type and class definitions.

## 8.3 ASN1BERDecodeStream.h File Reference

```
#include "rtbersrc/asn1BerCppType.h"
#include "rtbersrc/asn1berStream.h"
#include "rtxsrc/OSRTInputStreamIF.h"
```

## Classes

- class [ASN1BERDecodeStream](#)

### 8.3.1 Detailed Description

The C++ definitions for ASN.1 BER input streams.

## 8.4 ASN1BEREncodeStream.h File Reference

```
#include "rtsrc/asn1CppTypes.h"
#include "rtxsrc/OSRTOutputStreamIF.h"
#include "rtbersrc/asn1berStream.h"
```

## Classes

- class [ASN1BEREncodeStream](#)

### 8.4.1 Detailed Description

The C++ definitions for ASN.1 BER output streams.

## 8.5 asn1berSocket.h File Reference

```
#include "rtbersrc/asn1ber.h"
#include "rtxsrc/rxSocket.h"
```

## Functions

- int [berReadMsgFromSocket](#) (OSCTXT \*pctxt, OSRTSOCKET socket, OSOCTET \*buffer, int bufsiz, OSOCTET \*\*ppDestBuffer, int \*pMessageSize)

## 8.6 asn1berStream.h File Reference

```
#include "rtbersrc/asn1ber.h"
#include "rtxsrc/rxBuffer.h"
#include "rtxsrc/rxStream.h"
```

## Macros

- #define **cerEncCanonicalSort**
- #define **cerGetBufLocDescr**
- #define **cerAddBufLocDescr**
- #define **BS\_CHKEOB**(pctxt)
- #define **BS\_CHKEND**(pctxt, ccb\_p)

## Functions

- int **berStrmInitContext** (OSCTXT \*pctxt)
- int **berStrmInitContextUsingKey** (OSCTXT \*pctxt, const OSOCTET \*key, size\_t keylen)
- int **berStrmFreeContext** (OSCTXT \*pctxt)
- int **berEncStrmBigInt** (OSCTXT \*pctxt, const char \*pvalue, ASN1TagType tagging)
- int **berEncStrmBigIntNchars** (OSCTXT \*pctxt, const char \*pvalue, size\_t nchars, ASN1TagType tagging)
- int **berEncStrmBitStr** (OSCTXT \*pctxt, const OSOCTET \*object\_p, OSUINT32 numbits, ASN1TagType tagging)
- int **berEncStrmBMPStr** (OSCTXT \*pctxt, const Asn116BitCharString \*object\_p, ASN1TagType tagging)
- int **berEncStrmBool** (OSCTXT \*pctxt, OSBOOL value, ASN1TagType tagging)
- int **berEncStrmCharStr** (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int **berEncStrmDateStr** (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int **berEncStrmDateTimeStr** (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int **berEncStrmDurationStr** (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int **berEncStrmTimeStr** (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int **berEncStrmTimeOfDayStr** (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int **berEncStrmDefLength** (OSCTXT \*pctxt, size\_t length)
- int **berEncStrmEOC** (OSCTXT \*pctxt)
- int **berEncStrmEnum** (OSCTXT \*pctxt, OSINT32 value, ASN1TagType tagging)
- int **berEncStrmInt** (OSCTXT \*pctxt, OSINT32 value, ASN1TagType tagging)
- int **berEncStrmInt8** (OSCTXT \*pctxt, OSINT8 value, ASN1TagType tagging)
- int **berEncStrmInt16** (OSCTXT \*pctxt, OSINT16 value, ASN1TagType tagging)
- int **berEncStrmInt64** (OSCTXT \*pctxt, OSINT64 value, ASN1TagType tagging)
- int **berEncStrmLength** (OSCTXT \*pctxt, int length)
- int **berEncStrmNull** (OSCTXT \*pctxt, ASN1TagType tagging)
- int **berEncStrmObjId** (OSCTXT \*pctxt, const ASN1OBJID \*object\_p, ASN1TagType tagging)
- int **berEncStrmObjId64** (OSCTXT \*pctxt, const ASN1OID64 \*object\_p, ASN1TagType tagging)
- int **berEncStrmOctStr** (OSCTXT \*pctxt, const OSOCTET \*object\_p, OSSIZE numocts, ASN1TagType tagging)
- int **berEncStrmOpenTypeExt** (OSCTXT \*pctxt, OSRTDList \*pElemList)
- int **berEncStrmReal** (OSCTXT \*pctxt, OSREAL value, ASN1TagType tagging)
- int **berEncStrmReal10** (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging)
- int **cerEncStrmReal10** (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging)
- int **berEncStrmRelativeOID** (OSCTXT \*pctxt, const ASN1OBJID \*object\_p, ASN1TagType tagging)
- int **berEncStrmTag** (OSCTXT \*pctxt, ASN1TAG tag)
- int **berEncStrmTagAndLen** (OSCTXT \*pctxt, ASN1TAG tag, int length)
- int **berEncStrmTagAndDefLen** (OSCTXT \*pctxt, ASN1TAG tag, OSSIZE length)
- int **berEncStrmTagAndIndefLen** (OSCTXT \*pctxt, ASN1TAG tag)
- int **berEncStrmUInt** (OSCTXT \*pctxt, OSUINT32 value, ASN1TagType tagging)
- int **berEncStrmUInt8** (OSCTXT \*pctxt, OSUINT8 value, ASN1TagType tagging)
- int **berEncStrmUInt16** (OSCTXT \*pctxt, OSUINT16 value, ASN1TagType tagging)
- int **berEncStrmUInt64** (OSCTXT \*pctxt, OSUINT64 value, ASN1TagType tagging)
- int **berEncStrmUnivStr** (OSCTXT \*pctxt, const Asn132BitCharString \*object\_p, ASN1TagType tagging)



- int [berEncStrmXSDAny](#) (OSCTXT \*pctxt, OSXSDAny \*pvalue, ASN1TagType tagging)
- int [berEncStrmWriteOctet](#) (OSCTXT \*pctxt, OSOCTET octet)
- int [berEncStrmWriteOctets](#) (OSCTXT \*pctxt, const OSOCTET \*poctets, size\_t numocts)
- int [cerEncStrmBMPStr](#) (OSCTXT \*pctxt, const Asn116BitCharString \*object\_p, ASN1TagType tagging)
- int [cerEncStrmBitStr](#) (OSCTXT \*pctxt, const OSOCTET \*object\_p, OSUINT32 numbits, ASN1TagType tagging)
- int [cerEncStrmCharStr](#) (OSCTXT \*pctxt, const char \*object\_p, ASN1TagType tagging, ASN1TAG tag)
- int [cerEncStrmOctStr](#) (OSCTXT \*pctxt, const OSOCTET \*object\_p, OSUINT32 numocts, ASN1TagType tagging)
- int [cerEncStrmUnivStr](#) (OSCTXT \*pctxt, const Asn132BitCharString \*object\_p, ASN1TagType tagging)
- int [berDecStrmBMPStr](#) (OSCTXT \*pctxt, Asn116BitCharString \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmBigInt](#) (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmBigEnum](#) (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmBitStr](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnbits, ASN1TagType tagging, int length)
- int [berDecStrmBool](#) (OSCTXT \*pctxt, OSBOOL \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmCharStr](#) (OSCTXT \*pctxt, const char \*\*ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int [berDecStrmDateStr](#) (OSCTXT \*pctxt, const char \*\*ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int [berDecStrmDateTimeStr](#) (OSCTXT \*pctxt, const char \*\*ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int [berDecStrmDurationStr](#) (OSCTXT \*pctxt, const char \*\*ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int [berDecStrmTimeStr](#) (OSCTXT \*pctxt, const char \*\*ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int [berDecStrmTimeOfDayStr](#) (OSCTXT \*pctxt, const char \*\*ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- OSBOOL [berDecStrmCheckEnd](#) (OSCTXT \*pctxt, ASN1CCB \*pccb)
- int [berDecStrmDynBitStr](#) (OSCTXT \*pctxt, const OSOCTET \*\*ppvalue, OSUINT32 \*pnbits, ASN1TagType tagging, int length)
- int [berDecStrmDynBitStr64](#) (OSCTXT \*pctxt, const OSOCTET \*\*ppvalue, OSSIZE \*pnbits, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int [berDecStrmDynOctStr](#) (OSCTXT \*pctxt, const OSOCTET \*\*ppvalue, OSUINT32 \*pnocets, ASN1TagType tagging, int length)
- int [berDecStrmDynOctStr64](#) (OSCTXT \*pctxt, OSOCTET \*\*ppvalue, OSSIZE \*pnocets, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int [berDecStrmEnum](#) (OSCTXT \*pctxt, OSINT32 \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmFindTag](#) (OSCTXT \*pctxt, ASN1TAG tag, int \*len\_p, OSBOOL advance)
- int [berDecStrmFindTag2](#) (OSCTXT \*pctxt, ASN1TAG tag, OSSIZE \*len\_p, OSBOOL \*plndefLen, OSBOOL advance)
- int [berDecStrmGetTLVLength](#) (OSCTXT \*pctxt)
- int [berDecStrmInt](#) (OSCTXT \*pctxt, OSINT32 \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmInt8](#) (OSCTXT \*pctxt, OSINT8 \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmInt16](#) (OSCTXT \*pctxt, OSINT16 \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmInt64](#) (OSCTXT \*pctxt, OSINT64 \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmLength](#) (OSCTXT \*pctxt, int \*len\_p)
- int [berDecStrmLength2](#) (OSCTXT \*pctxt, OSSIZE \*pLength, OSBOOL \*plndefLen)
- int [berDecStrmMatchEOC](#) (OSCTXT \*pctxt)
- int [berDecStrmMatchTag](#) (OSCTXT \*pctxt, ASN1TAG tag, int \*len\_p, OSBOOL advance)
- int [berDecStrmMatchTag2](#) (OSCTXT \*pctxt, ASN1TAG tag, OSSIZE \*len\_p, OSBOOL \*plndefLen, OSBOOL advance)
- int [berDecStrmNextElement](#) (OSCTXT \*pctxt)
- int [berDecStrmNull](#) (OSCTXT \*pctxt, ASN1TagType tagging)
- int [berDecStrmObjId](#) (OSCTXT \*pctxt, ASN1OBJID \*object\_p, ASN1TagType tagging, int length)
- int [berDecStrmObjId64](#) (OSCTXT \*pctxt, ASN1OID64 \*object\_p, ASN1TagType tagging, int length)



- int `berDecStrmOctStr` (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, ASN1TagType tagging, int length)
- int `berDecStrmOpenType` (OSCTXT \*pctxt, const OSOCTET \*\*object\_p2, OSSIZE \*pnumocts)
- int `berDecStrmOpenTypeAppend` (OSCTXT \*pctxt, OSRTDList \*pElemList)
- int `berDecStrmOpenTypeExt` (OSCTXT \*pctxt, ASN1CCB \*ccb\_p, ASN1TAG \*tags, int tagCount, OSRTDList \*pElemList)
- int `berDecStrmPeekTagAndLen` (OSCTXT \*pctxt, ASN1TAG \*ptag, int \*plen)
- int `berDecStrmReadDynTLV` (OSCTXT \*pctxt, OSOCTET \*\*ppbuf)
- int `berDecStrmReadTLV` (OSCTXT \*pctxt, OSOCTET \*buf, OSSIZE bufsiz)
- int `berDecStrmReal` (OSCTXT \*pctxt, OSREAL \*object\_p, ASN1TagType tagging, int length)
- int `berDecStrmReal10` (OSCTXT \*pctxt, const char \*\*object\_p, ASN1TagType tagging, int length)
- int `berDecStrmRelativeOID` (OSCTXT \*pctxt, ASN1OBJID \*object\_p, ASN1TagType tagging, int length)
- int `berDecStrmTag` (OSCTXT \*pctxt, ASN1TAG \*tag\_p)
- int `berDecStrmTagAndLen` (OSCTXT \*pctxt, ASN1TAG \*tag\_p, int \*len\_p)
- int `berDecStrmTagAndLen2` (OSCTXT \*pctxt, ASN1TAG \*tag\_p, OSSIZE \*len\_p, OSBOOL \*pIndefLen)
- OSBOOL `berDecStrmTestEOC` (OSCTXT \*pctxt, ASN1CCB \*ccb\_p)
- OSBOOL `berDecStrmTestEOC2` (OSCTXT \*pctxt)
- int `berDecStrmTestTag` (OSCTXT \*pctxt, ASN1TAG tag, int \*len\_p, OSBOOL advance)
- int `berDecStrmUInt` (OSCTXT \*pctxt, OSUINT32 \*object\_p, ASN1TagType tagging, int length)
- int `berDecStrmUInt8` (OSCTXT \*pctxt, OSUINT8 \*object\_p, ASN1TagType tagging, int length)
- int `berDecStrmUInt16` (OSCTXT \*pctxt, OSUINT16 \*object\_p, ASN1TagType tagging, int length)
- int `berDecStrmUInt64` (OSCTXT \*pctxt, OSUINT64 \*object\_p, ASN1TagType tagging, int length)
- int `berDecStrmUnivStr` (OSCTXT \*pctxt, Asn132BitCharString \*object\_p, ASN1TagType tagging, int length)

### 8.6.1 Detailed Description

ASN.1 runtime constants, data structure definitions, and functions to support the streaming encoding/decoding of Basic Encoding Rules (BER) as defined in the ITU-T X.690 standard.



# Index

- ASN1BERDecodeBuffer, 167
  - ASN1BERDecodeBuffer, 168
  - findElement, 169
  - getMsgPtr, 170
  - init, 170
  - isA, 170
  - operator>>, 170
  - parseTagLen, 171, 172
  - readBinaryFile, 173
  - setBuffer, 173
- ASN1BERDecodeStream, 174
  - ASN1BERDecodeStream, 176
  - byteIndex, 176
  - chkend, 176
  - close, 177
  - currentPos, 177
  - decodeBMPStr, 179
  - decodeBigInt, 177
  - decodeBitStr, 178, 179
  - decodeBool, 180
  - decodeCharStr, 181
  - decodeEnum, 181
  - decodeEoc, 182
  - decodeInt, 182
  - decodeInt16, 183
  - decodeInt64, 184
  - decodeInt8, 184
  - decodeLength, 186
  - decodeNull, 186
  - decodeObj, 187
  - decodeObjId, 187
  - decodeObjId64, 188
  - decodeOctStr, 189, 190
  - decodeOpenType, 191
  - decodeReal, 191
  - decodeRelativeOID, 192
  - decodeTag, 193
  - decodeTagAndLen, 193, 194
  - decodeUInt, 194
  - decodeUInt16, 195
  - decodeUInt64, 196
  - decodeUInt8, 196
  - decodeUnivStr, 197
  - flush, 198
  - getAppInfo, 198
  - getContext, 198
  - getCtxtPtr, 198
  - getErrorInfo, 199
  - getPosition, 199
  - getStatus, 200
  - getTLVLength, 200
  - isOpened, 201
  - isA, 200
  - mark, 201
  - markSupported, 202
  - operator>>, 202
  - peekTagAndLen, 203
  - printErrorInfo, 203
  - read, 203
  - readBlocking, 204
  - readTLV, 204
  - reset, 205
  - resetErrorInfo, 205
  - setAppInfo, 205
  - setDiag, 205
  - setPosition, 206
  - skip, 206
- ASN1BERDecodeStream.h, 247
- ASN1BEREncodeBuffer, 207
  - ASN1BEREncodeBuffer, 208
  - encodeBigInt, 209
  - encodeBigIntNchars, 209
  - encodeBool, 210
  - encodeObjId, 210
  - freeBuffer, 211
  - getMsgCopy, 211
  - getMsgLen, 211
  - getMsgPtr, 212
  - init, 212
  - isA, 212
  - operator<<, 213
  - setBuffer, 213
- ASN1BEREncodeStream, 214
  - ASN1BEREncodeStream, 215
  - close, 215
  - encodeBMPStr, 218
  - encodeBigInt, 216
  - encodeBigIntNchars, 216
  - encodeBitStr, 217, 218
  - encodeBool, 219

[encodeCharStr, 219](#)  
[encodeEnum, 220](#)  
[encodeEoc, 221](#)  
[encodeIndefLen, 221](#)  
[encodeInt, 221](#)  
[encodeInt16, 222](#)  
[encodeInt64, 222](#)  
[encodeInt8, 223](#)  
[encodeLen, 224](#)  
[encodeNull, 224](#)  
[encodeObj, 225](#)  
[encodeObjId, 225](#)  
[encodeObjId64, 226](#)  
[encodeOctStr, 226, 227](#)  
[encodeReal, 227](#)  
[encodeRelativeOID, 228](#)  
[encodeTag, 229](#)  
[encodeTagAndIndefLen, 229](#)  
[encodeTagAndLen, 230](#)  
[encodeUInt, 230](#)  
[encodeUInt16, 231](#)  
[encodeUInt64, 232](#)  
[encodeUInt8, 232](#)  
[encodeUnivStr, 233](#)  
[flush, 233](#)  
[getAppInfo, 234](#)  
[getContext, 234](#)  
[getCtxtPtr, 234](#)  
[getErrorInfo, 234, 235](#)  
[getStatus, 235](#)  
[isOpened, 236](#)  
[isA, 235](#)  
[operator<<, 236](#)  
[printErrorInfo, 237](#)  
[resetErrorInfo, 237](#)  
[setAppInfo, 237](#)  
[setDiag, 237](#)  
[write, 237](#)  
[ASN1BEREncodeStream.h, 248](#)  
[ASN1BERLength, 238](#)  
[ASN1BERMessageBuffer, 239](#)  
    [ASN1BERMessageBuffer, 239, 240](#)  
    [binDump, 240](#)  
    [calcIndefLen, 240, 241](#)  
    [hexDump, 242](#)  
[asn1BerCppType.h, 247](#)  
[asn1ber.h, 243](#)  
[asn1berSocket.h, 248](#)  
[asn1berStream.h, 248](#)  
  
[BER Message Buffer Classes, 12](#)  
[BER Runtime Library Functions., 13](#)  
[BER/DER C Decode Functions., 14](#)  
    [berDecCharArray, 16](#)  
  
[xd\\_16BitCharStr, 17](#)  
[xd\\_16BitCharStr64, 18](#)  
[xd\\_32BitCharStr, 18](#)  
[xd\\_32BitCharStr64, 19](#)  
[xd\\_NextElement, 39](#)  
[xd\\_OpenType, 44](#)  
[xd\\_OpenTypeAppend, 45](#)  
[xd\\_OpenTypeExt, 45](#)  
[xd\\_OpenTypeExt64, 46](#)  
[xd\\_Tag1AndLen, 50](#)  
[xd\\_bigint, 20](#)  
[xd\\_bitstr, 20](#)  
[xd\\_bitstr64, 21](#)  
[xd\\_bitstr64\\_s, 22](#)  
[xd\\_bitstr64Ext\\_s, 22](#)  
[xd\\_bitstr\\_s, 23](#)  
[xd\\_bitstrExt\\_s, 24](#)  
[xd\\_boolean, 25](#)  
[xd\\_charstr, 25](#)  
[xd\\_charstr64, 26](#)  
[xd\\_chkend, 27](#)  
[xd\\_chkend64, 27](#)  
[xd\\_count, 28](#)  
[xd\\_count64, 28](#)  
[xd\\_datestr, 29](#)  
[xd\\_datetimestr, 30](#)  
[xd\\_durationstr, 30](#)  
[xd\\_enum, 31](#)  
[xd\\_enumUnsigned, 31](#)  
[xd\\_indeflen64, 32](#)  
[xd\\_indeflen\\_ex, 33](#)  
[xd\\_int16, 33](#)  
[xd\\_int64, 34](#)  
[xd\\_int8, 34](#)  
[xd\\_integer, 35](#)  
[xd\\_len, 36](#)  
[xd\\_len64, 36](#)  
[xd\\_match, 37](#)  
[xd\\_match1, 38](#)  
[xd\\_match64, 38](#)  
[xd\\_memcpy, 39](#)  
[xd\\_null, 40](#)  
[xd\\_objid, 40](#)  
[xd\\_octstr, 41](#)  
[xd\\_octstr64, 42](#)  
[xd\\_octstr64\\_s, 42](#)  
[xd\\_octstr\\_s, 43](#)  
[xd\\_oid64, 43](#)  
[xd\\_real, 46](#)  
[xd\\_real10, 47](#)  
[xd\\_reloid, 48](#)  
[xd\\_setp, 48](#)  
[xd\\_setp64, 49](#)  
[xd\\_tag, 50](#)

- xd\_tag\_len, 51
- xd\_tag\_len\_64, 52
- xd\_timeofdaystr, 52
- xd\_timestr, 53
- xd\_uint16, 53
- xd\_uint64, 54
- xd\_uint8, 55
- xd\_unsigned, 55
- xd\_utf8str, 16
- BER/DER C Encode Functions., 62
  - derEncBitString, 64
  - xe\_16BitCharStr, 64
  - xe\_32BitCharStr, 65
  - xe\_OpenType, 81
  - xe\_OpenTypeExt, 82
  - xe\_OpenTypeExtDer, 82
  - xe\_TagAndIndefLen, 85
  - xe\_bigint, 66
  - xe\_bitstr, 66
  - xe\_bitstrExt, 67
  - xe\_boolean, 67
  - xe\_charstr, 68
  - xe\_datestr, 69
  - xe\_datetimestr, 69
  - xe\_derCanSortSet, 70
  - xe\_derCanonicalSort, 70
  - xe\_derReal10, 71
  - xe\_durationstr, 72
  - xe\_enum, 72
  - xe\_enumUnsigned, 73
  - xe\_expandBuffer, 73
  - xe\_free, 74
  - xe\_getBufLocDescr, 74
  - xe\_getp, 75
  - xe\_identifier, 75
  - xe\_int16, 76
  - xe\_int64, 76
  - xe\_int8, 77
  - xe\_integer, 77
  - xe\_len, 78
  - xe\_len64, 78
  - xe\_memcpy, 79
  - xe\_null, 79
  - xe\_objid, 80
  - xe\_octstr, 80
  - xe\_oid64, 81
  - xe\_real, 82
  - xe\_real10, 83
  - xe\_reloid, 83
  - xe\_setp, 84
  - xe\_tag, 84
  - xe\_tag\_len, 85
  - xe\_timeofdaystr, 86
  - xe\_timestr, 87
  - xe\_uint16, 87
  - xe\_uint64, 88
  - xe\_uint8, 88
  - xe\_unsigned, 89
  - xe\_utf8str, 63
- BER/DER C File Functions., 57
  - xdf\_ReadContents, 58
  - xdf\_ReadPastEOC, 58
  - xdf\_TagAndLen, 60
  - xdf\_len, 57
  - xdf\_tag, 60
- BER/DER/CER C++ Run-Time Classes., 11
- BER/PER C Utility Functions, 90
  - berDefToIndefLen, 91
  - berErrAddTagParm, 92
  - berErrUnexpTag, 92
  - berGetLibInfo, 93
  - berGetLibVersion, 93
  - berIndefToDefLen, 93
  - berParseTagLen, 93
  - berReadMsgFromSocket, 94
  - berTagToDynStr, 95
  - berTagToString, 95
  - berValidateIso8601DateStr, 96
  - berValidateIso8601DurationStr, 96
  - berValidateIso8601TimeStr, 96
  - xu\_RestoreBufferState, 99
  - xu\_SaveBufferState, 100
  - xu\_alloc\_array, 97
  - xu\_dump, 97
  - xu\_dump2, 98
  - xu\_fdump, 99
  - xu\_hex\_dump, 91
- BS\_CHKEND
  - Streaming BER Runtime Library Functions., 101
- BS\_CHKEOB
  - Streaming BER Runtime Library Functions., 101
- berDecCharArray
  - BER/DER C Decode Functions., 16
- berDecStrmBMPStr
  - C Streaming BER Decode Functions., 133
- berDecStrmBigInt
  - C Streaming BER Decode Functions., 131
- berDecStrmBitStr
  - C Streaming BER Decode Functions., 132
- berDecStrmBool
  - C Streaming BER Decode Functions., 133
- berDecStrmCharStr
  - C Streaming BER Decode Functions., 134
- berDecStrmCheckEnd
  - C Streaming BER Decode Functions., 135
- berDecStrmDateStr
  - C Streaming BER Decode Functions., 135
- berDecStrmDateTimeStr

C Streaming BER Decode Functions., 136  
 berDecStrmDurationStr  
   C Streaming BER Decode Functions., 136  
 berDecStrmDynBitStr  
   C Streaming BER Decode Functions., 137  
 berDecStrmDynBitStr64  
   C Streaming BER Decode Functions., 138  
 berDecStrmDynOctStr  
   C Streaming BER Decode Functions., 138  
 berDecStrmDynOctStr64  
   C Streaming BER Decode Functions., 139  
 berDecStrmEnum  
   C Streaming BER Decode Functions., 140  
 berDecStrmFindTag  
   C Streaming BER Decode Functions., 140  
 berDecStrmFindTag2  
   C Streaming BER Decode Functions., 141  
 berDecStrmGetTLVLength  
   C Streaming BER Decode Functions., 142  
 berDecStrmInt  
   C Streaming BER Decode Functions., 142  
 berDecStrmInt16  
   C Streaming BER Decode Functions., 143  
 berDecStrmInt64  
   C Streaming BER Decode Functions., 143  
 berDecStrmInt8  
   C Streaming BER Decode Functions., 144  
 berDecStrmLength  
   C Streaming BER Decode Functions., 145  
 berDecStrmLength2  
   C Streaming BER Decode Functions., 145  
 berDecStrmMatchEOC  
   C Streaming BER Decode Functions., 146  
 berDecStrmMatchTag  
   C Streaming BER Decode Functions., 146  
 berDecStrmMatchTag2  
   C Streaming BER Decode Functions., 147  
 berDecStrmNextElement  
   C Streaming BER Decode Functions., 147  
 berDecStrmNull  
   C Streaming BER Decode Functions., 148  
 berDecStrmObjId  
   C Streaming BER Decode Functions., 148  
 berDecStrmObjId64  
   C Streaming BER Decode Functions., 149  
 berDecStrmOctStr  
   C Streaming BER Decode Functions., 149  
 berDecStrmOpenType  
   C Streaming BER Decode Functions., 150  
 berDecStrmOpenTypeAppend  
   C Streaming BER Decode Functions., 151  
 berDecStrmOpenTypeExt  
   C Streaming BER Decode Functions., 151  
 berDecStrmPeekTagAndLen  
   C Streaming BER Decode Functions., 152  
 berDecStrmReadDynTLV  
   C Streaming BER Decode Functions., 152  
 berDecStrmReadTLV  
   C Streaming BER Decode Functions., 153  
 berDecStrmReal  
   C Streaming BER Decode Functions., 153  
 berDecStrmReal10  
   C Streaming BER Decode Functions., 154  
 berDecStrmRelativeOID  
   C Streaming BER Decode Functions., 154  
 berDecStrmTag  
   C Streaming BER Decode Functions., 155  
 berDecStrmTagAndLen  
   C Streaming BER Decode Functions., 155  
 berDecStrmTagAndLen2  
   C Streaming BER Decode Functions., 156  
 berDecStrmTestEOC  
   C Streaming BER Decode Functions., 157  
 berDecStrmTestTag  
   C Streaming BER Decode Functions., 157  
 berDecStrmTimeOfDayStr  
   C Streaming BER Decode Functions., 158  
 berDecStrmTimeStr  
   C Streaming BER Decode Functions., 158  
 berDecStrmUInt  
   C Streaming BER Decode Functions., 160  
 berDecStrmUInt16  
   C Streaming BER Decode Functions., 160  
 berDecStrmUInt64  
   C Streaming BER Decode Functions., 161  
 berDecStrmUInt8  
   C Streaming BER Decode Functions., 162  
 berDecStrmUnivStr  
   C Streaming BER Decode Functions., 162  
 berDefToIndefLen  
   BER/PER C Utility Functions, 91  
 berEncStrmBMPStr  
   C Streaming BER Encode Functions., 106  
 berEncStrmBigInt  
   C Streaming BER Encode Functions., 105  
 berEncStrmBigIntNchars  
   C Streaming BER Encode Functions., 105  
 berEncStrmBitStr  
   C Streaming BER Encode Functions., 106  
 berEncStrmBool  
   C Streaming BER Encode Functions., 107  
 berEncStrmCharStr  
   C Streaming BER Encode Functions., 108  
 berEncStrmDateStr  
   C Streaming BER Encode Functions., 108  
 berEncStrmDateTimeStr  
   C Streaming BER Encode Functions., 109  
 berEncStrmDefLength  
   C Streaming BER Encode Functions., 109

- C Streaming BER Encode Functions., 109
- berEncStrmDurationStr
  - C Streaming BER Encode Functions., 110
- berEncStrmEOC
  - C Streaming BER Encode Functions., 111
- berEncStrmEnum
  - C Streaming BER Encode Functions., 110
- berEncStrmInt
  - C Streaming BER Encode Functions., 111
- berEncStrmInt16
  - C Streaming BER Encode Functions., 112
- berEncStrmInt64
  - C Streaming BER Encode Functions., 112
- berEncStrmInt8
  - C Streaming BER Encode Functions., 113
- berEncStrmLength
  - C Streaming BER Encode Functions., 114
- berEncStrmNull
  - C Streaming BER Encode Functions., 114
- berEncStrmObjId
  - C Streaming BER Encode Functions., 114
- berEncStrmObjId64
  - C Streaming BER Encode Functions., 115
- berEncStrmOctStr
  - C Streaming BER Encode Functions., 115
- berEncStrmOpenTypeExt
  - C Streaming BER Encode Functions., 116
- berEncStrmReal
  - C Streaming BER Encode Functions., 116
- berEncStrmReal10
  - C Streaming BER Encode Functions., 117
- berEncStrmRelativeOID
  - C Streaming BER Encode Functions., 118
- berEncStrmTag
  - C Streaming BER Encode Functions., 118
- berEncStrmTagAndDefLen
  - C Streaming BER Encode Functions., 119
- berEncStrmTagAndIndefLen
  - C Streaming BER Encode Functions., 119
- berEncStrmTagAndLen
  - C Streaming BER Encode Functions., 120
- berEncStrmTimeOfDayStr
  - C Streaming BER Encode Functions., 120
- berEncStrmTimeStr
  - C Streaming BER Encode Functions., 121
- berEncStrmUInt
  - C Streaming BER Encode Functions., 121
- berEncStrmUInt16
  - C Streaming BER Encode Functions., 122
- berEncStrmUInt64
  - C Streaming BER Encode Functions., 122
- berEncStrmUInt8
  - C Streaming BER Encode Functions., 123
- berEncStrmUnivStr
  - C Streaming BER Encode Functions., 124
- berEncStrmWriteOctet
  - C Streaming BER Encode Functions., 124
- berEncStrmWriteOctets
  - C Streaming BER Encode Functions., 125
- berEncStrmXSDAny
  - C Streaming BER Encode Functions., 125
- berErrAddTagParm
  - BER/PER C Utility Functions, 92
- berErrUnexpTag
  - BER/PER C Utility Functions, 92
- berGetLibInfo
  - BER/PER C Utility Functions, 93
- berGetLibVersion
  - BER/PER C Utility Functions, 93
- berIndefToDefLen
  - BER/PER C Utility Functions, 93
- berParseTagLen
  - BER/PER C Utility Functions, 93
- berReadMsgFromSocket
  - BER/PER C Utility Functions, 94
- berStrmFreeContext
  - Streaming BER Runtime Library Functions., 102
- berStrmInitContext
  - Streaming BER Runtime Library Functions., 103
- berTagToDynStr
  - BER/PER C Utility Functions, 95
- berTagToString
  - BER/PER C Utility Functions, 95
- berValidateIso8601DateStr
  - BER/PER C Utility Functions, 96
- berValidateIso8601DurationStr
  - BER/PER C Utility Functions, 96
- berValidateIso8601TimeStr
  - BER/PER C Utility Functions, 96
- binDump
  - ASN1BERMessageBuffer, 240
- byteIndex
  - ASN1BERDecodeStream, 176
- C Streaming BER Decode Functions., 130
  - berDecStrmBMPStr, 133
  - berDecStrmBigInt, 131
  - berDecStrmBitStr, 132
  - berDecStrmBool, 133
  - berDecStrmCharStr, 134
  - berDecStrmCheckEnd, 135
  - berDecStrmDateStr, 135
  - berDecStrmDateTimeStr, 136
  - berDecStrmDurationStr, 136
  - berDecStrmDynBitStr, 137
  - berDecStrmDynBitStr64, 138
  - berDecStrmDynOctStr, 138
  - berDecStrmDynOctStr64, 139

- berDecStrmEnum, 140
- berDecStrmFindTag, 140
- berDecStrmFindTag2, 141
- berDecStrmGetTLVLength, 142
- berDecStrmInt, 142
- berDecStrmInt16, 143
- berDecStrmInt64, 143
- berDecStrmInt8, 144
- berDecStrmLength, 145
- berDecStrmLength2, 145
- berDecStrmMatchEOC, 146
- berDecStrmMatchTag, 146
- berDecStrmMatchTag2, 147
- berDecStrmNextElement, 147
- berDecStrmNull, 148
- berDecStrmObjId, 148
- berDecStrmObjId64, 149
- berDecStrmOctStr, 149
- berDecStrmOpenType, 150
- berDecStrmOpenTypeAppend, 151
- berDecStrmOpenTypeExt, 151
- berDecStrmPeekTagAndLen, 152
- berDecStrmReadDynTLV, 152
- berDecStrmReadTLV, 153
- berDecStrmReal, 153
- berDecStrmReal10, 154
- berDecStrmRelativeOID, 154
- berDecStrmTag, 155
- berDecStrmTagAndLen, 155
- berDecStrmTagAndLen2, 156
- berDecStrmTestEOC, 157
- berDecStrmTestTag, 157
- berDecStrmTimeOfDayStr, 158
- berDecStrmTimeStr, 158
- berDecStrmUInt, 160
- berDecStrmUInt16, 160
- berDecStrmUInt64, 161
- berDecStrmUInt8, 162
- berDecStrmUnivStr, 162
- C Streaming BER Encode Functions., 104
  - berEncStrmBMPStr, 106
  - berEncStrmBigInt, 105
  - berEncStrmBigIntNchars, 105
  - berEncStrmBitStr, 106
  - berEncStrmBool, 107
  - berEncStrmCharStr, 108
  - berEncStrmDateStr, 108
  - berEncStrmDateTimeStr, 109
  - berEncStrmDefLength, 109
  - berEncStrmDurationStr, 110
  - berEncStrmEOC, 111
  - berEncStrmEnum, 110
  - berEncStrmInt, 111
  - berEncStrmInt16, 112
  - berEncStrmInt64, 112
  - berEncStrmInt8, 113
  - berEncStrmLength, 114
  - berEncStrmNull, 114
  - berEncStrmObjId, 114
  - berEncStrmObjId64, 115
  - berEncStrmOctStr, 115
  - berEncStrmOpenTypeExt, 116
  - berEncStrmReal, 116
  - berEncStrmReal10, 117
  - berEncStrmRelativeOID, 118
  - berEncStrmTag, 118
  - berEncStrmTagAndDefLen, 119
  - berEncStrmTagAndIndefLen, 119
  - berEncStrmTagAndLen, 120
  - berEncStrmTimeOfDayStr, 120
  - berEncStrmTimeStr, 121
  - berEncStrmUInt, 121
  - berEncStrmUInt16, 122
  - berEncStrmUInt64, 122
  - berEncStrmUInt8, 123
  - berEncStrmUnivStr, 124
  - berEncStrmWriteOctet, 124
  - berEncStrmWriteOctets, 125
  - berEncStrmXSDAny, 125
  - cerEncStrmBMPStr, 126
  - cerEncStrmBitStr, 126
  - cerEncStrmCharStr, 127
  - cerEncStrmOctStr, 127
  - cerEncStrmReal10, 128
  - cerEncStrmUnivStr, 128
  - C++ classes for streaming BER decoding., 165
  - C++ classes for streaming BER encoding., 164
  - calcIndefLen
    - ASN1BERMessageBuffer, 240, 241
  - cerAddBufLocDescr
    - Streaming BER Runtime Library Functions., 102
  - cerEncCanonicalSort
    - Streaming BER Runtime Library Functions., 102
  - cerEncStrmBMPStr
    - C Streaming BER Encode Functions., 126
  - cerEncStrmBitStr
    - C Streaming BER Encode Functions., 126
  - cerEncStrmCharStr
    - C Streaming BER Encode Functions., 127
  - cerEncStrmOctStr
    - C Streaming BER Encode Functions., 127
  - cerEncStrmReal10
    - C Streaming BER Encode Functions., 128
  - cerEncStrmUnivStr
    - C Streaming BER Encode Functions., 128
  - cerGetBufLocDescr
    - Streaming BER Runtime Library Functions., 102
  - chkend



- ASN1BERDecodeStream, [176](#)
- close
  - ASN1BERDecodeStream, [177](#)
  - ASN1BEREncodeStream, [215](#)
- currentPos
  - ASN1BERDecodeStream, [177](#)
- decodeBMPStr
  - ASN1BERDecodeStream, [179](#)
- decodeBigInt
  - ASN1BERDecodeStream, [177](#)
- decodeBitStr
  - ASN1BERDecodeStream, [178](#), [179](#)
- decodeBool
  - ASN1BERDecodeStream, [180](#)
- decodeCharStr
  - ASN1BERDecodeStream, [181](#)
- decodeEnum
  - ASN1BERDecodeStream, [181](#)
- decodeEoc
  - ASN1BERDecodeStream, [182](#)
- decodeInt
  - ASN1BERDecodeStream, [182](#)
- decodeInt16
  - ASN1BERDecodeStream, [183](#)
- decodeInt64
  - ASN1BERDecodeStream, [184](#)
- decodeInt8
  - ASN1BERDecodeStream, [184](#)
- decodeLength
  - ASN1BERDecodeStream, [186](#)
- decodeNull
  - ASN1BERDecodeStream, [186](#)
- decodeObj
  - ASN1BERDecodeStream, [187](#)
- decodeObjId
  - ASN1BERDecodeStream, [187](#)
- decodeObjId64
  - ASN1BERDecodeStream, [188](#)
- decodeOctStr
  - ASN1BERDecodeStream, [189](#), [190](#)
- decodeOpenType
  - ASN1BERDecodeStream, [191](#)
- decodeReal
  - ASN1BERDecodeStream, [191](#)
- decodeRelativeOID
  - ASN1BERDecodeStream, [192](#)
- decodeTag
  - ASN1BERDecodeStream, [193](#)
- decodeTagAndLen
  - ASN1BERDecodeStream, [193](#), [194](#)
- decodeUInt
  - ASN1BERDecodeStream, [194](#)
- decodeUInt16

- ASN1BERDecodeStream, [195](#)
- decodeUInt64
  - ASN1BERDecodeStream, [196](#)
- decodeUInt8
  - ASN1BERDecodeStream, [196](#)
- decodeUnivStr
  - ASN1BERDecodeStream, [197](#)
- derEncBitString
  - BER/DER C Encode Functions., [64](#)
- encodeBMPStr
  - ASN1BEREncodeStream, [218](#)
- encodeBigInt
  - ASN1BEREncodeBuffer, [209](#)
  - ASN1BEREncodeStream, [216](#)
- encodeBigIntNchars
  - ASN1BEREncodeBuffer, [209](#)
  - ASN1BEREncodeStream, [216](#)
- encodeBitStr
  - ASN1BEREncodeStream, [217](#), [218](#)
- encodeBool
  - ASN1BEREncodeBuffer, [210](#)
  - ASN1BEREncodeStream, [219](#)
- encodeCharStr
  - ASN1BEREncodeStream, [219](#)
- encodeEnum
  - ASN1BEREncodeStream, [220](#)
- encodeEoc
  - ASN1BEREncodeStream, [221](#)
- encodeIndefLen
  - ASN1BEREncodeStream, [221](#)
- encodeInt
  - ASN1BEREncodeStream, [221](#)
- encodeInt16
  - ASN1BEREncodeStream, [222](#)
- encodeInt64
  - ASN1BEREncodeStream, [222](#)
- encodeInt8
  - ASN1BEREncodeStream, [223](#)
- encodeLen
  - ASN1BEREncodeStream, [224](#)
- encodeNull
  - ASN1BEREncodeStream, [224](#)
- encodeObj
  - ASN1BEREncodeStream, [225](#)
- encodeObjId
  - ASN1BEREncodeBuffer, [210](#)
  - ASN1BEREncodeStream, [225](#)
- encodeObjId64
  - ASN1BEREncodeStream, [226](#)
- encodeOctStr
  - ASN1BEREncodeStream, [226](#), [227](#)
- encodeReal
  - ASN1BEREncodeStream, [227](#)

- encodeRelativeOID
  - ASN1BEREncodeStream, [228](#)
- encodeTag
  - ASN1BEREncodeStream, [229](#)
- encodeTagAndIndefLen
  - ASN1BEREncodeStream, [229](#)
- encodeTagAndLen
  - ASN1BEREncodeStream, [230](#)
- encodeUInt
  - ASN1BEREncodeStream, [230](#)
- encodeUInt16
  - ASN1BEREncodeStream, [231](#)
- encodeUInt64
  - ASN1BEREncodeStream, [232](#)
- encodeUInt8
  - ASN1BEREncodeStream, [232](#)
- encodeUnivStr
  - ASN1BEREncodeStream, [233](#)
  
- findElement
  - ASN1BERDecodeBuffer, [169](#)
- flush
  - ASN1BERDecodeStream, [198](#)
  - ASN1BEREncodeStream, [233](#)
- freeBuffer
  - ASN1BEREncodeBuffer, [211](#)
  
- getAppInfo
  - ASN1BERDecodeStream, [198](#)
  - ASN1BEREncodeStream, [234](#)
- getContext
  - ASN1BERDecodeStream, [198](#)
  - ASN1BEREncodeStream, [234](#)
- getCtxtPtr
  - ASN1BERDecodeStream, [198](#)
  - ASN1BEREncodeStream, [234](#)
- getErrorInfo
  - ASN1BERDecodeStream, [199](#)
  - ASN1BEREncodeStream, [234](#), [235](#)
- getMsgCopy
  - ASN1BEREncodeBuffer, [211](#)
- getMsgLen
  - ASN1BEREncodeBuffer, [211](#)
- getMsgPtr
  - ASN1BERDecodeBuffer, [170](#)
  - ASN1BEREncodeBuffer, [212](#)
- getPosition
  - ASN1BERDecodeStream, [199](#)
- getStatus
  - ASN1BERDecodeStream, [200](#)
  - ASN1BEREncodeStream, [235](#)
- getTLVLength
  - ASN1BERDecodeStream, [200](#)
  
- hexDump
  - ASN1BERMessageBuffer, [242](#)
  
- init
  - ASN1BERDecodeBuffer, [170](#)
  - ASN1BEREncodeBuffer, [212](#)
- isOpen
  - ASN1BERDecodeStream, [201](#)
  - ASN1BEREncodeStream, [236](#)
- isA
  - ASN1BERDecodeBuffer, [170](#)
  - ASN1BERDecodeStream, [200](#)
  - ASN1BEREncodeBuffer, [212](#)
  - ASN1BEREncodeStream, [235](#)
  
- mark
  - ASN1BERDecodeStream, [201](#)
- markSupported
  - ASN1BERDecodeStream, [202](#)
  
- operator<<
  - ASN1BEREncodeBuffer, [213](#)
  - ASN1BEREncodeStream, [236](#)
- operator>>
  - ASN1BERDecodeBuffer, [170](#)
  - ASN1BERDecodeStream, [202](#)
  
- parseTagLen
  - ASN1BERDecodeBuffer, [171](#), [172](#)
- peekTagAndLen
  - ASN1BERDecodeStream, [203](#)
- printErrorInfo
  - ASN1BERDecodeStream, [203](#)
  - ASN1BEREncodeStream, [237](#)
  
- read
  - ASN1BERDecodeStream, [203](#)
- readBinaryFile
  - ASN1BERDecodeBuffer, [173](#)
- readBlocking
  - ASN1BERDecodeStream, [204](#)
- readTLV
  - ASN1BERDecodeStream, [204](#)
- reset
  - ASN1BERDecodeStream, [205](#)
- resetErrorInfo
  - ASN1BERDecodeStream, [205](#)
  - ASN1BEREncodeStream, [237](#)
  
- setAppInfo
  - ASN1BERDecodeStream, [205](#)
  - ASN1BEREncodeStream, [237](#)
- setBuffer
  - ASN1BERDecodeBuffer, [173](#)
  - ASN1BEREncodeBuffer, [213](#)
- setDiag

- ASN1BERDecodeStream, [205](#)
- ASN1BEREncodeStream, [237](#)
- setPosition
  - ASN1BERDecodeStream, [206](#)
- skip
  - ASN1BERDecodeStream, [206](#)
- Streaming BER Runtime Library Functions., [101](#)
  - BS\_CHKEND, [101](#)
  - BS\_CHKEOB, [101](#)
  - berStrmFreeContext, [102](#)
  - berStrmInitContext, [103](#)
  - cerAddBufLocDescr, [102](#)
  - cerEncCanonicalSort, [102](#)
  - cerGetBufLocDescr, [102](#)
- write
  - ASN1BEREncodeStream, [237](#)
- xd\_16BitCharStr
  - BER/DER C Decode Functions., [17](#)
- xd\_16BitCharStr64
  - BER/DER C Decode Functions., [18](#)
- xd\_32BitCharStr
  - BER/DER C Decode Functions., [18](#)
- xd\_32BitCharStr64
  - BER/DER C Decode Functions., [19](#)
- xd\_NextElement
  - BER/DER C Decode Functions., [39](#)
- xd\_OpenType
  - BER/DER C Decode Functions., [44](#)
- xd\_OpenTypeAppend
  - BER/DER C Decode Functions., [45](#)
- xd\_OpenTypeExt
  - BER/DER C Decode Functions., [45](#)
- xd\_OpenTypeExt64
  - BER/DER C Decode Functions., [46](#)
- xd\_Tag1AndLen
  - BER/DER C Decode Functions., [50](#)
- xd\_bigint
  - BER/DER C Decode Functions., [20](#)
- xd\_bitstr
  - BER/DER C Decode Functions., [20](#)
- xd\_bitstr64
  - BER/DER C Decode Functions., [21](#)
- xd\_bitstr64\_s
  - BER/DER C Decode Functions., [22](#)
- xd\_bitstr64Ext\_s
  - BER/DER C Decode Functions., [22](#)
- xd\_bitstr\_s
  - BER/DER C Decode Functions., [23](#)
- xd\_bitstrExt\_s
  - BER/DER C Decode Functions., [24](#)
- xd\_boolean
  - BER/DER C Decode Functions., [25](#)
- xd\_charstr
  - BER/DER C Decode Functions., [25](#)
- xd\_charstr64
  - BER/DER C Decode Functions., [26](#)
- xd\_chkend
  - BER/DER C Decode Functions., [27](#)
- xd\_chkend64
  - BER/DER C Decode Functions., [27](#)
- xd\_count
  - BER/DER C Decode Functions., [28](#)
- xd\_count64
  - BER/DER C Decode Functions., [28](#)
- xd\_datestr
  - BER/DER C Decode Functions., [29](#)
- xd\_datetimestr
  - BER/DER C Decode Functions., [30](#)
- xd\_durationstr
  - BER/DER C Decode Functions., [30](#)
- xd\_enum
  - BER/DER C Decode Functions., [31](#)
- xd\_enumUnsigned
  - BER/DER C Decode Functions., [31](#)
- xd\_indeflen64
  - BER/DER C Decode Functions., [32](#)
- xd\_indeflen\_ex
  - BER/DER C Decode Functions., [33](#)
- xd\_int16
  - BER/DER C Decode Functions., [33](#)
- xd\_int64
  - BER/DER C Decode Functions., [34](#)
- xd\_int8
  - BER/DER C Decode Functions., [34](#)
- xd\_integer
  - BER/DER C Decode Functions., [35](#)
- xd\_len
  - BER/DER C Decode Functions., [36](#)
- xd\_len64
  - BER/DER C Decode Functions., [36](#)
- xd\_match
  - BER/DER C Decode Functions., [37](#)
- xd\_match1
  - BER/DER C Decode Functions., [38](#)
- xd\_match64
  - BER/DER C Decode Functions., [38](#)
- xd\_memcpy
  - BER/DER C Decode Functions., [39](#)
- xd\_null
  - BER/DER C Decode Functions., [40](#)
- xd\_objid
  - BER/DER C Decode Functions., [40](#)
- xd\_octstr
  - BER/DER C Decode Functions., [41](#)
- xd\_octstr64
  - BER/DER C Decode Functions., [42](#)
- xd\_octstr64\_s

BER/DER C Decode Functions.,	42	BER/DER C Encode Functions.,	82
xd_octstr_s		xe_TagAndIndefLen	
BER/DER C Decode Functions.,	43	BER/DER C Encode Functions.,	85
xd_oid64		xe_bigint	
BER/DER C Decode Functions.,	43	BER/DER C Encode Functions.,	66
xd_real		xe_bitstr	
BER/DER C Decode Functions.,	46	BER/DER C Encode Functions.,	66
xd_real10		xe_bitstrExt	
BER/DER C Decode Functions.,	47	BER/DER C Encode Functions.,	67
xd_reloid		xe_boolean	
BER/DER C Decode Functions.,	48	BER/DER C Encode Functions.,	67
xd_setp		xe_charstr	
BER/DER C Decode Functions.,	48	BER/DER C Encode Functions.,	68
xd_setp64		xe_datestr	
BER/DER C Decode Functions.,	49	BER/DER C Encode Functions.,	69
xd_tag		xe_datetimestr	
BER/DER C Decode Functions.,	50	BER/DER C Encode Functions.,	69
xd_tag_len		xe_derCanSortSet	
BER/DER C Decode Functions.,	51	BER/DER C Encode Functions.,	70
xd_tag_len_64		xe_derCanonicalSort	
BER/DER C Decode Functions.,	52	BER/DER C Encode Functions.,	70
xd_timeofdaystr		xe_derReal10	
BER/DER C Decode Functions.,	52	BER/DER C Encode Functions.,	71
xd_timestr		xe_durationstr	
BER/DER C Decode Functions.,	53	BER/DER C Encode Functions.,	72
xd_uint16		xe_enum	
BER/DER C Decode Functions.,	53	BER/DER C Encode Functions.,	72
xd_uint64		xe_enumUnsigned	
BER/DER C Decode Functions.,	54	BER/DER C Encode Functions.,	73
xd_uint8		xe_expandBuffer	
BER/DER C Decode Functions.,	55	BER/DER C Encode Functions.,	73
xd_unsigned		xe_free	
BER/DER C Decode Functions.,	55	BER/DER C Encode Functions.,	74
xd_utf8str		xe_getBufLocDescr	
BER/DER C Decode Functions.,	16	BER/DER C Encode Functions.,	74
xdf_ReadContents		xe_getp	
BER/DER C File Functions.,	58	BER/DER C Encode Functions.,	75
xdf_ReadPastEOC		xe_identifier	
BER/DER C File Functions.,	58	BER/DER C Encode Functions.,	75
xdf_TagAndLen		xe_int16	
BER/DER C File Functions.,	60	BER/DER C Encode Functions.,	76
xdf_len		xe_int64	
BER/DER C File Functions.,	57	BER/DER C Encode Functions.,	76
xdf_tag		xe_int8	
BER/DER C File Functions.,	60	BER/DER C Encode Functions.,	77
xe_16BitCharStr		xe_integer	
BER/DER C Encode Functions.,	64	BER/DER C Encode Functions.,	77
xe_32BitCharStr		xe_len	
BER/DER C Encode Functions.,	65	BER/DER C Encode Functions.,	78
xe_OpenType		xe_len64	
BER/DER C Encode Functions.,	81	BER/DER C Encode Functions.,	78
xe_OpenTypeExt		xe_memcpy	
BER/DER C Encode Functions.,	82	BER/DER C Encode Functions.,	79
xe_OpenTypeExtDer		xe_null	

BER/DER C Encode Functions., 79

xe\_objid  
BER/DER C Encode Functions., 80

xe\_octstr  
BER/DER C Encode Functions., 80

xe\_oid64  
BER/DER C Encode Functions., 81

xe\_real  
BER/DER C Encode Functions., 82

xe\_real10  
BER/DER C Encode Functions., 83

xe\_reloid  
BER/DER C Encode Functions., 83

xe\_setp  
BER/DER C Encode Functions., 84

xe\_tag  
BER/DER C Encode Functions., 84

xe\_tag\_len  
BER/DER C Encode Functions., 85

xe\_timeofdaystr  
BER/DER C Encode Functions., 86

xe\_timestr  
BER/DER C Encode Functions., 87

xe\_uint16  
BER/DER C Encode Functions., 87

xe\_uint64  
BER/DER C Encode Functions., 88

xe\_uint8  
BER/DER C Encode Functions., 88

xe\_unsigned  
BER/DER C Encode Functions., 89

xe\_utf8str  
BER/DER C Encode Functions., 63

xu\_RestoreBufferState  
BER/PER C Utility Functions, 99

xu\_SaveBufferState  
BER/PER C Utility Functions, 100

xu\_alloc\_array  
BER/PER C Utility Functions, 97

xu\_dump  
BER/PER C Utility Functions, 97

xu\_dump2  
BER/PER C Utility Functions, 98

xu\_fdump  
BER/PER C Utility Functions, 99

xu\_hex\_dump  
BER/PER C Utility Functions, 91