ASN1C

ASN.1 Compiler
Version 6.5
XER Runtime
Reference Manual

Objective Systems, Inc. version 6.5 — March 2012

Contents

Chapter 1

ASN1C XER 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 XML Encoding Rules (XER)as defined in the X.693 ITU-T standard.

These XER specific C++ runtime classes include:

- classes for streaming XER decoding
- classes for streaming XER encoding.

The **ASN.1 XER Runtime Library** contains 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 items:

- A global include file ("asn1xer.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 Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Class Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

Module Documentation

6.1 XER C++ Runtime Classes.

Modules

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

6.2 XER Message Buffer Classes

Classes

- class ASN1XERMessageBuffer
- class ASN1XEREncodeBuffer
- class ASN1XERShdMemHpEncBuf
- class ASN1XERSAXDecodeHandler
- class ASN1XERDecodeBuffer

6.2.1 Detailed Description

These classes manage the buffers for encoding and decoding ASN.1 XER messages.

6.3 XER Runtime Library Functions.

Classes

- struct XerElemInfo
- struct XmlNamedBitsDict

Modules

- XER C Decode Functions.
- XER C Encode Functions.
- XER C Utility Functions.
- XML C Decode Functions.
- XML C Encode Functions.

Defines

- #define **XERINDENT** 3
- #define **XERBYTECNT**(pctxt) (pctxt)->buffer.byteIndex
- #define **EXTERNXER**

Typedefs

- typedef struct XmlNamedBitsDict **XmlNamedBitsDict**

Enumerations

- enum **ASN1XERState** {
  XERINIT, **XERSTART**, **XERDATA**, **XEREND**,
  XERSTART0, **XEREND0** }

6.3.1 Detailed Description

The ASN.1 XML Encoding Rules (XER) runtime library contains low-level constants, types, and functions that are assembled by the ASN1C compiler to encode/decode more complex structures.

The XER low-level C encode/decode functions are identified by their prefixes: xerEnc for XER encode, xerDec for XER decode, and xer for XER utility functions.

6.4 XER C Decode Functions.

Functions

- int xerDecBMPStr (OSCTXT *pctxt, ASN1BMPString *outdata)
- int xerDecBase64Str (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSINT32 bufsize)
- int xerDecBigInt (OSCTXT *pctxt, char **ppvalue, int radix)
- int xerDecBitStr (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnbits, OSINT32 bufsize)
- int xerDecBitStrMemBuf (OSRTMEMBUF *pMemBuf, const XMLCHAR *inpdata, int length, OSBOOL skipWhitespaces)
- int xerDecBool (OSCTXT *pctxt, OSBOOL *pvalue)
- int xerDecCopyBitStr (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnbits, OSINT32 bufsize, int lastBitOffset)
- int xerDecCopyDynBitStr (OSCTXT *pctxt, ASN1DynBitStr *pvalue, int lastBitOffset)
- int xerDecCopyDynOctStr (OSCTXT *pctxt, ASN1DynOctStr *pvalue, int lastBitOffset)
- int xerDecCopyOctStr (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSINT32 bufsize, int lastBitOffset)
- int xerDecDynAscCharStr (OSCTXT *pctxt, const char **outdata)
- int xerDecDynBase64Str (OSCTXT *pctxt, ASN1DynOctStr *pvalue)
- int xerDecDynBitStr (OSCTXT *pctxt, ASN1DynBitStr *pvalue)
- int xerDecDynOctStr (OSCTXT *pctxt, ASN1DynOctStr *pvalue)
- int xerDecDynUTF8Str (OSCTXT *pctxt, ASN1UTF8String *outdata)
- int xerDecInt (OSCTXT *pctxt, OSINT32 *pvalue)
- int xerDecInt8 (OSCTXT *pctxt, OSINT8 *pvalue)
- int xerDecInt16 (OSCTXT *pctxt, OSINT16 *pvalue)
- int xerDecInt64 (OSCTXT *pctxt, OSINT64 *pvalue)
- int xerDecObjId (OSCTXT *pctxt, ASN1OBJID *pvalue)
- int xerDecObjId64 (OSCTXT *pctxt, ASN1OID64 *pvalue)
- int xerDecOctStr (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSINT32 bufsize)
- int xerDecOctStrMemBuf (OSRTMEMBUF *pMemBuf, const XMLCHAR *inpdata, int length, OSBOOL skipWhitespaces)
- int xerDecOpenType (OSCTXT *pctxt, ASN1OpenType *pvalue)
- int xerDecReal (OSCTXT *pctxt, OSREAL *pvalue)
- int xerDecReal10 (OSCTXT *pctxt, const OSUTF8CHAR **pvalue)
- int xerDecRelativeOID (OSCTXT *pctxt, ASN1OBJID *pvalue)
- int xerDecUInt (OSCTXT *pctxt, OSUINT32 *pvalue)
- int xerDecUInt8 (OSCTXT *pctxt, OSUINT8 *pvalue)
- int xerDecUInt16 (OSCTXT *pctxt, OSUINT16 *pvalue)
- int xerDecUInt64 (OSCTXT *pctxt, OSUINT64 *pvalue)
- int xerDecUnivStr (OSCTXT *pctxt, ASN1UniversalString *outdata)
- int xerSetDecBufPtr (OSCTXT *pCtxt, const OSOCTET *bufaddr, size_t bufsiz)

6.4.1 Detailed Description

XER runtime library decode functions handle the decoding of the primitive ASN.1 data types and length variables. Calls to these functions are assembled in the C source code generated by the ASN1C complier 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.

6.4.2 Function Documentation

6.4.2.1 int xerDecBase64Str (OSCTXT * *pctxt*, OSOCTET * *pvalue*, OSUINT32 * *pnocts*, OSINT32 *bufsize*)

This function will decode a variable of teh ASN.1 OCTET STRING type into a static memory structure. The octet string must be Base64 encoded. This function call is used to decode a sized octet string production.

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.

pvalue A 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 octets specified in the bufsize input parameter.

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

bufsize A 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.

Returns

Completion status of operation:

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

6.4.2.2 int xerDecBigInt (OSCTXT * *pctxt*, char ** *ppvalue*, int *radix*)

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. Depending on specified radix, they could be represented as binary, octal, decimal or hexadecimal strings starting with appropriate prefix. If it is necessary to convert to another radix, then use rtxBigIntSetStr or rtxBigIntToString functions.

Parameters

pctxt Pointer to context block structure.

ppvalue Pointer to a character pointer variable to receive the decoded unsigned value. Dynamic memory is allocated for the variable using the rtxMemAlloc function. The decoded variable is represented as a string starting with appropriate prefix.

radix The expected radix of the decoded value. The only radices 2, 8, 10 and 16 are supported.

Returns

Completion status of operation:

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

6.4.2.3 int xerDecBitStr (OSCTXT * *pctxt*, OSOCTET * *pvalue*, OSUINT32 * *pnbits*, OSINT32 *bufsize*)

This function will decode 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 production.

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.

pvalue A 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 octets specified in the bufsize input parameter.

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

bufsize An integer variable containing the size (in octets) of the sized ASN.1 bit string. An error will occur if the number of octets in the decoded string is larger than this value.

Returns

Completion status of operation:

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

6.4.2.4 int xerDecBitStrMemBuf (OSRTMEMBUF * *pMemBuf*, const XMLCHAR * *inpdata*, int *length*, OSBOOL *skipWhitespaces*)

This function decodes a variable of the ASN.1 BIT STRING type to a memory buffer. The decoded data will be put into the memory buffer starting from the current position and bit offset. After all data is decoded the bit string may be fetched out by call to xerDecCopyBitStr or xerDecCopyDynBitStr functions.

Usually, this function is used in the 'characters' SAX handler.

Parameters

pMemBuf Pointer to the destination memory buffer.

inpdata Pointer to a source string to be decoded.

length Length of the source string (in characters).

skipWhitespaces Indicates, could whitespaces be ignored or they are illegal.

Returns

Completion status of operation:

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

6.4.2.5 int xerDecBMPStr (OSCTXT * *pctxt*, ASN1BMPString * *outdata*)

This function will decode a variable ASN.1 16-bit character BMPString type. This function will allocate dynamic memory to store the decoded result.

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.

outdata A pointer to a structure variable to receive the decoded string. The string is stored as an array of short integer characters. The memory is allocated for the string by the rtxMemAlloc function.

Returns

Completion status of operation:

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

6.4.2.6 int xerDecBool (OSCTXT * *pctxt*, OSBOOL * *pvalue*)

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

Parameters

pctxt Pointer to context block structure.

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

Returns

Completion status of operation:

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

6.4.2.7 int xerDecCopyBitStr (OSCTXT * *pctxt*, OSOCTET * *pvalue*, OSUINT32 * *pnbits*, OSINT32 *bufsize*, int *lastBitOffset*)

This function copies the decoded BIT STRING from the memory buffer. This function call is generated by ASN1C to decode a sized bit string production.

Parameters

pctxt Pointer to context block structure. Its buffer should be set to point to the decoded data.

pvalue Pointer to a buffer to receive the decoded data.

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

bufsize An integer variable containing the size (in octets) of the sized ASN.1 bit string. An error will occur if the number of octets in the decoded string is larger than this value.

lastBitOffset A number of actual bits in the last octet.

Returns

Completion status of operation:

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

6.4.2.8 int xerDecCopyDynBitStr (OSCTXT * *pctxt*, ASN1DynBitStr * *pvalue*, int *lastBitOffset*)

This function copies the decoded BIT STRING from the memory buffer. This function will allocate dynamic memory to store the decoded result.

Parameters

pctxt Pointer to context block structure. Its buffer should be set to point to the decoded data.

pvalue Pointer to a dynamic bit string structure to receive the decoded bit string. Dynamic memory is allocated to hold the string using the rtxMemAlloc function.

lastBitOffset A number of actual bits in the last octet.

Returns

Completion status of operation:

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

6.4.2.9 int xerDecCopyDynOctStr (OSCTXT * *pctxt*, ASN1DynOctStr * *pvalue*, int *lastBitOffset*)

This function copies the decoded OCTET STRING from the memory buffer. This function will allocate dynamic memory to store the decoded result.

Parameters

pctxt Pointer to context block structure. Its buffer should be set to point to the decoded data.

pvalue Pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the rtxMemAlloc function.

lastBitOffset A number of actual bits in the last octet.

Returns

Completion status of operation:

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

6.4.2.10 int xerDecCopyOctStr (OSCTXT * *pctxt*, OSOCTET * *pvalue*, OSUINT32 * *pnocts*, OSINT32 *bufsize*, int *lastBitOffset*)

This function copies the decoded OCTET STRING from the memory buffer. This function call is generated by ASN1C to decode a sized octet string production.

Parameters

pctxt Pointer to context block structure. Its buffer should be set to point to the decoded data.

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 octets specified in the 'bufsize' input parameter.

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

bufsize 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.

lastBitOffset A number of actual bits in the last octet.

Returns

Completion status of operation:

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

6.4.2.11 int xerDecDynAscCharStr (OSCTXT * *pctxt*, const char ** *outdata*)

This function will decode a variable of one of the ASN.1 8-bit character string types. These types include IA5String, VisibleString, PrintableString, and NumericString. This function will allocate dynamic memory to store the result.

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.

outdata A pointer to a character string pointer variable to receive the decoded string. The string as stored as a standard null-terminated C string. Memory is allocated for the string by the rtxMemAlloc function.

Returns

Completion status of operation:

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

6.4.2.12 int xerDecDynBase64Str (OSCTXT * *pctxt*, ASN1DynOctStr * *pvalue*)

This function will decode a variable of the ASN.1 OCTET STRING type. The octet string must be Base64 encoded. This function will allocate dynamic memory to store the decoded result.

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.

pvalue A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the rtxMemAlloc function.

Returns

Completion status of operation:

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

6.4.2.13 int xerDecDynBitStr (OSCTXT * *pctxt*, ASN1DynBitStr * *pvalue*)

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

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.

pvalue A pointer to a dynamic bit string structure to receive the decoded bit string. Dynamic memory is allocated to hold the string using the rtxMemAlloc function.

Returns

Completion status of operation:

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

6.4.2.14 int xerDecDynOctStr (OSCTXT * *pctxt*, ASN1DynOctStr * *pvalue*)

This function will decode a variable of the ASN.1 OCTET STRING type. This function will allocate dynamic memory to store the decoded result.

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.

pvalue A pointer to a dynamic bit string structure to receive the decoded bit string. Dynamic memory is allocated to hold the string using the rtxMemAlloc function.

Returns

Completion status of operation:

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

6.4.2.15 int xerDecDynUTF8Str (OSCTXT * *pctxt*, ASN1UTF8String * *outdata*)

This function will decode a variable of UTF8String ASN.1 type. Generally, the SAX parser converts all UTF8 format strings to 16-bit Unicode format automatically. This function converts the Unicode string back to UTF8 format. This function will allocate dynamic memory to store the decoded result.

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.

outdata A pointer to a character string pointer variable to receive the decoded string. The string as stored as a UTF8 null-terminated string. Memory is allocated for the string by the rtxMemAlloc function.

Returns

Completion status of operation:

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

6.4.2.16 int xerDecInt (OSCTXT * *pctxt*, OSINT32 * *pvalue*)

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

Parameters

pctxt Pointer to context block structure.

pvalue Pointer to a variable to receive the decoded integer value.

Returns

Completion status of operation:

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

6.4.2.17 int xerDecInt16 (OSCTXT * *pctxt*, OSINT16 * *pvalue*)

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

Parameters

pctxt Pointer to context block structure.

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

Returns

Completion status of operation:

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

6.4.2.18 int xerDecInt64 (OSCTXT * *pctxt*, OSINT64 * *pvalue*)

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

Parameters

pctxt Pointer to context block structure.

pvalue Pointer to a 64-bit variable to receive the decoded integer value. 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).

Returns

Completion status of operation:

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

6.4.2.19 int xerDecInt8 (OSCTXT * *pctxt*, OSINT8 * *pvalue*)

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

Parameters

pctxt Pointer to context block structure.

pvalue Pointer to an 8-bit variable to receive the decoded integer value.

Returns

Completion status of operation:

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

6.4.2.20 int xerDecObjId (OSCTXT * *pctxt*, ASN1OBJID * *pvalue*)

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

Parameters

pctxt Pointer to context block structure.

pvalue 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.

Returns

Completion status of operation:

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

6.4.2.21 int xerDecObjId64 (OSCTXT * *pctxt*, ASN1OID64 * *pvalue*)

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

Parameters

pctxt Pointer to context block structure.

pvalue 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.

Returns

Completion status of operation:

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

6.4.2.22 int xerDecOctStr (OSCTXT * *pctxt*, OSOCTET * *pvalue*, OSUINT32 * *pnocts*, OSINT32 *bufsize*)

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

Parameters

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.

pvalue A 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 octets specified in the bufsize input parameter.

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

bufsize 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.

Returns

Completion status of operation:

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

6.4.2.23 int xerDecOctStrMemBuf (OSRTMEMBUF * *pMemBuf*, const XMLCHAR * *inpdata*, int *length*, OSBOOL *skipWhitespaces*)

This function decodes a variable of the ASN.1 OCTET STRING type to a memory buffer. The decoded data will be put into the memory buffer starting from the current position and bit offset. After all data is decoded the octet string may be fetched out by call to xerDecCopyOctStr or xerDecCopyDynOctStr functions.

Usually, this function is used in the 'characters' SAX handler.

Parameters

pMemBuf Pointer to the destination memory buffer.

inpdata Pointer to a source string to be decoded.

length Length of the source string (in characters).

skipWhitespaces Indicates, could whitespaces be ignored or they are illegal.

Returns

Completion status of operation:

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

6.4.2.24 int xerDecOpenType (OSCTXT * *pctxt*, ASN1OpenType * *pvalue*)

This function will decode an ASN.1 open type. This used to be the ASN.1 ANY type, but now is used in a variety of applications requiring an encoding that can be interpreted by a decoder without prior knowledge of the type of the variable.

Parameters

pctxt A 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.

pvalue A pointer to a structure variable to receive the decoded string. The string as stored as an array of unsigned integer characters. Memory is allocated for the string by the rtxMemAlloc function.

Returns

Completion status of operation:

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

6.4.2.25 int xerDecReal (OSCTXT * *pctxt*, OSREAL * *pvalue*)

This function will decode a variable of the ASN.1 32-bit character UniversalString type. This includes the Universal-String type.

Parameters

pctxt A 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.

pvalue A pointer to a structure variable to receive the decoded string. The string as stored as an array of unsigned integer characters. Memory is allocated for the string by the rtxMemAlloc function.

Returns

Completion status of operation:

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

6.4.2.26 int xerDecReal10 (OSCTXT * *pctxt*, const OSUTF8CHAR ** *pvalue*)

This function will decode a variable of the ASN.1 REAL type into a OSUTF8CHAR*. It is intended for use when the REAL type is constrained to base 10 and OSUTF8CHAR* is the chosen representation.

Parameters

pctxt A 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.

pvalue A 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.

Returns

Completion status of operation:

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

6.4.2.27 int xerDecRelativeOID (OSCTXT * *pctxt*, ASN1OBJID * *pvalue*)

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

Parameters

pctxt Pointer to context block structure.

pvalue 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.

Returns

Completion status of operation:

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

6.4.2.28 int xerDecUInt (OSCTXT * *pctxt*, OSUINT32 * *pvalue*)

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

Parameters

pctxt Pointer to context block structure.

pvalue Pointer to a variable to receive the decoded unsigned integer value.

Returns

Completion status of operation:

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

6.4.2.29 int xerDecUInt16 (OSCTXT * *pctxt*, OSUINT16 * *pvalue*)

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

Parameters

pctxt Pointer to context block structure.

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

Returns

Completion status of operation:

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

6.4.2.30 int xerDecUInt64 (OSCTXT * *pctxt*, OSUINT64 * *pvalue*)

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

Parameters

pctxt Pointer to context block structure.

pvalue Pointer to a 64-bit variable to receive the decoded unsigned integer value. 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).

Returns

Completion status of operation:

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

6.4.2.31 int xerDecUInt8 (OSCTXT * *pctxt*, OSUINT8 * *pvalue*)

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

Parameters

pctxt Pointer to context block structure.

pvalue Pointer to an 8-bit variable to receive the decoded unsigned integer value.

Returns

Completion status of operation:

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

6.4.2.32 int xerDecUnivStr (OSCTXT * *pctxt*, ASN1UniversalString * *outdata*)

This function will decode a variable an ASN.1 32-bit character UniversalString type. This includes the UniversalString type.

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.

outdata A pointer to a structure variable to receive the decoded string. The string as stored as an array of unsigned integer characters. Memory is allocated for the string by the rtxMemAlloc function.

6.4.2.33 int xerSetDecBufPtr (OSCTXT * *pCtxt*, const OSOCTET * *bufaddr*, size_t *bufsiz*)

This function is used to set the internal decode buffer pointer within the runtime library decode module. It must be called prior to calling any other complier generated or runtime library decode functions.

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.

bufaddr A pointer to a memory buffer containing the ASN.1 message. The pointer must point at the first byte in the message.

bufsiz The size of the message that was read. This is used to set an internal message size to check for field length errors. If this size is not known, a zero value can be passed to cause these checks to be bypassed.

6.5 XER C Encode Functions.

Functions

- int xerSetEncBufPtr (OSCTXT *pCtxt, OSOCTET *bufaddr, size_t bufsiz, OSBOOL canonical)
- int xerEncAscCharStr (OSCTXT *pctxt, const char *value, const char *elemName)
- int xerEncBase64Str (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *data, const char *elemName)
- int xerEncBigInt (OSCTXT *pctxt, const char *value, const char *elemName)
- int xerEncBitStr (OSCTXT *pctxt, OSUINT32 nbits, const OSOCTET *data, const char *elemName, ASN1StrType outputType)
- int **xerEncBoolValue** (OSCTXT *pctxt, OSBOOL value)
- int xerEncBool (OSCTXT *pctxt, OSBOOL value, const char *elemName)
- int xerEncEndDocument (OSCTXT *pctxt)
- int xerEncEndElement (OSCTXT *pctxt, const char *elemName)
- int xerEncIndent (OSCTXT *pctxt)
- int xerEncInt (OSCTXT *pctxt, OSINT32 value, const char *elemName)
- int xerEncInt64 (OSCTXT *pctxt, OSINT64 value, const char *elemName)
- int xerEncNewLine (OSCTXT *pctxt)
- int xerEncObjId (OSCTXT *pctxt, const ASN1OBJID *pvalue, const char *elemName)
- int xerEncObjId64 (OSCTXT *pctxt, const ASN1OID64 *pvalue, const char *elemName)
- int xerEncRelativeOID (OSCTXT *pctxt, const ASN1OBJID *pvalue, const char *elemName)
- int xerEncOctStr (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *data, const char *elemName)
- int xerEncReal (OSCTXT *pctxt, OSREAL value, const char *elemName)
- int xerEncReal10 (OSCTXT *pctxt, const OSUTF8CHAR *value, const char *elemName)
- int xerEncStartDocument (OSCTXT *pctxt)
- int xerEncStartElement (OSCTXT *pctxt, const char *elemName, const char *attributes)
- int xerEncEmptyElement (OSCTXT *pctxt, const char *elemName, const char *attributes)
- int xerEncNamedValue (OSCTXT *pctxt, const char *value, const char *elemName, const char *attributes)
- int xerEncUInt (OSCTXT *pctxt, OSUINT32 value, const char *elemName)
- int xerEncUInt64 (OSCTXT *pctxt, OSUINT64 value, const char *elemName)
- int xerEncBMPStr (OSCTXT *pctxt, const ASN1BMPString *value, const char *elemName)
- int xerEncUnivStr (OSCTXT *pctxt, const ASN1UniversalString *value, const char *elemName)
- int xerEncUniCharData (OSCTXT *pctxt, const OSUNICHAR *value, OSUINT32 nchars)
- int xerEncUniCharStr (OSCTXT *pctxt, OSUNICHAR *value, const char *elemName)
- int xerEncOpenType (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *data, const char *elemName)
- int xerEncNull (OSCTXT *pctxt, const char *elemName)
- int **xerEncXmlCharData** (OSCTXT *pctxt, const XMLCHAR *pvalue, int length)

6.5.1 Detailed Description

The XER low-level encode functions handle the XER encoding of primitive ASN.1 data types. 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 a low-level encode function is the same as the procedure to call a complier generated encode function described above. The rtInitContext and xerSetEncBufPtr functions must first be called to initialize a context and set a pointer to a buffer into which the variable is to be encoded. A static encode buffer is spcified 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 encode function is then invoked. The result of the encoding will start at the beginning of the specified buffer, or, if a dynamic buffer was used, it can be obtained by calling xerGetMsgPtr. The length of the encoded component is obtained by calling xerGetMsgLen.

6.5.2 Function Documentation

6.5.2.1 int xerEncAscCharStr (OSCTXT * *pctxt*, const char * *value*, const char * *elemName*)

This function encodes a variable one of the ASN.1 character string types that are based on 8-bit character sets. This includes IA5Sring, VisibleString, PrintableString, and UTF8String, and NumericString.

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.

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

elemName This argument specifies the name of the element that is wrapped around the encoded value. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.5.2.2 int xerEncBase64Str (OSCTXT * *pctxt*, OSUINT32 *nocts*, const OSOCTET * *data*, const char * *elemName*)

This function encodes a variable of the ASN.1 OCTET STRING type using Base64 encoding.

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.

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

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

elemName This argument specifies the name of the element that is wrapped around the encoded value. If the name is null, the default name for this type is (<OCTET_STRING>) is added. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.5.2.3 int xerEncBigInt (OSCTXT * *pctxt*, const char * *value*, const char * *elemName*)

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 are currently not supported.

Parameters

pctxt Pointer to context block structure.

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

elemName This argument specifies the name of the element that is wrapped around the encoded value. If the name is null, the default name for this type (<INTEGER>) is added. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.5.2.4 int xerEncBitStr (OSCTXT * *pctxt*, OSUINT32 *nbits*, const OSOCTET * *data*, const char * *elemName*, ASN1StrType *outputType*)

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

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.

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

data 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.

elemName This argument specifies the name of the element that is wrapped around the encoded value. If the name is null, the default name for this type (<BIT_STRING>) is added. If an empty string is passed (""), no element tag is added to the encoded value.

outputType An enumerated type whose value is set to either 'ASN1BIN' (for binary format) or 'ASN1HEX' (for hexadecimal format).

Returns

Completion status of operation:

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

6.5.2.5 int xerEncBMPStr (OSCTXT * *pctxt*, const ASN1BMPString * *value*, const char * *elemName*)

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

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.

value A pointer to a structure representing a 16-bit character string to 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.

elemName A pointer to a name of an element.

Returns

Completion status of operation:

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

6.5.2.6 int xerEncBool (OSCTXT * *pctxt*, OSBOOL *value*, const char * *elemName*)

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

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.

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.

elemName This argument specifies the name of the element that is wrapped around the encoded value. If the name is null, the default name for this type (<BOOLEAN>) is added. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.5.2.7 int xerEncEmptyElement (OSCTXT * *pctxt*, const char * *elemName*, const char * *attributes*)

This function encodes an empty element, such as <element/>.

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.

elemName A pointer to the name of the element.

attributes A pointer to the attributes of the element.

Returns

Completion status of operation:

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

6.5.2.8 int xerEncEndDocument (OSCTXT * *pctxt*)

This function should be called at the end of the document's encoding.

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

Completion status of operation:

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

6.5.2.9 int xerEncEndElement (OSCTXT * *pctxt*, const char * *elemName*)

This function encodes the ending tag of the XML element, such as </element>.

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.

elemName A pointer to the name of the element.

Returns

Completion status of operation:

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

6.5.2.10 int xerEncIndent (OSCTXT * *pctxt*)

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.

6.5.2.11 int xerEncInt (OSCTXT * *pctxt*, OSINT32 *value*, const char * *elemName*)

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

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.

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.

elemName This argument specifies the name of the element that is wrapped around the encoded value. If the name is null, the default name for this type (<INTEGER>) is added. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.5.2.12 int xerEncInt64 (OSCTXT * *pctxt*, OSINT64 *value*, const char * *elemName*)

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

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.

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.

elemName This argument specifies the name of the element that is wrapped around the encoded value. If the name is null, the default name for this type (<INTEGER>) is added. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.5.2.13 int xerEncNamedValue (OSCTXT * *pctxt*, const char * *value*, const char * *elemName*, const char * *attributes*)

This function encodes a named value, for example an enumerated value, such as <element><value/></element>.

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.

value A pointer to a value string

elemName A pointer to the name of the element.

attributes A pointer to the attributes of the element.

Returns

Completion status of operation:

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

6.5.2.14 int xerEncNewLine (OSCTXT * *pctxt*)

This function inserts a new line symbol int the XML document.

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

Completion status of operation:

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

6.5.2.15 int xerEncNull (OSCTXT * *pctxt*, const char * *elemName*)

This function encodes an ASN.1 NULL placeholder. In XER the NULL value is represented as an empty element, such as <null/>.

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.

elemName This argument specifies the name of the element that is wrapped around the encoded value. If a null or empty string is passes (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.5.2.16 int xerEncObjId (OSCTXT * *pctxt*, const ASN1OBJID * *pvalue*, const char * *elemName*)

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

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.

pvalue 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.

elemName This argument specifies the name of the element that is wrapped around the encoded value. If the name is null, the default name for this type (<OBJECT_IDENTIFIER>) is added. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.5.2.17 int xerEncObjId64 (OSCTXT * *pctxt*, const ASN1OID64 * *pvalue*, const char * *elemName*)

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

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.

pvalue 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.

elemName This argument specifies the name of the element that is wrapped around the encoded value. If the name is null, the default name for this type (<OBJECT_IDENTIFIER>) is added. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.5.2.18 int xerEncOctStr (OSCTXT * *pctxt*, OSUINT32 *nocts*, const OSOCTET * *data*, const char * *elemName*)

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

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.

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

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

elemName This argument specifies the name of the element that is wrapped around the encoded value. If the name is null, the default name for this type (<OCTET_STRING>) is added. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.5.2.19 int xerEncOpenType (OSCTXT * *pctxt*, OSUINT32 *nocts*, const OSOCTET * *data*, const char * *elemName*)

This function encodes 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 n X.681). A variable of this is considered to be a previously encoded ASN.1 message component.

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.

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

data A pointer to an OCTET STRING containing an encoded ASN.1 message component.

elemName This argument specifies the name of the element that is wrapped around the encoded value. If the name is null, the default name for this type (<REAL>) is added. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.5.2.20 int xerEncReal (OSCTXT * *pctxt*, OSREAL *value*, const char * *elemName*)

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

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.

value A value to be encoded. Special real values plus and minus infinity are encoded by using the rtxGetPlusInfinity and rtxGetMinusInfinity functions to set the real value to be encoded.

elemName A pointer to the name of the element.

Returns

Completion status of operation:

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

6.5.2.21 int xerEncReal10 (OSCTXT * *pctxt*, const OSUTF8CHAR * *value*, const char * *elemName*)

This function encodes a variable of the REAL base 10 data type. This function provides support for the plus-infinity and the minus-infinity special real values. Use the rtxGetPlusInfinity or rtxGetMinusInfinity functions to get these special values.

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.

value A value to be encoded.

elemName A pointer to the name of the element.

Returns

Completion status of operation:

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

6.5.2.22 int xerEncRelativeOID (OSCTXT * *pctxt*, const ASN1OBJID * *pvalue*, const char * *elemName*)

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

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.

pvalue 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.

elemName This argument specifies the name of the element that is wrapped around the encoded value. If the name is null, the default name for this type (<RELATIVE_OID>) is added. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.5.2.23 int xerEncStartDocument (OSCTXT * *pctxt*)

This function encodes the starting record of the XML document (such as the <?xml version = "1.0" encoding = "UTF-8"?>). This function should be called prior to encoding any other fields in the document. After all elements in the document are encoded, xerEncEndDocument should be called.

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

Completion status of operation:

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

6.5.2.24 int xerEncStartElement (OSCTXT * *pctxt*, const char * *elemName*, const char * *attributes*)

This function encodes the string tag of the XML element, such as <element>. After the element's data is encoded, the xerEncEndElement function should be called.

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.

elemName A pointer to the name of the element.

attributes A pointer to the attributes of the element.

Returns

Completion status of operation:

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

6.5.2.25 int xerEncUInt (OSCTXT * *pctxt*, OSUINT32 *value*, const char * *elemName*)

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

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.e.

value An unsigned INTEGER value to be encoded. The ASNU1INT type is set to the C type 'unsigned int' in the asn1type.h file. This is assumed to represent a 32-bit integer value.

elemName This argument specifies the name of the element that is wrapped around the encoded value. If the name is null, the default name for this type (<INTEGER>) is added. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.5.2.26 int xerEncUInt64 (OSCTXT * *pctxt*, OSUINT64 *value*, const char * *elemName*)

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

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.

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 an unsigned 64-bit integer value.

elemName This argument specifies the name of the element that is wrapped around the encoded value. If the name is null, the default name for this type (<INTEGER>) is added. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.5.2.27 int xerEncUniCharData (OSCTXT * *pctxt*, const OSUNICHAR * *value*, OSUINT32 *nchars*)

This function encodes a variable of the ASN.1 character string types that are based on 16-bit character sets and are represented as null-terminated Unicode string. This includes IA5String, VisibleString, PrintableString, NumericString, and BMPString.

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.

value A pointer to a null-terminated 16-bit string to be encoded.

nchars The number of characters to be encoded.

Returns

Completion status of operation:

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

6.5.2.28 int xerEncUniCharStr (OSCTXT * *pctxt*, OSUNICHAR * *value*, const char * *elemName*)

This function encodes a variable one of the ASN.1 character string types that are based on 16-bit character sets and are represented as null-terminated Unicode strings. This includes IA5String, VisibleString, PrintableString, NumericString, Adn BMPString.

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.

value A pointer to a null-terminated 16-bit character string to be encoded.

elemName This argument specifies the name of the element that is wrapped around the encoded value. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.5.2.29 int xerEncUnivStr (OSCTXT * *pctxt*, const ASN1UniversalString * *value*, const char * *elemName*)

This function encodes a variable of the ASN.1 Universal character string. This differs from the encode routines for the character strings previously described in that the Universal string type is based o te 32-bit characters. A 32-bit character string is modeled using an array of unsigned integers.

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.

value A pointer to a structure representing a 16-bit character string to encoded. This structure contains a character count element and a pointer to an array of 32-bit character elements represented as 32-bit short integers.

elemName A pointer to a name of an element.

Returns

Completion status of operation:

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

6.5.2.30 int xerSetEncBufPtr (OSCTXT * *pCtxt*, OSOCTET * *bufaddr*, size_t *bufsiz*, OSBOOL *canonical*)

This function is used to set the internal buffer within the runtime library encode module. It must be called prior to calling any other generated or runtime library encode functions.

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.

bufaddr A pointer to a memory buffer containing the ASN.1 message. The pointer must point at the first byte in the message.

bufsiz The size of the message that was read. This is used to set an internal message size to check for field length errors. If this size is not known, a zero value can be passed to cause these checks to be bypassed.

canonical TRUE, if canonical XML encoding rules (CXER) should be used. Otherwise FALSE.

Returns

Completion status of operation:

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

6.6 XER C Utility Functions.

Functions

- OSBOOL xerCmpText (const XMLCHAR *text1, const char *text2)
- int **xerCopyText** (OSCTXT *pctxt, const char *text)
- int xerTextLength (const XMLCHAR *text)
- const char * xerTextToCStr (OSCTXT *pctxt, const XMLCHAR *text)
- size_t xerGetMsgLen (OSCTXT *pctxt)
- OSOCTET * xerGetMsgPtr (OSCTXT *pctxt)
- int **xerGetElemIdx** (const XMLCHAR *elemName, XerElemInfo *pElemInfo, int numElems)
- int **xerGetSeqElemIdx** (const XMLCHAR *elemName, XerElemInfo *pElemInfo, int numElems, int startIndex)
- int **xerFinalizeMemBuf** (OSRTMEMBUF *pMemBuf)
- int xerGetLibVersion ()
- const char * xerGetLibInfo ()

6.6.1 Detailed Description

The XER utility functions are common routines used by both XER encode and decode functions.

6.6.2 Function Documentation

6.6.2.1 OSBOOL xerCmpText (const XMLCHAR * *text1*, const char * *text2*)

This function is used to compare two strings: the first is represented as a 16-bit character null-terminated string and the second is represented as an 8-bit standard null-terminated string.

Parameters

text1 A pointer to a 16-bit character null-terminated string.

text2 A pointer to an 8-bit character null-terminated string.

Returns

The result of the comparison: TRUE, if strings match, otherwise FALSE.

6.6.2.2 const char* xerGetLibInfo ()

Returns information string describing the library. The string contains name of library, its version and flags used for building the library.

Returns

Information string

6.6.2.3 int xerGetLibVersion ()

Returns numeric version of run-time library. The format of version is as follows: MmP, where: M - major version number; m - minor version number; p - patch release number. For example, the value 581 means the version 5.81.

Returns

Version of run-time library in numeric format.

6.6.2.4 size_t xerGetMsgLen (OSCTXT * *pctxt*)

This function is used to get the encoded message length.

Parameters

pctxt A pointer to a context structure.

Returns

The length of a message in the buffer.

6.6.2.5 OSOCTET* xerGetMsgPtr (OSCTXT * *pctxt*)

This function is used to obtain a pointer to the start of an encoded message. This function is called after a complier generated encode function to get the pointer to the start of the encoded message. It is normally used when dynamic encoding is specified because the message pointer is not known until the encoding is complete. If static encoding is used, the message starts at the beginning of the specified buffer and the xerGetMsgLen function can be used to obtain the length of the message.

Parameters

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

Returns

A pointer to the beginning of the encoded message.

6.6.2.6 int xerTextLength (const XMLCHAR * *text*)

This function returns the length of a 16-bit character null-terminated string.

Parameters

text A pointer to a 16-bit character null-terminated string.

6.6.2.7 const char* xerTextToCStr (OSCTXT * *pctxt*, const XMLCHAR * *text*)

This function converts a 16-bit character string to a standard null-terminated C-string. This function will allocate dynamic memory to store the decoded result.

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.

text A pointer to a 16-bit character null-terminated string.

Returns

The resulting C string. The memory is allocated by using rtxMemAlloc function.

6.7 XML C Decode Functions.

Functions

- int xmlDecBitStr (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnbits, OSINT32 bufsize)
- int xmlDecBool (OSCTXT *pctxt, OSBOOL *pvalue)
- int xmlDecDynBitStr (OSCTXT *pctxt, ASN1DynBitStr *pvalue)
- int xmlDecDynNamedBitStr (OSCTXT *pctxt, ASN1DynBitStr *pvalue, const XmlNamedBitsDict *pBitDict)
- int xmlDecDynOctStr (OSCTXT *pctxt, ASN1DynOctStr *pvalue)
- int xmlDecGeneralizedTime (OSCTXT *pctxt, const char **outdata)
- int xmlDecNamedBitStr (OSRTMEMBUF *pMemBuf, OSOCTET *pData, int dataSize, OSUINT32 *pNumbits, const XmlNamedBitsDict *pBitDict, const XMLCHAR *chars, int length)
- int xmlDecOctStr (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSINT32 bufsize)
- int xmlDecReal (OSCTXT *pctxt, OSREAL *pvalue)
- int xmlDecUTCTime (OSCTXT *pctxt, const char **outdata)

6.7.1 Detailed Description

XML runtime library decode functions handle the decoding of the primitive ASN.1 data types and length variables. Calls to these functions are assembled in the C source code generated by the ASN1C complier 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.

6.7.2 Function Documentation

6.7.2.1 int xmlDecBitStr (OSCTXT * *pctxt*, OSOCTET * *pvalue*, OSUINT32 * *pnbits*, OSINT32 *bufsize*)

This function will decode 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 production.

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.

pvalue A 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 octets specified in the bufsize input parameter.

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

bufsize An integer variable containing the size (in octets) of the sized ASN.1 bit string. An error will occur if the number of octets in the decoded string is larger than this value.

Returns

Completion status of operation:

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

6.7.2.2 int xmlDecBool (OSCTXT * *pctxt*, OSBOOL * *pvalue*)

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

Parameters

pctxt Pointer to context block structure.

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

Returns

Completion status of operation:

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

6.7.2.3 int xmlDecDynBitStr (OSCTXT * *pctxt*, ASN1DynBitStr * *pvalue*)

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

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.

pvalue A pointer to a dynamic bit string structure to receive the decoded bit string. Dynamic memory is allocated to hold the string using the rtxMemAlloc function.

Returns

Completion status of operation:

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

6.7.2.4 int xmlDecDynNamedBitStr (OSCTXT * *pctxt*, ASN1DynBitStr * *pvalue*, const XmlNamedBitsDict * *pBitDict*)

This function will decode a list of identifiers into the array of octets. Identifiers should represent named bits value for BIT STRING. This function call is generated by ASN1C to decode a named bit production.

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.

pvalue A pointer to a dynamic bit string structure to receive the decoded bit string. Dynamic memory is allocated to hold the string using the rtxMemAlloc function.

pBitDict Bits' dictionary to be used to decode each bit. It is an array of name-value pairs, represented by an array of XmlNamedBitsDict structure.

Returns

Completion status of operation:

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

6.7.2.5 int xmlDecDynOctStr (OSCTXT * *pctxt*, ASN1DynOctStr * *pvalue*)

This function will decode a variable of the ASN.1 OCTET STRING type. This function will allocate dynamic memory to store the decoded result.

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.

pvalue A pointer to a dynamic bit string structure to receive the decoded bit string. Dynamic memory is allocated to hold the string using the rtxMemAlloc function.

Returns

Completion status of operation:

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

6.7.2.6 int xmlDecGeneralizedTime (OSCTXT * *pctxt*, const char ** *outdata*)

This function will decode a variable of the ASN.1 GeneralizedTime type. This function performs conversion between XML format of dateTime into the ASN.1 format. This function will allocate dynamic memory to store the result.

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.

outdata A pointer to a character string pointer variable to receive the decoded string. The string as stored as a standard null-terminated C string. Memory is allocated for the string by the rtxMemAlloc function. It will contain time in ASN.1 format.

Returns

Completion status of operation:

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

6.7.2.7 int xmlDecNamedBitStr (OSRTMEMBUF * *pMemBuf*, OSOCTET * *pData*, int *dataSize*, OSUINT32 * *pNumbits*, const XmlNamedBitsDict * *pBitDict*, const XMLCHAR * *chars*, int *length*)

This function will decode a list of identifiers into the array of octets. Identifiers should represent named bits value for BIT STRING. This function call is generated by ASN1C to decode a named bit production.

Parameters

pMemBuf Pointer to the destination memory buffer.

pData A 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 octets specified in the bufsize input parameter.

dataSize An integer variable containing the size (in octets) of the sized ASN.1 bit string. An error will occur if the number of octets in the decoded string is larger than this value.

pNumbits A pointer to an integer value to receive the decoded number of bits.

pBitDict Bits' dictionary to be used to decode each bit. It is an array of name-value pairs, represented by an array of XmlNamedBitsDict structure.

chars XML data to be appended to memory buffer before parsing. Could be NULL, if it is final call to this function.

length Number of characters in XML data to be appended to memory buffer before parsing. Could be 0, if it is final call to this function.

Returns

Completion status of operation:

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

6.7.2.8 int xmlDecOctStr (OSCTXT * *pctxt*, OSOCTET * *pvalue*, OSUINT32 * *pnocts*, OSINT32 *bufsize*)

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

Parameters

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.

pvalue A 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 octets specified in the bufsize input parameter.

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

bufsize 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.

Returns

Completion status of operation:

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

6.7.2.9 int xmlDecReal (OSCTXT * *pctxt*, OSREAL * *pvalue*)

This function will decode a variable of the ASN.1 32-bit character UniversalString type. This includes the Universal-String type.

Parameters

pctxt A 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.

pvalue A pointer to a structure variable to receive the decoded string. The string as stored as an array of unsigned integer characters. Memory is allocated for the string by the rtxMemAlloc function.

Returns

Completion status of operation:

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

6.7.2.10 int xmlDecUTCTime (OSCTXT * *pctxt*, const char ** *outdata*)

This function will decode a variable of the ASN.1 UTCTime type. This function performs conversion between XML format of dateTime into the ASN.1 format. This function will allocate dynamic memory to store the result.

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.

outdata A pointer to a character string pointer variable to receive the decoded string. The string as stored as a standard null-terminated C string. Memory is allocated for the string by the rtxMemAlloc function. It will contain time in ASN.1 format.

Returns

Completion status of operation:

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

6.8 XML C Encode Functions.

Functions

- int xmlEncBitStr (OSCTXT *pctxt, XmlNamedBitsDict *namedbits, OSUINT32 noofnamedbits, OSUINT32 nbits, const OSOCTET *data, const char *elemName, ASN1StrType outputType)
- int **xmlEncBoolValue** (OSCTXT *pctxt, OSBOOL value)
- int xmlEncBool (OSCTXT *pctxt, OSBOOL value, const char *elemName)
- int xmlEncEnum (OSCTXT *pctxt, const char *value)
- int xmlEncGeneralizedTime (OSCTXT *pctxt, const char *value, const char *elemName)
- int xmlEncNamedValue (OSCTXT *pctxt, const char *value, const char *elemName, const char *attributes)
- int **xerEncOpenTypeExt** (OSCTXT *pctxt, OSRTDList *pElemList)
- int xmlEncReal (OSCTXT *pctxt, OSREAL value, const char *elemName)
- int xmlEncUTCTime (OSCTXT *pctxt, const char *value, const char *elemName)

6.8.1 Detailed Description

The XML low-level encode functions handle the XML encoding of primitive ASN.1 data types. In most cases XER encoding functions can be used for encoding XML. But there are some differences between XER and XML encodings, described in X.693 and X.694. These functions are very similar to XER ones, and only functions provide different encoding are added.

6.8.2 Function Documentation

6.8.2.1 int xmlEncBitStr (OSCTXT * *pctxt*, XmlNamedBitsDict * *namedbits*, OSUINT32 *noofnamedbits*, OSUINT32 *nbits*, const OSOCTET * *data*, const char * *elemName*, ASN1StrType *outputType*)

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

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.

namedbits Bits' dictionary to be used to encode each bit. It is an array of name-value pairs, represented by an array of XmlNamedBitsDict structure.

noofnamedbits Number of named bits in bits' dictionary.

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

data 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.

elemName This argument specifies the name of the element that is wrapped around the encoded value. If the name is null, the default name for this type (<BIT_STRING>) is added. If an empty string is passed (""), no element tag is added to the encoded value.

outputType An enumerated type whose value is set to either 'ASN1BIN' (for binary format) or 'ASN1HEX' (for hexadecimal format).

Returns

Completion status of operation:

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

6.8.2.2 int xmlEncBool (OSCTXT * *pctxt*, OSBOOL *value*, const char * *elemName*)

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

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.

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.

elemName This argument specifies the name of the element that is wrapped around the encoded value. If the name is null, the default name for this type (<BOOLEAN>) is added. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.8.2.3 int xmlEncEnum (OSCTXT * *pctxt*, const char * *value*)

This function encodes an enumeration value for the ASN.1 ENUMERATED and INTEGER types.

Parameters

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

value An enumeration identifier to be encoded.

Returns

Completion status of operation:

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

6.8.2.4 int xmlEncGeneralizedTime (OSCTXT * *pctxt*, const char * *value*, const char * *elemName*)

This function encodes a variable of the ASN.1 GeneralizedTime type. It performs conversion from ASN.1 time format into the XML dateTime format.

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.

value A pointer to a null-terminated C character string to be encoded. It should contain GeneralizedTime in ASN.1 format.

elemName This argument specifies the name of the element that is wrapped around the encoded value. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.8.2.5 int xmlEncNamedValue (OSCTXT * *pctxt*, const char * *value*, const char * *elemName*, const char * *attributes*)

This function encodes a named value, for example an enumerated value, such as <element>value</element>.

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.

value A pointer to a value string

elemName A pointer to the name of the element.

attributes A pointer to the attributes of the element.

Returns

Completion status of operation:

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

6.8.2.6 int xmlEncReal (OSCTXT * *pctxt*, OSREAL *value*, const char * *elemName*)

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

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.

value A value to be encoded. Special real values are encoded by using the apropriate functions to set the real value to be encoded.

elemName A pointer to the name of the element.

Returns

Completion status of operation:

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

6.8.2.7 int xmlEncUTCTime (OSCTXT * *pctxt*, const char * *value*, const char * *elemName*)

This function encodes a variable of the ASN.1 UTCTime type. It performs conversion from ASN.1 time format into the XML dateTime format.

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.

value A pointer to a null-terminated C character string to be encoded. It should contain UTCTime in ASN.1 format.

elemName This argument specifies the name of the element that is wrapped around the encoded value. If an empty string is passed (""), no element tag is added to the encoded value.

Returns

Completion status of operation:

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

6.9 C++ classes for streaming XER encoding.

Classes

- class ASN1XEREncodeStream

6.9.1 Detailed Description

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

6.10 C++ classes for streaming XER decoding.

Classes

- class ASN1XERDecodeStream

6.10.1 Detailed Description

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

Chapter 7

Class Documentation

7.1 ASN1SAX_XEROpenType Struct Reference

Public Attributes

- ASN1SAXCDecodeHandlerBase **mSaxBase**
- ASN1OpenType * **mpMsgData**
- OSCTXT **mEncCtxt**

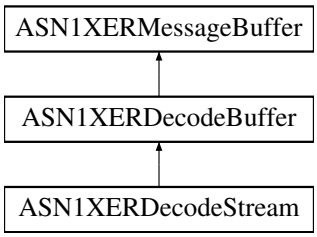
The documentation for this struct was generated from the following file:

- ASN1SAX_XEROpenType.h

7.2 ASN1XERDecodeBuffer Class Reference

`#include <asn1XerCppTypes.h>`

Inheritance diagram for ASN1XERDecodeBuffer:



Public Member Functions

- ASN1XERDecodeBuffer (const char *xmlFile)
- ASN1XERDecodeBuffer (const OSOCTET *msgbuf, size_t numocts, OSBOOL openType=FALSE)
- ASN1XERDecodeBuffer (OSRTInputStreamIF &inputStream, OSBOOL openType=FALSE)
- virtual int decodeXML (OSXMLReaderClass *pReader)
- const char * getXmlFileName ()
- virtual int initBuffer (OSRTMEMBUF &membuf)
- virtual int initBuffer (const OSUTF8CHAR *str)
- virtual int initBuffer (OSUNICHAR *str)
- virtual OSBOOL **isA** (int bufferType)
- void **setOpenType** ()

Protected Types

- enum { **INPUT_FILE**, **INPUT_STREAM**, **INPUT_STREAM_ATTACHED**, **INPUT_MEMORY** }

Protected Attributes

- union {
  const char * **fileName**
  OSRTInputStreamIF * **pInputStream**
  struct {
  const OSOCTET * **pMemBuf**
  int **bufSize**
  } **memBuf**
  } **mInput**

- enum ASN1XERDecodeBuffer:: { ... } **mInputId**

7.2.1 Detailed Description

The ANS1XERDecodeBuffer class is derived from the ASN1XERMessageBuffer base class. It contains variables and methods specific to decoding ASN.1 XER messages. It is used to manage the input buffer containing an ASN.1 message to be decoded.

Note that the XER decode buffer object does not take a message buffer argument because buffer management is handled by the XML parser.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 ASN1XERDecodeBuffer::ASN1XERDecodeBuffer (const char * *xmlFile*)

The ASN1XERDecodeBuffer class has three overloaded constructors. This version of the ASN1XERDecodeBuffer constructor takes a name of a file that contains the XML data to be decoded, and constructs a buffer. Use getStatus() method to determine has error occured during the initialization or not.

Parameters

xmlFile A pointer to name of file to be decoded.

7.2.2.2 ASN1XERDecodeBuffer::ASN1XERDecodeBuffer (const OSOCTET * *msgbuf*, size_t *numocts*, OSBOOL *openType* = `FALSE`)

This version of the ASN1XERDecodeBuffer constructor takes parameters describing the message to be decoded, and constructs a buffer describing an encoded ASN.1 message. Use getStatus() method to determine has error occured during the initialization or not.

Parameters

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

numocts Size of the message buffer.

openType Open Type decoding indicator. Set this argument as TRUE to decode an open type.

7.2.2.3 ASN1XERDecodeBuffer::ASN1XERDecodeBuffer (OSRTInputStreamIF & *inputStream*, OSBOOL *openType* = `FALSE`)

This version of the ASN1XERDecodeBuffer constructor takes a reference to the OSRTInputStream object. Use getStatus() method to determine has error occured during the initialization or not.

Parameters

inputStream reference to the OSRTInputStream object

openType Open Type decoding indicator. Set this argument as TRUE to decode an open type.

7.2.3 Member Function Documentation

7.2.3.1 virtual int ASN1XERDecodeBuffer::decodeXML (OSXMLReaderClass * *pReader*) `[virtual]`

This method decodes an XML message associated with this buffer.

Returns

stat 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++ runtime Common Functions Reference Manual.

Parameters

pReader Pointer to OSXMLReaderClass object.

7.2.3.2 const char* ASN1XERDecodeBuffer::getXmlFileName () [inline]

This method returns the name of the XML file that is associated with the current buffer.

Returns

Name of the XML file that is associated with the current buffer.

7.2.3.3 virtual int ASN1XERDecodeBuffer::initBuffer (OSUNICHAR * *str*) [inline, virtual]

This version of the overloaded initBuffer method initializes the message buffer to point at the given Unicode string. This is used mainly for XER (XML) message decoding.

Parameters

str Pointer to a Unicode character string.

Returns

Completion status of operation:

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

7.2.3.4 virtual int ASN1XERDecodeBuffer::initBuffer (const OSUTF8CHAR * *str*) [virtual]

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

Parameters

str Pointer to a null-terminated ASCII character string.

Returns

Completion status of operation:

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

7.2.3.5 virtual int ASN1XERDecodeBuffer::initBuffer (OSRTMEMBUF & *membuf*) [inline, virtual]

This method reinitializes the decode buffer pointer to allow new data to be decoded. This makes it possible to reuse one message buffer object in a loop to decode multiple data messages.

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++ Run-time Common Functions Reference Manual.

Parameters

membuf reference to OSMemBuf object.

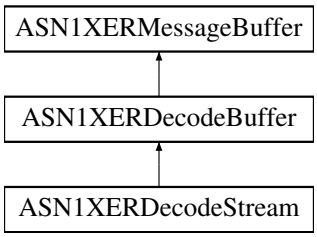
The documentation for this class was generated from the following file:

- asn1XerCppTypes.h

7.3 ASN1XERDecodeStream Class Reference

`#include <ASN1XERDecodeStream.h>`

Inheritance diagram for ASN1XERDecodeStream:



Public Member Functions

- ASN1XERDecodeStream (OSRTInputStreamIF &is, OSBOOL openType=FALSE)
- **ASN1XERDecodeStream** (OSRTInputStreamIF *pis, OSBOOL bOwnStream=TRUE, OSBOOL openType=FALSE)
- virtual void * getAppInfo ()
- virtual OSRTCtxtPtr getContext ()
- virtual OSCTXT * getCtxtPtr ()
- virtual char * getErrorInfo ()
- virtual char * getErrorInfo (char *pBuf, size_t &bufSize)
- virtual int getStatus () const
- virtual void printErrorInfo ()
- virtual void resetErrorInfo ()
- virtual void setAppInfo (void *pAppInfo)
- virtual void setDiag (OSBOOL value=TRUE)
- virtual int close ()
- virtual int flush ()
- virtual OSBOOL isOpened ()
- virtual 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 skip (size_t n)
- ASN1XERDecodeStream & operator>> (ASN1CType &val)
- int decodeObj (ASN1CType &val)
- OSBOOL isA (int bufferType)

Protected Attributes

- OSRTInputStreamIF * **mpStream**
- OSBOOL **mbOwnStream**

7.3.1 Detailed Description

This class is a base class for other ASN.1 XER input stream's classes. It is derived from the ASN1Stream base class. It contains variables and methods specific to streaming decoding of XER messages. It is used to manage the input stream containing the ASN.1 message to be decoded.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 ASN1XERDecodeStream::ASN1XERDecodeStream (OSRTInputStreamIF & *is*, OSBOOL *openType* = `FALSE`)

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

7.3.3 Member Function Documentation

7.3.3.1 virtual int ASN1XERDecodeStream::close () `[inline, virtual]`

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

Returns

Completion status of operation:

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

See also

rtxStreamClose

7.3.3.2 virtual size_t ASN1XERDecodeStream::currentPos () `[inline, virtual]`

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

Returns

The number of octets already read from the stream.

7.3.3.3 int ASN1XERDecodeStream::decodeObj (ASN1CType & *val*)

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

Parameters

val A reference to an object to be decoded.

Returns

Completion status of operation:

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

7.3.3.4 virtual int ASN1XERDecodeStream::flush () [inline, virtual]

Flushes the buffered data to the stream.

Returns

Completion status of operation:

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

See also

rtxStreamFlush

7.3.3.5 virtual void* ASN1XERDecodeStream::getAppInfo () [inline, virtual]

Returns a pointer to application-specific information block

7.3.3.6 virtual OSRTCtxtPtr ASN1XERDecodeStream::getContext () [inline, virtual]

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

Returns

Context smart pointer object.

7.3.3.7 virtual OSCTXT* ASN1XERDecodeStream::getCtxtPtr () [inline, virtual]

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

Returns

The pointer to C runtime context.

7.3.3.8 virtual char* ASN1XERDecodeStream::getErrorInfo (char * *pBuf*, size_t & *bufSize*) [inline, virtual]

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

Parameters

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

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

Returns

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

7.3.3.9 virtual char* ASN1XERDecodeStream::getErrorInfo () [inline, virtual]

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

Returns

A pointer to a newly allocated buffer with error text.

7.3.3.10 virtual int ASN1XERDecodeStream::getStatus () const [inline, virtual]

This method returns the completion status of previous operation. It can be used to check completion status of constructors or methods, which do not return completion status. If error occurs, use printErrorInfo method to print out the error's description and stack trace. Method resetError can be used to reset error to continue operations after recovering from the error.

Returns

Runtime status code:

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

7.3.3.11 OSBOOL ASN1XERDecodeStream::isA (int *bufferType*) [virtual]

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

Parameters

bufferType Enumerated identifier specifying a derived class. This type is defined as a public access type in the ASN1MessageBufferIF base interface. Possible values are: BEREncode, BERDecode, PEREncode, PERDecode, XEREncode, XERDecode, XMLEncode, XMLDecode, Stream.

Returns

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

Reimplemented from ASN1XERDecodeBuffer.

7.3.3.12 virtual OSBOOL ASN1XERDecodeStream::isOpened () [inline, virtual]

Checks, is the stream opened or not.

Returns

TRUE, if the stream is opened, FALSE otherwise.

See also

rtxStreamIsOpened

7.3.3.13 int ASN1XERDecodeStream::mark (size_t *readAheadLimit*) [inline]

This method marks the current position in this input stream. A subsequent call to the ASN1XERDecodeStream::reset function repositions this stream at the last marked position so that subsequent reads re-read the same bytes. The readAheadLimit argument tells this input stream to allow that many bytes to be read before the mark position gets invalidated.

Parameters

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

Returns

Completion status of operation:

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

See also

rtxStreamBufMark, rtxStreamBufReset

7.3.3.14 virtual OSBOOL ASN1XERDecodeStream::markSupported () [inline, virtual]

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

Returns

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

See also

rtxStreamIsMarkSupported

7.3.3.15 ASN1XERDecodeStream& ASN1XERDecodeStream::operator>> (ASN1CType & *val*)

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

Parameters

val A reference to an object to be decoded.

Returns

reference to this class to perform sequential decoding.

7.3.3.16 virtual void ASN1XERDecodeStream::printErrorInfo () [inline, virtual]

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

7.3.3.17 virtual long ASN1XERDecodeStream::read (OSOCTET * *pDestBuf*, size_t *maxToRead*) [inline, virtual]

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

Parameters

pDestBuf Pointer to a buffer to receive a data.

maxToRead Size of the buffer.

Returns

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

See also

rtxStreamRead

7.3.3.18 virtual long ASN1XERDecodeStream::readBlocking (OSOCTET * *pDestBuf*, size_t *toReadBytes*) [inline, virtual]

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

Parameters

pDestBuf Pointer to a buffer to receive a data.

toReadBytes Number of bytes to be read.

Returns

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

See also

rtxStreamRead

7.3.3.19 int ASN1XERDecodeStream::reset () [inline]

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

Returns

Completion status of operation:

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

See also

rtxStreamBufMark, rtxStreamBufReset

7.3.3.20 virtual void ASN1XERDecodeStream::resetErrorInfo () [inline, virtual]

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

7.3.3.21 virtual void ASN1XERDecodeStream::setAppInfo (void * *pAppInfo*) [inline, virtual]

Sets the application-specific information block.

7.3.3.22 virtual void ASN1XERDecodeStream::setDiag (OSBOOL *value* = TRUE) [inline, virtual]

The setDiag method will turn diagnostic tracing on or off.

Parameters

value - Boolean value (default = TRUE = on)

7.3.3.23 virtual int ASN1XERDecodeStream::skip (size_t *n*) [inline, virtual]

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

Parameters

n The number of octets to be skipped.

Returns

Completion status of operation:

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

See also

rtxStreamSkip

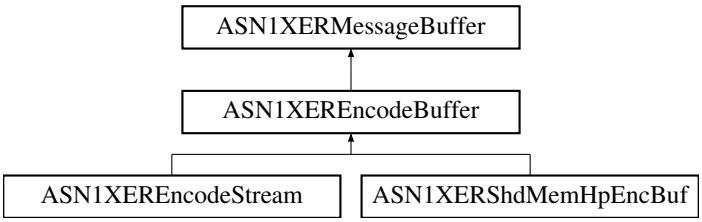
The documentation for this class was generated from the following file:

- ASN1XERDecodeStream.h

7.4 ASN1XEREncodeBuffer Class Reference

`#include <asn1XerCppTypes.h>`

Inheritance diagram for ASN1XEREncodeBuffer:



Public Member Functions

- ASN1XEREncodeBuffer ()
- ASN1XEREncodeBuffer (OSBOOL canonical)
- ASN1XEREncodeBuffer (OSBOOL canonical, OSRTContext *pContext, size_t initBufSize=0)
- ASN1XEREncodeBuffer (OSOCTET *pMsgBuf, size_t msgBufLen)
- ASN1XEREncodeBuffer (OSOCTET *pMsgBuf, size_t msgBufLen, OSBOOL canonical, OSBOOL openType=FALSE)
- size_t getMsgLen ()
- int init ()
- virtual OSBOOL isA (int bufferType)
- void setCanonical ()
- void setOpenType ()
- long write (const char *filename)
- long write (FILE *fp)

7.4.1 Detailed Description

The ASN1XEREncodeBuffer class is derived from the ASN1XERMessageBuffer base class. It contains variables and methods specific to encoding ASN.1 messages using the XML Encoding Rules (XER). It is used to manage the buffer into which an ASN.1 message is to be encoded.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 ASN1XEREncodeBuffer::ASN1XEREncodeBuffer ()

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

7.4.2.2 ASN1XEREncodeBuffer::ASN1XEREncodeBuffer (OSBOOL *canonical*)

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

Parameters

canonical Indicates the usage of Canonical XER(CXER).

7.4.2.3 ASN1XEREncodeBuffer::ASN1XEREncodeBuffer (OSBOOL *canonical*, OSRTContext * *pContext*, size_t *initBufSize* = 0)

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

Parameters

canonical Indicates the usage fo Canonical XER(CXER).

pContext Pointer to existing context to reference.

initBufSize Initial size of encode buffer

7.4.2.4 ASN1XEREncodeBuffer::ASN1XEREncodeBuffer (OSOCTET * *pMsgBuf*, size_t *msgBufLen*)

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

Parameters

pMsgBuf A pointer to a fixed size message buffer to receive the encoded message.

msgBufLen Size of the fixed-size message buffer.

7.4.2.5 ASN1XEREncodeBuffer::ASN1XEREncodeBuffer (OSOCTET * *pMsgBuf*, size_t *msgBufLen*, OSBOOL *canonical*, OSBOOL *openType* = FALSE)

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

Parameters

pMsgBuf A pointer to a fixed size message buffer to receive the encoded message.

msgBufLen Size of the fixed-size message buffer.

canonical Indicates the usage fo Canonical XER(CXER).

openType Open Type encoding indicator. Set this argument as TRUE to encode an open type.

7.4.3 Member Function Documentation

7.4.3.1 size_t ASN1XEREncodeBuffer::getMsgLen () [inline]

This method returns the length of a previously encoded XER message.

Returns

len Length of the XER message encapsulated within this buffer object.

7.4.3.2 int ASN1XEREncodeBuffer::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.

Returns

Completion status of operation:

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

7.4.3.3 virtual OSBOOL ASN1XEREncodeBuffer::isA (int *bufferType*) [inline, virtual]

This method checks the type of the message buffer.

Parameters

bufferType Enumerated identifier specifying a derived class. The only possible value for this class is XEREncode.

Returns

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

Reimplemented in ASN1XEREncodeStream.

7.4.3.4 void ASN1XEREncodeBuffer::setCanonical () [inline]

This method turns Canonical XML Encoding Rules (CXER) usage on.

7.4.3.5 void ASN1XEREncodeBuffer::setOpenType () [inline]

This method turns Open Type encoding on.

7.4.3.6 long ASN1XEREncodeBuffer::write (FILE * *fp*)

The second version writes to a file that is specified by a FILE pointer.

Parameters

fp The pointer to FILE structure for writing the encoded message.

Returns

Number of octets actually written, which may be less than real message length if an error occurs.

7.4.3.7 long ASN1XEREncodeBuffer::write (const char * *filename*)

There are two versions of this method. The first version writes to a file that is specified by file name.

Parameters

filename The name of file for writing the encoded message.

Returns

Number of octets actually written, which may be less than real message length if an error occurs.

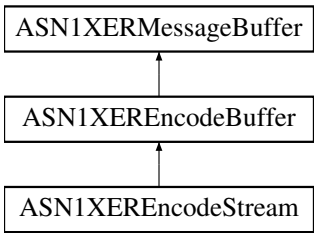
The documentation for this class was generated from the following file:

- asn1XerCppTypes.h

7.5 ASN1XEREncodeStream Class Reference

`#include <ASN1XEREncodeStream.h>`

Inheritance diagram for ASN1XEREncodeStream:



Public Member Functions

- ASN1XEREncodeStream (OSRTOutputStreamIF &os, OSBOOL canonical=FALSE)
- **ASN1XEREncodeStream** (OSRTOutputStreamIF *pos, OSBOOL bOwnStream=TRUE, OSBOOL canonical=FALSE)
- virtual void * getAppInfo ()
- virtual OSRTCtxtPtr getContext ()
- virtual OSCTXT * getCtxtPtr ()
- virtual char * getErrorInfo ()
- virtual char * getErrorInfo (char *pBuf, size_t &bufSize)
- virtual int getStatus () const
- virtual void printErrorInfo ()
- virtual void resetErrorInfo ()
- virtual void setAppInfo (void *pAppInfo)
- virtual void setDiag (OSBOOL value=TRUE)
- virtual int close ()
- virtual int flush ()
- virtual OSBOOL isOpened ()
- virtual long write (const OSOCTET *pdata, size_t size)
- ASN1XEREncodeStream & operator<< (ASN1CType &val)
- int encodeObj (ASN1CType &val)
- OSBOOL isA (int bufferType)

Protected Attributes

- OSRTOutputStreamIF * **mpStream**
- OSBOOL **mbOwnStream**

7.5.1 Detailed Description

This class is a base class for other ASN.1 XER output stream's classes. It is derived from the ASN1Stream base class. It contains variables and methods specific to streaming encoding of XER messages.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 ASN1XEREncodeStream::ASN1XEREncodeStream (OSRTOutputStreamIF & *os*, OSBOOL *canonical* = `FALSE`)

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

7.5.3 Member Function Documentation

7.5.3.1 virtual int ASN1XEREncodeStream::close () `[inline, virtual]`

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

Returns

Completion status of operation:

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

See also

rtxStreamClose

7.5.3.2 int ASN1XEREncodeStream::encodeObj (ASN1CType & *val*)

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

Parameters

val A reference to an object to be encoded.

Returns

Completion status of operation:

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

7.5.3.3 virtual int ASN1XEREncodeStream::flush () `[inline, virtual]`

Flushes the buffered data to the stream.

Returns

Completion status of operation:

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

See also

rtxStreamFlush

7.5.3.4 virtual void* ASN1XEREncodeStream::getAppInfo () [inline, virtual]

Returns a pointer to application-specific information block

7.5.3.5 virtual OSRTCtxtPtr ASN1XEREncodeStream::getContext () [inline, virtual]

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

Returns

Context smart pointer object.

7.5.3.6 virtual OSCTXT* ASN1XEREncodeStream::getCtxtPtr () [inline, virtual]

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

Returns

The pointer to C runtime context.

7.5.3.7 virtual char* ASN1XEREncodeStream::getErrorInfo (char * *pBuf*, size_t & *bufSize*) [inline, virtual]

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

Parameters

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

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

Returns

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

7.5.3.8 virtual char* ASN1XEREncodeStream::getErrorInfo () [inline, virtual]

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

Returns

A pointer to a newly allocated buffer with error text.

7.5.3.9 virtual int ASN1XEREncodeStream::getStatus () const [inline, virtual]

This method returns the completion status of previous operation. It can be used to check completion status of constructors or methods, which do not return completion status. If error occurs, use printErrorInfo method to print out the error's description and stack trace. Method resetError can be used to reset error to continue operations after recovering from the error.

Returns

Runtime status code:

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

7.5.3.10 OSBOOL ASN1XEREncodeStream::isA (int *bufferType*) [virtual]

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

Parameters

bufferType Enumerated identifier specifying a derived class. This type is defined as a public access type in the ASN1MessageBufferIF base interface. Possible values are: BEREncode, BERDecode, PEREncode, PERDecode, XEREncode, XERDecode, XMLEncode, XMLDecode, Stream.

Returns

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

Reimplemented from ASN1XEREncodeBuffer.

7.5.3.11 virtual OSBOOL ASN1XEREncodeStream::isOpened () [inline, virtual]

Checks, is the stream opened or not.

Returns

TRUE, if the stream is opened, FALSE otherwise.

See also

rtxStreamIsOpened

7.5.3.12 ASN1XEREncodeStream& ASN1XEREncodeStream::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.

Parameters

val A reference to an object to be encoded.

Returns

reference to this class to perform sequential encoding.

7.5.3.13 virtual void ASN1XEREncodeStream::printErrorInfo () [inline, virtual]

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

7.5.3.14 virtual void ASN1XEREncodeStream::resetErrorInfo () [inline, virtual]

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

7.5.3.15 virtual void ASN1XEREncodeStream::setAppInfo (void * *pAppInfo*) [inline, virtual]

Sets the application-specific information block.

7.5.3.16 virtual void ASN1XEREncodeStream::setDiag (OSBOOL *value* = TRUE) [inline, virtual]

The setDiag method will turn diagnostic tracing on or off.

Parameters

value - Boolean value (default = TRUE = on)

7.5.3.17 virtual long ASN1XEREncodeStream::write (const OSOCTET * *pdata*, size_t *size*) [inline, virtual]

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

Parameters

pdata Pointer to the data to be written.

size The number of octets to write.

Returns

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

See also

rtxStreamWrite

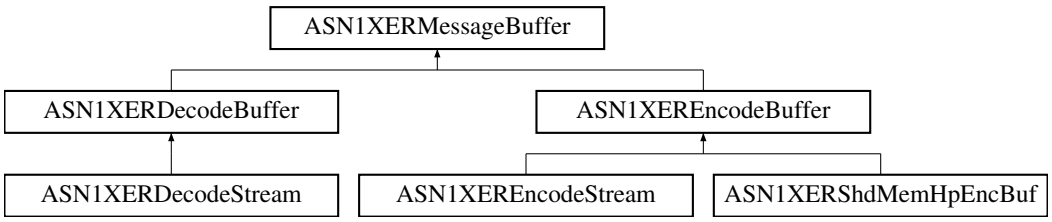
The documentation for this class was generated from the following file:

- ASN1XEREncodeStream.h

7.6 ASN1XERMessageBuffer Class Reference

`#include <asn1XerCppTypes.h>`

Inheritance diagram for ASN1XERMessageBuffer:



Protected Member Functions

- ASN1XERMessageBuffer (Type bufferType)
- ASN1XERMessageBuffer (Type bufferType, OSRTContext *pContext)

7.6.1 Detailed Description

The XER message buffer class is derived from the ASN1MessageBuffer base class. It is the base class for the ASN1XEREncodeBuffer and ASN1XERDecodeBuffer classes. It contains variables and methods specific to encoding or decoding ASN.1 messages using the XML Encoding Rules (XER). It is used to manage the buffer into which an ASN.1 message is to be encoded or decoded.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 ASN1XERMessageBuffer::ASN1XERMessageBuffer (Type *bufferType*) `[protected]`

This protected constructor creates a new context. Use the getStatus() method to determine if an error occured during initialization or not.

Parameters

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

7.6.2.2 ASN1XERMessageBuffer::ASN1XERMessageBuffer (Type *bufferType*, OSRTContext * *pContext*) `[inline, protected]`

This protected constructor creates uses an existing context to construct a message buffer. Use the getStatus() method to determine if an error occured during initialization or not.

Parameters

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

pContext Pointer to context to use.

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

- asn1XerCppTypes.h

7.7 ASN1XERSAXDecodeHandler Class Reference

`#include <asn1XerCppTypes.h>`

Classes

- struct ErrorInfo

Public Member Functions

- ASN1XERSAXDecodeHandler ()
- virtual int **startElement** (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const *attrs)
- virtual int **characters** (const OSUTF8CHAR *const chars, unsigned int length)
- virtual int **endElement** (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)
- virtual void **startDocument** ()
- virtual void **endDocument** ()
- virtual void **resetErrorInfo** ()
- virtual void **setErrorInfo** (int status, const char *file=0, int line=0)
- virtual int **getErrorInfo** (int *status, const char **file, int *line)
- ASN1XERState getState ()
- virtual int **finalize** ()
- OSCTXT * **finalizeMemBuf** (OSRTMessageBufferIF *msgBuf, OSRTMEMBUF &memBuf)
- virtual void **init** (int level=0)
- OSBOOL isComplete ()
- void **setTypeName** (const char *typeName)

Protected Attributes

- ASN1XERState **mCurrState**
- int **mCurrElemID**
- int **mLevel**
- int **mStartLevel**
- const char * **mpTypeName**
- struct ASN1XERSAXDecodeHandler::ErrorInfo **errorInfo**

7.7.1 Detailed Description

This class is derived from the SAX class DefaultHandler base class. It contains variables and methods specific to XML decoding ASN.1 messages. It is used to intercept standard SAX parser events, such as startElement, characters, endElement. This class is used as a base class for XER ASN1C generated ASN1C_* classes.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 ASN1XERSAXDecodeHandler::ASN1XERSAXDecodeHandler () `[inline]`

Default constructor without parameters.

7.7.3 Member Function Documentation

7.7.3.1 ASN1XERState ASN1XERSAXDecodeHandler::getState () [inline]

This method returns the current state of the decoding process.

Returns

The state of the decoding process as type ASN1XERState. Can be XERINIT, XERSTART, XERDATA, or XEREND.

7.7.3.2 OSBOOL ASN1XERSAXDecodeHandler::isComplete () [inline]

This method returns the completion status of the decoding process.

Returns

The completion state of decoding process, as OSBOOL. Can be TRUE (completed) or FALSE (not completed).

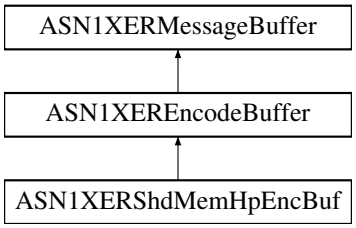
The documentation for this class was generated from the following file:

- asn1XerCppTypes.h

7.8 ASN1XERShdMemHpEncBuf Class Reference

`#include <asn1XerCppTypes.h>`

Inheritance diagram for ASN1XERShdMemHpEncBuf:



Public Member Functions

- ASN1XERShdMemHpEncBuf (OSBOOL canonical, OSRTContext *pContext, size_t initBufSize=0)
- ASN1XERShdMemHpEncBuf (OSBOOL canonical)
- ~ASN1XERShdMemHpEncBuf ()

Protected Attributes

- void * **mpSavedMemHeap**

7.8.1 Detailed Description

The ASN1XERShdMemHpEncBuf (shared memory heap) class is derived from the ASN1XEREncodeBuffer class. It is designed to use the memory heap of the given context object. Its primary use is for decoding open type data.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 ASN1XERShdMemHpEncBuf::ASN1XERShdMemHpEncBuf (OSBOOL *canonical*, OSRTContext * *pContext*, size_t *initBufSize* = 0)

This constructor creates a shared memory heap encode buffer by using the memory heap in the pContext object.

Parameters

canonical Indicates the usage fo Canonical XER(CXER).

pContext Pointer to existing context to reference.

initBufSize Initial size of encode buffer

7.8.2.2 ASN1XERShdMemHpEncBuf::ASN1XERShdMemHpEncBuf (OSBOOL *canonical*)

This constructor is a passthru to the underlying XER encode buffer constructor.

Parameters

canonical Indicates the usage of Canonical XER(CXER).

7.8.2.3 ASN1XERShdMemHpEncBuf::~ASN1XERShdMemHpEncBuf ()

The destructor frees the original context memory heap.

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

- asn1XerCppTypes.h

7.9 ASN1XERString Class Reference

Static Public Member Functions

- static unsigned int **stringLen** (const XMLCHAR ∗const str)
- static unsigned int **stringLen** (const XMLCHAR16 ∗const str)
- static XMLCHAR ∗ **replicate** (const XMLCHAR ∗const str)
- static XMLCHAR ∗ **transcode** (const char ∗const str)
- static char ∗ **transcode** (const XMLCHAR16 ∗const toTranscode)
- static int **compareString** (const XMLCHAR ∗const str1, const XMLCHAR ∗const str2)
- static XMLCHAR ∗ **catString** (XMLCHAR ∗destStr, int destStrSize, const XMLCHAR ∗const str)
- static XMLCHAR ∗ **catString** (XMLCHAR ∗destStr, int destStrSize, const XMLCHAR16 ∗const str)
- static int **stringUTF8Len** (const unsigned char ∗utf8Str, int strLen)
- static XMLCHAR16 ∗ **transUTF8** (const unsigned char ∗utf8Str, int ∗strSize, XMLCHAR16 ∗destBuf, int destBufSize)
- static XMLCHAR ∗ **safeTranscode** (const XMLCHAR16 ∗str, int ∗strSize, XMLCHAR ∗staticBuf, int bufSize)
- static XMLCHAR16 ∗ **safeTranscode** (const XMLCHAR ∗str, int ∗strSize, XMLCHAR16 ∗staticBuf, int bufSize)
- static const XMLCHAR ∗ **zeroLenString** ()

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

- ASN1XERString.h

7.10 ASN1XERSAXDecodeHandler::ErrorInfo Struct Reference

Public Attributes

- int **stat**
- const char * **file**
- int **line**

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

- asn1XerCppTypes.h

7.11 XerElemInfo Struct Reference

Public Attributes

- const char * **name**
- OSBOOL **optional**

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

- asn1xer.h

7.12 XmlNamedBitsDict Struct Reference

Public Attributes

- const char * **name**
- OSUINT32 **bitnum**

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

- asn1xer.h

Chapter 8

File Documentation

8.1 ASN1CXerOpenType.h File Reference

8.1.1 Detailed Description

Holds class that defines standard SAX event handlers for decoding XER open data type

8.2 ASN1SAX_XEROpenType.h File Reference

```
#include "asn1XerCTypes.h"
#include <stdio.h>
#include <stdlib.h>
#include <setjmp.h>
#include "asn1xer.h"
#include "rtxmlsrc/rtSaxCParser.h"
```

Classes

- struct ASN1SAX_XEROpenType

Typedefs

- typedef struct ASN1SAX_XEROpenType **ASN1SAX_XEROpenType**

Functions

- int **asn1Sax_XEROpenType_startElement** (void *userData, const OSUTF8CHAR *localname, const OSUTF8CHAR *qname, const OSUTF8CHAR *const *atts)
- int **asn1Sax_XEROpenType_characters** (void *userData, const OSUTF8CHAR *chars, int length)
- int **asn1Sax_XEROpenType_endElement** (void *userData, const OSUTF8CHAR *localname, const OSUTF8CHAR *qname)
- void **asn1Sax_XEROpenType_init** (OSCTXT *pctxt, ASN1SAX_XEROpenType *pSaxHandler, ASN1OpenType *pvalue, OSINT16 level)
- void **asn1Sax_XEROpenType_free** (OSCTXT *pctxt, ASN1SAX_XEROpenType *pSaxHandler)

8.2.1 Detailed Description

8.3 asn1xer.h File Reference

```
#include "rtsrc/asn1type.h"
#include "rtxsrc/rtxMemBuf.h"
#include "rtxersrc/rtSAXDefs.h"
```

Classes

- struct XerElemInfo
- struct XmlNamedBitsDict

Defines

- #define **XERINDENT** 3
- #define **XERBYTECNT**(pctxt) (pctxt)->buffer.byteIndex
- #define **EXTERNXER**

Typedefs

- typedef struct XmlNamedBitsDict **XmlNamedBitsDict**

Enumerations

- enum **ASN1XERState** {
  XERINIT, **XERSTART**, **XERDATA**, **XEREND**,
  XERSTART0, **XEREND0** }

Functions

- int xerDecBMPStr (OSCTXT *pctxt, ASN1BMPString *outdata)
- int xerDecBase64Str (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSINT32 bufsize)
- int xerDecBigInt (OSCTXT *pctxt, char **ppvalue, int radix)
- int xerDecBitStr (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnbits, OSINT32 bufsize)
- int xerDecBitStrMemBuf (OSRTMEMBUF *pMemBuf, const XMLCHAR *inpdata, int length, OSBOOL skipWhitespaces)
- int xerDecBool (OSCTXT *pctxt, OSBOOL *pvalue)
- int xerDecCopyBitStr (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnbits, OSINT32 bufsize, int lastBitOffset)
- int xerDecCopyDynBitStr (OSCTXT *pctxt, ASN1DynBitStr *pvalue, int lastBitOffset)
- int xerDecCopyDynOctStr (OSCTXT *pctxt, ASN1DynOctStr *pvalue, int lastBitOffset)
- int xerDecCopyOctStr (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSINT32 bufsize, int lastBitOffset)
- int xerDecDynAscCharStr (OSCTXT *pctxt, const char **outdata)
- int xerDecDynBase64Str (OSCTXT *pctxt, ASN1DynOctStr *pvalue)
- int xerDecDynBitStr (OSCTXT *pctxt, ASN1DynBitStr *pvalue)
- int xerDecDynOctStr (OSCTXT *pctxt, ASN1DynOctStr *pvalue)
- int xerDecDynUTF8Str (OSCTXT *pctxt, ASN1UTF8String *outdata)

- int xerDecInt (OSCTXT *pctxt, OSINT32 *pvalue)
- int xerDecInt8 (OSCTXT *pctxt, OSINT8 *pvalue)
- int xerDecInt16 (OSCTXT *pctxt, OSINT16 *pvalue)
- int xerDecInt64 (OSCTXT *pctxt, OSINT64 *pvalue)
- int xerDecObjId (OSCTXT *pctxt, ASN1OBJID *pvalue)
- int xerDecObjId64 (OSCTXT *pctxt, ASN1OID64 *pvalue)
- int xerDecOctStr (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSINT32 bufsize)
- int xerDecOctStrMemBuf (OSRTMEMBUF *pMemBuf, const XMLCHAR *inpdata, int length, OSBOOL skipWhitespaces)
- int xerDecOpenType (OSCTXT *pctxt, ASN1OpenType *pvalue)
- int xerDecReal (OSCTXT *pctxt, OSREAL *pvalue)
- int xerDecReal10 (OSCTXT *pctxt, const OSUTF8CHAR **pvalue)
- int xerDecRelativeOID (OSCTXT *pctxt, ASN1OBJID *pvalue)
- int xerDecUInt (OSCTXT *pctxt, OSUINT32 *pvalue)
- int xerDecUInt8 (OSCTXT *pctxt, OSUINT8 *pvalue)
- int xerDecUInt16 (OSCTXT *pctxt, OSUINT16 *pvalue)
- int xerDecUInt64 (OSCTXT *pctxt, OSUINT64 *pvalue)
- int xerDecUnivStr (OSCTXT *pctxt, ASN1UniversalString *outdata)
- int xerSetDecBufPtr (OSCTXT *pCtxt, const OSOCTET *bufaddr, size_t bufsiz)
- int xerSetEncBufPtr (OSCTXT *pCtxt, OSOCTET *bufaddr, size_t bufsiz, OSBOOL canonical)
- int xerEncAscCharStr (OSCTXT *pctxt, const char *value, const char *elemName)
- int xerEncBase64Str (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *data, const char *elemName)
- int xerEncBigInt (OSCTXT *pctxt, const char *value, const char *elemName)
- int xerEncBitStr (OSCTXT *pctxt, OSUINT32 nbits, const OSOCTET *data, const char *elemName, ASN1StrType outputType)
- int **xerEncBoolValue** (OSCTXT *pctxt, OSBOOL value)
- int xerEncBool (OSCTXT *pctxt, OSBOOL value, const char *elemName)
- int xerEncEndDocument (OSCTXT *pctxt)
- int xerEncEndElement (OSCTXT *pctxt, const char *elemName)
- int xerEncIndent (OSCTXT *pctxt)
- int xerEncInt (OSCTXT *pctxt, OSINT32 value, const char *elemName)
- int xerEncInt64 (OSCTXT *pctxt, OSINT64 value, const char *elemName)
- int xerEncNewLine (OSCTXT *pctxt)
- int xerEncObjId (OSCTXT *pctxt, const ASN1OBJID *pvalue, const char *elemName)
- int xerEncObjId64 (OSCTXT *pctxt, const ASN1OID64 *pvalue, const char *elemName)
- int xerEncRelativeOID (OSCTXT *pctxt, const ASN1OBJID *pvalue, const char *elemName)
- int xerEncOctStr (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *data, const char *elemName)
- int xerEncReal (OSCTXT *pctxt, OSREAL value, const char *elemName)
- int xerEncReal10 (OSCTXT *pctxt, const OSUTF8CHAR *value, const char *elemName)
- int xerEncStartDocument (OSCTXT *pctxt)
- int xerEncStartElement (OSCTXT *pctxt, const char *elemName, const char *attributes)
- int xerEncEmptyElement (OSCTXT *pctxt, const char *elemName, const char *attributes)
- int xerEncNamedValue (OSCTXT *pctxt, const char *value, const char *elemName, const char *attributes)
- int xerEncUInt (OSCTXT *pctxt, OSUINT32 value, const char *elemName)
- int xerEncUInt64 (OSCTXT *pctxt, OSUINT64 value, const char *elemName)
- int xerEncBMPStr (OSCTXT *pctxt, const ASN1BMPString *value, const char *elemName)
- int xerEncUnivStr (OSCTXT *pctxt, const ASN1UniversalString *value, const char *elemName)
- int xerEncUniCharData (OSCTXT *pctxt, const OSUNICHAR *value, OSUINT32 nchars)
- int xerEncUniCharStr (OSCTXT *pctxt, OSUNICHAR *value, const char *elemName)
- int xerEncOpenType (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *data, const char *elemName)
- int xerEncNull (OSCTXT *pctxt, const char *elemName)

- int **xerEncXmlCharData** (OSCTXT *pctxt, const XMLCHAR *pvalue, int length)
- OSBOOL xerCmpText (const XMLCHAR *text1, const char *text2)
- int **xerCopyText** (OSCTXT *pctxt, const char *text)
- int xerTextLength (const XMLCHAR *text)
- const char * xerTextToCStr (OSCTXT *pctxt, const XMLCHAR *text)
- size_t xerGetMsgLen (OSCTXT *pctxt)
- OSOCTET * xerGetMsgPtr (OSCTXT *pctxt)
- int **xerGetElemIdx** (const XMLCHAR *elemName, XerElemInfo *pElemInfo, int numElems)
- int **xerGetSeqElemIdx** (const XMLCHAR *elemName, XerElemInfo *pElemInfo, int numElems, int startIndex)
- int **xerFinalizeMemBuf** (OSRTMEMBUF *pMemBuf)
- int xerGetLibVersion ()
- const char * xerGetLibInfo ()
- int xmlDecBitStr (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnbits, OSINT32 bufsize)
- int xmlDecBool (OSCTXT *pctxt, OSBOOL *pvalue)
- int xmlDecDynBitStr (OSCTXT *pctxt, ASN1DynBitStr *pvalue)
- int xmlDecDynNamedBitStr (OSCTXT *pctxt, ASN1DynBitStr *pvalue, const XmlNamedBitsDict *pBitDict)
- int xmlDecDynOctStr (OSCTXT *pctxt, ASN1DynOctStr *pvalue)
- int xmlDecGeneralizedTime (OSCTXT *pctxt, const char **outdata)
- int xmlDecNamedBitStr (OSRTMEMBUF *pMemBuf, OSOCTET *pData, int dataSize, OSUINT32 *pNumbits, const XmlNamedBitsDict *pBitDict, const XMLCHAR *chars, int length)
- int xmlDecOctStr (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSINT32 bufsize)
- int xmlDecReal (OSCTXT *pctxt, OSREAL *pvalue)
- int xmlDecUTCTime (OSCTXT *pctxt, const char **outdata)
- int xmlEncBitStr (OSCTXT *pctxt, XmlNamedBitsDict *namedbits, OSUINT32 noofnamedbits, OSUINT32 nbits, const OSOCTET *data, const char *elemName, ASN1StrType outputType)
- int **xmlEncBoolValue** (OSCTXT *pctxt, OSBOOL value)
- int xmlEncBool (OSCTXT *pctxt, OSBOOL value, const char *elemName)
- int xmlEncEnum (OSCTXT *pctxt, const char *value)
- int xmlEncGeneralizedTime (OSCTXT *pctxt, const char *value, const char *elemName)
- int xmlEncNamedValue (OSCTXT *pctxt, const char *value, const char *elemName, const char *attributes)
- int **xerEncOpenTypeExt** (OSCTXT *pctxt, OSRTDList *pElemList)
- int xmlEncReal (OSCTXT *pctxt, OSREAL value, const char *elemName)
- int xmlEncUTCTime (OSCTXT *pctxt, const char *value, const char *elemName)

8.3.1 Detailed Description

ASN.1 runtime constants, data structure definitions, and functions to support the XML Encoding Rules (XER) as defined in the ITU-T X.693 standard.

8.4 asn1XerCppTypes.h File Reference

```
#include "rtxmlsrc/rtSaxCppParser.h"
#include "rtsrc/asn1CppTypes.h"
#include "rtxersrc/asn1xer.h"
#include "rtxsrc/rtxMemBuf.h"
```

Classes

- class ASN1XERMessageBuffer
- class ASN1XEREncodeBuffer
- class ASN1XERShdMemHpEncBuf
- class ASN1XERSAXDecodeHandler
- struct ASN1XERSAXDecodeHandler::ErrorInfo
- class ASN1XERDecodeBuffer

8.4.1 Detailed Description

XER C++ type and class definitions.

8.5 asn1XerCTypes.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <setjmp.h>
#include "asn1xer.h"
#include "rtsrc/asn1type.h"
#include "rtxsrc/rtxMemBuf.h"
#include "rtxersrc/rtSAXDefs.h"
#include "rtxmlsrc/rtSaxCParser.h"
```

Defines

- #define **ASN1SAXCTRY**(pctxt, stat)
- #define **ASN1SAXCTHROW**(pctxt, stat)
- #define **ASN1SAXCCATCH** else
- #define **STRX**(pctxt, pWideStr) xerTextToCStr ((pctxt), (pWideStr))
- #define **LSTRX**(pctxt, pLStr) strcpy ((char∗)rtxMemAlloc (pctxt, strlen (pLStr) + 1), (pLStr))
- #define **XERCDIAGSTRM2**(pctxt, a) RTDIAGSTRM2(pctxt,a)
- #define **XERCDIAGSTRM3**(pctxt, a, b) RTDIAGSTRM3(pctxt,a,b)
- #define **XERCDIAGSTRM4**(pctxt, a, b, c) RTDIAGSTRM4(pctxt,a,b,c)
- #define **DECLARE_NON_COMPACT_VAR**(type, var) type var

Typedefs

- typedef void(∗ **ASN1XERStartElementHandler**)(void ∗userData, const XMLCHAR ∗name, const XMLCHAR ∗∗atts)
- typedef void(∗ **ASN1XEREndElementHandler**)(void ∗userData, const XMLCHAR ∗name)
- typedef void(∗ **ASN1XERCharacterHandler**)(void ∗userData, const XMLCHAR ∗s, int len)
- typedef OSSAXHandlerBase **ASN1SAXCDecodeHandlerBase**

8.5.1 Detailed Description

8.5.2 Define Documentation

8.5.2.1 #define ASN1SAXCTHROW(pctxt, stat)

Value:

```
do { LOG_RTERR ((pctxt), stat); \
/* longjmp((pctxt)->jmpMark, stat); */ } while (0)
```

8.6 ASN1XERDecodeStream.h File Reference

```
#include "rtsrc/asn1CppTypes.h"
#include "rtxsrc/OSRTInputStreamIF.h"
#include "rtxersrc/asn1XerCppTypes.h"
```

Classes

- class ASN1XERDecodeStream

8.6.1 Detailed Description

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

8.7 ASN1XEREncodeStream.h File Reference

```
#include "rtsrc/asn1CppTypes.h"
#include "rtxsrc/OSRTOutputStreamIF.h"
#include "rtxersrc/asn1XerCppTypes.h"
```

Classes

- class ASN1XEREncodeStream

8.7.1 Detailed Description

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

Index