ASN1C BER Runtime

Version 7.4

Objective Systems, Inc.

January 2020

ASN1C BER Runtime

Author's Contact Information. Comments, suggestions, and inquiries regarding ASN1C or this document may be sent by electronic mail to <info@obj-sys.com>.

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 complier.

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

Chapter 2. Module Documentation

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

Detailed Description

Modules

- BER Message Buffer Classes
- C++ classes for streaming BER encoding.
- C++ classes for streaming BER decoding.

BER Message Buffer Classes

Detailed Description

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

Classes

- struct ASN1BERLength
- struct ASN1BERMessageBuffer
- struct ASN1BEREncodeBuffer
- struct ASN1BERDecodeBuffer

C++ classes for streaming BER encoding.

Detailed Description

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

Classes

- struct ASN1BEREncodeStream

C++ classes for streaming BER decoding.

Detailed Description

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

Classes

- struct ASN1BERDecodeStream

BER Runtime Library Functions.

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.

Typedefs

- typedef OSRTBufLocDescr Asn1BufLocDescr

Functions

- int xd_MovePastEOC (OSCTXT * pctxt)
- int xd_consStrIndefLenAndSize (OSCTXT * pctxt, ASN1TAG expectedTag, OSSIZE * length, OSSIZE * size)

BER/DER C Decode Functions.

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.

Functions

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

- int xd_match (OSCTXT * pctxt, ASN1TAG tag, int * len_p, OSOCTET flags)
- int xd_match64 (OSCTXT * pctxt, ASN1TAG tag, OSSIZE * len_p, OSBOOL * pindef, OSOCTET flags)
- int xd_boolean (OSCTXT * pctxt, OSBOOL * object_p, ASN1TagType tagging, int length)
- int xd_integer (OSCTXT * pctxt, OSINT32 * object_p, ASN1TagType tagging, int length)
- int xd_int8 (OSCTXT * pctxt, OSINT8 * object_p, ASN1TagType tagging, int length)
- int xd_int16 (OSCTXT * pctxt, OSINT16 * object_p, ASN1TagType tagging, int length)
- int xd_unsigned (OSCTXT * pctxt, OSUINT32 * object_p, ASN1TagType tagging, int length)
- int xd_uint8 (OSCTXT * pctxt, OSUINT8 * object_p, ASN1TagType tagging, int length)
- int xd_uint16 (OSCTXT * pctxt, OSUINT16 * object_p, ASN1TagType tagging, int length)
- int xd_int64 (OSCTXT * pctxt, OSINT64 * object_p, ASN1TagType tagging, int length)
- int xd_uint64 (OSCTXT * pctxt, OSUINT64 * object_p, ASN1TagType tagging, int length)
- int xd_bigint (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, int length)
- int xd_bitstr_s (OSCTXT * pctxt, OSOCTET * object_p, OSUINT32 * numbits_p, ASN1TagType tagging, int length)
- int xd_bitstrExt_s (OSCTXT * pctxt, OSOCTET * object_p, OSUINT32 * numbits_p, OSOCTET ** extdata, ASN1TagType tagging, int length)
- int xd_bitstr64_s (OSCTXT * pctxt, OSOCTET * object_p, OSSIZE * numbits_p, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int xd_bitstr64Ext_s (OSCTXT * pctxt, OSOCTET * object_p, OSSIZE * numbits_p, OSOCTET ** extdata, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int xd_bitstr (OSCTXT * pctxt, const OSOCTET ** object_p2, OSUINT32 * numbits_p, ASN1TagType tagging, int length)
- int xd_bitstr64 (OSCTXT * pctxt, const OSOCTET ** object_p2, OSSIZE * numbits_p, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int xd_octstr_s (OSCTXT * pctxt, OSOCTET * object_p, OSUINT32 * pnumocts, ASN1TagType tagging, int length)
- int xd_octstr64_s (OSCTXT * pctxt, OSOCTET * object_p, OSSIZE * pnumocts, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int xd_octstr (OSCTXT * pctxt, const OSOCTET ** object_p2, OSUINT32 * pnumocts, ASN1TagType tagging, int length)
- int xd_octstr64 (OSCTXT * pctxt, OSOCTET ** object_p2, OSSIZE * pnumocts, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int xd_charstr (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int xd_charstr64 (OSCTXT * pctxt, char ** object_p, ASN1TagType tagging, ASN1TAG tag, OSSIZE length, OSBOOL indefLen)
- int xd_datestr (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, ASN1TAG tag, int length)

- int xd_timestr (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int xd_datetimestr (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int xd_timeofdaystr (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int xd_durationstr (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int berDecCharArray (OSCTXT * pctxt, char * charArray, OSSIZE arraySize, ASN1TagType tagging, ASN1TAG tag, int length)
- int xd_16BitCharStr (OSCTXT * pctxt, Asn116BitCharString * object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int xd_16BitCharStr64 (OSCTXT * pctxt, Asn116BitCharString * object_p, ASN1TagType tagging, ASN1TAG tag, OSSIZE length, OSBOOL indefLen)
- int xd_32BitCharStr (OSCTXT * pctxt, Asn132BitCharString * object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int xd_32BitCharStr64 (OSCTXT * pctxt, Asn132BitCharString * object_p, ASN1TagType tagging, ASN1TAG tag, OSSIZE length, OSBOOL indefLen)
- int xd_null (OSCTXT * pctxt, ASN1TagType tagging)
- int xd_objid (OSCTXT * pctxt, ASN1OBJID * object_p, ASN1TagType tagging, int length)
- int xd_oid64 (OSCTXT * pctxt, ASN1OID64 * object_p, ASN1TagType tagging, int length)
- int xd_reloid (OSCTXT * pctxt, ASN1OBJID * object_p, ASN1TagType tagging, int length)
- int xd_real (OSCTXT * pctxt, OSREAL * object_p, ASN1TagType tagging, int length)
- int xd_real_bin (OSCTXT * pctxt, OSREAL * object_p, ASN1TagType tagging, int length)
- int xd_real_b10_content (OSCTXT * pctxt, OSREAL * object_p, const char * content, int length, int form)
- int xd_real_der (OSCTXT * pctxt, OSREAL * object_p, ASN1TagType tagging, int length)
- int xd_enum (OSCTXT * pctxt, OSINT32 * object_p, ASN1TagType tagging, int length)
- int xd_enumUnsigned (OSCTXT * pctxt, OSUINT32 * object_p, ASN1TagType tagging, int length)
- int xd_OpenType (OSCTXT * pctxt, const OSOCTET ** object_p2, OSSIZE * pnumocts)
- int xd_OpenTypeExt (OSCTXT * pctxt, ASN1CCB * ccb_p, ASN1TAG * tags, int tagCount, OSRTDList * pElemList)
- int xd_OpenTypeExt64 (OSCTXT * pctxt, const OSOCTET * consptr, OSSIZE conslen, OSBOOL indefLen, ASN1TAG * tags, OSSIZE tagCount, OSRTDList * pElemList)
- int xd_OpenTypeAppend (OSCTXT * pctxt, OSRTDList * pElemList)
- int xd_real10 (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, int length)
- int xd_setp (OSCTXT * pctxt, const OSOCTET * msg_p, int msglen, ASN1TAG * tag_p, int * len_p)
- int xd_setp64 (OSCTXT * pctxt, const OSOCTET * msg_p, OSSIZE msglen, ASN1TAG * tag_p, OSSIZE * len_p, OSBOOL * pIndefLen)

- int xd_indeflen_ex (const OSOCTET * msg_p, int bufSize)
- int xd_indeflen64 (const OSOCTET * msg_p, OSSIZE bufSize, OSSIZE * plength)
- int xd_len (OSCTXT * pctxt, int * len_p)
- int xd_len64 (OSCTXT * pctxt, OSSIZE * len_p, OSBOOL * pindef)
- OSBOOL xd_chkend (OSCTXT * pctxt, const ASN1CCB * ccb_p)
- OSBOOL xd_chkend64 (OSCTXT * pctxt, const OSOCTET * consptr, OSSIZE conslen, OSBOOL indef)
- int xd_count (OSCTXT * pctxt, int length, int * count_p)
- int xd_count64 (OSCTXT * pctxt, OSSIZE length, OSBOOL indefLen, OSSIZE * count_p)
- int xd_NextElement (OSCTXT * pctxt)
- int xd_Tag1AndLen (OSCTXT * pctxt, OSINT32 * len_p)
- int xd_memcpy (OSCTXT * pctxt, OSOCTET * object_p, int length)
- int xd_match1 (OSCTXT * pctxt, OSOCTET tag, int * len_p)
- int xd_match1_64 (OSCTXT * pctxt, OSOCTET tag, OSSIZE * len_p, OSBOOL * pindef)

Macros

- #define xd_utf8str xd_charstr (pctxt, (const char**)object_p, tagging, ASN_ID_UTF8String, length)
- #define xd_indeflen xd_indeflen_ex(m, INT_MAX)

Function Documentation

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

Table 2.1. 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.

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.

Table 2.2. 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 with no fatal errors, negative return value is a fatal error.

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

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

Table 2.3. Parameters

pctxt	Pointer to context block structure.
tag_p	Pointer to variable to receive decoded tag info.
len_p	Length of message component. The value returned is undefined if length is indefinite.
pIndefLen	Boolean flag indicating parsed length is indefinite.
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.

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.

Table 2.4. 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.

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

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

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

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

Table 2.5. Parameters

pctxt	Pointer to context block structure.
tag	Tag variable to match.
len_p	Pointer to variable to receive length of message component.
pindef	Pointer to boolean variable to indicate if parsed length value is indefinite.
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.

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.

Table 2.6. 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)

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.

BER requires that INTEGER be encoded in a minimal number of octets. This function does not enforce that rule because it may be used to decode integer values that are not ASN.1 INTEGER values. However, if the ASN1CANON or ASN1DER flags are set, this will log a non-fatal error if the minimal number of octets is not used.

Table 2.7. 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.

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.

Table 2.8. 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.

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.

Table 2.9. 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.

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.

Table 2.10. 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.

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.

Table 2.11. 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.

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.

Table 2.12. 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.

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.

Table 2.13. 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.

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.

Table 2.14. 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.

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

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

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

Table 2.15. 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 hexadecimal value. Dynamic memory is allocated for the variable using the rtxMemAlloc function. The decoded variable is represented as a hexadecimal string with prefix '0x'.

Returns: . Completion status of operation:

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

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.

Table 2.16. 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.

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

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

Table 2.17. Parameters

pctxt	Pointer to context block structure.
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.
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.
extdata	Pointer to byte array containing extension data.
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.

Returns: . Completion status of operation:

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

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

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

Table 2.18. Parameters

pctxt	Pointer to context block structure.
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.

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.
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.
indefLen	Boolean flag indicating length is indefinite.

Returns: . Completion status of operation:

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

int xd_bitstr64Ext_s (OSCTXT *pctxt, OSOCTET *object_p, OSSIZE *numbits_p, OSOCTET **extdata, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)

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

Table 2.19. Parameters

pctxt	Pointer to context block structure.
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.
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.
extdata	Pointer to byte array containing extension data.
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.
indefLen	Boolean flag indicating length is indefinite.

Returns: . Completion status of operation:

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

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.

Table 2.20. 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.

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

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

Table 2.21. 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.
indefLen	Boolean flag indicating length is indefinite.

Returns: . Completion status of operation:

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

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.

Table 2.22. 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.

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

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

Table 2.23. 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.

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.

Table 2.24. 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.

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

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

Table 2.25. 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.
indefLen	Indicates length of data passed in is indefinite. Valid for implicit case only.

Returns: . Completion status of operation:

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

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

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

Table 2.26. 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.

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

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

Table 2.27. 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.
indefLen	Indicates length of data passed in is indefinite. Valid for implicit case only.

Returns: . Completion status of operation:

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

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

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

Table 2.28. 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.

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

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

Table 2.29. 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.

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

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

Table 2.30. 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.

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

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

Table 2.31. 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.

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

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

Table 2.32. 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.

int berDecCharArray (OSCTXT *pctxt, char *charArray, OSSIZE arraySize, ASN1TagType tagging, ASN1TAG tag, int length)

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

Table 2.33. Parameters

pctxt	Pointer to ASN.1 context block structure
charArray	Static character array variable (char[]) large enough to hold decoded data plus a null-termination byte. If the array is not large enough, an RTERR_TOOBIG error status will be returned.
arraySize	Size (in bytes) of the charArray variable.
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.

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.

Table 2.34. 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.

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

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

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

Table 2.35. 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.
indefLen	Flag indicating length of string is indefinite.

Returns: . Completion status of operation:

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

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.

Table 2.36. 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.

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

64-bit version of xd_32BitCharStr.

Table 2.37. 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.
indefLen	Indefinite length indicator.

Returns: . Completion status of operation:

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

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.

Table 2.38. 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.

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

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

Table 2.39. 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.

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.

Table 2.40. 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.

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

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

Table 2.41. 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.

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

This function decodes an ASN.1 (unconstrained) REAL value. This function allows the REAL to be encoded in any base, but if the ASN1CANON or ASN1DER flags are set, it will log a non-fatal error if the CER/DER restrictions are not met.

Table 2.42. 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.

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

This function decodes an ASN.1 REAL value that is a base 2 value, -0, NaN, or +/-INF (base 10 values are not allowed).

This function allows a base 2 value to be encoded in bases 2, 8, or 16 (not base 10), but if the ASN1CANON or ASN1DER flags are set, it will log a non-fatal error if the encoding uses base 8 or 16 or if some other CER/DER restriction is not met.

Table 2.43. 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.

int xd_real_b10_content (OSCTXT *pctxt, OSREAL *object_p, const char *content, int length, int form)

This function decodes the content octets of an ASN.1 REAL base 10 encoding. This function is for internal use by the runtime.

If the ASN1CANON or ASN1DER flags are set, this will log a non-fatal error if the CER/DER restrictions are not met. This includes enforcing the use of the NR3 form.

Table 2.44. 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).
content	The content octets, excluding the first octet which encodes the base and form.
length	Length of data pointed to by content.
form	The NR form used (1, 2, or 3). This governs the set of valid characters and will be check for validity if the ASN1CANON or ASN1DER flag is set.

Returns: . Completion status of operation:

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

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

This function decodes a value of the binary encoded ASN.1 REAL type. This function requires the REAL to be encoded in base 2. This is used for CER and DER for REAL values whose base is 2.

Table 2.45. 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.

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.

Table 2.46. 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.

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

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

Table 2.47. 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.

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

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

Table 2.48. 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.

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

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

Table 2.49. 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.
tags	Array of next expected tag values (null if last field). The routine will loop through elements until a matching tag is found or some other error occurs.
tagCount	The number of tags in the tags array.
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.

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

64-bit version of xd_OpenTypeExt.

Table 2.50. Parameters

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

See also: . xd_OpenTypeExt

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 (...).

Table 2.51. 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.

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

This function decodes an ASN.1 base 10 REAL value (i.e. the type is constrained to base 10 values).

This function will log an error if the value is not encoded in base 10, is -0, NaN, or +/-INF.

Table 2.52. 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.

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.

Table 2.53. Parameters

pctxt	Pointer to context block structure.
msg_p	Pointer to message buffer containing data to be decoded.
msglen	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 recieve the value of the initial tag parsed from a message. This is an optional parameter. It can be set to NULL is 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.

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

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

Table 2.54. Parameters

pctxt	Pointer to context block structure.
msg_p	Pointer to message buffer containing data to be decoded.
msglen	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 recieve the value of the initial tag parsed from a message. This is an optional parameter. It can be set to NULL is 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.
pIndefLen	Boolean flag indicating parsed length is indefinite.

Returns: . Completion status of operation:

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

int xd_indeflen_ex (const OSOCTET *msg_p, int bufSize)

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

Table 2.55. Parameters

msg_p	Pointer to an indefinite length message component.
bufSize	Size of the message buffer

Returns: . Zero or positive = decoded length, negative = error status

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

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

Table 2.56. Parameters

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

Returns: . Completion status of operation:

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

int xd_len (OSCTXT *pctxt, int *len_p)

This function decodes an ASN.1 length value.

Table 2.57. 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.

int xd_len64 (OSCTXT *pctxt, OSSIZE *len_p, OSBOOL *pindef)

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

Table 2.58. Parameters

pctxt	Pointer to context block structure.
len_p	Length of message component. The value returned is undefined if length is indefinite.
pindef	Pointer to boolean indicating if length is indefinite.

Returns: . Completion status of operation:

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

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

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

Table 2.59. 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.

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

64-bit version of xd_chkend.

Table 2.60. Parameters

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

Returns: . Completion status of operation:

- 0 (0) = success,

- negative return value is error.

See also: . xd_chkend

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

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

Table 2.61. 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.

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

64-bit version of xd_count function

Table 2.62. Parameters

pctxt	Pointer to context block structure.
length	Length of the constructed type.
indefLen	Flag indicating length is indefinite.
count_p	Pointer to a size-typed variable to receive the element count.

Returns: . Completion status of operation:

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

See also: . xd_count

int xd_NextElement (OSCTXT *pctxt)

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

Table 2.63. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

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.

Table 2.64. 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.

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.

Table 2.65. 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.

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.

Table 2.66. 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.

Macro Definition Documentation

#define xd_utf8str

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.

Table 2.67. 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.

Definition at line 985 of file asn1ber.h

The Documentation for this define was generated from the following file:

- asn1ber.h

BER/DER C File Functions.

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)

Function Documentation

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

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

Table 2.68. 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.
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.

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

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

Table 2.69. 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.

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

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

Table 2.70. 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.
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.

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

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

Table 2.71. 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.
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 index after the parsed data is copied to the buffer.

Returns: . Completion status of operation:

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

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.

Table 2.72. 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.
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 index after the parsed data is copied to the buffer.

Returns: . Completion status of operation:

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

BER/DER C Encode Functions.

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 complier to accomplish the encoding of complex ASN.1 structures. These functions are also directly callable from within a user's application program if the need to accomplish a low level encoding function exists.

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

Functions

- int xe_identifier (OSCTXT * pctxt, OSUINT32 ident)
- int xe_tag (OSCTXT * pctxt, ASN1TAG tag)
- int xe_tag_len (OSCTXT * pctxt, ASN1TAG tag, int length)
- int xe_boolean (OSCTXT * pctxt, OSBOOL * object_p, ASN1TagType tagging)
- int xe_integer (OSCTXT * pctxt, int * object_p, ASN1TagType tagging)

- int xe_unsigned (OSCTXT * pctxt, OSUINT32 * object_p, ASN1TagType tagging)
- int xe_int8 (OSCTXT * pctxt, OSINT8 * object_p, ASN1TagType tagging)
- int xe_int16 (OSCTXT * pctxt, OSINT16 * object_p, ASN1TagType tagging)
- int xe_int64 (OSCTXT * pctxt, OSINT64 * object_p, ASN1TagType tagging)
- int xe_uint64 (OSCTXT * pctxt, OSUINT64 * object_p, ASN1TagType tagging)
- int xe_uint8 (OSCTXT * pctxt, OSUINT8 * object_p, ASN1TagType tagging)
- int xe_uint16 (OSCTXT * pctxt, OSUINT16 * object_p, ASN1TagType tagging)
- int xe_bigint (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging)
- int xe_bigintn (OSCTXT * pctxt, const char * object_p, size_t nchars, ASN1TagType tagging)
- int xe_bitstr (OSCTXT * pctxt, const OSOCTET * object_p, OSSIZE numbits, ASN1TagType tagging)
- int xe_bitstrExt (OSCTXT * pctxt, const OSOCTET * object_p, OSSIZE numbits, OSSIZE dataSize, const OSOCTET * extdata, ASN1TagType tagging)
- int xe_cerBitstr (OSCTXT * pctxt, const OSOCTET * object_p, OSSIZE numbits, ASN1TagType tagging)
- int xe_cerCharstr (OSCTXT * pctxt, const char * pvalue, ASN1TagType tagging, ASN1TAG tag)
- int xe_cerOctstr (OSCTXT * pctxt, const OSOCTET * object_p, OSSIZE numocts, ASN1TagType tagging)
- int xe_octstr (OSCTXT * pctxt, const OSOCTET * object_p, OSSIZE numocts, ASN1TagType tagging)
- int xe_charstr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_cer16BitCharStr (OSCTXT * pctxt, Asn116BitCharString * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_cer32BitCharStr (OSCTXT * pctxt, Asn132BitCharString * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_16BitCharStr (OSCTXT * pctxt, Asn116BitCharString * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_32BitCharStr (OSCTXT * pctxt, Asn132BitCharString * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_datestr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_timestr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_datetimestr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_timeofdaystr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_durationstr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_null (OSCTXT * pctxt, ASN1TagType tagging)
- int xe_objid (OSCTXT * pctxt, ASN1OBJID * object_p, ASN1TagType tagging)
- int xe_oid64 (OSCTXT * pctxt, ASN1OID64 * object_p, ASN1TagType tagging)
- int xe_reloid (OSCTXT * pctxt, ASN1OBJID * object_p, ASN1TagType tagging)

- int xe_enum (OSCTXT * pctxt, OSINT32 * object_p, ASN1TagType tagging)
- int xe_enumUnsigned (OSCTXT * pctxt, OSUINT32 * object_p, ASN1TagType tagging)
- int xe_real (OSCTXT * pctxt, OSREAL * object_p, ASN1TagType tagging)
- int xe_OpenType (OSCTXT * pctxt, const OSOCTET * object_p, OSSIZE numocts)
- int xe_OpenTypeExt (OSCTXT * pctxt, OSRTDList * pElemList)
- int xe_OpenTypeExtDer (OSCTXT * pctxt, OSRTDList * pElemList, OSRTSList * pBufList)
- int xe_real10 (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging)
- int xe_derReal10 (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging)
- int xe_setp (OSCTXT * pctxt, OSOCTET * buf_p, OSSIZE bufsiz)
- OSOCTET * xe_getp (OSCTXT * pctxt)
- void xe_free (OSCTXT * pctxt)
- int xe_expandBuffer (OSCTXT * pctxt, size_t length)
- int xe_memcpy (OSCTXT * pctxt, const OSOCTET * object_p, size_t length)
- int xe_len (OSCTXT * pctxt, int length)
- int xe_len64 (OSCTXT * pctxt, OSSIZE length, OSBOOL indef)
- int xe_derCanonicalSort (OSCTXT * pctxt, OSRTSList * pList)
- int xe_derCanSortSet (OSCTXT * pctxt, OSRTSList * pList)
- int xe_TagAndIndefLen (OSCTXT * pctxt, ASN1TAG tag, int length)
- void xe_getBufLocDescr (OSCTXT * pctxt, OSSIZE length, Asn1BufLocDescr * pDescr)
- int derEncBitString (OSCTXT * pctxt, const OSOCTET * pvalue, OSSIZE numbits, ASN1TagType tagging)

Macros

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

Function Documentation

int xe_identifier (OSCTXT *pctxt, OSUINT32 ident)

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

Table 2.73. 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.
ident	Identifier value to be encoded.

Returns: . Length of the encoded item.

int xe_tag (OSCTXT *pctxt, ASN1TAG tag)

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

Table 2.74. 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 ASN1C internal tag representation. It is passed as an unsigned 32-bit integer.

Returns: . Length of the encoded tag component.

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

This function is used to encode ASN.1 tag and length fields that preface each block of message data. The ASN1C complier 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.

Table 2.75. 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 ASN1C 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.

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

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

Table 2.76. 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.
*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.

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

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

Table 2.77. 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.

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

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

Table 2.78. 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.

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

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

Table 2.79. 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.

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

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

Table 2.80. 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.

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

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

Table 2.81. 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 (de-

	pends 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.

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

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

Table 2.82. 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.

int xe_uint8 (OSCTXT *pctxt, OSUINT8 *object_p, ASN1TagType tagging)

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

Table 2.83. 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.

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

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

Table 2.84. 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.

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 radixes currently are not supported. Non-decimal values are assumed to be in two's complement form, unless a leading zero is included. It is highly recommended to use the hexadecimal or binary strings for better performance.

Table 2.85. 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.

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

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

Table 2.86. 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.

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

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

Table 2.87. 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.
dataSize	Size, in octets, of the root bit string data.
extdata	Pointer to byte array containing extension 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: . Length of the encoded message component. A negative status value will be returned if encoding is not successful.

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

This function encodes an ASN.1 BIT STRING according to CER, except that it assumes any trailing zero bits that must be trimmed have already been trimmed.

A BIT STRING of 7992 bits or less is encoded in primitive form. All other values are encoded in constructed form. When encoding in constructed form, if tagging is ASN1IMPL, the caller is responsible for encoding the EOC (prior to calling this function), the tag (which must use constructed form), and the length (which must be indefinite); if tagging is ASN1EXPL (and constructed form applies), the full TLV-EOC is encoded.

Table 2.88. Parameters

pctxt	Pointer to ASN.1 context block structure
object_p	Bit string to be encoded.
numbits	Number of bits in the bit string.
tagging	Explicit or implicit tagging specification.

Returns: . Length of the encoded message component, or negative if encoding fails.

int xe_cerCharstr (OSCTXT *pctxt, const char *pvalue, ASN1TagType tagging, ASN1TAG tag)

This function encodes an ASN.1 restricted character string according to CER.

This assumes each C char corresponds to an octet in the encoding. An character string of 1000 characters or less is encoded in primitive form. This takes into account the ASN1ESCZTERM flag and "\0" escape sequence. All other values are encoded in constructed form. When encoding in constructed form, if tagging is ASN1IMPL, the caller is responsible for encoding the EOC (prior to calling this function), the tag (which must use constructed form), and the length (which must be indefinite); if tagging is ASN1EXPL (and constructed form applies), the full TLV-EOC is encoded.

Table 2.89. Parameters

pctxt	Pointer to ASN.1 context block structure
pvalue	null-terminated C string
tagging	Explicit or implicit tagging specification.
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, or negative if encoding fails.

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

This function encodes an ASN.1 OCTET STRING according to CER.

An OCTET STRING of 1000 octets or less is encoded in primitive form. All other values are encoded in constructed form. When encoding in constructed form, if tagging is ASN1IMPL, the caller is responsible for encoding the EOC (prior to calling this function), the tag (which must use constructed form), and the length (which must be indefinite); if tagging is ASN1EXPL (and constructed form applies), the full TLV-EOC is encoded.

Table 2.90. Parameters

pctxt	Pointer to ASN.1 context block structure
object_p	octet string to be encoded.
numocts	Number of octets in the octet string.
tagging	Explicit or implicit tagging specification.

Returns: . Length of the encoded message component, or negative if encoding fails.

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

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

Table 2.91. 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.

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

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

Table 2.92. 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 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.

int xe_cer16BitCharStr (OSCTXT *pctxt, Asn116BitCharString *object_p, ASN1TagType tagging, ASN1TAG tag)

This function will encode an ASN.1 BMPString value according to CER, in reverse order (VLT rather than TLV).

A BMPString of 500 or less characters is encoded in primitive form. All other values are encoded in constructed form. When encoding in constructed form, if tagging is ASN1IMPL, the caller is responsible for encoding the EOC (prior to calling this function), the tag (which must use constructed form), and the length (which must be indefinite); if tagging is ASN1EXPL (and constructed form applies), the full TLV-EOC is encoded.

Table 2.93. 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 t 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.

int xe_cer32BitCharStr (OSCTXT *pctxt, Asn132BitCharString *object_p, ASN1TagType tagging, ASN1TAG tag)

This function will encode an ASN.1 UniversalString value according to CER, in reverse order (VLT rather than TLV).

A UniversalString of 250 or less characters is encoded in primitive form. All other values are encoded in constructed form. When encoding in constructed form, if tagging is ASN1IMPL, the caller is responsible for encoding the EOC (prior to calling this function), the tag (which must use constructed form), and the length (which must be indefinite); if tagging is ASN1EXPL (and constructed form applies), the full TLV-EOC is encoded.

Table 2.94. 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 t 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.

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.

Table 2.95. 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 t 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.

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.

Table 2.96. 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 t 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.

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

This function will encode a variable one of the ASN.1 ISO 8601 Date character string types. If the ASN1CANON or ASN1DER flag is set, this will ensure the encoding complies with CER/DER.

Table 2.97. 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 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.

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

This function will encode a variable one of the ASN.1 ISO 8601 Time character string types. If the ASN1CANON or ASN1DER flag is set, this will ensure the encoding complies with CER/DER.

Table 2.98. 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 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.

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

This function will encode a variable one of the ASN.1 ISO 8601 Date/Time character string types. If the ASN1CANON or ASN1DER flag is set, this will ensure the encoding complies with CER/DER.

Table 2.99. 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 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.

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

This function will encode a variable one of the ASN.1 ISO 8601 Time-of-day character string types. If the ASN1CANON or ASN1DER flag is set, this will ensure the encoding complies with CER/DER.

Table 2.100. 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 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.

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

This function will encode a variable one of the ASN.1 ISO 8601 Duration character string types. If the ASN1CANON or ASN1DER flag is set, this will ensure the encoding complies with CER/DER.

Table 2.101. 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 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.

int xe_null (OSCTXT *pctxt, ASN1TagType tagging)

This function will encode an ASN.1 NULL placeholder.

Table 2.102. 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.

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

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

Table 2.103. 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: . Length of the encoded message component. A negative status value will be returned if encoding is not successful.

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.

Table 2.104. 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: . Length of the encoded message component. A negative status value will be returned if encoding is not successful.

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

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

Table 2.105. 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: . Length of the encoded message component. A negative status value will be returned if encoding is not successful.

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

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

Table 2.106. Parameters

pctxt	Pointer to a context structure. This provides a storage area for the function to store al 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: . Length of the encoded message component. A negative status value will be returned if encoding is not successful.

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

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

Table 2.107. Parameters

pctxt	Pointer to a context structure. This provides a storage area for the function to store al 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: . Length of the encoded message component. A negative status value will be returned if encoding is not successful.

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.

Table 2.108. Parameters

pctxt	Pointer to a context structure. This provides a storage area for the function to store al 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: . Length of the encoded message component. A negative status value will be returned if encoding is not successful.

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

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

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

Table 2.109. 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: . Length of the encoded message component. A negative status value will be returned if encoding is not successful.

int xe_OpenTypeExt (OSCTXT *pctxt, OSRTDList *pElemList)

Table 2.110. 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.

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

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

Table 2.111. 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 unknown extension elements, a list of ASN1OpenType.
pBufList	The list of buffer locations to add to, a list of Asn1BufLocDescr.

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.

Table 2.112. Parameters

pctxt	Pointer to a context structure. This provides a storage area for the function to store al 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: . Length of the encoded message component. A negative status value will be returned if encoding is not successful.

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.

Table 2.113. Parameters

pctxt	Pointer to a context structure. This provides a storage area for the function to store al 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: . Length of the encoded message component. A negative status value will be returned if encoding is not successful.

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

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

Table 2.114. 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.

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 BER messages are encoded from the end of a given buffer toward the beginning. Therefore, in practically all cases, the start of the message will not be at the beginning of the buffer.

Table 2.115. 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. If nothing has been encoded, NULL is returned.

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.

Table 2.116. 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.

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.

Table 2.117. 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.

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.

Table 2.118. 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: . Length of the encoded message component. A negative status value will be returned if encoding is not successful.

int xe_len (OSCTXT *pctxt, int length)

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

Table 2.119. Parameters

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

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

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

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

Table 2.120. Parameters

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

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

See also: . xe_len

int xe_derCanonicalSort (OSCTXT *pctxt, OSRTSList *pList)

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

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

Table 2.121. Parameters

pctxt	Pointer to a context structure. This provides a storage area for the function to store al 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.

int xe_derCanSortSet (OSCTXT *pctxt, OSRTSList *pList)

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

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

Table 2.122. Parameters

pctxt	Pointer to a context structure. This provides a storage area for the function to store al 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.

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.

Table 2.123. Parameters

pctxt	Pointer to a context structure. This provides a storage area for the function to store al 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: . Length of the encoded message component. A negative status value will be returned if encoding is not successful.

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

Table 2.124. Parameters

pctxt	Pointer to a context structure. This provides a storage area for the function to store al 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.

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

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

Table 2.125. Parameters

pctxt	Pointer to a context structure. This provides a storage area for the function to store al working variables that must be maintained between function calls.
pvalue	Bit string to be encoded.
numbits	Number of bits in the 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'.

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

Macro Definition Documentation

#define xe_utf8str

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

Table 2.126. 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.

Definition at line 2312 of file asn1ber.h

The Documentation for this define was generated from the following file:

- asn1ber.h

BER/PER C Utility Functions

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.

Classes

- struct ASN1TAGTEXT

Typedefs

- typedef struct ASN1TAGTEXT ASN1TAGTEXT

Functions

- int berDefToIndefLen (OSCTXT * pSrcCtxt, OSCTXT * pDstCtxt)
- int berIndefToDefLen (OSCTXT * pSrcCtxt, OSCTXT * pDstCtxt)
- OSBOOL berErrAddTagParm (OSCTXT * pctxt, ASN1TAG tag)
- int berErrUnexpTag (OSCTXT * pctxt, ASN1TAG exptag)
- int berGetLibVersion (OSVOIDARG)
- const char * berGetLibInfo (OSVOIDARG)
- int berParseTagLen (const OSOCTET * buffer, size_t bufidx, size_t bufsiz, ASN1TAG * ptag, size_t * plen)
- const char * berTagToString (ASN1TAG tag, char * buffer, size_t bufsiz)
- const char * berTagToTypeName (ASN1TAG tag)
- const char * berTagToDynStr (OSCTXT * pctxt, ASN1TAG tag)
- int berValidateIso8601DateStr (OSCTXT * pctxt, const char ** ppvalue)
- int berValidateIso8601DurationStr (OSCTXT * pctxt, const char ** ppvalue)
- int berValidateIso8601TimeStr (OSCTXT * pctxt, const char ** ppvalue)
- int xu_verify_len (OSOCTET * msg_p)
- void * xu_parse_mmbuf (OSOCTET ** buf_p2, int * buflen_p, OSOCTET * start_p, int bufsiz)
- void xu_alloc_array (OSCTXT * pctxt, ASN1SeqOf * seqOf_p, int recSize, int recCount)
- void xu_octscpy_s (OSUINT32 * nocts_p, OSOCTET * data_p, char * cstr, char zterm)
- void xu_octscpy_ss (ASN1OctStr * octStr_p, char * cstring, char zterm)
- void xu_octscpy_d (OSCTXT * pctxt, OSUINT32 * nocts_p, const OSOCTET ** data_p2, char * cstring, char zterm)
- void xu_octscpy_ds (OSCTXT * pctxt, ASN1DynOctStr * octStr_p, char * cstring, char zterm)
- void xu_octmcpy_s (ASN1OctStr * octStr_p, void * data_p, int datalen)
- void xu_octmcpy_d (OSCTXT * pctxt, ASN1DynOctStr * octStr_p, void * data_p, int datalen)

- char * xu_fetchstr (int numocts, char * data)
- int xu_hexstrcpy (char * data, char * hstring)
- int xu_binstrcpy (char * data, char * bstring)
- int xu_dump (const OSOCTET * msgptr, ASN1DumpCbFunc cb, void * cbArg_p)
- int xu_fdump (FILE * file_p, const OSOCTET * msgptr)
- int xu_dump2 (OSCTXT * pctxt, const OSOCTET * msgptr)
- void xu_fmt_tag (ASN1TAG * tag_p, char * class_p, char * form_p, char * id_code)
- void xu_fmt_tag_s (ASN1TAG * tag_p, ASN1TAGTEXT * pTagText)
- char * xu_fmt_tag2 (ASN1TAG * tag_p, char * bufp)
- char * xu_fmt_tag2_s (ASN1TAG * tag_p, char * bufp, OSSIZE bufsize)
- char * xu_fmt_contents (OSCTXT * pctxt, int len, int * count)
- int xu_fread (FILE * fp, OSOCTET * bufp, int bufsiz)
- void xu_SaveBufferState (OSCTXT * pCtxt, OSRTBufSave * pSavedInfo)
- void xu_RestoreBufferState (OSCTXT * pCtxt, OSRTBufSave * pSavedInfo)
- int berReadMsgFromSocket (OSCTXT * pctxt, OSRTSOCKET socket, OSOCTET * buffer, int bufsiz, OSOCTET ** ppDestBuffer, int * pMessageSize)

Macros

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

Function Documentation

int berDefToIndefLen (OSCTXT *pSrcCtxt, OSCTXT *pDstCtxt)

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

Table 2.127. Parameters

pSrcCtxt	A pointer to a source OSCTXT structure.
pDstCtxt	A pointer to a destination OSCTXT structure.

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

int berIndefToDefLen (OSCTXT *pSrcCtxt, OSCTXT *pDstCtxt)

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

Table 2.128. Parameters

pSrcCtxt	A pointer to a source OSCTXT structure.
pDstCtxt	A pointer to a destination OSCTXT structure.

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

OSBOOL berErrAddTagParm (OSCTXT *pctxt, ASN1TAG tag)

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

Table 2.129. Parameters

pctxt	Pointer to a context structure.
tag	The tag to add.

Returns: . Pointer to the text string (buffer).

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.

Table 2.130. Parameters

pctxt	Pointer to a context structure.
exptag	Expected tag.

Returns: . RTERR_IDNOTFOU status code.

int berGetLibVersion (OSVOIDARG)

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

const char* berGetLibInfo (OSVOIDARG)

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

int berParseTagLen (const OSOCTET *buffer, size_t bufidx, size_t bufsiz, ASN1TAG *ptag, size_t *plen)

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

Table 2.131. Parameters

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

Returns: . Pointer to the text string (buffer).

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.

Table 2.132. 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).

const char* berTagToTypeName (ASN1TAG tag)

This function returns the built-in type name associated with a BER UNIVERSAL tag. If the tag is not UNIVERSAL or the ID code is unknown, NULL is returned.

Table 2.133. Parameters

tag	The tag to convert.

Returns: . Universal type name or NULL if unknown.

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.

Table 2.134. Parameters

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

Returns: . Pointer to dynamic text string.

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

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

Table 2.135. 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.
ppvalue	A pointer to a null-terminated C character string to be validated. Note the string will be modified.

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

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

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

Table 2.136. 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.
ppvalue	A pointer to a null-terminated C character string to be validated. Note the string will be modified.

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

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

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

Table 2.137. 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.
ppvalue	A pointer to a null-terminated C character string to be validated. Note the string will be modified.

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

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.

Table 2.138. 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.

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

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

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

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

Table 2.139. 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.

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

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

Table 2.140. 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.

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

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

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

Table 2.141. Parameters

*pctxt	A pointer to an OSCTXT structure.
*msgptr	A pointer to an encoded ASN.1 message.

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

See also: . ::rtxSetPrintStream
::rtxSetGlobalPrintStream

void xu_fmt_tag (ASN1TAG *tag_p, char *class_p, char *form_p, char *id_code)

This function formats a tag in internal form into class, form, and ID code textual components. Its use is DEPRECATED because there is no mechanism to specify the size of text buffers for the components. Use xu_fmt_tag_s instead.

Table 2.142. Parameters

tag_p	Pointer to tag value in numeric form.
class_p	Pointer to buffer to receive class. It must be at least 5 bytes in size.
form_p	Pointer to buffer to receive form ('C' or 'P'). It must be at least 1 byte in size. Result is not null terminated.
id_code	Pointer to buffer to receive ID code. It must be at least 8 bytes in size.

void xu_fmt_tag_s (ASN1TAG *tag_p, ASN1TAGTEXT *pTagText)

This function formats a tag in internal form into class, form, and ID code textual components.

Table 2.143. Parameters

tag_p	Pointer to tag value in numeric form.
pTagText	Pointer to structure to receive tag text.

char* xu_fmt_tag2 (ASN1TAG *tag_p, char *bufp)

This function formats a tag in internal form into a string of format [CLASS FORM ID-CODE]. Its use is DEPRECATED because there is no mechanism to specify the size of the text buffer to receive the formatted tag. Use xu_fmt_tag2_s instead.

Table 2.144. Parameters

tag_p	Pointer to tag value in numeric form.
bufp	Pointer to buffer to receive formatted tag text. It must be at least 32 bytes in size.

char* xu_fmt_tag2_s (ASN1TAG *tag_p, char *bufp, OSSIZE bufsize)

This function formats a tag in internal form into a string of format [CLASS FORM ID-CODE].

Table 2.145. Parameters

tag_p	Pointer to tag value in numeric form.
bufp	Pointer to buffer to receive formatted tag text. It must be at least 32 bytes in size.
bufsize	Size of buffer to receive formatted text.

void xu_SaveBufferState (OSCTXT *pCtxt, OSRTBufSave *pSavedInfo)

This function is used to save the current buffer state.

Table 2.146. Parameters

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

void xu_RestoreBufferState (OSCTXT *pCtxt, OSRTBufSave *pSavedInfo)

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

Table 2.147. Parameters

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

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.

Table 2.148. 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.

Macro Definition Documentation

#define xu_hex_dump

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 depreciated and is only being maintained to provide backward compatibility with older versions of ASN1C.

Table 2.149. Parameters

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

Definition at line 3356 of file asn1ber.h

The Documentation for this define was generated from the following file:

- asn1ber.h

Streaming BER Runtime Library Functions.

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.

Functions

- int berStrmInitContext (OSCTXT * pctxt)
- int berStrmInitContextUsingKey (OSCTXT * pctxt, const OSOCTET * key, size_t keylen)
- int berStrmFreeContext (OSCTXT * pctxt)

Macros

- #define cerEncCanonicalSort (pctxt, pMemCtxt, pList) \ rtxEncCanonicalSort(pctxt, pMemCtxt, pList)
- #define cerGetBufLocDescr (pctxt, pDescr) \ rtxGetBufLocDescr(pctxt, pDescr)
- #define cerAddBufLocDescr (pctxt, pElemList, pDescr) \ rtxAddBufLocDescr(pctxt, pElemList, pDescr)
- #define BS_CHKEOB (((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))
- #define BS_CHKEND ((ccb_p)->stat = 0, \ (((ccb_p)->len == ASN_K_INDEFLEN) ? berDecStrmTestEOC (pctxt,ccb_p) : \ (((int)(OSRTSTREAM_BYTEINDEX(pctxt) - (ccb_p)->bytes) >= (ccb_p)->len))))

Function Documentation

int berStrmInitContext (OSCTXT *pctxt)

This function initializes an OSCTXT block for further streaming operations. It makes sure that if the block was not previoiusly 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.

Table 2.150. Parameters

pctxt	Pointer to context structure variable to be initialized.

Returns: . Completion status of operation:

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

int berStrmFreeContext (OSCTXT *pctxt)

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

Table 2.151. Parameters

pctxt	A pointer to a context structure.

Returns: . Completion status of operation:

- 0 (0) = success,

- negative return value is error.

C Streaming BER Encode Functions.

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 * pctxt, const char * pvalue, ASN1TagType tagging)
- int berEncStrmBigIntNchars (OSCTXT * pctxt, const char * pvalue, size_t nchars, ASN1TagType tagging)
- int berEncStrmBitStr (OSCTXT * pctxt, const OSOCTET * object_p, OSUINT32 numbits, ASN1TagType tagging)
- int berEncStrmBMPStr (OSCTXT * pctxt, const Asn116BitCharString * object_p, ASN1TagType tagging)
- int berEncStrmBool (OSCTXT * pctxt, OSBOOL value, ASN1TagType tagging)
- int berEncStrmCharStr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int berEncStrmDateStr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int berEncStrmDateTimeStr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int berEncStrmDurationStr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int berEncStrmTimeStr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int berEncStrmTimeOfDayStr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int berEncStrmDefLength (OSCTXT * pctxt, size_t length)
- int berEncStrmEOC (OSCTXT * pctxt)
- int berEncStrmEnum (OSCTXT * pctxt, OSINT32 value, ASN1TagType tagging)
- int berEncStrmInt (OSCTXT * pctxt, OSINT32 value, ASN1TagType tagging)
- int berEncStrmInt8 (OSCTXT * pctxt, OSINT8 value, ASN1TagType tagging)
- int berEncStrmInt16 (OSCTXT * pctxt, OSINT16 value, ASN1TagType tagging)
- int berEncStrmInt64 (OSCTXT * pctxt, OSINT64 value, ASN1TagType tagging)
- int berEncStrmLength (OSCTXT * pctxt, int length)
- int berEncStrmNull (OSCTXT * pctxt, ASN1TagType tagging)
- int berEncStrmObjId (OSCTXT * pctxt, const ASN1OBJID * object_p, ASN1TagType tagging)
- int berEncStrmObjId64 (OSCTXT * pctxt, const ASN1OID64 * object_p, ASN1TagType tagging)

- int berEncStrmOctStr (OSCTXT * pctxt, const OSOCTET * object_p, OSSIZE numocts, ASN1TagType tagging)
- int berEncStrmOpenTypeExt (OSCTXT * pctxt, OSRTDList * pElemList)
- int berEncStrmReal (OSCTXT * pctxt, OSREAL value, ASN1TagType tagging)
- int berEncStrmReal10 (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging)
- int cerEncStrmReal10 (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging)
- int berEncStrmRelativeOID (OSCTXT * pctxt, const ASN1OBJID * object_p, ASN1TagType tagging)
- int berEncStrmTag (OSCTXT * pctxt, ASN1TAG tag)
- int berEncStrmTagAndLen (OSCTXT * pctxt, ASN1TAG tag, int length)
- int berEncStrmTagAndDefLen (OSCTXT * pctxt, ASN1TAG tag, OSSIZE length)
- int berEncStrmTagAndIndefLen (OSCTXT * pctxt, ASN1TAG tag)
- int berEncStrmUInt (OSCTXT * pctxt, OSUINT32 value, ASN1TagType tagging)
- int berEncStrmUInt8 (OSCTXT * pctxt, OSUINT8 value, ASN1TagType tagging)
- int berEncStrmUInt16 (OSCTXT * pctxt, OSUINT16 value, ASN1TagType tagging)
- int berEncStrmUInt64 (OSCTXT * pctxt, OSUINT64 value, ASN1TagType tagging)
- int berEncStrmUnivStr (OSCTXT * pctxt, const Asn132BitCharString * object_p, ASN1TagType tagging)
- int berEncStrmXSDAny (OSCTXT * pctxt, OSXSDAny * pvalue, ASN1TagType tagging)
- int berEncStrmWriteOctet (OSCTXT * pctxt, OSOCTET octet)
- int berEncStrmWriteOctets (OSCTXT * pctxt, const OSOCTET * poctets, size_t numocts)
- int cerEncStrmBMPStr (OSCTXT * pctxt, const Asn116BitCharString * object_p, ASN1TagType tagging)
- int cerEncStrmBitStr (OSCTXT * pctxt, const OSOCTET * object_p, OSUINT32 numbits, ASN1TagType tagging)
- int cerEncStrmCharStr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int cerEncStrmOctStr (OSCTXT * pctxt, const OSOCTET * object_p, OSUINT32 numocts, ASN1TagType tagging)
- int cerEncStrmUnivStr (OSCTXT * pctxt, const Asn132BitCharString * object_p, ASN1TagType tagging)

Function Documentation

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 radixes currently are not supported. It is highly recommended to use the hexadecimal or binary strings for better performance.

Table 2.152. 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.

int berEncStrmBigIntNchars (OSCTXT *pctxt, const char *pvalue, size_t nchars, ASN1TagType tagging)

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

Table 2.153. 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.
nchars	Number of characters from pvalue to encode.
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.

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

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

Table 2.154. 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.

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.

Table 2.155. 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.

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

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

Table 2.156. 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.

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.

Table 2.157. 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.

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

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

Table 2.158. 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.

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

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

Table 2.159. 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.

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

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

Table 2.160. 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.

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

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

Table 2.161. 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.

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

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

Table 2.162. 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.

int berEncStrmDefLength (OSCTXT *pctxt, size_t length)

This function encodes a definite length field to the stream.

Table 2.163. 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	Length to be encoded.

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

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).

Table 2.164. 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.

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.

Table 2.165. 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.

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

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

Table 2.166. 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.

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

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

Table 2.167. 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.

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

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

Table 2.168. 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.

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

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

Table 2.169. 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.

int berEncStrmLength (OSCTXT *pctxt, int length)

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

Table 2.170. 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.

int berEncStrmNull (OSCTXT *pctxt, ASN1TagType tagging)

This function encodes an ASN.1 NULL placeholder.

Table 2.171. 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.

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

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

Table 2.172. 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 subidentifers 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.

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.

Table 2.173. 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 subidentifers 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.

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

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

Table 2.174. 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.

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.

Table 2.175. 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.

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.

Table 2.176. 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.

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.

Table 2.177. Parameters

pctxt	Pointer to a context structure. This provides a storage area for the function to store al 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.

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.

Table 2.178. Parameters

pctxt	Pointer to a context structure. This provides a storage area for the function to store al 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.

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

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

Table 2.179. 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 subidentifers 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.

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.

Table 2.180. 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.

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.

Table 2.181. 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.

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

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

Table 2.182. 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.

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

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.

Table 2.183. 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.

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

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

Table 2.184. 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.

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

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

Table 2.185. 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.

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

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

Table 2.186. 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.

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

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

Table 2.187. 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.

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.

Table 2.188. 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.

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.

Table 2.189. 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.

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).

Table 2.190. 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.

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.

Table 2.191. 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.

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

This function encodes a variable of the ASN.1 BMPString type with using Canonical Encoding Rules (CER). BMPString type is based on a 16-bit character sets.

Table 2.192. 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.

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).

Table 2.193. 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.

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.

Table 2.194. 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.

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).

Table 2.195. 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.

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.

Table 2.196. 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.

C Streaming BER Decode Functions.

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 berDecStrmBigEnum (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, int length)
- int berDecStrmBitStr (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnbits, ASN1TagType tagging, int length)
- int berDecStrmBool (OSCTXT * pctxt, OSBOOL * object_p, ASN1TagType tagging, int length)
- int berDecStrmCharStr (OSCTXT * pctxt, const char ** ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int berDecStrmCharArray (OSCTXT * pctxt, char * charArray, OSSIZE arraySize, ASN1TagType tagging, ASN1TAG tag, int length)

- int berDecStrmDateStr (OSCTXT * pctxt, const char ** ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int berDecStrmDateTimeStr (OSCTXT * pctxt, const char ** ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int berDecStrmDurationStr (OSCTXT * pctxt, const char ** ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int berDecStrmTimeStr (OSCTXT * pctxt, const char ** ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int berDecStrmTimeOfDayStr (OSCTXT * pctxt, const char ** ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- OSBOOL berDecStrmCheckEnd (OSCTXT * pctxt, ASN1CCB * pccb)
- int berDecStrmDynBitStr (OSCTXT * pctxt, const OSOCTET ** ppvalue, OSUINT32 * pnbits, ASN1TagType tagging, int length)
- int berDecStrmDynBitStr64 (OSCTXT * pctxt, const OSOCTET ** ppvalue, OSSIZE * pnbits, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int berDecStrmDynOctStr (OSCTXT * pctxt, const OSOCTET ** ppvalue, OSUINT32 * pnocts, ASN1TagType tagging, int length)
- int berDecStrmDynOctStr64 (OSCTXT * pctxt, OSOCTET ** ppvalue, OSSIZE * pnocts, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int berDecStrmEnum (OSCTXT * pctxt, OSINT32 * object_p, ASN1TagType tagging, int length)
- int berDecStrmFindTag (OSCTXT * pctxt, ASN1TAG tag, int * len_p, OSBOOL advance)
- int berDecStrmFindTag2 (OSCTXT * pctxt, ASN1TAG tag, OSSIZE * len_p, OSBOOL * pIndefLen, OSBOOL advance)
- int berDecStrmGetTLVLength (OSCTXT * pctxt)
- int berDecStrmInt (OSCTXT * pctxt, OSINT32 * object_p, ASN1TagType tagging, int length)
- int berDecStrmInt8 (OSCTXT * pctxt, OSINT8 * object_p, ASN1TagType tagging, int length)
- int berDecStrmInt16 (OSCTXT * pctxt, OSINT16 * object_p, ASN1TagType tagging, int length)
- int berDecStrmInt64 (OSCTXT * pctxt, OSINT64 * object_p, ASN1TagType tagging, int length)
- int berDecStrmLength (OSCTXT * pctxt, int * len_p)
- int berDecStrmLength2 (OSCTXT * pctxt, OSSIZE * pLength, OSBOOL * pIndefLen)
- int berDecStrmMatchEOC (OSCTXT * pctxt)
- int berDecStrmMatchTag (OSCTXT * pctxt, ASN1TAG tag, int * len_p, OSBOOL advance)
- int berDecStrmMatchTag2 (OSCTXT * pctxt, ASN1TAG tag, OSSIZE * len_p, OSBOOL * pIndefLen, OSBOOL advance)
- int berDecStrmNextElement (OSCTXT * pctxt)

- int berDecStrmNull (OSCTXT * pctxt, ASN1TagType tagging)
- int berDecStrmObjId (OSCTXT * pctxt, ASN1OBJID * object_p, ASN1TagType tagging, int length)
- int berDecStrmObjId64 (OSCTXT * pctxt, ASN1OID64 * object_p, ASN1TagType tagging, int length)
- int berDecStrmOctStr (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocts, ASN1TagType tagging, int length)
- int berDecStrmOpenType (OSCTXT * pctxt, const OSOCTET ** object_p2, OSSIZE * pnumocts)
- int berDecStrmOpenTypeAppend (OSCTXT * pctxt, OSRTDList * pElemList)
- int berDecStrmOpenTypeExt (OSCTXT * pctxt, ASN1CCB * ccb_p, ASN1TAG * tags, int tagCount, OSRTDList * pElemList)
- int berDecStrmPeekTagAndLen (OSCTXT * pctxt, ASN1TAG * ptag, int * plen)
- int berDecStrmReadDynTLV (OSCTXT * pctxt, OSOCTET ** ppbuf)
- int berDecStrmReadTLV (OSCTXT * pctxt, OSOCTET * buf, OSSIZE bufsiz)
- int berDecStrmReal (OSCTXT * pctxt, OSREAL * object_p, ASN1TagType tagging, int length)
- int berDecStrmRealBin (OSCTXT * pctxt, OSREAL * object_p, ASN1TagType tagging, int length)
- int berDecStrmRealDer (OSCTXT * pctxt, OSREAL * object_p, ASN1TagType tagging, int length)
- int berDecStrmReal10 (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, int length)
- int berDecStrmRelativeOID (OSCTXT * pctxt, ASN1OBJID * object_p, ASN1TagType tagging, int length)
- int berDecStrmTag (OSCTXT * pctxt, ASN1TAG * tag_p)
- int berDecStrmTagAndLen (OSCTXT * pctxt, ASN1TAG * tag_p, int * len_p)
- int berDecStrmTagAndLen2 (OSCTXT * pctxt, ASN1TAG * tag_p, OSSIZE * len_p, OSBOOL * pIndefLen)
- OSBOOL berDecStrmTestEOC (OSCTXT * pctxt, ASN1CCB * ccb_p)
- OSBOOL berDecStrmTestEOC2 (OSCTXT * pctxt)
- int berDecStrmTestTag (OSCTXT * pctxt, ASN1TAG tag, int * len_p, OSBOOL advance)
- int berDecStrmUInt (OSCTXT * pctxt, OSUINT32 * object_p, ASN1TagType tagging, int length)
- int berDecStrmUInt8 (OSCTXT * pctxt, OSUINT8 * object_p, ASN1TagType tagging, int length)
- int berDecStrmUInt16 (OSCTXT * pctxt, OSUINT16 * object_p, ASN1TagType tagging, int length)
- int berDecStrmUInt64 (OSCTXT * pctxt, OSUINT64 * object_p, ASN1TagType tagging, int length)
- int berDecStrmUnivStr (OSCTXT * pctxt, Asn132BitCharString * object_p, ASN1TagType tagging, int length)

Function Documentation

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.

Table 2.197. 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.

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.

Table 2.198. 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.

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.

Table 2.199. 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.

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

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

Table 2.200. 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.

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.

Table 2.201. 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.

int berDecStrmCharArray (OSCTXT *pctxt, char *charArray, OSSIZE arraySize, ASN1TagType tagging, ASN1TAG tag, int length)

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

Table 2.202. Parameters

pctxt	Pointer to ASN.1 context block structure
charArray	Static character array variable (char[]) large enough to hold decoded data plus a null-termination byte. If the array is not large enoungh, an RTERR_TOOBIG error status will be returned.
arraySize	Size (in bytes) of the charArray variable.
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.

int berDecStrmDateStr (OSCTXT *pctxt, const char **ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)

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

Table 2.203. 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.

int berDecStrmDateTimeStr (OSCTXT *pctxt, const char **ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)

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

Table 2.204. 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.

int berDecStrmDurationStr (OSCTXT *pctxt, const char **ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)

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

Table 2.205. 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.

int berDecStrmTimeStr (OSCTXT *pctxt, const char **ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)

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

Table 2.206. 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.

int berDecStrmTimeOfDayStr (OSCTXT *pctxt, const char **ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)

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

Table 2.207. 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.

OSBOOL berDecStrmCheckEnd (OSCTXT *pctxt, ASN1CCB *pccb)

This function tests for the end of a constructed context. It is used in loop control statements to determine when a block of repeating elements has ended.

Table 2.208. 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.
pccb	Pointer to a 'context control block' structure. This is basically a loop control mechanism to keep the variable associated with parsing a nested constructed element straight.

Returns: . A result of testing:

- TRUE, if end of current context has been reached;
- FALSE, otherwise.

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.

Table 2.209. 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.

int berDecStrmDynBitStr64 (OSCTXT *pctxt, const OSOCTET **ppvalue, OSSIZE *pnbits, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)

64-bit version of berDecStrmDynBitStr.

Table 2.210. 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.
indefLen	Boolean indicating component is indefinite length.

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

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.

Table 2.211. 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.

int berDecStrmDynOctStr64 (OSCTXT *pctxt, OSOCTET **ppvalue, OSSIZE *pnocts, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)

This version of berDecStrmDynOctStr provides true 64-bit support by using an OSSIZE type (by default, an abstraction for size_t) to hold the decoded number of octets. On 64-bit systems, this will be a 64-bit number. This also fixes the issue of using a const pointer for the returned data since the variable is mutable.

Table 2.212. 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	A pointer to a pointer (**) to hold the address of a byte buffer to receive decoded OCTET STRING content.
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.
indefLen	Boolean indicating component is indefinite length.

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

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

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

Table 2.213. 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.

int berDecStrmFindTag (OSCTXT *pctxt, ASN1TAG tag, int *len_p, OSBOOL advance)

This function searches the stream being decoded for the first match of the given tag from the current stream position. If the tag is fount, the decode stream position will be set to either the start of the tag or to the start of the contents

depending on the value of the advance flag. If not found, the stream positiion will be returned to the position from which the search started.

Table 2.214. 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.

int berDecStrmFindTag2 (OSCTXT *pctxt, ASN1TAG tag, OSSIZE *len_p, OSBOOL *pIndefLen, OSBOOL advance)

This version of berDecStrmFindTag supports messages with length fields longer than will fit in a signed 32-bit integer value (up to unsigned 64-bit lengths on 64-bit machines).

Table 2.215. 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.
pIndefLen	Pointer to boolean variable which will be set to true if length is indefinite. Length returned in pLength should be disregarded in this case.
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.

int berDecStrmGetTLVLength (OSCTXT *pctxt)

This function returns the length of the tag-length-value at the current position in the stream.

Table 2.216. 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: . Total length (in bytes) of TLV component or negative error code.

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

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

Table 2.217. 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.

int berDecStrmInt8 (OSCTXT *pctxt, OSINT8 *object_p, ASN1TagType tagging, int length)

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

Table 2.218. 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.

int berDecStrmInt16 (OSCTXT *pctxt, OSINT16 *object_p, ASN1TagType tagging, int length)

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

Table 2.219. 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.

int berDecStrmInt64 (OSCTXT *pctxt, OSINT64 *object_p, ASN1TagType tagging, int length)

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

Table 2.220. 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.

int berDecStrmLength (OSCTXT *pctxt, int *len_p)

This function decodes a BER length determinant value.

Table 2.221. 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_INDEFLEN', which indicates indefinite length.

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

int berDecStrmLength2 (OSCTXT *pctxt, OSSIZE *pLength, OSBOOL *pIndefLen)

This function decodes a BER length determinant value. This version of the function can support lengths larger than can be held in a signed 32-bit integer value (up to 64-bits unsigned on a 64-bit machine).

Table 2.222. 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.
pLength	Pointer to a variable to receive the decoded length of the tagged component.
pIndefLen	Pointer to boolean variable which will be set to true if length is indefinite. Length returned in pLength should be disregarded in this case.

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

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.

Table 2.223. 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.

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.

Table 2.224. 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.

int berDecStrmMatchTag2 (OSCTXT *pctxt, ASN1TAG tag, OSSIZE *len_p, OSBOOL *pIndefLen, OSBOOL advance)

This function does a comparison between the given tag and the tag at the current decode pointer position to determine if they match. It then returns the result of the match operation.

This version of the function can support lengths larger than can be held in a signed 32-bit integer value (up to 64-bits unsigned on a 64-bit machine).

Table 2.225. 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.
pIndefLen	Pointer to boolean variable which will be set to true if length is indefinite. Length returned in len_p should be disregarded in this case.
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.

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.

Table 2.226. 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.

int berDecStrmNull (OSCTXT *pctxt, ASN1TagType tagging)

This function decodes an ASN.1 NULL placeholder.

Table 2.227. 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.

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

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

Table 2.228. 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.

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

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

Table 2.229. 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 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.

int berDecStrmOctStr (OSCTXT *pctxt, 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.

Table 2.230. 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 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.

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

This function decodes a variable of an ASN.1 open type. This includes the now deprecated ANY and ANY DEFINED BY types from the 1990 standard as well as other types defined to be open in the new standards (for example, a variable type declaration in an X.681 Information Object Class definition).

Decoding is accomplished by returning a pointer to the encoded message component at the current decode pointer location and skipping to the next field. The caller must then call additional decode functions to further decode the component.

Table 2.231. 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 a size-typed variable to receive the decoded number of octets.

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

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.

Table 2.232. 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.

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

This function is similar to the berDecStrmOpenType function except that it is used in places where open type extensions are specified. An open type extension is defined as an extensibility marker on a constructed type without any extension elements defined (for example, SEQUENCE { a INTEGER, : }). The difference is that this is an implicit field that can span one or more elements whereas the standard Open Type is assumed to be a single tagged field.

Table 2.233. 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.
tags	Array of next expected tag values (null if last field). The routine will loop through elements until a matching tag is found or some other error occurs.
tagCount	The number of tags in the tags array.
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.

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.

Table 2.234. 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_INDEFLEN', which indicates indefinite length.

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

int berDecStrmReadDynTLV (OSCTXT *pctxt, OSOCTET **ppbuf)

This function reads a complete tag-length-value (TLV) from the decode stream into a dynamic memory buffer.

Table 2.235. 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.
ppbuf	Pointer to a pointer to a buffer to receive the allocated memory pointer. The memory is allocated using the rtxMem* memory management functions. This may be freed using rtxMemFree or rtxMemFreePtr.

Returns: . The total number of octets read into the buffer, or negative error code.

int berDecStrmReadTLV (OSCTXT *pctxt, OSOCTET *buf, OSSIZE bufsiz)

This function reads a complete tag-length-value (TLV) from the decode stream into the given memory buffer.

Table 2.236. 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	Pointer to a buffer to receive a data.
bufsiz	Size of the buffer.

Returns: . The total number of octets read into the buffer, or negative error code.

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

This function decodes a variable of the binary encoded ASN.1 REAL type. This function allows the encoding to be in any base. This is used for REAL values that could be base 2 or base 10, for BER.

Table 2.237. 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.

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

This function decodes a variable of the binary encoded ASN.1 REAL type. This function requires the encoding to be in base 2, 8, or 16. This is used for REAL values that must be base 2, for BER.

Table 2.238. 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.

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

This function decodes a variable of the binary encoded ASN.1 REAL type. This function requires the encoding to be in base 2. This is used for REAL values that must be base 2, for CER and DER.

Table 2.239. 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.

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.

Table 2.240. 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.

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

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

Table 2.241. 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.

int berDecStrmTag (OSCTXT *pctxt, ASN1TAG *tag_p)

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

Table 2.242. 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.

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.

Table 2.243. 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.

int berDecStrmTagAndLen2 (OSCTXT *pctxt, ASN1TAG *tag_p, OSSIZE *len_p, OSBOOL *pIndefLen)

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

This version of the function can support lengths larger than can be held in a signed 32-bit integer value (up to 64-bits unsigned on a 64-bit machine).

Table 2.244. 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.
pIndefLen	Pointer to boolean variable which will be set to true if length is indefinite. Length returned in len_p should be disregarded in this case.

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

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.

Table 2.245. 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.

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.

Table 2.246. 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.

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.

Table 2.247. 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.

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.

Table 2.248. 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.

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.

Table 2.249. 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.

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.

Table 2.250. 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.

int berDecStrmUnivStr (OSCTXT *pctxt, Asn132BitCharString *object_p, ASN1TagType tagging, int length)

This function decodes a variable an ASN.1 32-bit character UniversalString type.

Table 2.251. 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.

Chapter 3. Class Documentation

ASN1BERDecodeBuffer class Reference

`#include <asn1BerCppTypes.h>`

Protected Attributes

- const OSOCTET * mpMsgBuf
- OSSIZE mMsgBufLen
- OSBOOL mBufSetFlag

- ASN1BERDecodeBuffer ( )
- ASN1BERDecodeBuffer (const OSOCTET * pMsgBuf, OSSIZE msgBufLen)
- ASN1BERDecodeBuffer (const OSOCTET * pMsgBuf, OSSIZE msgBufLen, OSRTContext * pContext)
- OSOCTET * findElement (ASN1TAG tag, OSINT32 & elemLen, OSBOOL firstFlag)
- virtual const OSOCTET * getMsgPtr ( )
- int init ( )
- virtual OSBOOL isA (Type bufferType)
- int parseTagLen (ASN1TAG & tag, int & msglen)
- int parseTagLen (ASN1TAG & tag, OSSIZE & msglen, OSBOOL * pIndefLen)
- int parseTagLen (ASN1TAG & tag, ASN1BERLength & msglen)
- int readBinaryFile (const char * filePath)
- int setBuffer (const OSOCTET * pMsgBuf, OSSIZE msgBufLen)
- OSOCTET * FindElement (ASN1TAG tag, int & elemLen, int firstFlag)
- int ParseTagLen (ASN1TAG & tag, int & msglen)
- ASN1BERDecodeBuffer & operator>> (ASN1CType & val)

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.

Definition at line 431 of file asn1BerCppTypes.h

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

- asn1BerCppTypes.h

ASN1BERDecodeBuffer::ASN1BERDecodeBuffer ()

Default constructor. Use the getStatus() method to determine if an error occured during initialization or not.

ASN1BERDecodeBuffer::ASN1BERDecodeBuffer (const OSOCTET *pMsgBuf, OSSIZE msgBufLen)

Parameterized constructor. This constructor constructs a buffer describing an encoded ASN.1 message. Parameters describing the message to be decoded are passed as arguments.

Table 3.1. 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.

ASN1BERDecodeBuffer::ASN1BERDecodeBuffer (const OSOCTET *pMsgBuf, OSSIZE msgBufLen, OSRTContext *pContext)

Parameterized constructor. This constructor constructs a buffer describing an encoded ASN.1 message. Parameters describing the message to be decoded are passed as arguments.

Table 3.2. 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.
pContext	A pointer to an existing OSRTContext structure.

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.

Table 3.3. 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.

virtual const OSOCTET* ASN1BERDecodeBuffer::getMsgPtr ()

This method returns the internal pointer to the current message that has been set to be decoded.

Returns: . Pointer to message.

int ASN1BERDecodeBuffer::init ()

Initializes message buffer.

Returns: . Completion status of operation:

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

virtual OSBOOL ASN1BERDecodeBuffer::isA (Type bufferType)

This method checks the type of the message buffer.

Table 3.4. 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.

int ASN1BERDecodeBuffer::parseTagLen (ASN1TAG &tag, int &msglen)

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.

Table 3.5. 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.

int ASN1BERDecodeBuffer::parseTagLen (ASN1TAG &tag, OSSIZE &msglen, OSBOOL *pIndefLen)

This method overloaded version of parseTagLen will parse the initial tag-length pair from the message. In this case, the length is returned in a size-typed variable which will hold lengths up to 64 bits in size.

Calling Sequence:

```
stat = decodeBuffer.parseTagLen (tag, msglen);
```

where decodeBuffer is an ASN1BERDecodeBuffer object.

Returns: . Status of the operation. Possible values are 0 if successful or one of the negative error status codes defined in Appendix A of the C/C++ Common Functions Reference Manual.

Table 3.6. 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.
pIndefLen	Boolean flag indicating parsed length is indefinite.

Returns: . Completion status of operation:

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

int ASN1BERDecodeBuffer::parseTagLen (ASN1TAG &tag, ASN1BERLength &msglen)

This method overloaded version of parseTagLen will parse the initial tag-length pair from the message. This variant allows a reference to a BER length object to be used for the length. Lengths up to 64 bits in size are supported.

Calling Sequence:

```
stat = decodeBuffer.parseTagLen (tag, msglen);
```

where decodeBuffer is an ASN1BERDecodeBuffer object.

Returns: . Status of the operation. Possible values are 0 if successful or one of the negative error status codes defined in Appendix A of the C/C++ Common Functions Reference Manual.

Table 3.7. Parameters

tag	Reference to a tag structure to receive the outer level tag value parsed from the message.
msglen	Reference to an object used to describe the length of the message.

Returns: . Completion status of operation:

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

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.

Table 3.8. 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.

int ASN1BERDecodeBuffer::setBuffer (const OSOCTET *pMsgBuf, OSSIZE msgBufLen)

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

Table 3.9. 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.

ASN1BERDecodeBuffer& ASN1BERDecodeBuffer::operator>> (ASN1CType &val)

This operator decodes an instance of an ASN1CType derived class. Use the getStatus() method to determine if an error occured during the operation or not.

ASN1BERDecodeStream class Reference

`#include <ASN1BERDecodeStream.h>`

Protected Attributes

- OSRTInputStreamIF * mpStream
- OSBOOL mbOwnStream

- ASN1BERDecodeStream (OSRTInputStreamIF & is)
- ASN1BERDecodeStream (OSRTInputStreamIF * pis, OSBOOL bOwnStream)
- ~ASN1BERDecodeStream ( )
- virtual void * getAppInfo ( )
- virtual OSRTCtxtPtr getContext ( )
- virtual OSCTXT * getCtxtPtr ( )
- virtual char * getErrorInfo ( )
- virtual char * getErrorInfo (char * pBuf, size_t & bufSize)
- virtual int getStatus ( )
- virtual void printErrorInfo ( )
- virtual void resetErrorInfo ( )
- virtual void setAppInfo (void * pAppInfo)
- virtual void setDiag (OSBOOL value)
- virtual int close ( )
- virtual int flush ( )
- virtual int getPosition (size_t * ppos)
- virtual OSBOOL isOpened ( )

- virtual size_t currentPos ( )
- virtual OSBOOL markSupported ( )
- int mark (size_t readAheadLimit)
- virtual long read (OSOCTET * pDestBuf, size_t maxToRead)
- virtual long readBlocking (OSOCTET * pDestBuf, size_t toReadBytes)
- int reset ( )
- virtual int setPosition (size_t pos)
- virtual int skip (size_t n)
- ASN1BERDecodeStream & operator>> (ASN1CType & val)
- size_t byteIndex ( )
- OSBOOL chkend (ASN1CCB & ccb)
- int decodeBigInt (const char *& pval, ASN1TagType tagging, int length)
- int decodeBitStr (OSOCTET * pbits, OSUINT32 & numbits, ASN1TagType tagging, int length)
- int decodeBitStr (ASN1DynBitStr & val, ASN1TagType tagging, int length)
- int decodeBMPStr (Asn116BitCharString & val, ASN1TagType tagging, int length)
- int decodeBool (OSBOOL & val, ASN1TagType tagging, int length)
- int decodeCharStr (const char *& pval, ASN1TagType tagging, ASN1TAG tag, int length)
- int decodeEnum (OSINT32 & val, ASN1TagType tagging, int length)
- int decodeEoc ( )
- int decodeInt (OSINT32 & val, ASN1TagType tagging, int length)
- int decodeInt8 (OSINT8 & val, ASN1TagType tagging, int length)
- int decodeInt16 (OSINT16 & val, ASN1TagType tagging, int length)
- int decodeInt64 (OSINT64 & val, ASN1TagType tagging, int length)
- int decodeLength (OSINT32 & length)
- int decodeNull (ASN1TagType tagging)
- int decodeObj (ASN1CType & val)
- int decodeObjId (ASN1OBJID & val, ASN1TagType tagging, int length)
- int decodeObjId64 (ASN1OID64 & val, ASN1TagType tagging, int length)
- int decodeOctStr (OSOCTET * pocts, OSUINT32 & numocts, ASN1TagType tagging, int length)

- int decodeOctStr (ASN1DynOctStr & val, ASN1TagType tagging, int length)
- int decodeOctStr (OSDynOctStr64 & val, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int decodeOpenType (ASN1OpenType & val)
- int decodeReal (OSREAL & val, ASN1TagType tagging, int length)
- int decodeRelativeOID (ASN1OBJID & val, ASN1TagType tagging, int length)
- int decodeTag (ASN1TAG & tag)
- int decodeTagAndLen (ASN1TAG & tag, OSINT32 & len)
- int decodeTagAndLen (ASN1TAG & tag, ASN1BERLength & len)
- int decodeUInt (OSUINT32 & val, ASN1TagType tagging, int length)
- int decodeUInt8 (OSUINT8 & val, ASN1TagType tagging, int length)
- int decodeUInt16 (OSUINT16 & val, ASN1TagType tagging, int length)
- int decodeUInt64 (OSUINT64 & val, ASN1TagType tagging, int length)
- int decodeUnivStr (Asn132BitCharString & val, ASN1TagType tagging, int length)
- int getTLVLength ( )
- OSBOOL isA (Type bufferType)
- int peekTagAndLen (ASN1TAG & tag, int & len)
- int readTLV (OSOCTET * pDestBuf, size_t bufsiz)

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.

Definition at line 50 of file ASN1BERDecodeStream.h

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

- ASN1BERDecodeStream.h

ASN1BERDecodeStream::ASN1BERDecodeStream (OSRTInputStreamIF &is)

A default constructor. Use getStatus() method to determine if an error occured during the initialization or not.

virtual void* ASN1BERDecodeStream::getAppInfo ()

Returns a pointer to application-specific information block

virtual OSRTCtxtPtr ASN1BERDecodeStream::getContext ()

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

Returns: . Context smart pointer object.

virtual OSCTXT* ASN1BERDecodeStream::getCtxtPtr ()

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.

virtual char* ASN1BERDecodeStream::getErrorInfo ()

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

Returns: . A pointer to a newly allocated buffer with error text.

virtual char* ASN1BERDecodeStream::getErrorInfo (char *pBuf, size_t &bufSize)

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

Table 3.10. 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 or NULL, if error.

virtual int ASN1BERDecodeStream::getStatus () const

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.

virtual void ASN1BERDecodeStream::printErrorInfo ()

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

virtual void ASN1BERDecodeStream::resetErrorInfo ()

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

virtual void ASN1BERDecodeStream::setAppInfo (void *pAppInfo)

Sets the application-specific information block.

virtual void ASN1BERDecodeStream::setDiag (OSBOOL value=TRUE)

The setDiag method will turn diagnostic tracing on or off.

Table 3.11. Parameters

value	- Boolean value (default = TRUE = on)

virtual int ASN1BERDecodeStream::close ()

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

virtual int ASN1BERDecodeStream::flush ()

Flushes the buffered data to the stream.

Returns: . Completion status of operation:

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

See also: . ::rtxStreamFlush

virtual int ASN1BERDecodeStream::getPosition (size_t *ppos)

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

Table 3.12. Parameters

ppos	Pointer to a variable to receive position.

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

virtual OSBOOL ASN1BERDecodeStream::isOpened ()

Checks, is the stream opened or not.

Returns: . TRUE, if the stream is opened, FALSE otherwise.

See also: . ::rtxStreamIsOpened

virtual size_t ASN1BERDecodeStream::currentPos ()

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

Returns: . The number of octets already read from the stream.

virtual OSBOOL ASN1BERDecodeStream::markSupported ()

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

int ASN1BERDecodeStream::mark (size_t readAhead-Limit)

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

Table 3.13. 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

virtual long ASN1BERDecodeStream::read (OSOCTET *pDestBuf, size_t maxToRead)

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

Table 3.14. 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

virtual long ASN1BERDecodeStream::readBlocking (OSOCTET *pDestBuf, size_t toReadBytes)

Read data from the stream. This method reads `maxToRead` bytes from the stream. It will block until either the bytes are available or an error occurs.

Table 3.15. 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 error code.

See also: . ::rtxStreamReadBlocking

int ASN1BERDecodeStream::reset ()

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

virtual int ASN1BERDecodeStream::setPosition (size_t pos)

Sets the current stream position to the given offset.

Table 3.16. Parameters

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

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

virtual int ASN1BERDecodeStream::skip (size_t n)

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

Table 3.17. Parameters

n	The number of octets to be skipped.

Returns: . Completion status of operation:

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

See also: . ::rtxStreamSkip

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

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

Table 3.18. Parameters

val	A reference to an object to be decoded.

Returns: . reference to this class to perform sequential decoding.

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.

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.

Table 3.19. 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.

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).

Table 3.20. 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

int ASN1BERDecodeStream::decodeBitStr (OSOCTET *pbits, OSUINT32 &numbits, ASN1TagType tagging=ASN1EXPL, int length=0)

This method decodes a variable of the ASN.1 BIT STRING type into a static memory structure. It is used to decode a sized bit string production.

Table 3.21. 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

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.

Table 3.22. 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

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

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

Table 3.23. 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

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

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

Table 3.24. 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

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.

Table 3.25. 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

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

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

Table 3.26. 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

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

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

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

Table 3.27. 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

int ASN1BERDecodeStream::decodeInt8 (OSINT8 &val, ASN1TagType tagging=ASN1EXPL, int length=0)

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

Table 3.28. 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

int ASN1BERDecodeStream::decodeInt16 (OSINT16 &val, ASN1TagType tagging=ASN1EXPL, int length=0)

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

Table 3.29. 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

int ASN1BERDecodeStream::decodeInt64 (OSINT64 &val, ASN1TagType tagging=ASN1EXPL, int length=0)

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

Table 3.30. 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

int ASN1BERDecodeStream::decodeLength (OSINT32 &length)

This method decodes a BER length determinant value.

Table 3.31. 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

int ASN1BERDecodeStream::decodeNull (ASN1TagType tagging=ASN1EXPL)

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

Table 3.32. 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

int ASN1BERDecodeStream::decodeObj (ASN1CType &val)

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

Table 3.33. Parameters

val	A reference to an object to be decoded.

Returns: . Completion status of operation:

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

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

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

Table 3.34. 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

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..

Table 3.35. 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

int ASN1BERDecodeStream::decodeOctStr (OSOCTET *pocts, OSUINT32 &numocts, ASN1TagType tagging=ASN1EXPL, int length=0)

This method decodes a variable of the ASN.1 OCTET STRING type into a static memory structure. It is used to decode a sized octet string production.

Table 3.36. 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

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.

Table 3.37. 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

int ASN1BERDecodeStream::decodeOctStr (OSDynOctStr64 &val, ASN1TagType tagging=ASN1EXPL, OSSIZE length=0, OSBOOL indefLen=FALSE)

This method decodes a variable of the ASN.1 OCTET STRING type. It will allocate dynamic memory to store the decoded result. It supports 64-bit length variables.

Table 3.38. 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.
indefLen	Flag indicating if component to be decoded is indefinite length. This parameter only has meaning if the tagging parameter specifies implicit decoding. If explicit, the length is obtained from the decoded length field.

Returns: . Completion status of operation:

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

See also: . berDecStrmDynOctStr64

int ASN1BERDecodeStream::decodeOpenType (ASN1OpenType &val)

This method decodes an ASN.1 open type value. This is a value of any ASN.1 data type. It may be constructed and contain multiple elements including elements encoded with indefinite lengths. This method will allocate dynamic memory to store the decoded result.

Table 3.39. Parameters

val	Reference to an ASN1OpenType variable to receive the decoded open type data.

Returns: . Completion status of operation:

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

See also: . berDecStrmDynOctStr

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

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

Table 3.40. 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

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

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

Table 3.41. 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

int ASN1BERDecodeStream::decodeTag (ASN1TAG &tag)

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

Table 3.42. 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

int ASN1BERDecodeStream::decodeTagAndLen (ASN1TAG &tag, OSINT32 &len)

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

Table 3.43. 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

int ASN1BERDecodeStream::decodeTagAndLen (ASN1TAG &tag, ASN1BERLength &len)

This method decodes the tag and length at the current decode pointer location and returns the results. This variant of the method supports 64-bit length values.

Table 3.44. Parameters

tag	Reference to a variable to receive the decoded ASN.1 tag value.
len	Reference to an object to receive the decoded length of the tagged component.

Returns: . Completion status of operation:

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

See also: . berDecStrmTagAndLen

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.

Table 3.45. 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

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.

Table 3.46. 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

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.

Table 3.47. 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

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.

Table 3.48. 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

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

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

Table 3.49. 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

int ASN1BERDecodeStream::getTLVLength ()

Get the total length of a tag-length-value (TLV) component. This is not the length stored in the L field. It is the total length of the component which is equal to the parsed length plus the number of bytes in the tag and length fields.

Returns: . The total number of octets in the TLV or a negative error code.

See also: . berDecStrmGetTLVLength

OSBOOL ASN1BERDecodeStream::isA (Type bufferType)

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

Table 3.50. 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, XMLEncode, XMLDecode, Stream.

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

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.

Table 3.51. 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

int ASN1BERDecodeStream::readTLV (OSOCTET *pDestBuf, size_t bufsiz)

Read a complete tag-length-value (TLV) from the decode stream into the given memory buffer.

Table 3.52. Parameters

pDestBuf	Pointer to a buffer to receive a data.
bufsiz	Size of the buffer.

Returns: . The total number of octets read into the buffer, or negative error code.

See also: . berDecStrmReadTLV

ASN1BEREncodeBuffer class Reference

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

- ASN1BEREncodeBuffer ( )
- ASN1BEREncodeBuffer (OSOCTET * pMsgBuf, OSSIZE msgBufLen)
- ASN1BEREncodeBuffer (OSOCTET * pMsgBuf, OSSIZE msgBufLen, OSRTContext * pContext)
- int encodeBool (OSBOOL val, ASN1TagType tagging)
- int encodeBigIntNchars (const char * pval, size_t nchars, ASN1TagType tagging)
- int encodeBigInt (const char * pval, ASN1TagType tagging)
- int encodeObjId (ASN1OBJID * pval, ASN1TagType tagging)
- void freeBuffer ( )
- virtual OSOCTET * getMsgCopy ( )
- virtual const OSOCTET * getMsgPtr ( )
- virtual size_t getMsgLen ( )
- int init ( )
- virtual OSBOOL isA (Type bufferType)
- int setBuffer (OSOCTET * pMsgBuf, OSSIZE msgBufLen)
- ASN1BEREncodeBuffer & operator<< (ASN1CType & val)

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.

Definition at line 199 of file asn1BerCppTypes.h

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

- asn1BerCppTypes.h

ASN1BEREncodeBuffer::ASN1BEREncodeBuffer ()

Default constructor. This sets all internal variables to their initial values. Use the getStatus() method to determine if an error occured during initialization or not.

ASN1BEREncodeBuffer::ASN1BEREncodeBuffer (OSOCTET *pMsgBuf, OSSIZE msgBufLen)

Parameterized constructor. This version takes a message buffer and size argument (static encoding version). Use the getStatus() method to determine if an error occured during initialization or not.

Table 3.53. Parameters

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

ASN1BEREncodeBuffer::ASN1BEREncodeBuffer (OSOCTET *pMsgBuf, OSSIZE msgBufLen, OSRTContext *pContext)

Parameterized constructor. This version takes a message buffer, a size argument (static encoding version), and a context. Use the getStatus() method to determine if an error occured during initialization or not.

Table 3.54. Parameters

pMsgBuf	A pointer to a fixed size message buffer to recieve the encoded message.
msgBufLen	Size of the fixed-size message buffer.
pContext	A pointer to an existing OSRTContext structure.

int ASN1BEREncodeBuffer::encodeBool (OSBOOL val, ASN1TagType tagging=ASN1EXPL)

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

Table 3.55. Parameters

val	BOOLEAN 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.

int ASN1BEREncodeBuffer::encodeBigIntNchars (const char *pval, size_t nchars, ASN1TagType tagging=ASN1EXPL)

This method encodes a variable of the ASN.1 INTEGER type. In this case, the integer may be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits). This variant of the method allows the number of characters to encode to be passed in.

Table 3.56. Parameters

*pval	A pointer to a character string containing the value to be encoded.
nchars	Number of characters from pval to encode.
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.

int ASN1BEREncodeBuffer::encodeBigInt (const char *pval, ASN1TagType tagging=ASN1EXPL)

This method encodes a variable of the ASN.1 INTEGER type. In this case, the integer may be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits).

Table 3.57. 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.

int ASN1BEREncodeBuffer::encodeObjId (ASN1OBJID *pval, ASN1TagType tagging=ASN1EXPL)

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

Table 3.58. Parameters

val	Pointer to object identifier structure. This structure contains an integer to hold the number of subidentifers 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

void ASN1BEREncodeBuffer::freeBuffer ()

This method releases memory that was allocated for a dynamic encode buffer.

virtual OSOCTET* ASN1BEREncodeBuffer::getMsgCopy ()

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;).

virtual const OSOCTET* ASN1BEREncodeBuffer::getMsgPtr ()

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.

virtual size_t ASN1BEREncodeBuffer::getMsgLen ()

This method returns the length of the current encoded message. Calling Sequence:

```
len = encodeBuffer.getMsgLen ();
```

where encodeBuffer is an ASN1BEREncodeBuffer object.

Returns: . Encoded message length;

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.

virtual OSBOOL ASN1BEREncodeBuffer::isA (Type bufferType)

This method checks the type of the message buffer.

Table 3.59. 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.

int ASN1BEREncodeBuffer::setBuffer (OSOCTET *pMsg-Buf, OSSIZE msgBufLen)

This method sets a buffer to receive the encoded message.

Table 3.60. 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 (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).
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.

ASN1BEREncodeBuffer& ASN1BEREncodeBuffer::operator<< (ASN1CType &val)

This operator encodes instance of ASN1CType derived class. Use getStatus() method to determine has error occured during the operation or not.

ASN1BEREncodeStream class Reference

`#include <ASN1BEREncodeStream.h>`

Protected Attributes

- OSRTOutputStreamIF * mpStream
- OSBOOL mbOwnStream

- ASN1BEREncodeStream (OSRTOutputStreamIF & os)
- ASN1BEREncodeStream (OSRTOutputStreamIF * pos, OSBOOL bOwnStream)
- ~ASN1BEREncodeStream ( )
- virtual void * getAppInfo ( )
- virtual OSRTCtxtPtr getContext ( )
- virtual OSCTXT * getCtxtPtr ( )
- virtual char * getErrorInfo ( )
- virtual char * getErrorInfo (char * pBuf, size_t & bufSize)
- virtual int getStatus ( )
- virtual void printErrorInfo ( )
- virtual void resetErrorInfo ( )

- virtual void setAppInfo (void * pAppInfo)
- virtual void setDiag (OSBOOL value)
- virtual int close ( )
- virtual int flush ( )
- virtual OSBOOL isOpened ( )
- virtual long write (const OSOCTET * pdata, size_t size)
- ASN1BEREncodeStream & operator<< (ASN1CType & val)
- int encodeBMPStr (const Asn116BitCharString & val, ASN1TagType tagging)
- int encodeBigInt (const char * pval, ASN1TagType tagging)
- int encodeBigIntNchars (const char * pval, size_t nchars, ASN1TagType tagging)
- int encodeBitStr (const OSOCTET * pbits, OSUINT32 numbits, ASN1TagType tagging)
- int encodeBitStr (const ASN1DynBitStr & val, ASN1TagType tagging)
- int encodeBool (OSBOOL val, ASN1TagType tagging)
- int encodeCharStr (const char * pval, ASN1TagType tagging, ASN1TAG tag)
- int encodeEnum (OSINT32 val, ASN1TagType tagging)
- int encodeEoc ( )
- int encodeIndefLen ( )
- int encodeInt (OSINT32 val, ASN1TagType tagging)
- int encodeInt8 (OSINT8 val, ASN1TagType tagging)
- int encodeInt16 (OSINT16 val, ASN1TagType tagging)
- int encodeInt64 (OSINT64 val, ASN1TagType tagging)
- int encodeLen (size_t len)
- int encodeNull (ASN1TagType tagging)
- int encodeObj (ASN1CType & val)
- int encodeObjId (const ASN1OBJID & val, ASN1TagType tagging)
- int encodeObjId64 (const ASN1OID64 & val, ASN1TagType tagging)
- int encodeOctStr (const OSOCTET * pocts, OSSIZE numocts, ASN1TagType tagging)
- int encodeOctStr (const ASN1DynOctStr & val, ASN1TagType tagging)
- int encodeReal (OSREAL val, ASN1TagType tagging)

- int encodeRelativeOID (const ASN1OBJID & val, ASN1TagType tagging)
- int encodeTag (ASN1TAG tag)
- int encodeTagAndIndefLen (ASN1TAG tag)
- int encodeTagAndLen (ASN1TAG tag, OSINT32 len)
- int encodeUInt (OSUINT32 val, ASN1TagType tagging)
- int encodeUInt8 (OSUINT8 val, ASN1TagType tagging)
- int encodeUInt16 (OSUINT16 val, ASN1TagType tagging)
- int encodeUInt64 (OSUINT64 val, ASN1TagType tagging)
- int encodeUnivStr (const Asn132BitCharString & val, ASN1TagType tagging)
- OSBOOL isA (Type bufferType)

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.

Definition at line 49 of file ASN1BEREncodeStream.h

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

- ASN1BEREncodeStream.h

ASN1BEREncodeStream::ASN1BEREncodeStream (OSRTOutputStreamIF &os)

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

virtual void* ASN1BEREncodeStream::getAppInfo ()

Returns a pointer to application-specific information block

virtual OSRTCtxtPtr ASN1BEREncodeStream::getContext ()

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

Returns: . Context smart pointer object.

virtual OSCTXT* ASN1BEREncodeStream::getCtxtPtr ()

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.

virtual char* ASN1BEREncodeStream::getErrorInfo ()

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.

virtual char* ASN1BEREncodeStream::getErrorInfo (char *pBuf, size_t &bufSize)

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

Table 3.61. 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 occured.

virtual int ASN1BEREncodeStream::getStatus () const

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.

virtual void ASN1BEREncodeStream::printErrorInfo ()

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

virtual void ASN1BEREncodeStream::resetErrorInfo ()

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

virtual void ASN1BEREncodeStream::setAppInfo (void *pAppInfo)

Sets the application-specific information block.

virtual void ASN1BEREncodeStream::setDiag (OSBOOL value=TRUE)

The setDiag method will turn diagnostic tracing on or off.

Table 3.62. Parameters

value	- Boolean value (default = TRUE = on)

virtual int ASN1BEREncodeStream::close ()

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

virtual int ASN1BEREncodeStream::flush ()

Flushes the buffered data to the stream.

Returns: . Completion status of operation:

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

See also: . ::rtxStreamFlush

virtual OSBOOL ASN1BEREncodeStream::isOpened ()

Checks, is the stream opened or not.

Returns: . TRUE, if the stream is opened, FALSE otherwise.

See also: . ::rtxStreamIsOpened

virtual long ASN1BEREncodeStream::write (const OSOCTET *pdata, size_t size)

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

Table 3.63. 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

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

Encodes an ASN.1 constructed object to the stream. Use getStatus() method to determine has error occured during the operation or not.

Table 3.64. Parameters

val	A reference to an object to be encoded.

Returns: . reference to this class to perform sequential encoding.

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.

Table 3.65. 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

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).

Table 3.66. 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

int ASN1BEREncodeStream::encodeBigIntNchars (const char *pval, size_t nchars, ASN1TagType tagging=ASN1EXPL)

This method encodes a variable of the ASN.1 INTEGER type. In this case, the integer is assumed to be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits).

Table 3.67. Parameters

*pval	A pointer to a character string containing the value to be encoded.
nchars	Number of characters from pval to encode.
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

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

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

Table 3.68. 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

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

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

Table 3.69. 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

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

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

Table 3.70. 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

int ASN1BEREncodeStream::encodeCharStr (const char *pval, ASN1TagType tagging=ASN1EXPL, ASN1TAG tag=0)

This method encodes a variable of the ASN.1 character string type.

Table 3.71. 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

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.

Table 3.72. 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

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,

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

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

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

Table 3.73. 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

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

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

Table 3.74. 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

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

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

Table 3.75. 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

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

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

Table 3.76. 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

int ASN1BEREncodeStream::encodeLen (size_t len)

This method is used to encode a length in BER format.

Table 3.77. Parameters

len	The length of the contents field.

Returns: . Completion status of operation:

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

See also: . berEncStrmLength

int ASN1BEREncodeStream::encodeNull (ASN1TagType tagging=ASN1EXPL)

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

Table 3.78. 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

int ASN1BEREncodeStream::encodeObj (ASN1CType &val)

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

Table 3.79. Parameters

val	A reference to an object to be encoded.

Returns: . Completion status of operation:

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

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

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

Table 3.80. Parameters

val	A reference to an object identifier structure. This structure contains an integer to hold the number of subidentifers 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

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.

Table 3.81. Parameters

val	A reference to a 64-bit object identifier structure. This structure contains an integer to hold the number of subidentifers 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

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

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

Table 3.82. Parameters

pocts	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

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

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

Table 3.83. 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

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.

Table 3.84. 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

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

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

Table 3.85. Parameters

val	A reference to an object identifier structure. This structure contains an integer to hold the number of subidentifers 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

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.

Table 3.86. 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

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.

Table 3.87. 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

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.

Table 3.88. 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

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

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

Table 3.89. 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

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

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

Table 3.90. 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

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

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

Table 3.91. 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

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

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

Table 3.92. 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

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.

Table 3.93. 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

OSBOOL ASN1BEREncodeStream::isA (Type bufferType)

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

Table 3.94. 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, XMLEncode, XMLDecode, Stream.

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

ASN1BERLength class Reference

Protected Attributes

- OSSIZE mLength
- OSBOOL mbIndef
-
-
- ASN1BERLength ( )
- ASN1BERLength (size_t length, OSBOOL indef)
- ASN1BERLength (int length)

- int decodeLen (OSCTXT * pctxt)
- int decodeTagLen (OSCTXT * pctxt, ASN1TAG * ptag, OSOCTET flags)
- int getIntLength ( )
- int getElemCount (OSCTXT * pctxt, OSSIZE & count)
- OSSIZE getLength ( )
- OSBOOL isEOC (ASN1TAG tag)
- OSBOOL isIndef ( )
- int matchTag (OSCTXT * pctxt, OSOCTET tag)
- int matchTag (OSCTXT * pctxt, ASN1TAG tag, OSOCTET flags)
- void setCCBLength (ASN1CCB & ccb)
- void setLength (size_t length)
- void setIndef (OSBOOL value)
- operator int ( )
- operator size_t ( )

ASN1BERMessageBuffer class Reference

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

- ASN1BERMessageBuffer (Type bufferType)
- ASN1BERMessageBuffer (Type bufferType, OSRTContext * pContext)

- int calcIndefLen (OSOCTET * buf_p, OSSIZE bufSize, OSSIZE * pSize)
- int calcIndefLen (OSOCTET * buf_p, int bufSize)
- void binDump ( )
- void hexDump (OSSIZE numocts)
- int CalcIndefLen (OSOCTET * buf_p)
- void BinDump ( )
- void HexDump (OSSIZE numocts)

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.

Definition at line 87 of file asn1BerCppTypes.h

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

- asn1BerCppTypes.h

ASN1BERMessageBuffer::ASN1BERMessageBuffer (Type bufferType)

The protected constructor creates a new context and sets the buffer class type. Use getStatus() method to determine has error occured during the initialization or not.

Table 3.95. Parameters

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

ASN1BERMessageBuffer::ASN1BERMessageBuffer (Type bufferType, OSRTContext *pContext)

The protected constructor uses an existing context and sets the buffer class type. Use getStatus() method to determine has error occured during the initialization or not.

Table 3.96. Parameters

bufferType	Type of message buffer that is being created (for example, BEREncode or BERDecode).
pContext	A pointer to the context structure with which this buffer is associated.

int ASN1BERMessageBuffer::calcIndefLen (OSOCTET *buf_p, OSSIZE bufSize, OSSIZE *pSize)

This method calculates the actual length of an indefinite length message component. This version of the method will calculate lengths up the full size of a size type (64 bits on 64-bit systems).

Calling Sequence:

```
len=messageBuffer.calcIndefLen (buf_p, bufSize, pSize);
```

where messageBuffer is an ASN1BERMessageBuffer derived class object.

Table 3.97. Parameters

buf_p	A pointer to a message component encoded using indefinite length encoding.
bufSize	Size of the buffer `buf_p` (in octets).
pSize	Pointer to size-typed variable to receive size.

Returns: . Zero if successful, or a negative status value if error.

int ASN1BERMessageBuffer::calcIndefLen (OSOCTET *buf_p, int bufSize=INT_MAX)

This method calculates the actual length of an indefinite length message component. This version of the method will only calculate lengths up the size of a signed integer (INT_MAX). It is considered to be deprecated.

Calling Sequence:

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

where messageBuffer is an ASN1BERMessageBuffer derived class object.

Table 3.98. 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.

void ASN1BERMessageBuffer::binDump ()

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.

void ASN1BERMessageBuffer::hexDump (OSSIZE numocts)

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.

ASN1MessageBuffer class Reference

ASN1TAGTEXT struct Reference

Public Attributes

- const char * class_p

- const char * form_p
- char idCode[20]

Chapter 4. File Documentation

asn1ber.h File Reference

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

#include "rtbersrc/berMacros.h"

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

Classes

- struct ASN1TAGTEXT

Macros

- #define xd_utf8str xd_charstr (pctxt, (const char**)object_p, tagging, ASN_ID_UTF8String, length)
- #define xd_indeflen xd_indeflen_ex(m, INT_MAX)
- #define xe_utf8str xe_charstr (pctxt, (const char*)object_p, tagging, ASN_ID_UTF8String)
- #define xu_addTagErrParm berErrAddTagParm
- #define xu_hex_dump rtxHexDump(msg,numoct)

Typedefs

- typedef OSRTBufLocDescr Asn1BufLocDescr
- typedef struct ASN1TAGTEXT ASN1TAGTEXT

Functions

- int xd_tag (OSCTXT * pctxt, ASN1TAG * tag_p)
- int xd_tag_len (OSCTXT * pctxt, ASN1TAG * tag_p, int * len_p, OSOCTET flags)
- int xd_tag_len_64 (OSCTXT * pctxt, ASN1TAG * tag_p, OSSIZE * len_p, OSBOOL * pIndefLen, OSOCTET flags)
- int xd_match (OSCTXT * pctxt, ASN1TAG tag, int * len_p, OSOCTET flags)
- int xd_match64 (OSCTXT * pctxt, ASN1TAG tag, OSSIZE * len_p, OSBOOL * pindef, OSOCTET flags)
- int xd_boolean (OSCTXT * pctxt, OSBOOL * object_p, ASN1TagType tagging, int length)
- int xd_integer (OSCTXT * pctxt, OSINT32 * object_p, ASN1TagType tagging, int length)
- int xd_int8 (OSCTXT * pctxt, OSINT8 * object_p, ASN1TagType tagging, int length)
- int xd_int16 (OSCTXT * pctxt, OSINT16 * object_p, ASN1TagType tagging, int length)
- int xd_unsigned (OSCTXT * pctxt, OSUINT32 * object_p, ASN1TagType tagging, int length)

- int xd_uint8 (OSCTXT * pctxt, OSUINT8 * object_p, ASN1TagType tagging, int length)
- int xd_uint16 (OSCTXT * pctxt, OSUINT16 * object_p, ASN1TagType tagging, int length)
- int xd_int64 (OSCTXT * pctxt, OSINT64 * object_p, ASN1TagType tagging, int length)
- int xd_uint64 (OSCTXT * pctxt, OSUINT64 * object_p, ASN1TagType tagging, int length)
- int xd_bigint (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, int length)
- int xd_bitstr_s (OSCTXT * pctxt, OSOCTET * object_p, OSUINT32 * numbits_p, ASN1TagType tagging, int length)
- int xd_bitstrExt_s (OSCTXT * pctxt, OSOCTET * object_p, OSUINT32 * numbits_p, OSOCTET ** extdata, ASN1TagType tagging, int length)
- int xd_bitstr64_s (OSCTXT * pctxt, OSOCTET * object_p, OSSIZE * numbits_p, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int xd_bitstr64Ext_s (OSCTXT * pctxt, OSOCTET * object_p, OSSIZE * numbits_p, OSOCTET ** extdata, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int xd_bitstr (OSCTXT * pctxt, const OSOCTET ** object_p2, OSUINT32 * numbits_p, ASN1TagType tagging, int length)
- int xd_bitstr64 (OSCTXT * pctxt, const OSOCTET ** object_p2, OSSIZE * numbits_p, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int xd_octstr_s (OSCTXT * pctxt, OSOCTET * object_p, OSUINT32 * pnumocts, ASN1TagType tagging, int length)
- int xd_octstr64_s (OSCTXT * pctxt, OSOCTET * object_p, OSSIZE * pnumocts, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int xd_octstr (OSCTXT * pctxt, const OSOCTET ** object_p2, OSUINT32 * pnumocts, ASN1TagType tagging, int length)
- int xd_octstr64 (OSCTXT * pctxt, OSOCTET ** object_p2, OSSIZE * pnumocts, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int xd_charstr (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int xd_charstr64 (OSCTXT * pctxt, char ** object_p, ASN1TagType tagging, ASN1TAG tag, OSSIZE length, OSBOOL indefLen)
- int xd_datestr (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int xd_timestr (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int xd_datetimestr (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int xd_timeofdaystr (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int xd_durationstr (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int berDecCharArray (OSCTXT * pctxt, char * charArray, OSSIZE arraySize, ASN1TagType tagging, ASN1TAG tag, int length)

- int xd_16BitCharStr (OSCTXT * pctxt, Asn116BitCharString * object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int xd_16BitCharStr64 (OSCTXT * pctxt, Asn116BitCharString * object_p, ASN1TagType tagging, ASN1TAG tag, OSSIZE length, OSBOOL indefLen)
- int xd_32BitCharStr (OSCTXT * pctxt, Asn132BitCharString * object_p, ASN1TagType tagging, ASN1TAG tag, int length)
- int xd_32BitCharStr64 (OSCTXT * pctxt, Asn132BitCharString * object_p, ASN1TagType tagging, ASN1TAG tag, OSSIZE length, OSBOOL indefLen)
- int xd_null (OSCTXT * pctxt, ASN1TagType tagging)
- int xd_objid (OSCTXT * pctxt, ASN1OBJID * object_p, ASN1TagType tagging, int length)
- int xd_oid64 (OSCTXT * pctxt, ASN1OID64 * object_p, ASN1TagType tagging, int length)
- int xd_reloid (OSCTXT * pctxt, ASN1OBJID * object_p, ASN1TagType tagging, int length)
- int xd_real (OSCTXT * pctxt, OSREAL * object_p, ASN1TagType tagging, int length)
- int xd_real_bin (OSCTXT * pctxt, OSREAL * object_p, ASN1TagType tagging, int length)
- int xd_real_b10_content (OSCTXT * pctxt, OSREAL * object_p, const char * content, int length, int form)
- int xd_real_der (OSCTXT * pctxt, OSREAL * object_p, ASN1TagType tagging, int length)
- int xd_enum (OSCTXT * pctxt, OSINT32 * object_p, ASN1TagType tagging, int length)
- int xd_enumUnsigned (OSCTXT * pctxt, OSUINT32 * object_p, ASN1TagType tagging, int length)
- int xd_OpenType (OSCTXT * pctxt, const OSOCTET ** object_p2, OSSIZE * pnumocts)
- int xd_OpenTypeExt (OSCTXT * pctxt, ASN1CCB * ccb_p, ASN1TAG * tags, int tagCount, OSRTDList * pElemList)
- int xd_OpenTypeExt64 (OSCTXT * pctxt, const OSOCTET * consptr, OSSIZE conslen, OSBOOL indefLen, ASN1TAG * tags, OSSIZE tagCount, OSRTDList * pElemList)
- int xd_OpenTypeAppend (OSCTXT * pctxt, OSRTDList * pElemList)
- int xd_real10 (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, int length)
- int xd_setp (OSCTXT * pctxt, const OSOCTET * msg_p, int msglen, ASN1TAG * tag_p, int * len_p)
- int xd_setp64 (OSCTXT * pctxt, const OSOCTET * msg_p, OSSIZE msglen, ASN1TAG * tag_p, OSSIZE * len_p, OSBOOL * pIndefLen)
- int xd_indeflen_ex (const OSOCTET * msg_p, int bufSize)
- int xd_indeflen64 (const OSOCTET * msg_p, OSSIZE bufSize, OSSIZE * plength)
- int xd_len (OSCTXT * pctxt, int * len_p)
- int xd_len64 (OSCTXT * pctxt, OSSIZE * len_p, OSBOOL * pindef)
- OSBOOL xd_chkend (OSCTXT * pctxt, const ASN1CCB * ccb_p)

- OSBOOL xd_chkend64 (OSCTXT * pctxt, const OSOCTET * consptr, OSSIZE conslen, OSBOOL indef)
- int xd_count (OSCTXT * pctxt, int length, int * count_p)
- int xd_count64 (OSCTXT * pctxt, OSSIZE length, OSBOOL indefLen, OSSIZE * count_p)
- int xd_NextElement (OSCTXT * pctxt)
- int xd_Tag1AndLen (OSCTXT * pctxt, OSINT32 * len_p)
- int xd_memcpy (OSCTXT * pctxt, OSOCTET * object_p, int length)
- int xd_match1 (OSCTXT * pctxt, OSOCTET tag, int * len_p)
- int xd_match1_64 (OSCTXT * pctxt, OSOCTET tag, OSSIZE * len_p, OSBOOL * pindef)
- 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_identifier (OSCTXT * pctxt, OSUINT32 ident)
- int xe_tag (OSCTXT * pctxt, ASN1TAG tag)
- int xe_tag_len (OSCTXT * pctxt, ASN1TAG tag, int length)
- int xe_boolean (OSCTXT * pctxt, OSBOOL * object_p, ASN1TagType tagging)
- int xe_integer (OSCTXT * pctxt, int * object_p, ASN1TagType tagging)
- int xe_unsigned (OSCTXT * pctxt, OSUINT32 * object_p, ASN1TagType tagging)
- int xe_int8 (OSCTXT * pctxt, OSINT8 * object_p, ASN1TagType tagging)
- int xe_int16 (OSCTXT * pctxt, OSINT16 * object_p, ASN1TagType tagging)
- int xe_int64 (OSCTXT * pctxt, OSINT64 * object_p, ASN1TagType tagging)
- int xe_uint64 (OSCTXT * pctxt, OSUINT64 * object_p, ASN1TagType tagging)
- int xe_uint8 (OSCTXT * pctxt, OSUINT8 * object_p, ASN1TagType tagging)
- int xe_uint16 (OSCTXT * pctxt, OSUINT16 * object_p, ASN1TagType tagging)
- int xe_bigint (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging)
- int xe_bigintn (OSCTXT * pctxt, const char * object_p, size_t nchars, ASN1TagType tagging)
- int xe_bitstr (OSCTXT * pctxt, const OSOCTET * object_p, OSSIZE numbits, ASN1TagType tagging)
- int xe_bitstrExt (OSCTXT * pctxt, const OSOCTET * object_p, OSSIZE numbits, OSSIZE dataSize, const OSOCTET * extdata, ASN1TagType tagging)

- int xe_cerBitstr (OSCTXT * pctxt, const OSOCTET * object_p, OSSIZE numbits, ASN1TagType tagging)
- int xe_cerCharstr (OSCTXT * pctxt, const char * pvalue, ASN1TagType tagging, ASN1TAG tag)
- int xe_cerOctstr (OSCTXT * pctxt, const OSOCTET * object_p, OSSIZE numocts, ASN1TagType tagging)
- int xe_octstr (OSCTXT * pctxt, const OSOCTET * object_p, OSSIZE numocts, ASN1TagType tagging)
- int xe_charstr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_cer16BitCharStr (OSCTXT * pctxt, Asn116BitCharString * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_cer32BitCharStr (OSCTXT * pctxt, Asn132BitCharString * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_16BitCharStr (OSCTXT * pctxt, Asn116BitCharString * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_32BitCharStr (OSCTXT * pctxt, Asn132BitCharString * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_datestr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_timestr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_datetimestr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_timeofdaystr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_durationstr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int xe_null (OSCTXT * pctxt, ASN1TagType tagging)
- int xe_objid (OSCTXT * pctxt, ASN1OBJID * object_p, ASN1TagType tagging)
- int xe_oid64 (OSCTXT * pctxt, ASN1OID64 * object_p, ASN1TagType tagging)
- int xe_reloid (OSCTXT * pctxt, ASN1OBJID * object_p, ASN1TagType tagging)
- int xe_enum (OSCTXT * pctxt, OSINT32 * object_p, ASN1TagType tagging)
- int xe_enumUnsigned (OSCTXT * pctxt, OSUINT32 * object_p, ASN1TagType tagging)
- int xe_real (OSCTXT * pctxt, OSREAL * object_p, ASN1TagType tagging)
- int xe_OpenType (OSCTXT * pctxt, const OSOCTET * object_p, OSSIZE numocts)
- int xe_OpenTypeExt (OSCTXT * pctxt, OSRTDList * pElemList)
- int xe_OpenTypeExtDer (OSCTXT * pctxt, OSRTDList * pElemList, OSRTSList * pBufList)
- int xe_real10 (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging)
- int xe_derReal10 (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging)
- int xe_setp (OSCTXT * pctxt, OSOCTET * buf_p, OSSIZE bufsiz)
- OSOCTET * xe_getp (OSCTXT * pctxt)
- void xe_free (OSCTXT * pctxt)

- int xe_expandBuffer (OSCTXT * pctxt, size_t length)
- int xe_memcpy (OSCTXT * pctxt, const OSOCTET * object_p, size_t length)
- int xe_len (OSCTXT * pctxt, int length)
- int xe_len64 (OSCTXT * pctxt, OSSIZE length, OSBOOL indef)
- int xe_derCanonicalSort (OSCTXT * pctxt, OSRTSList * pList)
- int xe_derCanSortSet (OSCTXT * pctxt, OSRTSList * pList)
- int xe_TagAndIndefLen (OSCTXT * pctxt, ASN1TAG tag, int length)
- void xe_getBufLocDescr (OSCTXT * pctxt, OSSIZE length, Asn1BufLocDescr * pDescr)
- int derEncBitString (OSCTXT * pctxt, const OSOCTET * pvalue, OSSIZE numbits, ASN1TagType tagging)
- int berDefToIndefLen (OSCTXT * pSrcCtxt, OSCTXT * pDstCtxt)
- int berIndefToDefLen (OSCTXT * pSrcCtxt, OSCTXT * pDstCtxt)
- OSBOOL berErrAddTagParm (OSCTXT * pctxt, ASN1TAG tag)
- int berErrUnexpTag (OSCTXT * pctxt, ASN1TAG exptag)
- int berGetLibVersion (OSVOIDARG)
- const char * berGetLibInfo (OSVOIDARG)
- int berParseTagLen (const OSOCTET * buffer, size_t bufidx, size_t bufsiz, ASN1TAG * ptag, size_t * plen)
- const char * berTagToString (ASN1TAG tag, char * buffer, size_t bufsiz)
- const char * berTagToTypeName (ASN1TAG tag)
- const char * berTagToDynStr (OSCTXT * pctxt, ASN1TAG tag)
- int berValidateIso8601DateStr (OSCTXT * pctxt, const char ** ppvalue)
- int berValidateIso8601DurationStr (OSCTXT * pctxt, const char ** ppvalue)
- int berValidateIso8601TimeStr (OSCTXT * pctxt, const char ** ppvalue)
- int xu_verify_len (OSOCTET * msg_p)
- void * xu_parse_mmbuf (OSOCTET ** buf_p2, int * buflen_p, OSOCTET * start_p, int bufsiz)
- void xu_alloc_array (OSCTXT * pctxt, ASN1SeqOf * seqOf_p, int recSize, int recCount)
- void xu_octscpy_s (OSUINT32 * nocts_p, OSOCTET * data_p, char * cstr, char zterm)
- void xu_octscpy_ss (ASN1OctStr * octStr_p, char * cstring, char zterm)
- void xu_octscpy_d (OSCTXT * pctxt, OSUINT32 * nocts_p, const OSOCTET ** data_p2, char * cstring, char zterm)
- void xu_octscpy_ds (OSCTXT * pctxt, ASN1DynOctStr * octStr_p, char * cstring, char zterm)

- void xu_octmcpy_s (ASN1OctStr * octStr_p, void * data_p, int datalen)
- void xu_octmcpy_d (OSCTXT * pctxt, ASN1DynOctStr * octStr_p, void * data_p, int datalen)
- char * xu_fetchstr (int numocts, char * data)
- int xu_hexstrcpy (char * data, char * hstring)
- int xu_binstrcpy (char * data, char * bstring)
- int xu_dump (const OSOCTET * msgptr, ASN1DumpCbFunc cb, void * cbArg_p)
- int xu_fdump (FILE * file_p, const OSOCTET * msgptr)
- int xu_dump2 (OSCTXT * pctxt, const OSOCTET * msgptr)
- void xu_fmt_tag (ASN1TAG * tag_p, char * class_p, char * form_p, char * id_code)
- void xu_fmt_tag_s (ASN1TAG * tag_p, ASN1TAGTEXT * pTagText)
- char * xu_fmt_tag2 (ASN1TAG * tag_p, char * bufp)
- char * xu_fmt_tag2_s (ASN1TAG * tag_p, char * bufp, OSSIZE bufsize)
- char * xu_fmt_contents (OSCTXT * pctxt, int len, int * count)
- int xu_fread (FILE * fp, OSOCTET * bufp, int bufsiz)
- void xu_SaveBufferState (OSCTXT * pCtxt, OSRTBufSave * pSavedInfo)
- void xu_RestoreBufferState (OSCTXT * pCtxt, OSRTBufSave * pSavedInfo)
- int xd_MovePastEOC (OSCTXT * pctxt)
- int xd_consStrIndefLenAndSize (OSCTXT * pctxt, ASN1TAG expectedTag, OSSIZE * length, OSSIZE * size)

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.

Definition in file asn1ber.h

asn1BerCppTypes.h File Reference

```
#include "rtsrc/asn1CppTypes.h"

#include "rtbersrc/asn1ber.h"

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

Classes

- struct ASN1BERLength
- struct ASN1BERMessageBuffer

- struct ASN1BEREncodeBuffer
- struct ASN1BERDecodeBuffer

Detailed Description

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

Definition in file asn1BerCppTypes.h

ASN1BERDecodeStream.h File Reference

```
#include "rtbersrc/asn1BerCppTypes.h"
#include "rtbersrc/asn1berStream.h"
#include "rtxsrc/OSRTInputStreamIF.h"
```

Classes

- struct ASN1BERDecodeStream

Detailed Description

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

Definition in file ASN1BERDecodeStream.h

ASN1BEREncodeStream.h File Reference

```
#include "rtsrc/asn1CppTypes.h"
#include "rtxsrc/OSRTOutputStreamIF.h"
#include "rtbersrc/asn1berStream.h"
```

Classes

- struct ASN1BEREncodeStream

Detailed Description

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

Definition in file ASN1BEREncodeStream.h

asn1berSocket.h File Reference

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

Functions

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

Detailed Description

Definition in file asn1berSocket.h

asn1berStream.h File Reference

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

#include "rtxsrc/rtxBuffer.h"

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

Macros

- #define cerEncCanonicalSort (pctxt, pMemCtxt, pList) \ rtxEncCanonicalSort(pctxt, pMemCtxt, pList)
- #define cerGetBufLocDescr (pctxt, pDescr) \ rtxGetBufLocDescr(pctxt, pDescr)
- #define cerAddBufLocDescr (pctxt, pElemList, pDescr) \ rtxAddBufLocDescr(pctxt, pElemList, pDescr)
- #define BS_CHKEOB (((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))
- #define BS_CHKEND ((ccb_p)->stat = 0, \ (((ccb_p)->len == ASN_K_INDEFLEN) ? berDecStrmTestEOC (pctxt,ccb_p) : \ (((int)(OSRTSTREAM_BYTEINDEX(pctxt) - (ccb_p)->bytes) >= (ccb_p)->len))))

Functions

- int berStrmInitContext (OSCTXT * pctxt)
- int berStrmInitContextUsingKey (OSCTXT * pctxt, const OSOCTET * key, size_t keylen)
- int berStrmFreeContext (OSCTXT * pctxt)
- int berEncStrmBigInt (OSCTXT * pctxt, const char * pvalue, ASN1TagType tagging)
- int berEncStrmBigIntNchars (OSCTXT * pctxt, const char * pvalue, size_t nchars, ASN1TagType tagging)
- int berEncStrmBitStr (OSCTXT * pctxt, const OSOCTET * object_p, OSUINT32 numbits, ASN1TagType tagging)
- int berEncStrmBMPStr (OSCTXT * pctxt, const Asn116BitCharString * object_p, ASN1TagType tagging)
- int berEncStrmBool (OSCTXT * pctxt, OSBOOL value, ASN1TagType tagging)
- int berEncStrmCharStr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int berEncStrmDateStr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int berEncStrmDateTimeStr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)

- int berEncStrmDurationStr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int berEncStrmTimeStr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int berEncStrmTimeOfDayStr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int berEncStrmDefLength (OSCTXT * pctxt, size_t length)
- int berEncStrmEOC (OSCTXT * pctxt)
- int berEncStrmEnum (OSCTXT * pctxt, OSINT32 value, ASN1TagType tagging)
- int berEncStrmInt (OSCTXT * pctxt, OSINT32 value, ASN1TagType tagging)
- int berEncStrmInt8 (OSCTXT * pctxt, OSINT8 value, ASN1TagType tagging)
- int berEncStrmInt16 (OSCTXT * pctxt, OSINT16 value, ASN1TagType tagging)
- int berEncStrmInt64 (OSCTXT * pctxt, OSINT64 value, ASN1TagType tagging)
- int berEncStrmLength (OSCTXT * pctxt, int length)
- int berEncStrmNull (OSCTXT * pctxt, ASN1TagType tagging)
- int berEncStrmObjId (OSCTXT * pctxt, const ASN1OBJID * object_p, ASN1TagType tagging)
- int berEncStrmObjId64 (OSCTXT * pctxt, const ASN1OID64 * object_p, ASN1TagType tagging)
- int berEncStrmOctStr (OSCTXT * pctxt, const OSOCTET * object_p, OSSIZE numocts, ASN1TagType tagging)
- int berEncStrmOpenTypeExt (OSCTXT * pctxt, OSRTDList * pElemList)
- int berEncStrmReal (OSCTXT * pctxt, OSREAL value, ASN1TagType tagging)
- int berEncStrmReal10 (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging)
- int cerEncStrmReal10 (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging)
- int berEncStrmRelativeOID (OSCTXT * pctxt, const ASN1OBJID * object_p, ASN1TagType tagging)
- int berEncStrmTag (OSCTXT * pctxt, ASN1TAG tag)
- int berEncStrmTagAndLen (OSCTXT * pctxt, ASN1TAG tag, int length)
- int berEncStrmTagAndDefLen (OSCTXT * pctxt, ASN1TAG tag, OSSIZE length)
- int berEncStrmTagAndIndefLen (OSCTXT * pctxt, ASN1TAG tag)
- int berEncStrmUInt (OSCTXT * pctxt, OSUINT32 value, ASN1TagType tagging)
- int berEncStrmUInt8 (OSCTXT * pctxt, OSUINT8 value, ASN1TagType tagging)
- int berEncStrmUInt16 (OSCTXT * pctxt, OSUINT16 value, ASN1TagType tagging)
- int berEncStrmUInt64 (OSCTXT * pctxt, OSUINT64 value, ASN1TagType tagging)
- int berEncStrmUnivStr (OSCTXT * pctxt, const Asn132BitCharString * object_p, ASN1TagType tagging)
- int berEncStrmXSDAny (OSCTXT * pctxt, OSXSDAny * pvalue, ASN1TagType tagging)

- int berEncStrmWriteOctet (OSCTXT * pctxt, OSOCTET octet)
- int berEncStrmWriteOctets (OSCTXT * pctxt, const OSOCTET * poctets, size_t numocts)
- int cerEncStrmBMPStr (OSCTXT * pctxt, const Asn116BitCharString * object_p, ASN1TagType tagging)
- int cerEncStrmBitStr (OSCTXT * pctxt, const OSOCTET * object_p, OSUINT32 numbits, ASN1TagType tagging)
- int cerEncStrmCharStr (OSCTXT * pctxt, const char * object_p, ASN1TagType tagging, ASN1TAG tag)
- int cerEncStrmOctStr (OSCTXT * pctxt, const OSOCTET * object_p, OSUINT32 numocts, ASN1TagType tagging)
- int cerEncStrmUnivStr (OSCTXT * pctxt, const Asn132BitCharString * object_p, ASN1TagType tagging)
- int berDecStrmBMPStr (OSCTXT * pctxt, Asn116BitCharString * object_p, ASN1TagType tagging, int length)
- int berDecStrmBigInt (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, int length)
- int berDecStrmBigEnum (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, int length)
- int berDecStrmBitStr (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnbits, ASN1TagType tagging, int length)
- int berDecStrmBool (OSCTXT * pctxt, OSBOOL * object_p, ASN1TagType tagging, int length)
- int berDecStrmCharStr (OSCTXT * pctxt, const char ** ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int berDecStrmCharArray (OSCTXT * pctxt, char * charArray, OSSIZE arraySize, ASN1TagType tagging, ASN1TAG tag, int length)
- int berDecStrmDateStr (OSCTXT * pctxt, const char ** ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int berDecStrmDateTimeStr (OSCTXT * pctxt, const char ** ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int berDecStrmDurationStr (OSCTXT * pctxt, const char ** ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int berDecStrmTimeStr (OSCTXT * pctxt, const char ** ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- int berDecStrmTimeOfDayStr (OSCTXT * pctxt, const char ** ppvalue, ASN1TagType tagging, ASN1TAG tag, int length)
- OSBOOL berDecStrmCheckEnd (OSCTXT * pctxt, ASN1CCB * pccb)
- int berDecStrmDynBitStr (OSCTXT * pctxt, const OSOCTET ** ppvalue, OSUINT32 * pnbits, ASN1TagType tagging, int length)
- int berDecStrmDynBitStr64 (OSCTXT * pctxt, const OSOCTET ** ppvalue, OSSIZE * pnbits, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int berDecStrmDynOctStr (OSCTXT * pctxt, const OSOCTET ** ppvalue, OSUINT32 * pnocts, ASN1TagType tagging, int length)

- int berDecStrmDynOctStr64 (OSCTXT * pctxt, OSOCTET ** ppvalue, OSSIZE * pnocts, ASN1TagType tagging, OSSIZE length, OSBOOL indefLen)
- int berDecStrmEnum (OSCTXT * pctxt, OSINT32 * object_p, ASN1TagType tagging, int length)
- int berDecStrmFindTag (OSCTXT * pctxt, ASN1TAG tag, int * len_p, OSBOOL advance)
- int berDecStrmFindTag2 (OSCTXT * pctxt, ASN1TAG tag, OSSIZE * len_p, OSBOOL * pIndefLen, OSBOOL advance)
- int berDecStrmGetTLVLength (OSCTXT * pctxt)
- int berDecStrmInt (OSCTXT * pctxt, OSINT32 * object_p, ASN1TagType tagging, int length)
- int berDecStrmInt8 (OSCTXT * pctxt, OSINT8 * object_p, ASN1TagType tagging, int length)
- int berDecStrmInt16 (OSCTXT * pctxt, OSINT16 * object_p, ASN1TagType tagging, int length)
- int berDecStrmInt64 (OSCTXT * pctxt, OSINT64 * object_p, ASN1TagType tagging, int length)
- int berDecStrmLength (OSCTXT * pctxt, int * len_p)
- int berDecStrmLength2 (OSCTXT * pctxt, OSSIZE * pLength, OSBOOL * pIndefLen)
- int berDecStrmMatchEOC (OSCTXT * pctxt)
- int berDecStrmMatchTag (OSCTXT * pctxt, ASN1TAG tag, int * len_p, OSBOOL advance)
- int berDecStrmMatchTag2 (OSCTXT * pctxt, ASN1TAG tag, OSSIZE * len_p, OSBOOL * pIndefLen, OSBOOL advance)
- int berDecStrmNextElement (OSCTXT * pctxt)
- int berDecStrmNull (OSCTXT * pctxt, ASN1TagType tagging)
- int berDecStrmObjId (OSCTXT * pctxt, ASN1OBJID * object_p, ASN1TagType tagging, int length)
- int berDecStrmObjId64 (OSCTXT * pctxt, ASN1OID64 * object_p, ASN1TagType tagging, int length)
- int berDecStrmOctStr (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocts, ASN1TagType tagging, int length)
- int berDecStrmOpenType (OSCTXT * pctxt, const OSOCTET ** object_p2, OSSIZE * pnumocts)
- int berDecStrmOpenTypeAppend (OSCTXT * pctxt, OSRTDList * pElemList)
- int berDecStrmOpenTypeExt (OSCTXT * pctxt, ASN1CCB * ccb_p, ASN1TAG * tags, int tagCount, OSRTDList * pElemList)
- int berDecStrmPeekTagAndLen (OSCTXT * pctxt, ASN1TAG * ptag, int * plen)
- int berDecStrmReadDynTLV (OSCTXT * pctxt, OSOCTET ** ppbuf)
- int berDecStrmReadTLV (OSCTXT * pctxt, OSOCTET * buf, OSSIZE bufsiz)
- int berDecStrmReal (OSCTXT * pctxt, OSREAL * object_p, ASN1TagType tagging, int length)
- int berDecStrmRealBin (OSCTXT * pctxt, OSREAL * object_p, ASN1TagType tagging, int length)

- int berDecStrmRealDer (OSCTXT * pctxt, OSREAL * object_p, ASN1TagType tagging, int length)
- int berDecStrmReal10 (OSCTXT * pctxt, const char ** object_p, ASN1TagType tagging, int length)
- int berDecStrmRelativeOID (OSCTXT * pctxt, ASN1OBJID * object_p, ASN1TagType tagging, int length)
- int berDecStrmTag (OSCTXT * pctxt, ASN1TAG * tag_p)
- int berDecStrmTagAndLen (OSCTXT * pctxt, ASN1TAG * tag_p, int * len_p)
- int berDecStrmTagAndLen2 (OSCTXT * pctxt, ASN1TAG * tag_p, OSSIZE * len_p, OSBOOL * pIndefLen)
- OSBOOL berDecStrmTestEOC (OSCTXT * pctxt, ASN1CCB * ccb_p)
- OSBOOL berDecStrmTestEOC2 (OSCTXT * pctxt)
- int berDecStrmTestTag (OSCTXT * pctxt, ASN1TAG tag, int * len_p, OSBOOL advance)
- int berDecStrmUInt (OSCTXT * pctxt, OSUINT32 * object_p, ASN1TagType tagging, int length)
- int berDecStrmUInt8 (OSCTXT * pctxt, OSUINT8 * object_p, ASN1TagType tagging, int length)
- int berDecStrmUInt16 (OSCTXT * pctxt, OSUINT16 * object_p, ASN1TagType tagging, int length)
- int berDecStrmUInt64 (OSCTXT * pctxt, OSUINT64 * object_p, ASN1TagType tagging, int length)
- int berDecStrmUnivStr (OSCTXT * pctxt, Asn132BitCharString * object_p, ASN1TagType tagging, int length)

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.

Definition in file asn1berStream.h