

ASN1C

ASN.1 Compiler
Version 6.0
BER Runtime
Reference Manual

Contents

1	ASN1C BER/DER Runtime Classes and Library Functions	1
2	ASN1C BER Runtime Module Index	2
2.1	ASN1C BER Runtime Modules	2
3	ASN1C BER Runtime Hierarchical Index	3
3.1	ASN1C BER Runtime Class Hierarchy	3
4	ASN1C BER Runtime Class Index	4
4.1	ASN1C BER Runtime Class List	4
5	ASN1C BER Runtime File Index	5
5.1	ASN1C BER Runtime File List	5
6	ASN1C BER Runtime Module Documentation	6
6.1	BER/DER/CER C++ Run-Time Classes.	6
6.2	BER Message Buffer Classes	7
6.3	BER Runtime Library Functions.	8
6.4	BER/DER C Decode Functions.	12
6.5	BER/DER C File Functions.	29
6.6	BER/DER C Encode Functions.	32
6.7	BER/PER C Utility Functions	46
6.8	Streaming BER Runtime Library Functions.	51
6.9	C Streaming BER Encode Functions.	53
6.10	C Streaming BER Decode Functions.	67
6.11	C++ classes for streaming BER encoding.	82
6.12	C++ classes for streaming BER decoding.	83
7	ASN1C BER Runtime Class Documentation	84
7.1	_Asn1BufLocDescr Struct Reference	84
7.2	ASN1BERDecodeBuffer Class Reference	85

7.3	ASN1BERDecodeStream Class Reference	89
7.4	ASN1BEREncodeBuffer Class Reference	108
7.5	ASN1BEREncodeStream Class Reference	111
7.6	ASN1BERMessageBuffer Class Reference	127
8	ASN1C BER Runtime File Documentation	129
8.1	asn1ber.h File Reference	129
8.2	asn1BerCppTypes.h File Reference	133
8.3	ASN1BERDecodeStream.h File Reference	134
8.4	ASN1BEREncodeStream.h File Reference	135
8.5	asn1berSocket.h File Reference	136
8.6	asn1berStream.h File Reference	137

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

ASN1C BER Runtime Module Index

2.1 ASN1C BER Runtime Modules

Here is a list of all modules:

BER/DER/CER C++ Run-Time Classes.	6
BER Message Buffer Classes	7
C++ classes for streaming BER encoding.	82
C++ classes for streaming BER decoding.	83
BER Runtime Library Functions.	8
BER/DER C Decode Functions.	12
BER/DER C File Functions.	29
BER/DER C Encode Functions.	32
BER/PER C Utility Functions	46
Streaming BER Runtime Library Functions.	51
C Streaming BER Encode Functions.	53
C Streaming BER Decode Functions.	67

Chapter 3

ASN1C BER Runtime Hierarchical Index

3.1 ASN1C BER Runtime Class Hierarchy

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

<code>_Asn1BufLocDescr</code>	84
<code>ASN1BERDecodeStream</code>	89
<code>ASN1BEREncodeStream</code>	111
<code>ASN1BERMessageBuffer</code>	127
<code>ASN1BERDecodeBuffer</code>	85
<code>ASN1BEREncodeBuffer</code>	108

Chapter 4

ASN1C BER Runtime Class Index

4.1 ASN1C BER Runtime Class List

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

_Asn1BufLocDescr	84
ASN1BERDecodeBuffer	85
ASN1BERDecodeStream	89
ASN1BEREncodeBuffer	108
ASN1BEREncodeStream	111
ASN1BERMessageBuffer	127

Chapter 5

ASN1C BER Runtime File Index

5.1 ASN1C BER Runtime File List

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

asn1ber.h	129
asn1BerCppType.h	133
ASN1BERDecodeStream.h	134
ASN1BEREncodeStream.h	135
asn1berSocket.h	136
asn1berStream.h	137

Chapter 6

ASN1C BER Runtime 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.](#)

6.2 BER Message Buffer Classes

6.2.1 Detailed Description

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

Classes

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

6.3 BER Runtime Library Functions.

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.

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.](#)

Classes

- [struct _Asn1BufLocDescr](#)

Defines

- `#define ASN_K_INDEFLEN -9999`
- `#define XU_DUMP(msg) xu_dump(msg,0,0)`
- `#define xd_resetp(pctx) rtxResetContext(pctx)`
- `#define ASN1TAG2BYTE(tag) ((OSOCTET)(((tag)&TM_B_IDCODE)|((tag)>>ASN1TAG_LSHIFT)))`
- `#define XD_MEMCPY1(pctx, object_p)`
- `#define XD_FETCH1(pctx) ((pctx) → buffer.data[(pctx) → buffer.byteIndex++]`
- `#define XD_PEEKTAG(pctx, tag) (((pctx) → buffer.data[(pctx) → buffer.byteIndex] & (~0x20)) == (tag & (~0x20)))`
- `#define XD_PEEKPC(pctx) (((pctx) → buffer.data[(pctx) → buffer.byteIndex] & 0x20) == 0x20)`
- `#define XD_MATCHEOC(pctx)`
- `#define XD_MATCHBYTES1(pctx, b1) ((pctx) → buffer.data[(pctx) → buffer.byteIndex] == b1)`
- `#define XD_MATCHBYTES2(pctx, b1, b2)`
- `#define XD_MATCHBYTES3(pctx, b1, b2, b3)`
- `#define XD_MATCHBYTES4(pctx, b1, b2, b3, b4)`
- `#define XD_MATCHBYTES5(pctx, b1, b2, b3, b4, b5)`
- `#define XD_BUMPIDX(pctx, nbytes) ((pctx) → buffer.byteIndex += nbytes)`
- `#define XD_CHKBUFEND(pctx)`
- `#define XD_CHKDEFLEN(pctx, len)`
- `#define XD_CHKEOB(pctx)`
- `#define XD_CHKEND(pctx, ccb_p)`
- `#define XE_CHKBUF(pctx, len)`
- `#define XE_PUT1(pctx, ch) (pctx) → buffer.data[-(pctx) → buffer.byteIndex] = ch;`
- `#define XE_PUT2(pctx, ch1, ch2)`
- `#define XE_SAFEPUT1(pctx, ch) XE_CHKBUF(pctx,1); (pctx) → buffer.data[-(pctx) → buffer.byteIndex] = ch;`

Typedefs

- typedef [_Asn1BufLocDescr](#) [Asn1BufLocDescr](#)

6.3.2 Define Documentation

6.3.2.1 #define XD_CHKBUFEND(pctxt)

Value:

```
((ASN1BUF_INDFLEN(pctxt)) || \
((pctxt)->buffer.byteIndex <= (pctxt)->buffer.size) ? 0 : RTERR_ENDOFBUF)
```

6.3.2.2 #define XD_CHKDFLEN(pctxt, len)

Value:

```
((ASN1BUF_INDFLEN(pctxt)) || \
((pctxt)->buffer.byteIndex + len) <= (pctxt)->buffer.size) ? \
0 : RTERR_ENDOFBUF)
```

6.3.2.3 #define XD_CHKEND(pctxt, ccb_p)

Value:

```
((ccb_p)->len == ASN_K_INDFLEN) ? XD_CHKEOB(pctxt) : \
(OSRTBUFPTR(pctxt) - (ccb_p)->ptr >= (ccb_p)->len) || \
((pctxt)->buffer.byteIndex >= (pctxt)->buffer.size))
```

6.3.2.4 #define XD_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))
```

6.3.2.5 #define XD_MATCHBYTES2(pctxt, b1, b2)

Value:

```
((pctxt)->buffer.data[(pctxt)->buffer.byteIndex] == b1) && \
((pctxt)->buffer.data[(pctxt)->buffer.byteIndex+1] == b2))
```

6.3.2.6 #define XD_MATCHBYTES3(pctxt, b1, b2, b3)

Value:

```
((pctxt)->buffer.data[(pctxt)->buffer.byteIndex] == b1) && \
((pctxt)->buffer.data[(pctxt)->buffer.byteIndex+1] == b2) && \
((pctxt)->buffer.data[(pctxt)->buffer.byteIndex+2] == b3))
```

6.3.2.7 #define XD_MATCHBYTES4(pctxt, b1, b2, b3, b4)

Value:

```
((pctxt)->buffer.data[(pctxt)->buffer.byteIndex] == b1) && \
((pctxt)->buffer.data[(pctxt)->buffer.byteIndex+1] == b2) && \
((pctxt)->buffer.data[(pctxt)->buffer.byteIndex+2] == b3) && \
((pctxt)->buffer.data[(pctxt)->buffer.byteIndex+3] == b4))
```

6.3.2.8 #define XD_MATCHBYTES5(pctxt, b1, b2, b3, b4, b5)

Value:

```
((pctxt)->buffer.data[(pctxt)->buffer.byteIndex] == b1) && \
((pctxt)->buffer.data[(pctxt)->buffer.byteIndex+1] == b2) && \
((pctxt)->buffer.data[(pctxt)->buffer.byteIndex+2] == b3) && \
((pctxt)->buffer.data[(pctxt)->buffer.byteIndex+3] == b4) && \
((pctxt)->buffer.data[(pctxt)->buffer.byteIndex+4] == b5))
```

6.3.2.9 #define XD_MATCHEOC(pctxt)

Value:

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

6.3.2.10 #define XD_MEMCPY1(pctxt, object_p)

Value:

```
(ASN1BUF_INDEFLen(pctxt) || \
(pctxt)->buffer.byteIndex < (pctxt)->buffer.size) ? \
(*object_p = *OSRTBUPPTR(pctxt), (pctxt)->buffer.byteIndex ++, 0) : \
RTERR_ENDOFBUF)
```

6.3.2.11 #define XE_CHKBUF(pctxt, len)

Value:

```
if (len > (pctxt)->buffer.byteIndex) { \
int stat = xe_expandBuffer (pctxt, len); \
if (stat != 0) return stat; }
```

6.3.2.12 #define XE_PUT2(pctxt, ch1, ch2)

Value:

```
(pctxt)->buffer.byteIndex -= 2; \  
(pctxt)->buffer.data[(pctxt)->buffer.byteIndex] = ch1; \  
(pctxt)->buffer.data[(pctxt)->buffer.byteIndex+1] = ch2;
```

6.3.3 Typedef Documentation

6.3.3.1 typedef struct [_Asn1BufLocDescr](#) Asn1BufLocDescr

Buffer location descriptor

6.4 BER/DER C Decode Functions.

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.

Defines

- #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_match` (OSCTXT *pctxt, ASN1TAG tag, int *len_p, 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_bitstr` (OSCTXT *pctxt, const OSOCTET **object_p2, OSUINT32 *numbits_p, ASN1TagType tagging, int length)
- int `xd_octstr_s` (OSCTXT *pctxt, OSOCTET *object_p, OSUINT32 *pnumocts, ASN1TagType tagging, int length)
- int `xd_octstr` (OSCTXT *pctxt, const OSOCTET **object_p2, OSUINT32 *pnumocts, ASN1TagType tagging, int length)
- int `xd_charstr` (OSCTXT *pctxt, const char **object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int `xd_16BitCharStr` (OSCTXT *pctxt, Asn16BitCharString *object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int `xd_32BitCharStr` (OSCTXT *pctxt, Asn132BitCharString *object_p, ASN1TagType tagging, ASN1TAG tag, int length)

- 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_OpenType` (OSCTXT *pctxt, const OSOCTET **object_p2, OSUINT32 *pnumocts)
- int `xd_OpenTypeExt` (OSCTXT *pctxt, ASN1CCB *ccb_p, ASN1TAG tag, OSRTDList *pElemList)
- int `xd_OpenTypeAppend` (OSCTXT *pctxt, OSRTDList *pElemList)
- int `xd_real10` (OSCTXT *pctxt, const char **object_p, ASN1TagType tagging, int length)
- int `xd_decimal` (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_indeflen_ex` (const OSOCTET *msg_p, int bufSize)
- int `xd_len` (OSCTXT *pctxt, int *len_p)
- int `xd_chkend` (OSCTXT *pctxt, ASN1CCB *ccb_p)
- int `xd_count` (OSCTXT *pctxt, int length, int *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)

6.4.2 Define Documentation

6.4.2.1 #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:

pctxt Pointer to ASN.1 context block structure

object_p 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.

tagging Specifies whether element is implicitly or explicitly tagged.

length 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 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:

pctxt Pointer to ASN.1 context block structure

object_p 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.

tagging Specifies whether element is implicitly or explicitly tagged.

tag Tag variable to match

length 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 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:

pctxt Pointer to ASN.1 context block structure

object_p 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.

tagging Specifies whether element is implicitly or explicitly tagged.

tag Tag variable to match.

length 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 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 decimal strings starting with no prefix. If it is necessary to convert a decimal string to another radix, then use `rtxBigIntSetStr` / `rtxBigIntToString` functions.

Parameters:

pctxt Pointer to context block structure.

tagging Specifies whether element is implicitly or explicitly tagged.

length Length of data to retrieve. Valid for implicit case only.

object_p Pointer to a character pointer variable to receive the decoded unsigned value. Dynamic memory is allocated for the variable using the `rtxMemAlloc` function. The decoded variable is represented as a decimal string starting with no prefix.

Returns:

Completion status of operation:

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

6.4.3.4 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:

pctxt Pointer to context block structure.

tagging Specifies whether element is implicitly or explicitly tagged.

length 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.

numbits_p Pointer to an integer value to receive the decoded number of bits.

object_p2 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.

6.4.3.5 int xd_bitstr_s (OSCTXT * *pctxt*, OSOCTET * *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:

pctxt Pointer to context block structure.

tagging Specifies whether element is implicitly or explicitly tagged.

length 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.

numbits_p 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.

object_p 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.

6.4.3.6 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:

pctxt Pointer to context block structure.

tagging Specifies whether element is implicitly or explicitly tagged.

length Length of data to retrieve. Valid for implicit case only.

object_p 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)

6.4.3.7 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, Visible-String, etc. This function allocates memory for the decoded string and returns a pointer to the data.

Parameters:

pctxt Pointer to ASN.1 context block structure

object_p 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.

tagging Specifies whether element is implicitly or explicitly tagged.

tag Tag variable to match

length Length of data to retrieve. Valid for implicit case only.

Returns:

Completion status of operation:

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

6.4.3.8 int xd_chkend (OSCTXT * *pctxt*, 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:

pctxt Pointer to context block structure.

ccb_p 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.

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

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

Parameters:

pctxt Pointer to context block structure.

length Length of the constructed type.

count_p Pointer to an integer variable to receive the element count.

Returns:

Completion status of operation:

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

6.4.3.10 int xd_decimal (OSCTXT * *pctxt*, const char ** *object_p*, ASN1TagType *tagging*, int *length*)

This function decodes a value of the decimal encoded ASN.1 REAL type. Result will be presented in xsd:decimal format.

Parameters:

pctxt Pointer to context block structure.

object_p Pointer to a character pointer variable to receive the decoded result. Dynamic memory is allocated for the variable using the rtxMemAlloc function.

tagging Specifies whether element is implicitly or explicitly tagged.

length Length of data to retrieve. Valid for implicit case only.

Returns:

Completion status of operation:

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

6.4.3.11 `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:

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

Returns:

Completion status of operation:

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

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

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

Parameters:

- msg_p* Pointer to an indefinite length message component.
- bufSize* Size of the message's buffer

6.4.3.13 `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:

- pctxt* Pointer to context block structure.
- tagging* Specifies whether element is implicitly or explicitly tagged.
- length* Length of data to retrieve. Valid for implicit case only.
- object_p* Pointer to decoded 16-bit integer value.

Returns:

Completion status of operation:

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

6.4.3.14 `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:

pctxt Pointer to context block structure.

tagging Specifies whether element is implicitly or explicitly tagged.

length Length of data to retrieve. Valid for implicit case only.

object_p Pointer to decoded 64-bit integer value.

Returns:

Completion status of operation:

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

6.4.3.15 `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:

pctxt Pointer to context block structure.

tagging Specifies whether element is implicitly or explicitly tagged.

length Length of data to retrieve. Valid for implicit case only.

object_p Pointer to decoded 8-bit integer value.

Returns:

Completion status of operation:

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

6.4.3.16 `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:

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

Returns:

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

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

This function decodes an ASN.1 length value.

Parameters:

- pctxt* Pointer to context block structure.
- len_p* 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.

6.4.3.18 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:

- pctxt* Pointer to context block structure.
- tag* Tag variable to match.
- len_p* Length of message component. Returned as follows: ≥ 0 component is fixed length ASN_K_INDEFLEN component is indefinite length
- flags* Bit flags used to set the following options:
 - XM_ADVANCE: Advance decode pointer on match.
 - XM_SEEK : Seek until match found or EOM.
 - XM_SKIP : Skip to next tag before search

Returns:

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

6.4.3.19 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:

pctxt Pointer to context block structure.

tag Tag variable to match in octet format.

len_p Length of message component. Returned as follows: ≥ 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.

6.4.3.20 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:

pctxt 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.

**object_p* A pointer to a memory structure to receive the copied data.

length The number of bytes to copy from the contents field.

Returns:

Completion status of operation:

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

6.4.3.21 int xd_NextElement (OSCTXT * *pctxt*)

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

Parameters:

pctxt Pointer to context block structure.

Returns:

Completion status of operation:

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

6.4.3.22 `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:

pctxt Pointer to ASN.1 context block structure

tagging 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.

6.4.3.23 `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:

pctxt Pointer to context block structure.

object_p 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.

tagging Specifies whether element is implicitly or explicitly tagged.

length 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 `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:

pctxt Pointer to ASN.1 context block structure

object_p2 Pointer to a pointer to receive the address of the allocated memory into which the decoded data will be stored.

pnumocts Pointer to an integer variable to receive length of the decoded octet string.

tagging Specifies whether element is implicitly or explicitly tagged.

length 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 `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:

pctxt Pointer to ASN.1 context block structure

object_p Pointer to static octet array to receive decoded data

pnumocts 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.

tagging Specifies whether element is implicitly or explicitly tagged.

length Length of data to retrieve. Valid for implicit case only.

Returns:

Completion status of operation:

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

6.4.3.26 `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:

pctxt Pointer to context block structure.

object_p 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.

tagging Specifies whether element is implicitly or explicitly tagged.

length Length of data to retrieve. Valid for implicit case only.

Returns:

Completion status of operation:

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

6.4.3.27 `int xd_OpenType (OSCTXT * pctxt, const OSOCTET ** object_p2, OSUINT32 * 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:

pctxt Pointer to context block structure.

object_p2 Pointer to value to receive decoded result.

pnumocts 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.

6.4.3.28 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:

pctxt Pointer to context block structure.

pElemList 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.

6.4.3.29 int xd_OpenTypeExt (OSCTXT * *pctxt*, ASN1CCB * *ccb_p*, ASN1TAG *tag*, 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:

pctxt Pointer to context block structure.

ccb_p 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.

tag Next expected tag value (or ASN_K_NOTAG value if last field). The routine will loop through elements until matching tag found or some other error occurs.

pElemList 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.

6.4.3.30 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:

pctxt Pointer to context block structure.

object_p Pointer to value to receive decoded result. The OSREAL data type is a double-precision floating point number type (C double type).

tagging Specifies whether element is implicitly or explicitly tagged.

length Length of data to retrieve. Valid for implicit case only.

Returns:

Completion status of operation:

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

6.4.3.31 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:

pctxt Pointer to context block structure.

object_p Pointer to a character pointer variable to receive the decoded result. Dynamic memory is allocated for the variable using the rtxMemAlloc function.

tagging Specifies whether element is implicitly or explicitly tagged.

length Length of data to retrieve. Valid for implicit case only.

Returns:

Completion status of operation:

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

6.4.3.32 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:

pctxt Pointer to context block structure.

object_p 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.

tagging Specifies whether element is implicitly or explicitly tagged.

length Length of data to retrieve. Valid for implicit case only.

Returns:

Completion status of operation:

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

6.4.3.33 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:

pctxt Pointer to context block structure.

msg_p Pointer to message buffer containing data to be decoded.

msglen Pointer to 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.

tag_p 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.

len_p 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.

6.4.3.34 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:

pctxt Pointer to context block structure.

tag_p Pointer to variable to receive decoded tag info.

Returns:

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

6.4.3.35 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:

pctxt Pointer to context block structure.

len_p Length of message component. Returned as follows: ≥ 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.

6.4.3.36 `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:

pctxt Pointer to context block structure.

tag_p Pointer to variable to receive decoded tag info.

len_p Length of message component. Returned as follows: ≥ 0 component is fixed length ASN_K_INDEFLEN component is indefinite length

flags 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.

6.4.3.37 `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:

pctxt Pointer to context block structure.

tagging Specifies whether element is implicitly or explicitly tagged.

length Length of data to retrieve. Valid for implicit case only.

object_p Pointer to decoded 16-bit unsigned integer value.

Returns:

Completion status of operation:

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

6.4.3.38 `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:

pctxt Pointer to context block structure.

tagging Specifies whether element is implicitly or explicitly tagged.

length Length of data to retrieve. Valid for implicit case only.

object_p Pointer to decoded 64-bit unsigned integer value.

Returns:

Completion status of operation:

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

6.4.3.39 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:

pctxt Pointer to context block structure.

tagging Specifies whether element is implicitly or explicitly tagged.

length Length of data to retrieve. Valid for implicit case only.

object_p Pointer to decoded 8-bit unsigned integer value.

Returns:

Completion status of operation:

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

6.4.3.40 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:

pctxt Pointer to context block structure.

tagging Specifies whether element is implicitly or explicitly tagged.

length Length of data to retrieve. Valid for implicit case only.

object_p 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.

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.

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)

6.5.2 Function Documentation

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

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

Parameters:

fp 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.

plen A pointer to an integer to receive the decoded length value.

buffer The buffer to receive the parsed data.

pbufidx 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 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:

fp 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).

len 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.

buffer The buffer to receive the parsed data.

bufsiz The size of the buffer to receive the parsed data.

pbuflidx 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 int xdf_ReadPastEOC (FILE * *fp*, OSOCTET * *buffer*, int *bufsiz*, int * *pbuflidx*)

This function consumes bytes from the file stream until a matching end-of-context (EOC) maker 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:

fp 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).

buffer The buffer to receive the parsed data.

bufsiz The size of the buffer to receive the parsed data.

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

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

Parameters:

fp 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.

ptag A pointer to an ASN.1 tag structure to receive the decoded tag.

buffer The buffer to receive the parsed data.

pbuflidx 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 int xdf_TagAndLen (FILE * *fp*, ASN1TAG * *ptag*, OSINT32 * *plen*, OSOCTET * *buffer*, int * *pbufix*)

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

Parameters:

fp 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.

ptag A pointer to an ASN1TAG variable to receive parsed the tag value.

plen A pointer to an integer to receive the decoded length value.

buffer The buffer to receive the parsed data.

pbufix 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.

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 compound.

Defines

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

Functions

- `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_bitstr (OSCTXT *pctxt, const OSOCTET *object_p, OSUINT32 numbits, ASN1TagType tagging)`
- `int xe_octstr (OSCTXT *pctxt, const OSOCTET *object_p, OSUINT32 numocts, ASN1TagType tagging)`
- `int xe_charstr (OSCTXT *pctxt, const char *object_p, ASN1TagType tagging, ASN1TAG tag)`
- `int xe_16BitCharStr (OSCTXT *pctxt, Asn16BitCharString *object_p, ASN1TagType tagging, ASN1TAG tag)`
- `int xe_32BitCharStr (OSCTXT *pctxt, Asn132BitCharString *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_real (OSCTXT *pctxt, OSREAL *object_p, ASN1TagType tagging)`
- `int xe_OpenType (OSCTXT *pctxt, const OSOCTET *object_p, OSUINT32 numocts)`
- `int xe_OpenTypeExt (OSCTXT *pctxt, OSRTDList *pElemList)`
- `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, int 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_derCanonicalSort` (OSCTXT *pctxt, OSRTSList *pList)
- int `xe_TagAndIndefLen` (OSCTXT *pctxt, ASN1TAG tag, int length)
- void `xe_getBufLocDescr` (OSCTXT *pctxt, OSUINT32 length, `Asn1BufLocDescr` *pDescr)

6.6.2 Define Documentation

6.6.2.1 #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:

- pctxt* 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.
- object_p* A pointer to a null-terminated UTF-8 C character string to be encoded.
- tagging* 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 int `xe_16BitCharStr(OSCTXT *pctxt, 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:

- pctxt* 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.
- object_p* 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.
- tagging* 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'.
- tag* 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.

6.6.3.2 **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:

pctxt 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.

object_p 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.

tagging 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'.

tag 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.

6.6.3.3 **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. It is highly recommended to use the hexadecimal or binary strings for better performance.

Parameters:

pctxt Pointer to context block structure.

tagging 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'.

object_p 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.

6.6.3.4 **int xe_bitstr (OSCTXT * *pctxt*, const OSOCTET * *object_p*, OSUINT32 *numbits*, ASN1TagType *tagging*)**

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

Parameters:

pctxt 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.

object_p 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.

numbits The number of bits within the bit string to be encoded.

tagging 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.

6.6.3.5 int xe_boolean (OSCTXT *pctx, OSBOOL *object_p, ASN1TagType tagging)

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

Parameters:

pctx 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.

**object_p* 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.

tagging 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.

6.6.3.6 int xe_charstr (OSCTXT *pctx, 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:

pctx 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.

object_p A pointer to a null-terminated C character string to be encoded

tagging 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'.

tag 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.

6.6.3.7 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 prior to encoding.

Parameters:

pctxt 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.

pList 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.

6.6.3.8 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:

pctxt 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.

object_p Value to be encoded.

tagging 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.

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

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

Parameters:

pctxt 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.

object_p 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.)

tagging 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.

6.6.3.10 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:

pctxt 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.

length 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 `ASN_K_ENCBUFSIZ` for small requests to limit the required number of expansions.

6.6.3.11 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:

pctxt 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.

6.6.3.12 void xe_getBufLocDescr (OSCTXT * *pctxt*, OSUINT32 *length*, Asn1BufLocDescr * *pDescr*)**Parameters:**

pctxt 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.

length The actual length of existing message components.

pDescr A pointer to a buffer location descriptor.

6.6.3.13 OSOCTET* 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 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:

pctxt 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.

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

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

Parameters:

pctxt Pointer to context block structure.

tagging 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'.

object_p 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.

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

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

Parameters:

pctxt Pointer to context block structure.

tagging 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'.

object_p 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.

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

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

Parameters:

pctxt Pointer to context block structure.

tagging 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'.

object_p 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.

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

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

Parameters:

pctxt Pointer to context block structure.

tagging 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'.

object_p 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.

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

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

Parameters:

pctxt 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.

length The length variable to encode. A negative number is interpreted that an indefinite length integer should be encoded.

Returns:

Completion status of operation:

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

6.6.3.19 `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:

pctxt 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.

object_p A pointer to a buffer containing the bytes to be copied.

length The number of bytes to copy.

Returns:

Completion status of operation:

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

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

This function will encode an ASN.1 NULL placeholder.

Parameters:

pctxt 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.

tagging 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.

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

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

Parameters:

pctxt Pointer to context block structure.

tagging 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'.

object_p 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:

Completion status of operation:

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

6.6.3.22 `int xe_octstr (OSCTXT * pctxt, const OSOCTET * object_p, OSUINT32 numocts, ASN1TagType tagging)`

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

Parameters:

- pctxt* 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.
- object_p* 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.
- numocts* The number of octets (bytes) within the OCTET STRING to be encoded.
- tagging* 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.

6.6.3.23 `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:

- pctxt* Pointer to context block structure.
- tagging* 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'.
- object_p* 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:

Completion status of operation:

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

6.6.3.24 `int xe_OpenType (OSCTXT * pctxt, const OSOCTET * object_p, OSUINT32 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:

- pctxt* 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.
- object_p* A pointer to a buffer containing an encoded ASN.1 message component.
- numocts* The number of octets to be encoded.

Returns:

Completion status of operation:

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

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

Parameters:

pctxt 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.

pElemList A pointer to open type to be encoded.

6.6.3.26 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:

pctxt 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.

object_p 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 `rtxGetPlusInfinity` and `rtxGetMinusInfinity` functions to set the real value to be encoded.

tagging 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.

6.6.3.27 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:

pctxt 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.

object_p Value to be encoded.

tagging 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.

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

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

Parameters:

pctxt Pointer to context block structure.

tagging 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'.

object_p 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:

Completion status of operation:

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

6.6.3.29 `int xe_setp (OSCTXT * pctxt, OSOCTET * buf_p, int 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:

pctxt 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.

**buf_p* 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).

bufsiz The length of the memory buffer in bytes.

6.6.3.30 `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 ANS1C 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:

**pctxt* 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.

tag The ASN.1 tag to be encoded in the message. This parameter is passed using the ANS1C internal tag representation. It is passed as an unsigned 32-bit integer.

length The length of the contents field previously encoded. 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:

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.

6.6.3.31 `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:

- pctxt* 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.
- tag* ASN.1 tag value to be encoded.
- length* The actual length of existing message components.

Returns:

Completion status of operation:

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

6.6.3.32 `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:

- pctxt* Pointer to context block structure.
- tagging* 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'.
- object_p* 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.

6.6.3.33 `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:

- pctxt* Pointer to context block structure.
- tagging* 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'.
- object_p* 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.

6.6.3.34 `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:

pctxt Pointer to context block structure.

tagging 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'.

object_p 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.

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

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

Parameters:

pctxt Pointer to context block structure.

tagging 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'.

object_p 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

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.

Defines

- `#define xu_addTagErrParm` `berErrAddTagParm`

Functions

- OSBOOL `berErrAddTagParm` (OSCTXT *pctxt, ASN1TAG tag)
- int `berErrUnexpTag` (OSCTXT *pctxt, ASN1TAG exptag)
- const char * `berTagToString` (ASN1TAG tag, char *buffer, size_t bufsiz)
- const char * `berTagToDynStr` (OSCTXT *pctxt, ASN1TAG tag)
- int `xu_verify_len` (OSOCKET *msg_p)
- void * `xu_parse_mmbuf` (OSOCKET **buf_p2, int *buflen_p, OSOCKET *start_p, int bufsiz)
- void `xu_alloc_array` (OSCTXT *pctxt, ASN1SeqOf *seqOf_p, int recSize, int recCount)
- void `xu_octscopy_s` (OSUIN32 *nocts_p, OSOCKET *data_p, char *cstr, char zterm)
- void `xu_octscopy_ss` (ASN1OctStr *octStr_p, char *cstring, char zterm)
- void `xu_octscopy_d` (OSCTXT *pctxt, OSUIN32 *nocts_p, const OSOCKET **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_hexscopy` (char *data, char *hstring)
- int `xu_binscopy` (char *data, char *bstring)
- int `xu_dump` (OSOCKET *msgptr, ASN1DumpCbFunc cb, void *cbArg_p)
- int `xu_fdump` (FILE *file_p, OSOCKET *msgptr)
- void `xu_hex_dump` (OSOCKET *data, int numocts, OSBOOL hdrflg)
- void `xu_fmt_tag` (ASN1TAG *tag_p, char *class_p, char *form_p, char *id_code)
- char * `xu_fmt_contents` (OSCTXT *pctxt, int len, int *count)
- int `xu_fread` (FILE *fp, OSOCKET *bufp, int bufsiz)
- void `xu_SaveBufferState` (OSCTXT *pCtxt, OSRTBufSave *pSavedInfo)
- void `xu_RestoreBufferState` (OSCTXT *pCtxt, OSRTBufSave *pSavedInfo)
- int `berReadMsgFromSocket` (OSCTXT *pctxt, OSRTSOCKET socket, OSOCKET *buffer, int bufsiz, OSOCKET **ppDestBuffer, int *pMessageSize)

6.7.2 Function Documentation

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

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

Parameters:

pctxt Pointer to a context structure.

tag The tag to add.

Returns:

Pointer to the text string (buffer).

6.7.2.2 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:

pctxt Pointer to a context structure.

exptag Expected tag.

Returns:

RTERR_IDNOTFOU status code.

6.7.2.3 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:

pctxt 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.

socket The socket to read from. It should be already connected TCP socket.

buffer The static buffer to receive the parsed data. Can be NULL.

bufsiz The size of the buffer to receive the parsed data. Should be 0, if *buffer* is NULL.

ppDestBuffer The pointer to receive the destination buffer pointer. If *buffer* is static, this parameter can be NULL.

pMessageSize The pointer to integer to receive the size of read message.

Returns:

Completion status of operation:

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

6.7.2.4 **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 `rtxMemAlloc` function. This memory will be release when context memory is freed.

Parameters:

- pctxt* Pointer to a context structure.
- tag* The tag to convert.

Returns:

Pointer to dynamic text string.

6.7.2.5 **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:

- tag* The tag to convert.
- buffer* Text buffer into which the string will be written.
- bufsiz* Size of the text buffer.

Returns:

Pointer to the text string (*buffer*).

6.7.2.6 **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:

- pctxt* 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.
- seqOf_p* 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 `recCount` passed into the function. The data pointer is set to the value that is returned from the memory allocation function.
- recSize* The number of bytes in one record in the array.
- recCount* 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 `recCount` passed into the function. The data pointer is set to the value that is returned from the memory allocation function.

6.7.2.7 **int xu_dump (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 `rtxsrc/rtxCommon.h` can be used for this purpose).

Parameters:

**msgptr* A pointer to an encoded ASN.1 message.

cb Callback function that gets invoked for each line of formatted output. For a dump to standard output (`stdout`), this parameter can be specified as `NULL`.

cbArg_p 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.

6.7.2.8 int xu_fdump (FILE *file_p, 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:

**file_p* A text file pointer.

**msgptr* 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.

6.7.2.9 void xu_hex_dump (OSOCTET * data, int numocts, OSBOOL hdrflg)

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:

**data* A pointer to the start of the block of memory to be dumped.

numocts The number of octets (bytes) to be dumped.

hdrflg A Boolean variable indicating whether or not a head line should be dumped as the first line of the display.

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

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

Parameters:

pCtxt - A pointer to a context structure.

pSavedInfo - A pointer to a structure holding the saved info.

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

This function is used to save the current buffer state.

Parameters:

pCtxt - A pointer to a context structure.

pSavedInfo - A pointer to a structure to hold the saved info.

6.8 Streaming BER Runtime Library Functions.

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.

Modules

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

Defines

- `#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 cerEncCanonicalSort (OSCTXT *pctxt, OSCTXT *pMemCtxt, OSRTSList *pList)`
- `void cerGetBufLocDescr (OSCTXT *pctxt, Asn1BufLocDescr *pDescr)`
- `void cerAddBufLocDescr (OSCTXT *pctxt, OSRTSList *pElemList, Asn1BufLocDescr *pDescr)`

6.8.2 Define Documentation

6.8.2.1 `#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 `#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))
```

6.8.3 Function Documentation

6.8.3.1 `int berStrmFreeContext (OSCTXT * pctxt)`

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

Parameters:

pctxt A pointer to a context structure.

Returns:

Completion status of operation:

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

6.8.3.2 `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:

pctxt 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.

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.

Functions

- int `berEncStrmBigInt` (OSCTXT *pctx, const char *pvalue, 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 `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, OSUINT32 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 `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)

6.9.2 Function Documentation

6.9.2.1 `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:

pctxt 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.

pvalue A pointer to a character string containing the value to be encoded.

tagging 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 `int berEncStrmBitStr (OSCTXT * pctxt, const OSOCTET * object_p, OSUINT32 numbits, ASN1TagType tagging)`

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

Parameters:

pctxt 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.

object_p 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.

numbits The number of bits within the bit string to be encoded.

tagging 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 `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:

pctxt 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.

object_p 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.

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

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

Parameters:

pctxt 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.

value 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.

tagging 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 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:

pctxt 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.

object_p A pointer to a null-terminated C character string to be encoded.

tagging 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'.

tag 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.

6.9.2.6 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:

pctxt 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.

value An integer containing the enumerated value to be encoded.

tagging 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.

6.9.2.7 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:

pctxt 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.

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

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

Parameters:

pctxt 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.

value 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.

tagging 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.

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

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

Parameters:

- pctxt* 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.
- value* 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.
- tagging* 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.

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

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

Parameters:

- pctxt* 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.
- value* 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.
- tagging* 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.

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

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

Parameters:

- pctxt* 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.
- value* 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.
- tagging* 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 int berEncStrmLength (OSCTXT * *pctxt*, int *length*)

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

Parameters:

pctxt 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.

length 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.

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

This function encodes an ASN.1 NULL placeholder.

Parameters:

pctxt 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.

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

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

Parameters:

pctxt 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.

object_p 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.

tagging 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 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:

pctxt 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.

object_p 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.

tagging 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 int berEncStrmOctStr (OSCTXT * *pctxt*, const OSOCTET * *object_p*, OSUINT32 *numocts*, ASN1TagType *tagging*)

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

Parameters:

pctxt 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.

object_p A pointer to an OCTET STRING containing the octet data to be encoded.

numocts The number of octets (bytes) within the OCTET STRING to be encoded.

tagging 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 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:

pctxt 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.

pElemList 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.

6.9.2.18 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:

pctxt 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.

value 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.

tagging 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 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:

pctxt 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.

object_p Value to be encoded.

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

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

Parameters:

pctxt 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.

object_p 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.

tagging 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 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:

- pctxt* 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.
- tag* 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.

6.9.2.22 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:

- pctxt* 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.
- tag* 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.

6.9.2.23 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:

- pctxt* 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.
- tag* 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.
- length* 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.

6.9.2.24 `int berEncStrmUInt (OSCTXT * pctxt, OSUINT32 value, ASN1TagType tagging)`

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

Parameters:

- pctxt* 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.
- value* An unsigned INTEGER value to be encoded. The OSUINT32 type is set to the C type 'unsigned int' in the `asn1type.h` file. This is assumed to represent a 32-bit integer value.
- tagging* 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 `int berEncStrmUInt16 (OSCTXT * pctxt, OSUINT16 value, ASN1TagType tagging)`

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

Parameters:

- pctxt* 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.
- value* An unsigned 16-bit INTEGER value to be encoded. The OSUINT16 type is set to the C type 'unsigned short' in the `asn1type.h` file. This is assumed to represent a 16-bit integer value.
- tagging* 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 `int berEncStrmUInt64 (OSCTXT * pctxt, OSUINT64 value, ASN1TagType tagging)`

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

Parameters:

- pctxt* 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.
- value* 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.
- tagging* 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.

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

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

Parameters:

- pctxt* 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.
- value* 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.
- tagging* 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.

6.9.2.28 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:

- pctxt* 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.
- object_p* 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.
- tagging* 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.

6.9.2.29 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:

- pctxt* 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.
- octet* The octet to be encoded.

Returns:

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

6.9.2.30 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:

- pctxt* 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.
- poctets* The array of octets to be encoded.
- numocts* The number of octets in the array.

Returns:

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

6.9.2.31 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:

- pctxt* 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.
- pvalue* 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.
- tagging* 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.

6.9.2.32 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:

- pctxt* 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.
- object_p* 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.
- numbits* The number of bits within the bit string to be encoded.
- tagging* 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 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). BMP-String type is based on a 16-bit character sets.

Parameters:

pctxt 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.

object_p 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.

tagging 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 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:

pctxt 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.

object_p A pointer to a null-terminated C character string to be encoded.

tagging 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'.

tag 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.

6.9.2.35 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:

pctxt 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.

object_p A pointer to an OCTET STRING containing the octet data to be encoded.

numocts The number of octets (bytes) within the OCTET STRING to be encoded.

tagging 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 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:

pctxt 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.

object_p Value to be encoded.

tagging 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 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:

pctxt 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.

object_p 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.

tagging 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.

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`, `rtxStreamMemoryCreate`, `rtxStreamMemoryAttach`.

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 `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 `berDecStrmDecimal` (OSCTXT *pctxt, const char **object_p, ASN1TagType tagging, int length)
- int `berDecStrmDynBitStr` (OSCTXT *pctxt, const OSOCTET **ppvalue, OSUINT32 *pnbits, ASN1TagType tagging, int length)
- int `berDecStrmDynOctStr` (OSCTXT *pctxt, const OSOCTET **ppvalue, OSUINT32 *pnocts, ASN1TagType tagging, int length)
- int `berDecStrmEnum` (OSCTXT *pctxt, OSINT32 *object_p, ASN1TagType tagging, int length)
- 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 `berDecStrmMatchEOC` (OSCTXT *pctxt)
- int `berDecStrmMatchTag` (OSCTXT *pctxt, ASN1TAG tag, int *len_p, 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, OSUINT32 *pnumocts)
- int `berDecStrmOpenTypeAppend` (OSCTXT *pctxt, OSRTDList *pElemList)
- int `berDecStrmOpenTypeExt` (OSCTXT *pctxt, ASN1CCB *ccb_p, ASN1TAG tag, OSRTDList *pElemList)
- int `berDecStrmPeekTagAndLen` (OSCTXT *pctxt, ASN1TAG *ptag, int *plen)
- 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)
- OSBOOL `berDecStrmTestEOC` (OSCTXT *pctxt, ASN1CCB *ccb_p)
- 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)

6.10.2 Function Documentation

6.10.2.1 `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:

pctxt 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.

object_p Pointer to a character pointer variable to receive the decoded value. Dynamic memory is allocated for the variable using the `rtxMemAlloc` function. The decoded variable is represented as a hexadecimal string starting with a "0x" prefix.

tagging 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'.

length 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 `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:

pctxt 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.

pvalue 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 *pnbits* input parameter.

pnbits 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.

tagging 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'.

length 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 **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:

pctxt 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.

object_p 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.

tagging 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'.

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

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

Parameters:

pctxt 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.

object_p Pointer to a variable to receive the decoded BOOLEAN value.

tagging 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'.

length 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 **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:

pctxt 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.

ppvalue 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.

tagging 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'.

tag 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.

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

This function decodes a value of the decimal encoded ASN.1 REAL type. Result will be presented in xsd:decimal format.

Parameters:

pctxt 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.

object_p Pointer to a character pointer variable to receive the decoded result. Dynamic memory is allocated for the variable using the rtxMemAlloc function.

tagging 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'.

length 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.

6.10.2.7 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:

pctxt 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.

ppvalue Pointer to a pointer variable to receive the decoded bit string. Dynamic memory is allocated to hold the string.

pnbits Pointer to an integer value to receive the decoded number of bits.

tagging 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'.

length 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 `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:

pctxt 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.

ppvalue Pointer to a pointer variable to receive the decoded octet string. Dynamic memory is allocated to hold the string.

pnocts Pointer to an integer value to receive the decoded number of octets.

tagging 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'.

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

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

Parameters:

pctxt 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.

object_p Pointer to a variable to receive the decoded enumerated value.

tagging 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'.

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

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

Parameters:

pctxt 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.

object_p Pointer to a variable to receive a decoded 32-bit integer value.

tagging 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'.

length 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 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:

pctxt 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.

object_p Pointer to a variable to receive a decoded 16-bit integer value.

tagging 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'.

length 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.

6.10.2.12 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:

pctxt 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.

object_p Pointer to a variable to receive a decoded 64-bit integer value.

tagging 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'.

length 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 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:

pctxt 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.

object_p Pointer to a variable to receive a decoded 8-bit integer value.

tagging 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'.

length 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.

6.10.2.14 int berDecStrmLength (OSCTXT * *pctxt*, int * *len_p*)

This function decodes a BER length determinant value.

Parameters:

pctxt 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.

len_p 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_INDEFLN', which indicates indefinite length.

Returns:

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

6.10.2.15 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:

pctxt 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.

6.10.2.16 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:

pctxt 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.

tag Tag variable to match.

len_p 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.

advance 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 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:

pctxt 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.

6.10.2.18 int berDecStrmNull (OSCTXT * *pctxt*, ASN1TagType *tagging*)

This function decodes an ASN.1 NULL placeholder.

Parameters:

pctxt 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.

tagging 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.

6.10.2.19 int berDecStrmObjId (OSCTXT * *pctxt*, ASN1OBJID * *object_p*, ASN1TagType *tagging*, int *length*)

This function decodes a value of the ASN.1 OBJECT IDENTIFIER type.

Parameters:

pctxt 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.

object_p 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.

tagging 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'.

length 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 int berDecStrmObjId64 (OSCTXT *pctx, ASN1OID64 *object_p, ASN1TagType tagging, int length)

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

Parameters:

pctx 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.

object_p 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.

tagging 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'.

length 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 int berDecStrmOctStr (OSCTXT *pctx, OSOCTET *pvalue, OSUINT32 *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:

pctx 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.

pvalue 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.

pnocts 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.

tagging 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'.

length 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 **int berDecStrmOpenType** (OSCTXT * *pctxt*, const OSOCTET ** *object_p2*, OSUINT32 * *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:

pctxt 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.

object_p2 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.

pnumocts Pointer to an integer value to receive the decoded number of octets.

Returns:

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

6.10.2.23 **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:

pctxt 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.

pElemList 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.

6.10.2.24 **int berDecStrmOpenTypeExt** (OSCTXT * *pctxt*, ASN1CCB * *ccb_p*, ASN1TAG *tag*, 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:

pctxt 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.

ccb_p 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.

tag Next expected tag value (or ASN_K_NOTAG value if last field). The routine will loop through elements until matching tag found or some other error occurs.

pElemList 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.

6.10.2.25 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:

pctxt 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.

ptag Pointer to a variable to receive the decoded ASN.1 tag value.

plen 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_INDFLEN', which indicates indefinite length.

Returns:

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

6.10.2.26 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:

pctxt 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.

object_p Pointer to a variable to receive the decoded real value.

tagging 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'.

length 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.

6.10.2.27 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:

pctxt 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.

object_p Pointer to a character pointer variable to receive the decoded result. Dynamic memory is allocated for the variable using the `rtxMemAlloc` function.

tagging 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'.

length 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.

6.10.2.28 int berDecStrmRelativeOID (OSCTXT * *pctxt*, ASN1OBJID * *object_p*, ASN1TagType *tagging*, int *length*)

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

Parameters:

pctxt 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.

object_p 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.

tagging 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'.

length 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.

6.10.2.29 int berDecStrmTag (OSCTXT * *pctxt*, ASN1TAG * *tag_p*)

This function decodes the tag at the current decode pointer location and returns the results.

Parameters:

pctxt 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.

tag_p 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.

6.10.2.30 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:

pctxt 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.

tag_p Pointer to a variable to receive the decoded ASN.1 tag value.

len_p 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.

6.10.2.31 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:

pctxt 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.

ccb_p 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.

6.10.2.32 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:

pctxt 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.

tag Tag variable to match.

len_p 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.

advance 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.

6.10.2.33 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:

pctxt 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.

object_p Pointer to a variable to receive a decoded unsigned 32-bit integer value.

tagging 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'.

length 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.

6.10.2.34 int berDecStrmUInt16 (OSCTXT * *pctxt*, OSUINT16 * *object_p*, ASN1TagType *tagging*, int *length*)

This function decodes a 16-bit variable of the unsigned variant of ASN.1 INTEGER type.

Parameters:

pctxt 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.

object_p Pointer to a variable to receive a decoded unsigned 16-bit integer value.

tagging 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'.

length 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.

6.10.2.35 int berDecStrmUInt64 (OSCTXT * *pctxt*, OSUINT64 * *object_p*, ASN1TagType *tagging*, int *length*)

This function decodes a 64-bit variable of the unsigned variant of ASN.1 INTEGER type.

Parameters:

pctxt 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.

object_p Pointer to a variable to receive a decoded unsigned 64-bit integer value.

tagging 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'.

length 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.

6.10.2.36 `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:

- pctxt* 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.
- object_p* Pointer to a variable to receive a decoded unsigned 8-bit integer value.
- tagging* 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'.
- length* 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.

6.10.2.37 `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:

- pctxt* 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.
- object_p* 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.
- tagging* 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'.
- length* 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.

6.11.1 Detailed Description

These classes are used to perform BER encoding directly to a stream (file, network, memory).

Classes

- class [ASN1BEREncodeStream](#)

6.12 C++ classes for streaming BER decoding.

6.12.1 Detailed Description

These classes are used to perform BER decoding directly from a stream (file, network, memory).

Classes

- class [ASN1BERDecodeStream](#)

Chapter 7

ASN1C BER Runtime Class Documentation

7.1 `_Asn1BufLocDescr` Struct Reference

```
#include <asn1ber.h>
```

7.1.1 Detailed Description

Buffer location descriptor

Public Attributes

- OSUINT32 **numocts**
- OSINT32 **offset**

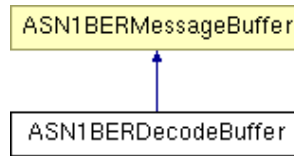
The documentation for this struct was generated from the following file:

- [asn1ber.h](#)

7.2 ASN1BERDecodeBuffer Class Reference

```
#include <asn1BerCppTypes.h>
```

Inheritance diagram for ASN1BERDecodeBuffer::



7.2.1 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.

Public Member Functions

- [ASN1BERDecodeBuffer](#) ()
- [ASN1BERDecodeBuffer](#) (const OSOCTET *pMsgBuf, size_t msgBufLen)
- OSOCTET * [findElement](#) (ASN1TAG tag, OSINT32 &elemLen, OSBOOL firstFlag=TRUE)
- virtual const OSOCTET * [getMsgPtr](#) ()
- int [init](#) ()
- virtual OSBOOL [isA](#) (int bufferType)
- int [parseTagLen](#) (ASN1TAG &tag, int &msglen)
- int [readBinaryFile](#) (const char *filePath)
- int [setBuffer](#) (const OSOCTET *pMsgBuf, size_t 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](#)
- size_t [mMsgBufLen](#)
- OSBOOL [mBufSetFlag](#)

7.2.2 Constructor & Destructor Documentation

7.2.2.1 ASN1BERDecodeBuffer::ASN1BERDecodeBuffer ()

Default constructor. Use the [getStatus\(\)](#) method to determine if an error occurred during initialization or not.

7.2.2.2 ASN1BERDecodeBuffer::ASN1BERDecodeBuffer (const OSOCTET * pMsgBuf, size_t 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:

pMsgBuf A pointer to a buffer containing an encoded ASN.1 message.

msgBufLen 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.

7.2.3 Member Function Documentation

7.2.3.1 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:

tag ASN.1 tag value to search for.

firstFlag 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

elemLen Reference to an integer value to receive the length of the found element.

7.2.3.2 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.

7.2.3.3 int ASN1BERDecodeBuffer::init ()

Initializes message buffer.

Returns:

Completion status of operation:

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

7.2.3.4 virtual OSBOOL ASN1BERDecodeBuffer::isA (int *bufferType*) [inline, virtual]

This method checks the type of the message buffer.

Parameters:

bufferType 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.

7.2.3.5 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.

7.2.3.6 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:

tag Reference to a tag structure to receive the outer level tag value parsed from the message.

msglen 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.

7.2.3.7 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:

filePath The zero-terminated string containing the path to the file.

Returns:

Completion status of operation:

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

7.2.3.8 `int ASN1BERDecodeBuffer::setBuffer (const OSOCTET * pMsgBuf, size_t msgBufLen)`

This method sets a buffer containing a message to be decoded.

Parameters:

pMsgBuf A pointer to a memory buffer containing a message to be decoded. The buffer should be declared as an array of unsigned characters (OSOCTETs).

msgBufLen 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:

- [asn1BerCppType.h](#)

7.3 ASN1BERDecodeStream Class Reference

```
#include <ASN1BERDecodeStream.h>
```

7.3.1 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.

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 OSBOOL [isOpen](#) ()
- 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 [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 `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 `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)
- OSBOOL `isA` (int bufferType)
- int `peekTagAndLen` (ASN1TAG &tag, int &len)

Protected Attributes

- OSRTInputStreamIF * `mpStream`
- OSBOOL `mbOwnStream`

7.3.2 Constructor & Destructor Documentation

7.3.2.1 ASN1BERDecodeStream::ASN1BERDecodeStream (OSRTInputStreamIF & *is*)

A default constructor. Use `getStatus()` method to determine has error occurred during the initialization or not.

7.3.3 Member Function Documentation

7.3.3.1 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.

7.3.3.2 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:

ccb 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.

7.3.3.3 `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:

`rtxStreamClose`

7.3.3.4 `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.

7.3.3.5 `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:

pval Reference to a pointer to a variable to receive a decoded big integer value.

tagging 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'.

length 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](#)

7.3.3.6 `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:

val Reference to an ASN1DynBitStr variable to receive the decoded bit string.

tagging 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'.

length 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](#)

7.3.3.7 int ASN1BERDecodeStream::decodeBitStr (OSOCKET * *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:

pbits 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.

numbits 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.

tagging 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'.

length 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](#)

7.3.3.8 int ASN1BERDecodeStream::decodeBMPStr (Asn116BitCharString & *val*, ASN1TagType *tagging* = ASN1EXPL, int *length* = 0)

This method decodes a variable of the ASN.1 BMPString type.

Parameters:

val Reference to a variable to receive the decoded BMPString value.

tagging 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'.

length 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](#)

7.3.3.9 int ASN1BERDecodeStream::decodeBool (OSBOOL & val, ASN1TagType tagging = ASN1EXPL, int length = 0)

This method decodes a variable of the ASN.1 BOOLEAN type.

Parameters:

val Reference to a variable to receive the decoded boolean value.

tagging 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'.

length 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](#)

7.3.3.10 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:

pval 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 rtxMemAlloc function.

tagging 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'.

tag 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.

length 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](#)

7.3.3.11 int ASN1BERDecodeStream::decodeEnum (OSINT32 & val, ASN1TagType tagging = ASN1EXPL, int length = 0)

This method decodes a variable of the ASN.1 ENUMERATED type.

Parameters:

val Reference to a variable to receive the decoded enumerated value.

tagging 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'.

length 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](#)

7.3.3.12 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](#)

7.3.3.13 `int ASN1BERDecodeStream::decodeInt (OSINT32 & val, ASN1TagType tagging = ASN1EXPL, int length = 0)`

This method decodes a variable of the ASN.1 INTEGER type.

Parameters:

val Reference to a variable to receive a decoded 32-bit integer value.

tagging 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'.

length 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:

[berDecStrmInt](#)

7.3.3.14 `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:

val Reference to a variable to receive a decoded 16-bit integer value.

tagging 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'.

length 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](#)

7.3.3.15 `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:

val Reference to a variable to receive a decoded 64-bit integer value.

tagging 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'.

length 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](#)

7.3.3.16 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:

val Reference to a variable to receive a decoded 8-bit integer value.

tagging 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'.

length 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](#)

7.3.3.17 int ASN1BERDecodeStream::decodeLength (OSINT32 & length)

This method decodes a BER length determinant value.

Parameters:

length 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](#)

7.3.3.18 `int ASN1BERDecodeStream::decodeNull (ASN1TagType tagging = ASN1EXPL)`

This method decodes a variable of the ASN.1 NULL type.

Parameters:

tagging 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](#)

7.3.3.19 `int ASN1BERDecodeStream::decodeObj (ASN1CType & val)`

This method decodes an ASN.1 constructed object from the stream.

Parameters:

val A reference to an object to be decoded.

Returns:

Completion status of operation:

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

7.3.3.20 `int ASN1BERDecodeStream::decodeObjId (ASN1OBJID & val, ASN1TagType tagging = ASN1EXPL, int length = 0)`

This method decodes a variable of the ASN.1 OBJECT IDENTIFIER type.

Parameters:

val Reference to a variable to receive the decoded OBJECT IDENTIFIER value.

tagging 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'.

length 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](#)

7.3.3.21 `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:

val Reference to a variable to receive the decoded 64-bit OBJECT IDENTIFIER value.

tagging 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'.

length 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](#)

7.3.3.22 `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:

val Reference to an ASN1DynOctStr variable to receive the decoded octet string.

tagging 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'.

length 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](#)

7.3.3.23 `int ASN1BERDecodeStream::decodeOctStr (OSOCKET * popts, 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:

pocts 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.

numocts 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.

tagging 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'.

length 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](#)

7.3.3.24 int ASN1BERDecodeStream::decodeReal (OSREAL & val, ASN1TagType tagging = ASN1EXPL, int length = 0)

This method decodes a variable of the ASN.1 REAL type.

Parameters:

val Reference to a variable to receive the decoded REAL value.

tagging 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'.

length 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](#)

7.3.3.25 int ASN1BERDecodeStream::decodeRelativeOID (ASN1OBJID & val, ASN1TagType tagging = ASN1EXPL, int length = 0)

This method decodes a variable of the ASN.1 RELATIVE-OID type.

Parameters:

val Reference to a variable to receive the decoded RELATIVE-OID value.

tagging 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'.

length 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](#)

7.3.3.26 int ASN1BERDecodeStream::decodeTag (ASN1TAG & tag)

This method decodes the tag at the current decode pointer location and returns the results.

Parameters:

tag 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](#)

7.3.3.27 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:

tag Reference to a variable to receive the decoded ASN.1 tag value.

len 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](#)

7.3.3.28 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:

val Reference to a variable to receive a decoded unsigned 32-bit integer value.

tagging 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'.

length 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](#)

7.3.3.29 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:

val Reference to a variable to receive a decoded unsigned 16-bit integer value.

tagging 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'.

length 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](#)

7.3.3.30 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:

val Reference to a variable to receive a decoded unsigned 64-bit integer value.

tagging 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'.

length 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](#)

7.3.3.31 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:

val Reference to a variable to receive a decoded unsigned 8-bit integer value.

tagging 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'.

length 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](#)

7.3.3.32 int ASN1BERDecodeStream::decodeUnivStr (Asn132BitCharString & val, ASN1TagType tagging = ASN1EXPL, int length = 0)

This method decodes a variable of the ASN.1 UniversalString type.

Parameters:

val Reference to a variable to receive the decoded UniversalString value.

tagging 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'.

length 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](#)

7.3.3.33 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:

[rtxStreamFlush](#)

7.3.3.34 virtual void* ASN1BERDecodeStream::getAppInfo () [inline, virtual]

Returns a pointer to application-specific information block

7.3.3.35 virtual OSRTCtxtPtr ASN1BERDecodeStream::getContext () [inline, virtual]

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

Returns:

Context smart pointer object.

7.3.3.36 virtual OSCTXT* ASN1BERDecodeStream::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.

7.3.3.37 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:

pBuf A pointer to a destination buffer to obtain the error text. If NULL, dynamic buffer will be allocated.
bufSize A reference to buffer size. If pBuf is NULL it will receive the size of allocated dynamic buffer.

Returns:

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

7.3.3.38 virtual char* ASN1BERDecodeStream::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.

7.3.3.39 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.

7.3.3.40 OSBOOL ASN1BERDecodeStream::isA (int bufferType)

This method matches an enumerated identifier defined in the base class. One identifier is declared for each of the derived classes.

Parameters:

bufferType 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.

7.3.3.41 virtual OSBOOL ASN1BERDecodeStream::isOpened () [inline, virtual]

Checks, is the stream opened or not.

Returns:

TRUE, if the stream is opened, FALSE otherwise.

See also:

rtxStreamIsOpened

7.3.3.42 `int ASN1BERDecodeStream::mark (size_t readAheadLimit)` [inline]

This method marks the current position in this input stream. A subsequent call to the `ASN1BERDecodeStream::reset` function repositions this stream at the last marked position so that subsequent reads re-read the same bytes. The `readAheadLimit` argument tells this input stream to allow that many bytes to be read before the mark position gets invalidated.

Parameters:

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

Returns:

Completion status of operation:

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

See also:

`rtxStreamBufMark`, `rtxStreamBufReset`

7.3.3.43 `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`

7.3.3.44 `ASN1BERDecodeStream& ASN1BERDecodeStream::operator>> (ASN1CType & val)`

Decodes an ASN.1 constructed object from the stream. Use `getStatus()` method to determine has error occurred during the operation or not.

Parameters:

val A reference to an object to be decoded.

Returns:

reference to this class to perform sequential decoding.

7.3.3.45 `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:

tag Reference to a variable to receive the decoded ASN.1 tag value.

len 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](#)

7.3.3.46 virtual void ASN1BERDecodeStream::printErrorInfo () [inline, virtual]

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

7.3.3.47 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:

pDestBuf Pointer to a buffer to receive a data.

maxToRead Size of the buffer.

Returns:

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

See also:

[rtxStreamRead](#)

7.3.3.48 virtual long ASN1BERDecodeStream::readBlocking (OSOCKET * *pDestBuf*, size_t *toReadBytes*) [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:

pDestBuf Pointer to a buffer to receive a data.

toReadBytes Number of bytes to be read.

Returns:

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

See also:

[rtxStreamRead](#)

7.3.3.49 `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`

7.3.3.50 `virtual void ASN1BERDecodeStream::resetErrorInfo ()` [inline, virtual]

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

7.3.3.51 `virtual void ASN1BERDecodeStream::setAppInfo (void * pAppInfo)` [inline, virtual]

Sets the application-specific information block.

7.3.3.52 `virtual void ASN1BERDecodeStream::setDiag (OSBOOL value = TRUE)` [inline, virtual]

The `setDiag` method will turn diagnostic tracing on or off.

Parameters:

value - Boolean value (default = TRUE = on)

7.3.3.53 `virtual int ASN1BERDecodeStream::skip (size_t n)` [inline, virtual]

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

Parameters:

n The number of octets to be skipped.

Returns:

Completion status of operation:

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

See also:

`rtxStreamSkip`

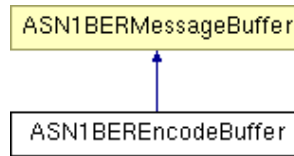
The documentation for this class was generated from the following file:

- [ASN1BERDecodeStream.h](#)

7.4 ASN1BEREncodeBuffer Class Reference

```
#include <asn1BerCppTypes.h>
```

Inheritance diagram for ASN1BEREncodeBuffer::



7.4.1 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.

Public Member Functions

- [ASN1BEREncodeBuffer](#) ()
- [ASN1BEREncodeBuffer](#) (OSOCKET *pMsgBuf, size_t msgBufLen)
- virtual OSOCKET * [getMsgCopy](#) ()
- virtual const OSOCKET * [getMsgPtr](#) ()
- int [init](#) ()
- virtual OSOOL [isA](#) (int bufferType)
- int [setBuffer](#) (OSOCKET *pMsgBuf, size_t msgBufLen)
- [ASN1BEREncodeBuffer](#) & [operator<<](#) (ASN1CType &val)

7.4.2 Constructor & Destructor Documentation

7.4.2.1 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.

7.4.2.2 ASN1BEREncodeBuffer::ASN1BEREncodeBuffer (OSOCKET *pMsgBuf, size_t 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:

- pMsgBuf* A pointer to a fixed size message buffer to receive the encoded message.
- msgBufLen* Size of the fixed-size message buffer.

7.4.3 Member Function Documentation

7.4.3.1 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;).

7.4.3.2 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.

7.4.3.3 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.

7.4.3.4 virtual OSBOOL ASN1BEREncodeBuffer::isA (int *bufferType*) [inline, virtual]

This method checks the type of the message buffer.

Parameters:

bufferType 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.

7.4.3.5 [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.

7.4.3.6 `int ASN1BEREncodeBuffer::setBuffer (OSOCTET * pMsgBuf, size_t msgBufLen)`

This method sets a buffer to receive the encoded message.

Parameters:

pMsgBuf A pointer to a memory buffer to use to encode a message. The buffer should be declared as an array of unsigned characters ([OSOCTET](#)s). This parameter can be set to `NULL` to specify dynamic encoding (i.e., the encode functions will dynamically allocate a buffer for the message).

msgBufLen 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:

- [asn1BerCppType.h](#)

7.5 ASN1BEREncodeStream Class Reference

```
#include <ASN1BEREncodeStream.h>
```

7.5.1 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.

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 Asn16BitCharString &val, ASN1TagType tagging=ASN1EXPL)
- int [encodeBigInt](#) (const char *pval, 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 [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` (int bufferType)

Protected Attributes

- OSRTOutputStreamIF * `mpStream`
- OSBOOL `mbOwnStream`

7.5.2 Constructor & Destructor Documentation

7.5.2.1 ASN1BEREncodeStream::ASN1BEREncodeStream (OSRTOutputStreamIF & os)

A default constructor. Use `getStatus()` method to determine has error occurred during the initialization or not.

7.5.3 Member Function Documentation

7.5.3.1 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`

7.5.3.2 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:

pval A pointer to a character string containing the value to be encoded.

tagging 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](#)

7.5.3.3 int ASN1BEREncodeStream::encodeBitStr (const ASN1DynBitStr & *val*, ASN1TagType *tagging* = ASN1EXPL)

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

Parameters:

val A reference to the ASN1DynBitStr structure containing a bit data and number of bits to be encoded.

tagging 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](#)

7.5.3.4 int ASN1BEREncodeStream::encodeBitStr (const OSOCTET * *pbits*, OSUINT32 *numbits*, ASN1TagType *tagging* = ASN1EXPL)

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

Parameters:

pbits 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.

numbits The number of bits within the bit string to be encoded.

tagging 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](#)

7.5.3.5 `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:

- val* 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.
- tagging* 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](#)

7.5.3.6 `int ASN1BEREncodeStream::encodeBool (OSBOOL val, ASN1TagType tagging = ASN1EXPL)`

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

Parameters:

- val* 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.
- tagging* 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](#)

7.5.3.7 `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:

- pval* A pointer to a null-terminated C character string to be encoded.
- tagging* 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'.

tag 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](#)

7.5.3.8 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:

val An integer containing the enumerated value to be encoded.

tagging 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](#)

7.5.3.9 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](#),

7.5.3.10 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](#)

7.5.3.11 int ASN1BEREncodeStream::encodeInt (OSINT32 val, ASN1TagType tagging = ASN1EXPL)

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

Parameters:

val A 32-bit INTEGER value to be encoded.

tagging 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](#)

7.5.3.12 int ASN1BEREncodeStream::encodeInt16 (OSINT16 val, ASN1TagType tagging = ASN1EXPL)

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

Parameters:

val A 16-bit INTEGER value to be encoded.

tagging 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](#)

7.5.3.13 `int ASN1BEREncodeStream::encodeInt64 (OSINT64 val, ASN1TagType tagging = ASN1EXPL)`

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

Parameters:

val A 64-bit INTEGER value to be encoded.

tagging 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](#)

7.5.3.14 `int ASN1BEREncodeStream::encodeInt8 (OSINT8 val, ASN1TagType tagging = ASN1EXPL)`

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

Parameters:

val An 8-bit INTEGER value to be encoded.

tagging 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](#)

7.5.3.15 `int ASN1BEREncodeStream::encodeNull (ASN1TagType tagging = ASN1EXPL)`

This method encodes a variable of the ASN.1 NULL type.

Parameters:

tagging 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](#)

7.5.3.16 `int ASN1BEREncodeStream::encodeObj (ASN1CType & val)`

This method encodes an ASN.1 constructed object to the stream.

Parameters:

val A reference to an object to be encoded.

Returns:

Completion status of operation:

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

7.5.3.17 `int ASN1BEREncodeStream::encodeObjId (const ASN1OBJID & val, ASN1TagType tagging = ASN1EXPL)`

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

Parameters:

val 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.

tagging 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](#)

7.5.3.18 `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:

val 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.

tagging 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](#)

7.5.3.19 `int ASN1BEREncodeStream::encodeOctStr (const ASN1DynOctStr & val, ASN1TagType tagging = ASN1EXPL)`

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

Parameters:

val A reference to the ASN1DynOctStr structure containing an octet data and number of octets to be encoded.

tagging 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](#)

7.5.3.20 `int ASN1BEREncodeStream::encodeOctStr (const OSOCTET * pocts, OSUINT32 numocts, ASN1TagType tagging = ASN1EXPL)`

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

Parameters:

pocets A pointer to an OCTET STRING containing the octet data to be encoded.

numocts The number of octets (bytes) within the OCTET STRING to be encoded.

tagging 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](#)

7.5.3.21 `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:

val 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.

tagging 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](#)

7.5.3.22 int ASN1BEREncodeStream::encodeRelativeOID (const ASN1OBJID & val, ASN1TagType tagging = ASN1EXPL)

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

Parameters:

val 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.

tagging 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](#)

7.5.3.23 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:

tag 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](#)

7.5.3.24 `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:

tag 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](#)

7.5.3.25 `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:

tag 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.

len 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](#)

7.5.3.26 `int ASN1BEREncodeStream::encodeUInt (OSUINT32 val, ASN1TagType tagging = ASN1EXPL)`

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

Parameters:

val An unsigned INTEGER value to be encoded.

tagging 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](#)

7.5.3.27 `int ASN1BEREncodeStream::encodeUInt16 (OSUINT16 val, ASN1TagType tagging = ASN1EXPL)`

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

Parameters:

val A 16-bit unsigned INTEGER value to be encoded.

tagging 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](#)

7.5.3.28 `int ASN1BEREncodeStream::encodeUInt64 (OSUINT64 val, ASN1TagType tagging = ASN1EXPL)`

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

Parameters:

val A 64-bit unsigned INTEGER value to be encoded.

tagging 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](#)

7.5.3.29 `int ASN1BEREncodeStream::encodeUInt8 (OSUINT8 val, ASN1TagType tagging = ASN1EXPL)`

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

Parameters:

val An 8-bit unsigned INTEGER value to be encoded.

tagging 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](#)

7.5.3.30 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:

val 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.

tagging 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](#)

7.5.3.31 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](#)

7.5.3.32 virtual void* ASN1BEREncodeStream::getAppInfo () [inline, virtual]

Returns a pointer to application-specific information block

7.5.3.33 virtual OSRTCtxtPtr ASN1BEREncodeStream::getContext () [inline, virtual]

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

Returns:

Context smart pointer object.

7.5.3.34 `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.

7.5.3.35 `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:

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

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

Returns:

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

7.5.3.36 `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.

7.5.3.37 `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.

7.5.3.38 OSBOOL ASN1BEREncodeStream::isA (int *bufferType*)

This method matches an enumerated identifier defined in the base class. One identifier is declared for each of the derived classes.

Parameters:

bufferType 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.

7.5.3.39 virtual OSBOOL ASN1BEREncodeStream::isOpen () [inline, virtual]

Checks, is the stream opened or not.

Returns:

TRUE, if the stream is opened, FALSE otherwise.

See also:

rtxStreamIsOpened

7.5.3.40 ASN1BEREncodeStream& ASN1BEREncodeStream::operator<< (ASN1CType & *val*)

Encodes an ASN.1 constructed object to the stream. Use [getStatus\(\)](#) method to determine has error occurred during the operation or not.

Parameters:

val A reference to an object to be encoded.

Returns:

reference to this class to perform sequential encoding.

7.5.3.41 virtual void ASN1BEREncodeStream::printErrorInfo () [inline, virtual]

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

7.5.3.42 virtual void ASN1BEREncodeStream::resetErrorInfo () [inline, virtual]

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

7.5.3.43 virtual void ASN1BEREncodeStream::setAppInfo (void * *pAppInfo*) [inline, virtual]

Sets the application-specific information block.

7.5.3.44 `virtual void ASN1BEREncodeStream::setDiag (OSBOOL value = TRUE) [inline, virtual]`

The setDiag method will turn diagnostic tracing on or off.

Parameters:

value - Boolean value (default = TRUE = on)

7.5.3.45 `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:

pdata Pointer to the data to be written.

size The number of octets to write.

Returns:

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

See also:

rtxStreamWrite

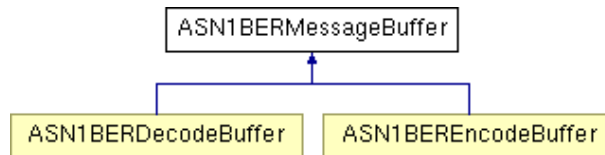
The documentation for this class was generated from the following file:

- [ASN1BEREncodeStream.h](#)

7.6 ASN1BERMessageBuffer Class Reference

```
#include <asn1BerCppTypes.h>
```

Inheritance diagram for ASN1BERMessageBuffer::



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.

Public Member Functions

- int [calcIndefLen](#) (OSOCKET *buf_p, int bufSize=INT_MAX)
- void [binDump](#) ()
- void [hexDump](#) (int numocts)
- int [CalcIndefLen](#) (OSOCKET *buf_p)
- void [BinDump](#) ()
- void [HexDump](#) (int numocts)

Protected Member Functions

- [ASN1BERMessageBuffer](#) (Type *bufferType*)

7.6.2 Constructor & Destructor Documentation

7.6.2.1 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:

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

7.6.3 Member Function Documentation

7.6.3.1 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.

7.6.3.2 int ASN1BERMessageBuffer::calcIndefLen (OSOCKET * buf_p, int bufSize = INT_MAX)
[inline]

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

Calling Sequence:

```
len=messageBuffer.calcIndefLen (buf_p);
```

where messageBuffer is an [ASN1BERMessageBuffer](#) derived class object.

Parameters:

buf_p A pointer to a message component encoded using indefinite length encoding.

bufSize Size of the buffer buf_p (in octets).

Returns:

Length, in octets, of message component, as int.

7.6.3.3 void ASN1BERMessageBuffer::hexDump (int 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

ASN1C BER Runtime File Documentation

8.1 asn1ber.h File Reference

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.

```
#include "rtsrc/asn1type.h"
```

Classes

- struct [_Asn1BufLocDescr](#)

Defines

- #define **ASN_K_INDEFLEN** -9999
- #define **XU_DUMP**(msg) xu_dump(msg,0,0)
- #define **xd_resetp**(pctx) rtxResetContext(pctx)
- #define **ASN1TAG2BYTE**(tag) ((OSOCTET)(((tag)&TM_B_IDCODE)|((tag)>>ASN1TAG_LSHIFT)))
- #define **xd_utf8str**(pctx, object_p, tagging, length) xd_charstr (pctx, (const char**)object_p, tagging, ASN_ID_UTF8String, length)
- #define **xd_indeflen**(m) xd_indeflen_ex(m, INT_MAX)
- #define **xe_utf8str**(pctx, object_p, tagging) xe_charstr (pctx, (const char*)object_p, tagging, ASN_ID_UTF8String)
- #define **xu_addTagErrParm** berErrAddTagParm
- #define **XD_MEMCPY1**(pctx, object_p)
- #define **XD_FETCH1**(pctx) ((pctx) → buffer.data[(pctx) → buffer.byteIndex++])
- #define **XD_PEEKTAG**(pctx, tag) (((pctx) → buffer.data[(pctx) → buffer.byteIndex] & (~0x20)) == (tag & (~0x20)))
- #define **XD_PEEKPC**(pctx) (((pctx) → buffer.data[(pctx) → buffer.byteIndex] & 0x20) == 0x20)
- #define **XD_MATCHEOC**(pctx)
- #define **XD_MATCHBYTES1**(pctx, b1) ((pctx) → buffer.data[(pctx) → buffer.byteIndex] == b1)
- #define **XD_MATCHBYTES2**(pctx, b1, b2)
- #define **XD_MATCHBYTES3**(pctx, b1, b2, b3)
- #define **XD_MATCHBYTES4**(pctx, b1, b2, b3, b4)

- #define **XD_MATCHBYTES5**(pctxt, b1, b2, b3, b4, b5)
- #define **XD_BUMPIDX**(pctxt, nbytes) ((pctxt) → buffer.byteIndex += nbytes)
- #define **XD_CHKBUFEND**(pctxt)
- #define **XD_CHKDEFLEN**(pctxt, len)
- #define **XD_CHKEOB**(pctxt)
- #define **XD_CHKEND**(pctxt, ccb_p)
- #define **XE_CHKBUF**(pctxt, len)
- #define **XE_PUT1**(pctxt, ch) (pctxt) → buffer.data[-(pctxt) → buffer.byteIndex] = ch;
- #define **XE_PUT2**(pctxt, ch1, ch2)
- #define **XE_SAFEPUT1**(pctxt, ch) XE_CHKBUF(pctxt,1); (pctxt) → buffer.data[-(pctxt) → buffer.byteIndex] = ch;

Typedefs

- typedef [_Asn1BufLocDescr](#) [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_match](#) (OSCTXT *pctxt, ASN1TAG tag, int *len_p, 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_bitstr](#) (OSCTXT *pctxt, const OSOCTET **object_p2, OSUINT32 *numbits_p, ASN1TagType tagging, int length)
- int [xd_octstr_s](#) (OSCTXT *pctxt, OSOCTET *object_p, OSUINT32 *pnumocts, ASN1TagType tagging, int length)
- int [xd_octstr](#) (OSCTXT *pctxt, const OSOCTET **object_p2, OSUINT32 *pnumocts, ASN1TagType tagging, int length)
- int [xd_charstr](#) (OSCTXT *pctxt, const char **object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int [xd_16BitCharStr](#) (OSCTXT *pctxt, Asn116BitCharString *object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int [xd_32BitCharStr](#) (OSCTXT *pctxt, Asn132BitCharString *object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- 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_OpenType](#) (OSCTXT *pctxt, const OSOCTET **object_p2, OSUINT32 *pnumocts)
- int [xd_OpenTypeExt](#) (OSCTXT *pctxt, ASN1CCB *ccb_p, ASN1TAG tag, OSRTDList *pElemList)
- int [xd_OpenTypeAppend](#) (OSCTXT *pctxt, OSRTDList *pElemList)
- int [xd_real10](#) (OSCTXT *pctxt, const char **object_p, ASN1TagType tagging, int length)
- int [xd_decimal](#) (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_indeflen_ex](#) (const OSOCTET *msg_p, int bufSize)
- int [xd_len](#) (OSCTXT *pctxt, int *len_p)
- int [xd_chkend](#) (OSCTXT *pctxt, ASN1CCB *ccb_p)
- int [xd_count](#) (OSCTXT *pctxt, int length, int *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 [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_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_bitstr](#) (OSCTXT *pctxt, const OSOCTET *object_p, OSUINT32 numbits, ASN1TagType tagging)
- int [xe_octstr](#) (OSCTXT *pctxt, const OSOCTET *object_p, OSUINT32 numocts, ASN1TagType tagging)
- int [xe_charstr](#) (OSCTXT *pctxt, const char *object_p, ASN1TagType tagging, ASN1TAG tag)
- int [xe_16BitCharStr](#) (OSCTXT *pctxt, Asn16BitCharString *object_p, ASN1TagType tagging, ASN1TAG tag)
- int [xe_32BitCharStr](#) (OSCTXT *pctxt, Asn132BitCharString *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_real](#) (OSCTXT *pctxt, OSREAL *object_p, ASN1TagType tagging)
- int [xe_OpenType](#) (OSCTXT *pctxt, const OSOCTET *object_p, OSUINT32 numocts)
- int [xe_OpenTypeExt](#) (OSCTXT *pctxt, OSRTDList *pElemList)
- 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, int bufsiz)
- OSOCTET * [xe_getp](#) (OSCTXT *pctxt)
- void [xe_free](#) (OSCTXT *pctxt)
- int [xe_expandBuffer](#) (OSCTXT *pctxt, size_t length)

- int [xe_memcpy](#) (OSCTXT *pctx, const OSOCTET *object_p, size_t length)
- int [xe_len](#) (OSCTXT *pctx, int length)
- int [xe_derCanonicalSort](#) (OSCTXT *pctx, OSRTSList *pList)
- int [xe_TagAndIndefLen](#) (OSCTXT *pctx, ASN1TAG tag, int length)
- void [xe_getBufLocDescr](#) (OSCTXT *pctx, OSUINT32 length, [Asn1BufLocDescr](#) *pDescr)
- OSBOOL [berErrAddTagParm](#) (OSCTXT *pctx, ASN1TAG tag)
- int [berErrUnexpTag](#) (OSCTXT *pctx, ASN1TAG exptag)
- const char * [berTagToString](#) (ASN1TAG tag, char *buffer, size_t bufsiz)
- const char * [berTagToDynStr](#) (OSCTXT *pctx, ASN1TAG tag)
- 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 *pctx, ASN1SeqOf *seqOf_p, int recSize, int recCount)
- void [xu_octcpy_s](#) (OSUINT32 *nocts_p, OSOCTET *data_p, char *cstr, char zterm)
- void [xu_octcpy_ss](#) (ASN1OctStr *octStr_p, char *cstring, char zterm)
- void [xu_octcpy_d](#) (OSCTXT *pctx, OSUINT32 *nocts_p, const OSOCTET **data_p2, char *cstring, char zterm)
- void [xu_octcpy_ds](#) (OSCTXT *pctx, ASN1DynOctStr *octStr_p, char *cstring, char zterm)
- void [xu_octmcpy_s](#) (ASN1OctStr *octStr_p, void *data_p, int datalen)
- void [xu_octmcpy_d](#) (OSCTXT *pctx, ASN1DynOctStr *octStr_p, void *data_p, int datalen)
- char * [xu_fetchstr](#) (int numocts, char *data)
- int [xu_hexstcpy](#) (char *data, char *hstring)
- int [xu_binstcpy](#) (char *data, char *bstring)
- int [xu_dump](#) (OSOCTET *msgptr, ASN1DumpCbFunc cb, void *cbArg_p)
- int [xu_fdump](#) (FILE *file_p, OSOCTET *msgptr)
- void [xu_hex_dump](#) (OSOCTET *data, int numocts, OSBOOL hdrflg)
- void [xu_fmt_tag](#) (ASN1TAG *tag_p, char *class_p, char *form_p, char *id_code)
- char * [xu_fmt_contents](#) (OSCTXT *pctx, 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)

8.2 `asn1BerCppType.h` File Reference

8.2.1 Detailed Description

BER/DER/CER C++ type and class definitions.

```
#include "rtsrc/asn1CppType.h"  
#include "rtbersrc/asn1ber.h"  
#include "rtxsrc/rtxPrint.h"
```

Classes

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

8.3 ASN1BERDecodeStream.h File Reference

8.3.1 Detailed Description

The C++ definitions for ASN.1 BER input streams.

```
#include "rtsrc/asn1CppType.h"  
#include "rtxsrc/OSRTInputStreamIF.h"  
#include "rtbersrc/asn1berStream.h"
```

Classes

- class [ASN1BERDecodeStream](#)

8.4 ASN1BEREncodeStream.h File Reference

8.4.1 Detailed Description

The C++ definitions for ASN.1 BER output streams.

```
#include "rtsrc/asn1CppType.h"  
#include "rtxsrc/OSRTOutputStreamIF.h"  
#include "rtbersrc/asn1berStream.h"
```

Classes

- class [ASN1BEREncodeStream](#)

8.5 `asn1berSocket.h` File Reference

8.5.1 Detailed Description

```
#include "rtbersrc/asn1ber.h"
```

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

Functions

- int [berReadMsgFromSocket](#) (OSCTXT *pctx, OSRTSOCKET socket, OSOCTET *buffer, int bufsiz, OSOCTET **ppDestBuffer, int *pMessageSize)

8.6 asn1berStream.h File Reference

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.

```
#include "rtbersrc/asn1ber.h"
#include "rtxsrc/rtxStream.h"
```

Defines

- #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 [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 [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, OSUINT32 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 [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 *pobjectets, 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 [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 [berDecStrmDecimal](#) (OSCTXT *pctxt, const char **object_p, ASN1TagType tagging, int length)
- int [berDecStrmDynBitStr](#) (OSCTXT *pctxt, const OSOCTET **ppvalue, OSUINT32 *pnbits, ASN1TagType tagging, int length)
- int [berDecStrmDynOctStr](#) (OSCTXT *pctxt, const OSOCTET **ppvalue, OSUINT32 *pnocts, ASN1TagType tagging, int length)
- int [berDecStrmEnum](#) (OSCTXT *pctxt, OSINT32 *object_p, ASN1TagType tagging, int length)
- 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 [berDecStrmMatchEOC](#) (OSCTXT *pctxt)
- int [berDecStrmMatchTag](#) (OSCTXT *pctxt, ASN1TAG tag, int *len_p, 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, OSUINT32 *pnumocts)
- int [berDecStrmOpenTypeAppend](#) (OSCTXT *pctxt, OSRTDList *pElemList)
- int [berDecStrmOpenTypeExt](#) (OSCTXT *pctxt, ASN1CCB *ccb_p, ASN1TAG tag, OSRTDList *pElemList)
- int [berDecStrmPeekTagAndLen](#) (OSCTXT *pctxt, ASN1TAG *ptag, int *plen)
- 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)
- OSBOOL [berDecStrmTestEOC](#) (OSCTXT *pctxt, ASN1CCB *ccb_p)
- 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 *pctx, OSUINT64 *object_p, ASN1TagType tagging, int length)
- int **berDecStrmUnivStr** (OSCTXT *pctx, Asn132BitCharString *object_p, ASN1TagType tagging, int length)
- int **cerEncCanonicalSort** (OSCTXT *pctx, OSCTXT *pMemCtx, OSRTSList *pList)
- void **cerGetBufLocDescr** (OSCTXT *pctx, [Asn1BufLocDescr](#) *pDescr)
- void **cerAddBufLocDescr** (OSCTXT *pctx, OSRTSList *pElemList, [Asn1BufLocDescr](#) *pDescr)

Index

- [_Asn1BufLocDescr](#), 84
- [asn1ber.h](#), 129
- [asn1BerCppType.h](#), 133
- [ASN1BERDecodeBuffer](#), 85
 - [ASN1BERDecodeBuffer](#), 85
- [ASN1BERDecodeBuffer](#)
 - [ASN1BERDecodeBuffer](#), 85
 - [findElement](#), 86
 - [getMsgPtr](#), 86
 - [init](#), 86
 - [isA](#), 86
 - [operator>>](#), 87
 - [parseTagLen](#), 87
 - [readBinaryFile](#), 87
 - [setBuffer](#), 87
- [ASN1BERDecodeStream](#), 89
 - [ASN1BERDecodeStream](#), 90
- [ASN1BERDecodeStream](#)
 - [ASN1BERDecodeStream](#), 90
 - [byteIndex](#), 90
 - [chkend](#), 90
 - [close](#), 90
 - [currentPos](#), 91
 - [decodeBigInt](#), 91
 - [decodeBitStr](#), 91, 92
 - [decodeBMPStr](#), 92
 - [decodeBool](#), 93
 - [decodeCharStr](#), 93
 - [decodeEnum](#), 94
 - [decodeEoc](#), 94
 - [decodeInt](#), 94
 - [decodeInt16](#), 95
 - [decodeInt64](#), 95
 - [decodeInt8](#), 96
 - [decodeLength](#), 96
 - [decodeNull](#), 96
 - [decodeObj](#), 97
 - [decodeObjId](#), 97
 - [decodeObjId64](#), 97
 - [decodeOctStr](#), 98
 - [decodeReal](#), 99
 - [decodeRelativeOID](#), 99
 - [decodeTag](#), 100
 - [decodeTagAndLen](#), 100
 - [decodeUInt](#), 100
 - [decodeUInt16](#), 101
 - [decodeUInt64](#), 101
 - [decodeUInt8](#), 102
 - [decodeUnivStr](#), 102
 - [flush](#), 103
 - [getAppInfo](#), 103
 - [getContext](#), 103
 - [getCtxtPtr](#), 103
 - [getErrorInfo](#), 103, 104
 - [getStatus](#), 104
 - [isA](#), 104
 - [isOpened](#), 104
 - [mark](#), 104
 - [markSupported](#), 105
 - [operator>>](#), 105
 - [peekTagAndLen](#), 105
 - [printErrorInfo](#), 106
 - [read](#), 106
 - [readBlocking](#), 106
 - [reset](#), 106
 - [resetErrorInfo](#), 107
 - [setAppInfo](#), 107
 - [setDiag](#), 107
 - [skip](#), 107
- [ASN1BERDecodeStream.h](#), 134
- [ASN1BEREncodeBuffer](#), 108
 - [ASN1BEREncodeBuffer](#), 108
- [ASN1BEREncodeBuffer](#)
 - [ASN1BEREncodeBuffer](#), 108
 - [getMsgCopy](#), 109
 - [getMsgPtr](#), 109
 - [init](#), 109
 - [isA](#), 109
 - [operator<<](#), 109
 - [setBuffer](#), 110
- [ASN1BEREncodeStream](#), 111
 - [ASN1BEREncodeStream](#), 112
- [ASN1BEREncodeStream](#)
 - [ASN1BEREncodeStream](#), 112
 - [close](#), 112
 - [encodeBigInt](#), 112
 - [encodeBitStr](#), 113
 - [encodeBMPStr](#), 113
 - [encodeBool](#), 114

- encodeCharStr, 114
- encodeEnum, 115
- encodeEoc, 115
- encodeIndefLen, 115
- encodeInt, 116
- encodeInt16, 116
- encodeInt64, 116
- encodeInt8, 117
- encodeNull, 117
- encodeObj, 117
- encodeObjId, 118
- encodeObjId64, 118
- encodeOctStr, 118, 119
- encodeReal, 119
- encodeRelativeOID, 120
- encodeTag, 120
- encodeTagAndIndefLen, 120
- encodeTagAndLen, 121
- encodeUInt, 121
- encodeUInt16, 121
- encodeUInt64, 122
- encodeUInt8, 122
- encodeUnivStr, 123
- flush, 123
- getAppInfo, 123
- getContext, 123
- getCtxtPtr, 123
- getErrorInfo, 124
- getStatus, 124
- isA, 124
- isOpened, 125
- operator<<, 125
- printErrorInfo, 125
- resetErrorInfo, 125
- setAppInfo, 125
- setDiag, 125
- write, 126
- ASN1BEREncodeStream.h, 135
- ASN1BERMessageBuffer, 127
 - ASN1BERMessageBuffer, 127
- ASN1BERMessageBuffer
 - ASN1BERMessageBuffer, 127
 - binDump, 127
 - calcIndefLen, 128
 - hexDump, 128
- asn1berSocket.h, 136
- asn1berStream.h, 137
- Asn1BufLocDescr
 - berruntime, 11
- BER Message Buffer Classes, 7
- BER Runtime Library Functions., 8
- BER/DER C Decode Functions., 12
- BER/DER C Encode Functions., 32
- BER/DER C File Functions., 29
- BER/DER/CER C++ Run-Time Classes., 6
- BER/PER C Utility Functions, 46
- berdecruntime
 - xd_16BitCharStr, 13
 - xd_32BitCharStr, 14
 - xd_bigint, 14
 - xd_bitstr, 15
 - xd_bitstr_s, 15
 - xd_boolean, 16
 - xd_charstr, 16
 - xd_chkend, 16
 - xd_count, 17
 - xd_decimal, 17
 - xd_enum, 17
 - xd_indeflen_ex, 18
 - xd_int16, 18
 - xd_int64, 18
 - xd_int8, 19
 - xd_integer, 19
 - xd_len, 20
 - xd_match, 20
 - xd_match1, 20
 - xd_memcpy, 21
 - xd_NextElement, 21
 - xd_null, 21
 - xd_objid, 22
 - xd_octstr, 22
 - xd_octstr_s, 22
 - xd_oid64, 23
 - xd_OpenType, 23
 - xd_OpenTypeAppend, 24
 - xd_OpenTypeExt, 24
 - xd_real, 24
 - xd_real10, 25
 - xd_reloid, 25
 - xd_setp, 25
 - xd_tag, 26
 - xd_Tag1AndLen, 26
 - xd_tag_len, 26
 - xd_uint16, 27
 - xd_uint64, 27
 - xd_uint8, 28
 - xd_unsigned, 28
 - xd_utf8str, 13
- berDecStrmBigInt
 - berstreamdec, 68
- berDecStrmBitStr
 - berstreamdec, 68
- berDecStrmBMPStr
 - berstreamdec, 68
- berDecStrmBool
 - berstreamdec, 69
- berDecStrmCharStr

- berstreamdec, 69
- berDecStrmDecimal
 - berstreamdec, 70
- berDecStrmDynBitStr
 - berstreamdec, 70
- berDecStrmDynOctStr
 - berstreamdec, 70
- berDecStrmEnum
 - berstreamdec, 71
- berDecStrmInt
 - berstreamdec, 71
- berDecStrmInt16
 - berstreamdec, 72
- berDecStrmInt64
 - berstreamdec, 72
- berDecStrmInt8
 - berstreamdec, 72
- berDecStrmLength
 - berstreamdec, 73
- berDecStrmMatchEOC
 - berstreamdec, 73
- berDecStrmMatchTag
 - berstreamdec, 73
- berDecStrmNextElement
 - berstreamdec, 74
- berDecStrmNull
 - berstreamdec, 74
- berDecStrmObjId
 - berstreamdec, 74
- berDecStrmObjId64
 - berstreamdec, 75
- berDecStrmOctStr
 - berstreamdec, 75
- berDecStrmOpenType
 - berstreamdec, 75
- berDecStrmOpenTypeAppend
 - berstreamdec, 76
- berDecStrmOpenTypeExt
 - berstreamdec, 76
- berDecStrmPeekTagAndLen
 - berstreamdec, 77
- berDecStrmReal
 - berstreamdec, 77
- berDecStrmReal10
 - berstreamdec, 77
- berDecStrmRelativeOID
 - berstreamdec, 78
- berDecStrmTag
 - berstreamdec, 78
- berDecStrmTagAndLen
 - berstreamdec, 78
- berDecStrmTestEOC
 - berstreamdec, 79
- berDecStrmTestTag

- berstreamdec, 79
- berDecStrmUInt
 - berstreamdec, 79
- berDecStrmUInt16
 - berstreamdec, 80
- berDecStrmUInt64
 - berstreamdec, 80
- berDecStrmUInt8
 - berstreamdec, 80
- berDecStrmUnivStr
 - berstreamdec, 81
- berderfilefun
 - xdf_len, 29
 - xdf_ReadContents, 29
 - xdf_ReadPastEOC, 30
 - xdf_tag, 30
 - xdf_TagAndLen, 30
- berencruntime
 - xe_16BitCharStr, 33
 - xe_32BitCharStr, 33
 - xe_bigint, 34
 - xe_bitstr, 34
 - xe_boolean, 35
 - xe_charstr, 35
 - xe_derCanonicalSort, 35
 - xe_derReal10, 36
 - xe_enum, 36
 - xe_expandBuffer, 37
 - xe_free, 37
 - xe_getBufLocDescr, 37
 - xe_getp, 37
 - xe_int16, 38
 - xe_int64, 38
 - xe_int8, 38
 - xe_integer, 39
 - xe_len, 39
 - xe_memcpy, 39
 - xe_null, 40
 - xe_objid, 40
 - xe_octstr, 40
 - xe_oid64, 41
 - xe_OpenType, 41
 - xe_OpenTypeExt, 41
 - xe_real, 42
 - xe_real10, 42
 - xe_reloid, 42
 - xe_setp, 43
 - xe_tag_len, 43
 - xe_TagAndIndefLen, 43
 - xe_uint16, 44
 - xe_uint64, 44
 - xe_uint8, 44
 - xe_unsigned, 45
 - xe_utf8str, 33

- berEncStrmBigInt
 - berstreamenc, 54
- berEncStrmBitStr
 - berstreamenc, 54
- berEncStrmBMPStr
 - berstreamenc, 54
- berEncStrmBool
 - berstreamenc, 55
- berEncStrmCharStr
 - berstreamenc, 55
- berEncStrmEnum
 - berstreamenc, 55
- berEncStrmEOC
 - berstreamenc, 56
- berEncStrmInt
 - berstreamenc, 56
- berEncStrmInt16
 - berstreamenc, 56
- berEncStrmInt64
 - berstreamenc, 57
- berEncStrmInt8
 - berstreamenc, 57
- berEncStrmLength
 - berstreamenc, 57
- berEncStrmNull
 - berstreamenc, 58
- berEncStrmObjId
 - berstreamenc, 58
- berEncStrmObjId64
 - berstreamenc, 58
- berEncStrmOctStr
 - berstreamenc, 59
- berEncStrmOpenTypeExt
 - berstreamenc, 59
- berEncStrmReal
 - berstreamenc, 59
- berEncStrmReal10
 - berstreamenc, 60
- berEncStrmRelativeOID
 - berstreamenc, 60
- berEncStrmTag
 - berstreamenc, 60
- berEncStrmTagAndIndefLen
 - berstreamenc, 61
- berEncStrmTagAndLen
 - berstreamenc, 61
- berEncStrmUInt
 - berstreamenc, 61
- berEncStrmUInt16
 - berstreamenc, 62
- berEncStrmUInt64
 - berstreamenc, 62
- berEncStrmUInt8
 - berstreamenc, 62
- berEncStrmUnivStr
 - berstreamenc, 63
- berEncStrmWriteOctet
 - berstreamenc, 63
- berEncStrmWriteOctets
 - berstreamenc, 63
- berEncStrmXSDAny
 - berstreamenc, 64
- berErrAddTagParm
 - berperUtil, 47
- berErrUnexpTag
 - berperUtil, 47
- berperUtil
 - berErrAddTagParm, 47
 - berErrUnexpTag, 47
 - berReadMsgFromSocket, 47
 - berTagToDynStr, 47
 - berTagToString, 48
 - xu_alloc_array, 48
 - xu_dump, 48
 - xu_fdump, 49
 - xu_hex_dump, 49
 - xu_RestoreBufferState, 49
 - xu_SaveBufferState, 49
- berReadMsgFromSocket
 - berperUtil, 47
- berruntime
 - Asn1BufLocDescr, 11
 - XD_CHKBUFEND, 9
 - XD_CHKDEFLEN, 9
 - XD_CHKEND, 9
 - XD_CHKEOB, 9
 - XD_MATCHBYTES2, 9
 - XD_MATCHBYTES3, 9
 - XD_MATCHBYTES4, 10
 - XD_MATCHBYTES5, 10
 - XD_MATCHEOC, 10
 - XD_MEMCPY1, 10
 - XE_CHKBUF, 10
 - XE_PUT2, 10
- berstream
 - berStrmFreeContext, 52
 - berStrmInitContext, 52
 - BS_CHKEND, 51
 - BS_CHKEOB, 51
- berstreamdec
 - berDecStrmBigInt, 68
 - berDecStrmBitStr, 68
 - berDecStrmBMPStr, 68
 - berDecStrmBool, 69
 - berDecStrmCharStr, 69
 - berDecStrmDecimal, 70
 - berDecStrmDynBitStr, 70
 - berDecStrmDynOctStr, 70

- berDecStrmEnum, [71](#)
- berDecStrmInt, [71](#)
- berDecStrmInt16, [72](#)
- berDecStrmInt64, [72](#)
- berDecStrmInt8, [72](#)
- berDecStrmLength, [73](#)
- berDecStrmMatchEOC, [73](#)
- berDecStrmMatchTag, [73](#)
- berDecStrmNextElement, [74](#)
- berDecStrmNull, [74](#)
- berDecStrmObjId, [74](#)
- berDecStrmObjId64, [75](#)
- berDecStrmOctStr, [75](#)
- berDecStrmOpenType, [75](#)
- berDecStrmOpenTypeAppend, [76](#)
- berDecStrmOpenTypeExt, [76](#)
- berDecStrmPeekTagAndLen, [77](#)
- berDecStrmReal, [77](#)
- berDecStrmReal10, [77](#)
- berDecStrmRelativeOID, [78](#)
- berDecStrmTag, [78](#)
- berDecStrmTagAndLen, [78](#)
- berDecStrmTestEOC, [79](#)
- berDecStrmTestTag, [79](#)
- berDecStrmUInt, [79](#)
- berDecStrmUInt16, [80](#)
- berDecStrmUInt64, [80](#)
- berDecStrmUInt8, [80](#)
- berDecStrmUnivStr, [81](#)
- berstreamenc
 - berEncStrmBigInt, [54](#)
 - berEncStrmBitStr, [54](#)
 - berEncStrmBMPStr, [54](#)
 - berEncStrmBool, [55](#)
 - berEncStrmCharStr, [55](#)
 - berEncStrmEnum, [55](#)
 - berEncStrmEOC, [56](#)
 - berEncStrmInt, [56](#)
 - berEncStrmInt16, [56](#)
 - berEncStrmInt64, [57](#)
 - berEncStrmInt8, [57](#)
 - berEncStrmLength, [57](#)
 - berEncStrmNull, [58](#)
 - berEncStrmObjId, [58](#)
 - berEncStrmObjId64, [58](#)
 - berEncStrmOctStr, [59](#)
 - berEncStrmOpenTypeExt, [59](#)
 - berEncStrmReal, [59](#)
 - berEncStrmReal10, [60](#)
 - berEncStrmRelativeOID, [60](#)
 - berEncStrmTag, [60](#)
 - berEncStrmTagAndIndefLen, [61](#)
 - berEncStrmTagAndLen, [61](#)
 - berEncStrmUInt, [61](#)
 - berEncStrmUInt16, [62](#)
 - berEncStrmUInt64, [62](#)
 - berEncStrmUInt8, [62](#)
 - berEncStrmUnivStr, [63](#)
 - berEncStrmWriteOctet, [63](#)
 - berEncStrmWriteOctets, [63](#)
 - berEncStrmXSDAny, [64](#)
 - cerEncStrmBitStr, [64](#)
 - cerEncStrmBMPStr, [64](#)
 - cerEncStrmCharStr, [65](#)
 - cerEncStrmOctStr, [65](#)
 - cerEncStrmReal10, [66](#)
 - cerEncStrmUnivStr, [66](#)
- berStrmFreeContext
 - berstream, [52](#)
- berStrmInitContext
 - berstream, [52](#)
- berTagToDynStr
 - berperUtil, [47](#)
- berTagToString
 - berperUtil, [48](#)
- binDump
 - ASN1BERMessageBuffer, [127](#)
- BS_CHKEND
 - berstream, [51](#)
- BS_CHKEOB
 - berstream, [51](#)
- byteIndex
 - ASN1BERDecodeStream, [90](#)
- C Streaming BER Decode Functions., [67](#)
- C Streaming BER Encode Functions., [53](#)
- C++ classes for streaming BER decoding., [83](#)
- C++ classes for streaming BER encoding., [82](#)
- calcIndefLen
 - ASN1BERMessageBuffer, [128](#)
- cerEncStrmBitStr
 - berstreamenc, [64](#)
- cerEncStrmBMPStr
 - berstreamenc, [64](#)
- cerEncStrmCharStr
 - berstreamenc, [65](#)
- cerEncStrmOctStr
 - berstreamenc, [65](#)
- cerEncStrmReal10
 - berstreamenc, [66](#)
- cerEncStrmUnivStr
 - berstreamenc, [66](#)
- chkend
 - ASN1BERDecodeStream, [90](#)
- close
 - ASN1BERDecodeStream, [90](#)
 - ASN1BEREncodeStream, [112](#)
- currentPos

- ASN1BERDecodeStream, [91](#)
- decodeBigInt
 - ASN1BERDecodeStream, [91](#)
- decodeBitStr
 - ASN1BERDecodeStream, [91, 92](#)
- decodeBMPStr
 - ASN1BERDecodeStream, [92](#)
- decodeBool
 - ASN1BERDecodeStream, [93](#)
- decodeCharStr
 - ASN1BERDecodeStream, [93](#)
- decodeEnum
 - ASN1BERDecodeStream, [94](#)
- decodeEoc
 - ASN1BERDecodeStream, [94](#)
- decodeInt
 - ASN1BERDecodeStream, [94](#)
- decodeInt16
 - ASN1BERDecodeStream, [95](#)
- decodeInt64
 - ASN1BERDecodeStream, [95](#)
- decodeInt8
 - ASN1BERDecodeStream, [96](#)
- decodeLength
 - ASN1BERDecodeStream, [96](#)
- decodeNull
 - ASN1BERDecodeStream, [96](#)
- decodeObj
 - ASN1BERDecodeStream, [97](#)
- decodeObjId
 - ASN1BERDecodeStream, [97](#)
- decodeObjId64
 - ASN1BERDecodeStream, [97](#)
- decodeOctStr
 - ASN1BERDecodeStream, [98](#)
- decodeReal
 - ASN1BERDecodeStream, [99](#)
- decodeRelativeOID
 - ASN1BERDecodeStream, [99](#)
- decodeTag
 - ASN1BERDecodeStream, [100](#)
- decodeTagAndLen
 - ASN1BERDecodeStream, [100](#)
- decodeUInt
 - ASN1BERDecodeStream, [100](#)
- decodeUInt16
 - ASN1BERDecodeStream, [101](#)
- decodeUInt64
 - ASN1BERDecodeStream, [101](#)
- decodeUInt8
 - ASN1BERDecodeStream, [102](#)
- decodeUnivStr
 - ASN1BERDecodeStream, [102](#)
- encodeBigInt
 - ASN1BEREncodeStream, [112](#)
- encodeBitStr
 - ASN1BEREncodeStream, [113](#)
- encodeBMPStr
 - ASN1BEREncodeStream, [113](#)
- encodeBool
 - ASN1BEREncodeStream, [114](#)
- encodeCharStr
 - ASN1BEREncodeStream, [114](#)
- encodeEnum
 - ASN1BEREncodeStream, [115](#)
- encodeEoc
 - ASN1BEREncodeStream, [115](#)
- encodeIndefLen
 - ASN1BEREncodeStream, [115](#)
- encodeInt
 - ASN1BEREncodeStream, [116](#)
- encodeInt16
 - ASN1BEREncodeStream, [116](#)
- encodeInt64
 - ASN1BEREncodeStream, [116](#)
- encodeInt8
 - ASN1BEREncodeStream, [117](#)
- encodeNull
 - ASN1BEREncodeStream, [117](#)
- encodeObj
 - ASN1BEREncodeStream, [117](#)
- encodeObjId
 - ASN1BEREncodeStream, [118](#)
- encodeObjId64
 - ASN1BEREncodeStream, [118](#)
- encodeOctStr
 - ASN1BEREncodeStream, [118, 119](#)
- encodeReal
 - ASN1BEREncodeStream, [119](#)
- encodeRelativeOID
 - ASN1BEREncodeStream, [120](#)
- encodeTag
 - ASN1BEREncodeStream, [120](#)
- encodeTagAndIndefLen
 - ASN1BEREncodeStream, [120](#)
- encodeTagAndLen
 - ASN1BEREncodeStream, [121](#)
- encodeUInt
 - ASN1BEREncodeStream, [121](#)
- encodeUInt16
 - ASN1BEREncodeStream, [121](#)
- encodeUInt64
 - ASN1BEREncodeStream, [122](#)
- encodeUInt8
 - ASN1BEREncodeStream, [122](#)
- encodeUnivStr
 - ASN1BEREncodeStream, [123](#)

- findElement
 - ASN1BERDecodeBuffer, 86
- flush
 - ASN1BERDecodeStream, 103
 - ASN1BEREncodeStream, 123
- getAppInfo
 - ASN1BERDecodeStream, 103
 - ASN1BEREncodeStream, 123
- getContext
 - ASN1BERDecodeStream, 103
 - ASN1BEREncodeStream, 123
- getCtxtPtr
 - ASN1BERDecodeStream, 103
 - ASN1BEREncodeStream, 123
- getErrorInfo
 - ASN1BERDecodeStream, 103, 104
 - ASN1BEREncodeStream, 124
- getMsgCopy
 - ASN1BEREncodeBuffer, 109
- getMsgPtr
 - ASN1BERDecodeBuffer, 86
 - ASN1BEREncodeBuffer, 109
- getStatus
 - ASN1BERDecodeStream, 104
 - ASN1BEREncodeStream, 124
- hexDump
 - ASN1BERMessageBuffer, 128
- init
 - ASN1BERDecodeBuffer, 86
 - ASN1BEREncodeBuffer, 109
- isA
 - ASN1BERDecodeBuffer, 86
 - ASN1BERDecodeStream, 104
 - ASN1BEREncodeBuffer, 109
 - ASN1BEREncodeStream, 124
- isOpened
 - ASN1BERDecodeStream, 104
 - ASN1BEREncodeStream, 125
- mark
 - ASN1BERDecodeStream, 104
- markSupported
 - ASN1BERDecodeStream, 105
- operator<<
 - ASN1BEREncodeBuffer, 109
 - ASN1BEREncodeStream, 125
- operator>>
 - ASN1BERDecodeBuffer, 87
 - ASN1BERDecodeStream, 105
- parseTagLen
 - ASN1BERDecodeBuffer, 87
- peekTagAndLen
 - ASN1BERDecodeStream, 105
- printErrorInfo
 - ASN1BERDecodeStream, 106
 - ASN1BEREncodeStream, 125
- read
 - ASN1BERDecodeStream, 106
- readBinaryFile
 - ASN1BERDecodeBuffer, 87
- readBlocking
 - ASN1BERDecodeStream, 106
- reset
 - ASN1BERDecodeStream, 106
- resetErrorInfo
 - ASN1BERDecodeStream, 107
 - ASN1BEREncodeStream, 125
- setAppInfo
 - ASN1BERDecodeStream, 107
 - ASN1BEREncodeStream, 125
- setBuffer
 - ASN1BERDecodeBuffer, 87
 - ASN1BEREncodeBuffer, 110
- setDiag
 - ASN1BERDecodeStream, 107
 - ASN1BEREncodeStream, 125
- skip
 - ASN1BERDecodeStream, 107
- Streaming BER Runtime Library Functions., 51
- write
 - ASN1BEREncodeStream, 126
- xd_16BitCharStr
 - berdecruntime, 13
- xd_32BitCharStr
 - berdecruntime, 14
- xd_bigint
 - berdecruntime, 14
- xd_bitstr
 - berdecruntime, 15
- xd_bitstr_s
 - berdecruntime, 15
- xd_boolean
 - berdecruntime, 16
- xd_charstr
 - berdecruntime, 16
- XD_CHKBUFEND
 - berruntime, 9
- XD_CHKDEFLEN
 - berruntime, 9
- XD_CHKEND
 - berruntime, 9

- xd_chkend
 - berdecruntime, 16
- XD_CHKEOB
 - berruntime, 9
- xd_count
 - berdecruntime, 17
- xd_decimal
 - berdecruntime, 17
- xd_enum
 - berdecruntime, 17
- xd_indeflen_ex
 - berdecruntime, 18
- xd_int16
 - berdecruntime, 18
- xd_int64
 - berdecruntime, 18
- xd_int8
 - berdecruntime, 19
- xd_integer
 - berdecruntime, 19
- xd_len
 - berdecruntime, 20
- xd_match
 - berdecruntime, 20
- xd_match1
 - berdecruntime, 20
- XD_MATCHBYTES2
 - berruntime, 9
- XD_MATCHBYTES3
 - berruntime, 9
- XD_MATCHBYTES4
 - berruntime, 10
- XD_MATCHBYTES5
 - berruntime, 10
- XD_MATCHEOC
 - berruntime, 10
- xd_memcpy
 - berdecruntime, 21
- XD_MEMCPY1
 - berruntime, 10
- xd_NextElement
 - berdecruntime, 21
- xd_null
 - berdecruntime, 21
- xd_objid
 - berdecruntime, 22
- xd_octstr
 - berdecruntime, 22
- xd_octstr_s
 - berdecruntime, 22
- xd_oid64
 - berdecruntime, 23
- xd_OpenType
 - berdecruntime, 23
- xd_OpenTypeAppend
 - berdecruntime, 24
- xd_OpenTypeExt
 - berdecruntime, 24
- xd_real
 - berdecruntime, 24
- xd_real10
 - berdecruntime, 25
- xd_reloid
 - berdecruntime, 25
- xd_setp
 - berdecruntime, 25
- xd_tag
 - berdecruntime, 26
- xd_Tag1AndLen
 - berdecruntime, 26
- xd_tag_len
 - berdecruntime, 26
- xd_uint16
 - berdecruntime, 27
- xd_uint64
 - berdecruntime, 27
- xd_uint8
 - berdecruntime, 28
- xd_unsigned
 - berdecruntime, 28
- xd_utf8str
 - berdecruntime, 13
- xdf_len
 - berderfilefun, 29
- xdf_ReadContents
 - berderfilefun, 29
- xdf_ReadPastEOC
 - berderfilefun, 30
- xdf_tag
 - berderfilefun, 30
- xdf_TagAndLen
 - berderfilefun, 30
- xe_16BitCharStr
 - berencruntime, 33
- xe_32BitCharStr
 - berencruntime, 33
- xe_bigint
 - berencruntime, 34
- xe_bitstr
 - berencruntime, 34
- xe_boolean
 - berencruntime, 35
- xe_charstr
 - berencruntime, 35
- XE_CHKBUF
 - berruntime, 10
- xe_derCanonicalSort
 - berencruntime, 35

- xe_derReal10
 - berencruntime, 36
- xe_enum
 - berencruntime, 36
- xe_expandBuffer
 - berencruntime, 37
- xe_free
 - berencruntime, 37
- xe_getBufLocDescr
 - berencruntime, 37
- xe_getp
 - berencruntime, 37
- xe_int16
 - berencruntime, 38
- xe_int64
 - berencruntime, 38
- xe_int8
 - berencruntime, 38
- xe_integer
 - berencruntime, 39
- xe_len
 - berencruntime, 39
- xe_memcpy
 - berencruntime, 39
- xe_null
 - berencruntime, 40
- xe_objid
 - berencruntime, 40
- xe_octstr
 - berencruntime, 40
- xe_oid64
 - berencruntime, 41
- xe_OpenType
 - berencruntime, 41
- xe_OpenTypeExt
 - berencruntime, 41
- XE_PUT2
 - berruntime, 10
- xe_real
 - berencruntime, 42
- xe_real10
 - berencruntime, 42
- xe_reloid
 - berencruntime, 42
- xe_setp
 - berencruntime, 43
- xe_tag_len
 - berencruntime, 43
- xe_TagAndIndefLen
 - berencruntime, 43
- xe_uint16
 - berencruntime, 44
- xe_uint64
 - berencruntime, 44
- xe_uint8
 - berencruntime, 44
- xe_unsigned
 - berencruntime, 45
- xe_utf8str
 - berencruntime, 33
- xu_alloc_array
 - berperUtil, 48
- xu_dump
 - berperUtil, 48
- xu_fdump
 - berperUtil, 49
- xu_hex_dump
 - berperUtil, 49
- xu_RestoreBufferState
 - berperUtil, 49
- xu_SaveBufferState
 - berperUtil, 49