# ASN1C

ASN.1 Compiler
Version 7.2
OER Runtime
Reference Manual

*Objective Systems, Inc. version 7.2 — January 2018*

**Copyright Notice**



**Author's Contact Information**

Comments, suggestions, and inquiries regarding ASN1C may be submitted via electronic mail to info@obj-sys.com.

# Contents

# Chapter 1

# Module Index

## 1.1 Modules

Here is a list of all modules:

# Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

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

# Chapter 3

# Class Index

## 3.1 Class List

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

# Chapter 4

# File Index

## 4.1 File List

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

# Chapter 5

# Module Documentation

## 5.1 OER C++ Runtime Classes.

### Modules

- OER Message Buffer Classes

### 5.1.1 Detailed Description

## 5.2 OER Message Buffer Classes

### Classes

- class ASN1OERMessageBuffer
- class ASN1OEREncodeBuffer
- class ASN1OERDecodeBuffer

### 5.2.1 Detailed Description

The ASN.1 C++ runtime classes are wrapper classes that provide an object-oriented interface to the ASN.1 C runtime library functions. These classes are derived from the common classes documented in the ASN1C C/C++ Common Runtime Functions manual and are specific the Octet Encoding Rules (OER). These classes manage the buffers for encoding and decoding ASN.1 OER messages.

## 5.3 OER Runtime Library Functions.

### Modules

- OER C Decode Functions.
- OER C Encode Functions.

### Functions

- int **oerEncIdent** (OSCTXT *pctxt, OSUINT64 ident)

#### 5.3.1 Detailed Description

The ASN.1 Octet Encoding Rules (OER) runtime library contains the low-level constants, types, and functions that are assembled by the compiler to encode/decode more complex structures. The OER low-level C encode/decode functions are identified by their prefixes: oerEnc for encode, oerDec for decode, and oerUtil for utility functions.

## 5.4 OER C Decode Functions.

### Macros

- #define oerDecInt8(pctxt, pvalue) rtxReadBytes(pctxt,pvalue,1)
- #define oerDecUInt8(pctxt, pvalue) rtxReadBytes(pctxt,pvalue,1)
- #define **oerDecUnrestInt** oerDecUnrestInt32
- #define **oerDecUnrestUInt** oerDecUnrestUInt32

### Functions

- int oerDecBitStr (OSCTXT *pctxt, OSOCTET *pvalue, size_t bufsiz, OSUINT32 *pnbits)
- int oerDecBitStrExt (OSCTXT *pctxt, OSOCTET *pvalue, size_t bufsiz, OSUINT32 *pnbits, OSOCTET **extdata)
- int oerDecDynBitStr (OSCTXT *pctxt, const OSOCTET **ppvalue, OSUINT32 *pnbits)
- int oerDecDynCharStr (OSCTXT *pctxt, char **ppvalue)
- int oerDecDynOctStr (OSCTXT *pctxt, OSOCTET **ppvalue, OSUINT32 *pnocts, size_t len)
- int oerDecDynOctStr64 (OSCTXT *pctxt, OSOCTET **ppvalue, OSSIZE *pnocts, size_t len)
- int oerDecInt16 (OSCTXT *pctxt, OSINT16 *pvalue)
- int oerDecInt32 (OSCTXT *pctxt, OSINT32 *pvalue)
- int oerDecInt64 (OSCTXT *pctxt, OSINT64 *pvalue)
- int oerDecUnrestInt64 (OSCTXT *pctxt, OSINT64 *pvalue)
- int oerDecLen (OSCTXT *pctxt, OSSIZE *plength)
- int oerDecLen32 (OSCTXT *pctxt, OSUINT32 *plength)
- int oerDecObjId (OSCTXT *pctxt, ASN1OBJID *pvalue)
- int oerDecSignedEnum (OSCTXT *pctxt, OSINT32 *pvalue)
- int oerDecTag (OSCTXT *pctxt, ASN1TAG *ptag)
- int oerDecUInt16 (OSCTXT *pctxt, OSUINT16 *pvalue)
- int oerDecUInt32 (OSCTXT *pctxt, OSUINT32 *pvalue)
- int oerDecUInt64 (OSCTXT *pctxt, OSUINT64 *pvalue)
- int oerDecUnrestSignedUInt64 (OSCTXT *pctxt, OSUINT64 *pvalue)
- int oerDecUnrestUInt64 (OSCTXT *pctxt, OSUINT64 *pvalue)
- int oerDecUnrestInt32 (OSCTXT *pctxt, OSINT32 *pvalue)
- int oerDecUnrestSignedUInt32 (OSCTXT *pctxt, OSUINT32 *pvalue)
- int oerDecUnrestUInt32 (OSCTXT *pctxt, OSUINT32 *pvalue)
- int oerDecBigInt (OSCTXT *pctxt, const char **ppvalue, int radix)
- int oerDecBigUInt (OSCTXT *pctxt, const char **ppvalue, int radix)
- int oerDecUnrestSize (OSCTXT *pctxt, OSSIZE *pvalue)
- int oerDecUnsignedEnum (OSCTXT *pctxt, OSUINT32 *pvalue)

### 5.4.1 Detailed Description

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

### 5.4.2 Macro Definition Documentation

#### 5.4.2.1 oerDecInt8

```
#define oerDecInt8(
              pctxt,
              pvalue ) rtxReadBytes(pctxt,pvalue,1)
```

This macro decodes an OER 8-bit signed integer at the current message buffer/stream location and advances the pointer to the next field.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to decoded 16-bit integer value. |

**Returns**

Completion status of operation:

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

#### 5.4.2.2 oerDecUInt8

```
#define oerDecUInt8(
              pctxt,
              pvalue ) rtxReadBytes(pctxt,pvalue,1)
```

This macro decodes an OER 8-bit unsigned integer at the current message buffer/stream location and advances the pointer to the next field.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to decoded 16-bit integer value. |

**Returns**

Completion status of operation:

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

### 5.4.3 Function Documentation

#### 5.4.3.1 oerDecBigInt()

```
int oerDecBigInt (
             OSCTXT * pctxt,
             const char ** ppvalue,
             int radix )
```

This funcion decodes an OER length determinant and unbound signed integer at the current message buffer/stream location and advances the pointer to the next field.

**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 memory might be allocated despite the negative return status. |
| *radix* | Radix to be used for decoded string. Valid values are 2, 8, 10, or 16. |

**Returns**

Completion status of operation:

- 0 (0) = success,
- ASN_E_NOTCANON successful but encoding was non-canonical (caller may treat this as success or failure, as appropriate)
- negative return value is error.

#### 5.4.3.2 oerDecBigUInt()

```
int oerDecBigUInt (
             OSCTXT * pctxt,
             const char ** ppvalue,
             int radix )
```

This funcion decodes an OER length determinant and unbound unsigned integer at the current message buffer/stream location and advances the pointer to the next field.

**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 memory might be allocated despite the negative return status. |
| *radix* | Radix to be used for decoded string. Valid values are 2, 8, 10, or 16. |

**Returns**

Completion status of operation:

- 0 (0) = success,
- ASN_E_NOTCANON successful but encoding was non-canonical (caller may treat this as success or failure, as appropriate)
- negative return value is error.

#### 5.4.3.3 oerDecBitStr()

```
int oerDecBitStr (
             OSCTXT * pctxt,
             OSOCTET * pvalue,
             size_t bufsiz,
             OSUINT32 * pnbits )
```

This function decodes an OER bit string. This assumes a variable length string. Fixed-sized strings (i.e SIZE(N)) are encoded with no length or unused bit descriptors. This is handled by the compiler.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to static byte array into which data is to be read. The buffer must be large enough to hold the decoded value. |
| *bufsiz* | Size of the static byte array into which data is to be read. |
| *pnbits* | Pointer to a variable to receive the decoded number of bits. |

**Returns**

Completion status of operation:

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

#### 5.4.3.4 oerDecBitStrExt()

```
int oerDecBitStrExt (
             OSCTXT * pctxt,
             OSOCTET * pvalue,
             size_t bufsiz,
             OSUINT32 * pnbits,
             OSOCTET ** extdata )
```

This function decodes an OER bit string. This assumes a variable length string. Fixed-sized strings (i.e SIZE(N)) are encoded with no length or unused bit descriptors. This is handled by the compiler. This method handles bit strings with an extdata member present.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to static byte array into which data is to be read. The buffer must be large enough to hold the decoded value. |
| *bufsiz* | Size of the static byte array into which data is to be read. |
| *pnbits* | Pointer to a variable to receive the decoded number of bits. |
| *extdata* | Pointer to byte array containing extension data. |

**Returns**

Completion status of operation:

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

#### 5.4.3.5 oerDecDynBitStr()

```
int oerDecDynBitStr (
             OSCTXT * pctxt,
             const OSOCTET ** ppvalue,
             OSUINT32 * pnbits )
```

This function decodes an OER bit string into a dynamic memory buffer. Memory for the buffer is allocated using the rtx↩ MemAlloc function and should be subsequently freed using the rtxMemFree or rtxMemFreePtr functions. This assumes a variable length string. Fixed-sized strings (i.e SIZE(N)) are encoded with no length or unused bit descriptors. This is handled by the compiler.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *ppvalue* | Pointer to receive pointer to allocated byte buffer. |
| *pnbits* | Pointer to a variable to receive the decoded number of bits. |

**Returns**

Completion status of operation:

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

#### 5.4.3.6 oerDecDynCharStr()

```
int oerDecDynCharStr (
             OSCTXT * pctxt,
             char ** ppvalue )
```

This function decodes an OER character string into a dynamic memory buffer. Memory for the buffer is allocated using the rtxMemAlloc function and should be subsequently freed using the rtxMemFree or rtxMemFreePtr functions.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *ppvalue* | Pointer to receive pointer to null-terminated char string. |

**Returns**

Completion status of operation:

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

#### 5.4.3.7 oerDecDynOctStr()

```
int oerDecDynOctStr (
             OSCTXT * pctxt,
             OSOCTET ** ppvalue,
             OSUINT32 * pnocts,
             size_t len )
```

This function decodes an OER octet string into a dynamic memory buffer. Memory for the buffer is allocated using the rtxMemAlloc function and should be subsequently freed using the rtxMemFree or rtxMemFreePtr functions.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *ppvalue* | Pointer to receive pointer to allocated byte buffer. |
| *pnocts* | Pointer to a variable to receive the decoded number of octets (bytes). |
| *len* | Length of string to decoded. If zero, it is assumed the string is variable length and the length is decoded within the funtion. |

**Returns**

Completion status of operation:

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

#### 5.4.3.8 oerDecDynOctStr64()

```
int oerDecDynOctStr64 (
             OSCTXT * pctxt,
             OSOCTET ** ppvalue,
             OSSIZE * pnocts,
             size_t len )
```

This function decodes an OER octet string into a dynamic memory buffer. Memory for the buffer is allocated using the rtxMemAlloc function and should be subsequently freed using the rtxMemFree or rtxMemFreePtr functions.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *ppvalue* | Pointer to receive pointer to allocated byte buffer. |
| *pnocts* | Pointer to a variable to receive the decoded number of octets (bytes). |
| *len* | Length of string to decoded. If zero, it is assumed the string is variable length and the length is decoded within the funtion. |

**Returns**

Completion status of operation:

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

#### 5.4.3.9 oerDecInt16()

```
int oerDecInt16 (
             OSCTXT * pctxt,
             OSINT16 * pvalue )
```

This function decodes an OER 16-bit signed integer at the current message buffer/stream location and advances the pointer to the next field.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to decoded 16-bit integer value. |

**Returns**

Completion status of operation:

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

#### 5.4.3.10 oerDecInt32()

```
int oerDecInt32 (
             OSCTXT * pctxt,
             OSINT32 * pvalue )
```

This function decodes an OER 32-bit signed integer at the current message buffer/stream location and advances the pointer to the next field.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to decoded 32-bit integer value. |

**Returns**

Completion status of operation:

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

#### 5.4.3.11 oerDecInt64()

```
int oerDecInt64 (
             OSCTXT * pctxt,
             OSINT64 * pvalue )
```

This function decodes an OER 64-bit signed integer at the current message buffer/stream location and advances the pointer to the next field.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to decoded 64-bit integer value. |

**Returns**

Completion status of operation:

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

#### 5.4.3.12 oerDecLen()

```
int oerDecLen (
            OSCTXT * pctxt,
            OSSIZE * plength )
```

This function is used to decode an OER length determinant value.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *plength* | Pointer to variable to receive decoded length. |

**Returns**

Completion status of operation:

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

#### 5.4.3.13 oerDecLen32()

```
int oerDecLen32 (
            OSCTXT * pctxt,
            OSUINT32 * plength )
```

This function is used to decode an OER length determinant value. In this case, the length is restricted to a 32-bit unsigned integer maximum value.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *plength* | Pointer to variable to receive decoded length. |

**Returns**

Completion status of operation:

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

#### 5.4.3.14 oerDecObjId()

```
int oerDecObjId (
             OSCTXT * pctxt,
             ASN1OBJID * pvalue )
```

This function is used to decode an OER object identifier value.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to variable to receive decoded value. |

**Returns**

Completion status of operation:

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

#### 5.4.3.15 oerDecSignedEnum()

```
int oerDecSignedEnum (
             OSCTXT * pctxt,
             OSINT32 * pvalue )
```

This function is used to decode a signed enumerated value.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to variable to receive decoded value. |

**Returns**

Completion status of operation:

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

#### 5.4.3.16 oerDecTag()

```
int oerDecTag (
             OSCTXT * pctxt,
             ASN1TAG * ptag )
```

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

Bit Fields:

- 31-30 Class (00 = UNIV, 01 = APPL, 10 = CTXT, 11 = PRIV)
- 29 Form (not used for OER)
- 28-0 ID code value

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *ptag* | Pointer to variable to receive decoded tag info. |

**Returns**

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

#### 5.4.3.17 oerDecUInt16()

```
int oerDecUInt16 (
             OSCTXT * pctxt,
             OSUINT16 * pvalue )
```

This function decodes an OER 16-bit unsigned integer at the current message buffer/stream location and advances the pointer to the next field.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to decoded 16-bit integer value. |

**Returns**

Completion status of operation:

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

#### 5.4.3.18 oerDecUInt32()

```
int oerDecUInt32 (
             OSCTXT * pctxt,
             OSUINT32 * pvalue )
```

This function decodes an OER 32-bit unsigned integer at the current message buffer/stream location and advances the pointer to the next field.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to decoded 32-bit integer value. |

**Returns**

Completion status of operation:

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

#### 5.4.3.19 oerDecUInt64()

```
int oerDecUInt64 (
             OSCTXT * pctxt,
             OSUINT64 * pvalue )
```

This function decodes an OER 64-bit unsigned integer at the current message buffer/stream location and advances the pointer to the next field.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to decoded 64-bit integer value. |

**Returns**

Completion status of operation:

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

#### 5.4.3.20 oerDecUnrestInt32()

```
int oerDecUnrestInt32 (
             OSCTXT * pctxt,
             OSINT32 * pvalue )
```

This function decodes an OER unrestricted signed integer at the current message buffer/stream location and advances the pointer to the next field. It is assumed that the value will fit in a 32-bit integer variable.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to decoded 32-bit integer value. |

**Returns**

Completion status of operation:

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

#### 5.4.3.21 oerDecUnrestInt64()

```
int oerDecUnrestInt64 (
             OSCTXT * pctxt,
             OSINT64 * pvalue )
```

This function decodes an OER unrestricted signed integer at the current message buffer/stream location and advances the pointer to the next field. It is assumed that the value will fit in a 64-bit signed integer variable.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to decoded 64-bit integer value. |

**Returns**

Completion status of operation:

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

#### 5.4.3.22 oerDecUnrestSignedUInt32()

```
int oerDecUnrestSignedUInt32 (
             OSCTXT * pctxt,
             OSUINT32 * pvalue )
```

This function decodes an OER unrestricted signed integer at the current message buffer/stream location and advances the pointer to the next field. The value must fit in an unsigned 32-bit integer variable.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to decoded 32-bit integer value. |

**Returns**

Completion status of operation:

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

#### 5.4.3.23 oerDecUnrestSignedUInt64()

```
int oerDecUnrestSignedUInt64 (
             OSCTXT * pctxt,
             OSUINT64 * pvalue )
```

This function decodes an OER unrestricted signed integer at the current message buffer/stream location and advances the pointer to the next field. The value must fit in a 64-bit unsigned integer variable.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to decoded 64-bit integer value. |

**Returns**

Completion status of operation:

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

#### 5.4.3.24 oerDecUnrestSize()

```
int oerDecUnrestSize (
             OSCTXT * pctxt,
             OSSIZE * pvalue )
```

This function decodes an OER unrestricted size-typed value at the current message buffer/stream location and advances the pointer to the next field.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to decoded size-typed value. |

**Returns**

Completion status of operation:

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

#### 5.4.3.25 oerDecUnrestUInt32()

```
int oerDecUnrestUInt32 (
             OSCTXT * pctxt,
             OSUINT32 * pvalue )
```

This function decodes an OER unrestricted unsigned integer at the current message buffer/stream location and advances the pointer to the next field. It is assumed that the value will fit in a 32-bit integer variable.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to decoded 32-bit integer value. |

**Returns**

Completion status of operation:

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

#### 5.4.3.26 oerDecUnrestUInt64()

```
int oerDecUnrestUInt64 (
             OSCTXT * pctxt,
             OSUINT64 * pvalue )
```

This function decodes an OER unrestricted unsigned integer at the current message buffer/stream location and advances the pointer to the next field. It is assumed that the value will fit in a 64-bit unsigned integer variable.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to decoded 64-bit integer value. |

**Returns**

Completion status of operation:

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

#### 5.4.3.27 oerDecUnsignedEnum()

```
int oerDecUnsignedEnum (
             OSCTXT * pctxt,
             OSUINT32 * pvalue )
```

This function is used to decode an unsigned enumerated value.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to variable to receive decoded value. |

**Returns**

Completion status of operation:

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

## 5.5 OER C Encode Functions.

### Functions

- int oerEncBigInt (OSCTXT *pctxt, const char *pvalue, int radix)
- int **oerEncBigIntValue** (OSCTXT *pctxt, struct OSBigInt *pvalue)
- int oerEncBitStr (OSCTXT *pctxt, const OSOCTET *pvalue, size_t numbits)
- int oerEncBitStrExt (OSCTXT *pctxt, const OSOCTET *pvalue, size_t numbits, const OSOCTET *extdata, size_t dataSize)
- int oerEncExtElem (OSCTXT *pctxt, OSRTBuffer *pbuffer)
- int oerEncInt (OSCTXT *pctxt, OSINT64 value, size_t size)
- int oerEncUnrestInt64 (OSCTXT *pctxt, OSINT64 value)
- int oerEncUnrestSignedUInt64 (OSCTXT *pctxt, OSUINT64 value)
- int oerEncLen (OSCTXT *pctxt, size_t length)
- int oerEncObjId (OSCTXT *pctxt, const ASN1OBJID *pvalue)
- int **oerEncObjId64** (OSCTXT *pctxt, const ASN1OID64 *pvalue)
- int oerEncOpenExt (OSCTXT *pctxt, OSRTDList *pElemList)
- int oerEncRelOID64 (OSCTXT *pctxt, const ASN1OID64 *pvalue)
- int oerEncReal (OSCTXT *pctxt, OSREAL value)
- int oerEncSignedEnum (OSCTXT *pctxt, OSINT32 value)
- int oerEncTag (OSCTXT *pctxt, ASN1TAG tag)
- int oerEncUInt (OSCTXT *pctxt, OSUINT64 value, size_t size)
- int oerEncUnrestUInt64 (OSCTXT *pctxt, OSUINT64 value)
- int oerEncUnrestInt32 (OSCTXT *pctxt, OSINT32 value)
- int oerEncUnrestSignedUInt32 (OSCTXT *pctxt, OSUINT32 value)
- int oerEncUnrestUInt32 (OSCTXT *pctxt, OSUINT32 value)
- int oerEncUnrestSize (OSCTXT *pctxt, OSSIZE value)
- int oerEncUnsignedEnum (OSCTXT *pctxt, OSUINT32 value)

### 5.5.1 Detailed Description

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

### 5.5.2 Function Documentation

#### 5.5.2.1 oerEncBigInt()

```
int oerEncBigInt (
             OSCTXT * pctxt,
             const char * pvalue,
             int radix )
```

This function encodes an OER length determinant and signed integer at the current message buffer/stream location. The encoding will be canonical.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to a character string representing the integer to be encoded, in the given radix. |
| *radix* | Radix of the given integer character string. Valid values are 2, 8, 10, or 16. |

**Returns**

Completion status of operation:

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

#### 5.5.2.2 oerEncBitStr()

```
int oerEncBitStr (
             OSCTXT * pctxt,
             const OSOCTET * pvalue,
             size_t numbits )
```

This function encodes an OER bit string. This assumes a variable length string. Fixed-sized strings (i.e SIZE(N)) are encoded with no length or unused bit descriptors. This is handled by the compiler.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to binary data to encode. |
| *numbits* | Number of bits in the bit string. |

**Returns**

Completion status of operation:

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

#### 5.5.2.3 oerEncBitStrExt()

```
int oerEncBitStrExt (
             OSCTXT * pctxt,
             const OSOCTET * pvalue,
             size_t numbits,
```

```
            const OSOCTET * extdata,
            size_t dataSize )
```

This function encodes an OER bit string. This assumes a variable length string. Fixed-sized strings (i.e SIZE(N)) are encoded with no length or unused bit descriptors. This is handled by the compiler. This method handles bit strings with an extdata member present.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to binary data to encode. |
| *numbits* | Number of bits in the bit string. |
| *extdata* | Pointer to byte array containing extension data. |
| *dataSize* | Size, in octets, of the root bit string data. |

**Returns**

Completion status of operation:

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

#### 5.5.2.4 oerEncExtElem()

```
int oerEncExtElem (
            OSCTXT * pctxt,
            OSRTBuffer * pbuffer )
```

This function encodes a known extesnion element in a SEQUENCE or similar construct.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pbuffer* | Pointer to run-time buffer containing encoded element. |

**Returns**

Completion status of operation:

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

#### 5.5.2.5 oerEncInt()

```
int oerEncInt (
             OSCTXT * pctxt,
             OSINT64 value,
             size_t size )
```

This function encodes an OER fixed-size integer (1, 2, 4, or 8 bytes) and writes the encoded value to the output buffer/stream.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *value* | Integer value to be encoded. |
| *size* | Size of the encoded field (1, 2, 4, or 8 bytes). |

**Returns**

Completion status of operation:

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

#### 5.5.2.6 oerEncLen()

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

This function is used to encode an OER length determinant value.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *length* | The length to encode. |

**Returns**

Completion status of operation:

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

#### 5.5.2.7 oerEncObjId()

```
int oerEncObjId (
             OSCTXT * pctxt,
             const ASN1OBJID * pvalue )
```

This function is used to encode an OER object identifier value.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to variable containing value to encode. |

**Returns**

Completion status of operation:

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

#### 5.5.2.8 oerEncOpenExt()

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

This function will encode an ASN.1 open type extension. An open type extension field is the data that potentially resides after the ... marker in a version-1 message. The open type structure contains a complete encoded bit set including option element bits or choice index, length, and data. Typically, this data is populated when a version-1 system decodes a version-2 message. The extension fields are retained and can then be re-encoded if a new message is to be sent out (for example, in a store and forward system).

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pElemList* | A pointer to the open type to be encoded. |

**Returns**

Completion status of operation:

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

#### 5.5.2.9 oerEncReal()

```
int oerEncReal (
             OSCTXT * pctxt,
             OSREAL value )
```

This function is used to encode a floating point value.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *value* | The variable containing the value to encode. |

**Returns**

Completion status of operation:

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

#### 5.5.2.10 oerEncRelOID64()

```
int oerEncRelOID64 (
             OSCTXT * pctxt,
             const ASN1OID64 * pvalue )
```

This function is used to encode an OER relative object identifier value. In this case, the arc values can be up to 64-bits in size.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *pvalue* | Pointer to variable to receive decoded value. |

**Returns**

Completion status of operation:

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

#### 5.5.2.11 oerEncSignedEnum()

```
int oerEncSignedEnum (
             OSCTXT * pctxt,
             OSINT32 value )
```

This function is used to encode a signed OER enumerated value.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *value* | The variable containing the value to encode. |

**Returns**

Completion status of operation:

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

#### 5.5.2.12 oerEncTag()

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

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

Bit Fields:

- 31-30 Class (00 = UNIV, 01 = APPL, 10 = CTXT, 11 = PRIV)
- 29 Form (not used for OER)
- 28-0 ID code value

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *tag* | Tag value to be encoded. |

**Returns**

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

#### 5.5.2.13 oerEncUInt()

```
int oerEncUInt (
             OSCTXT * pctxt,
             OSUINT64 value,
             size_t size )
```

This function encodes an OER unsigned fixed-size integer (1, 2, 4, or 8 bytes) and writes the encoded value to the output buffer/stream.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *value* | Integer value to be encoded. |
| *size* | Size of the encoded field (1, 2, 4, or 8 bytes). |

**Returns**

Completion status of operation:

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

#### 5.5.2.14 oerEncUnrestInt32()

```
int oerEncUnrestInt32 (
             OSCTXT * pctxt,
             OSINT32 value )
```

This function encodes an OER unrestricted signed integer value and writes the encoded value to the output buffer/stream.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *value* | Integer value to be encoded. |

**Returns**

Completion status of operation:

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

#### 5.5.2.15 oerEncUnrestInt64()

```
int oerEncUnrestInt64 (
            OSCTXT * pctxt,
            OSINT64 value )
```

This function encodes a 64-bit signed integer value as an OER unrestricted signed integer value and writes the encoded value to the output buffer/stream.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *value* | Integer value to be encoded. |

**Returns**

Completion status of operation:

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

#### 5.5.2.16 oerEncUnrestSignedUInt32()

```
int oerEncUnrestSignedUInt32 (
            OSCTXT * pctxt,
            OSUINT32 value )
```

This function encodes an OER unrestricted signed integer value and writes the encoded value to the output buffer/stream.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *value* | Integer value to be encoded. |

**Returns**

Completion status of operation:

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

#### 5.5.2.17 oerEncUnrestSignedUInt64()

```
int oerEncUnrestSignedUInt64 (
             OSCTXT * pctxt,
             OSUINT64 value )
```

This function encodes a 64-bit unsigned integer value as an OER unrestricted signed integer value and writes the encoded value to the output buffer/stream.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *value* | Integer value to be encoded. |

**Returns**

Completion status of operation:

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

#### 5.5.2.18 oerEncUnrestSize()

```
int oerEncUnrestSize (
             OSCTXT * pctxt,
             OSSIZE value )
```

This function encodes an OER unrestricted size-typed value and writes the encoded value to the output buffer/stream.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *value* | Integer value to be encoded. |

**Returns**

Completion status of operation:

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

#### 5.5.2.19 oerEncUnrestUInt32()

```
int oerEncUnrestUInt32 (
             OSCTXT * pctxt,
             OSUINT32 value )
```

This function encodes an OER unrestricted unsigned integer value and writes the encoded value to the output buffer/stream.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *value* | Integer value to be encoded. |

**Returns**

Completion status of operation:

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

#### 5.5.2.20 oerEncUnrestUInt64()

```
int oerEncUnrestUInt64 (
             OSCTXT * pctxt,
             OSUINT64 value )
```

This function encodes an unsigned 64-bit integer value as an OER unrestricted unsigned integer value and writes the encoded value to the output buffer/stream.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *value* | Integer value to be encoded. |

**Returns**

Completion status of operation:

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

#### 5.5.2.21 oerEncUnsignedEnum()

```
int oerEncUnsignedEnum (
             OSCTXT * pctxt,
             OSUINT32 value )
```

This function is used to encode an unsigned OER enumerated value.

**Parameters**

| | |
|---|---|
| *pctxt* | Pointer to context block structure. |
| *value* | The variable containing the value to encode. |

**Returns**

Completion status of operation:

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

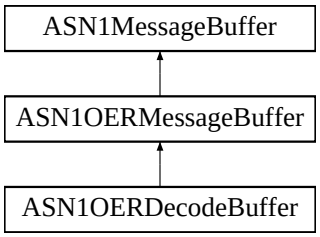
# Chapter 6

# Class Documentation

## 6.1 ASN1OERDecodeBuffer Class Reference

`#include <asn1OerCppTypes.h>`

Inheritance diagram for ASN1OERDecodeBuffer:




### Public Member Functions

- ASN1OERDecodeBuffer ()
- ASN1OERDecodeBuffer (const OSOCTET ∗pMsgBuf, size_t msgBufLen)
- ASN1OERDecodeBuffer (const OSOCTET ∗pMsgBuf, size_t msgBufLen, OSRTContext ∗pContext)
- EXTOERMETHOD ASN1OERDecodeBuffer (OSRTInputStream &istream)
- EXTOERMETHOD ASN1OERDecodeBuffer (const char ∗filePath)
- virtual OSBOOL isA (Type bufferType)
- EXTOERMETHOD int peekByte (OSOCTET &ub)
- EXTOERMETHOD int readBinaryFile (const char ∗filePath)
- EXTOERMETHOD int readBytes (OSOCTET ∗buffer, size_t bufsize, size_t nbytes)

### Additional Inherited Members

#### 6.1.1 Detailed Description

The ASN1OERDecodeBuffer class is derived from the ASN1OERMessageBuffer base class. It contains variables and methods specific to decoding ASN.1 OER messages. It is used to manage the input buffer containing the ASN.1 message to be decoded. This class has 3 overloaded constructors.

### 6.1.2 Constructor & Destructor Documentation

#### 6.1.2.1 ASN1OERDecodeBuffer() [1/5]

```
ASN1OERDecodeBuffer::ASN1OERDecodeBuffer ( )  [inline]
```

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

#### 6.1.2.2 ASN1OERDecodeBuffer() [2/5]

```
ASN1OERDecodeBuffer::ASN1OERDecodeBuffer (
            const OSOCTET * pMsgBuf,
            size_t msgBufLen )  [inline]
```

This constructor is used to describe the message to be decoded. Use getStatus() method to determine has error occured during the initialization or not.

**Parameters**

| | |
|---|---|
| *pMsgBuf* | A pointer to the message to be decoded. |
| *msgBufLen* | Length of the message buffer. |

#### 6.1.2.3 ASN1OERDecodeBuffer() [3/5]

```
ASN1OERDecodeBuffer::ASN1OERDecodeBuffer (
            const OSOCTET * pMsgBuf,
            size_t msgBufLen,
            OSRTContext * pContext )  [inline]
```

This constructor is used to describe the message to be decoded. Use getStatus() method to determine has error occured during the initialization or not.

**Parameters**

| | |
|---|---|
| *pMsgBuf* | A pointer to the message to be decoded. |
| *msgBufLen* | Length of the message buffer. |
| *pContext* | A pointer to an OSRTContext structure created by the user. |

#### 6.1.2.4 ASN1OERDecodeBuffer() [4/5]

```
EXTOERMETHOD ASN1OERDecodeBuffer::ASN1OERDecodeBuffer (
             OSRTInputStream & istream )
```

This version of the ASN1OERDecodeBuffer constructor takes a reference to an input stream object (stream decoding version).

**Parameters**

| | |
|---|---|
| *istream* | A reference to an input stream object. |

#### 6.1.2.5 ASN1OERDecodeBuffer() [5/5]

```
EXTOERMETHOD ASN1OERDecodeBuffer::ASN1OERDecodeBuffer (
             const char * filePath )
```

This constructor takes a pointer to the path of a file containing a binary OER message to be decoded.

**Parameters**

| | |
|---|---|
| *filePath* | Complete file path and name of file to read. |

### 6.1.3 Member Function Documentation

#### 6.1.3.1 isA()

```
virtual OSBOOL ASN1OERDecodeBuffer::isA (
             Type 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 OERDecode. |

**Returns**

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

#### 6.1.3.2 peekByte()

```
EXTOERMETHOD int ASN1OERDecodeBuffer::peekByte (
            OSOCTET & ub )
```

This method is used to peek at the next available byte in the decode buffer/stream without advancing the cursor.

**Parameters**

| | |
|---|---|
| *ub* | Single byte buffer to receive peeked byte. |

**Returns**

Completion status of operation:

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

#### 6.1.3.3 readBinaryFile()

```
EXTOERMETHOD int ASN1OERDecodeBuffer::readBinaryFile (
            const char * filePath )
```

This method reads the file into the buffer to decode.

**Parameters**

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

**Returns**

Completion status of operation:

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

#### 6.1.3.4 readBytes()

```
EXTOERMETHOD int ASN1OERDecodeBuffer::readBytes (
            OSOCTET * buffer,
```

```
size_t bufsize,
size_t nbytes )
```

This method is used to read the given number of bytes from the underlying buffer/stream into the given buffer.

**Parameters**

| | |
|---|---|
| *buffer* | Buffer into which data should be read. |
| *bufsize* | Size of the buffer |
| *nbytes* | Number of bytes to read. Must be <= bufsize. |

**Returns**

Completion status of operation:

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

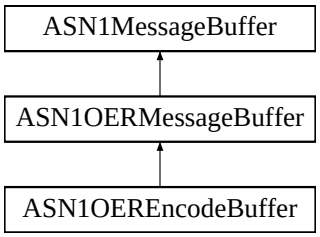
The documentation for this class was generated from the following file:

- asn1OerCppTypes.h

## 6.2 ASN1OEREncodeBuffer Class Reference

```
#include <asn1OerCppTypes.h>
```

Inheritance diagram for ASN1OEREncodeBuffer:




### Public Member Functions

- ASN1OEREncodeBuffer ()
- ASN1OEREncodeBuffer (OSOCTET *pMsgBuf, size_t msgBufLen)
- ASN1OEREncodeBuffer (OSOCTET *pMsgBuf, size_t msgBufLen, OSRTContext *pContext)
- EXTOERMETHOD ASN1OEREncodeBuffer (OSRTOutputStream &ostream)
- int encodeBit (OSBOOL value)
- int encodeBits (const OSOCTET *pvalue, size_t nbits, OSUINT32 bitOffset=0)
- virtual EXTOERMETHOD OSOCTET * getMsgCopy ()
- virtual EXTOERMETHOD const OSOCTET * getMsgPtr ()
- EXTOERMETHOD int init ()
- virtual OSBOOL isA (Type bufferType)
- EXTOERMETHOD int writeBytes (const OSOCTET *buffer, size_t nbytes)

### Additional Inherited Members

### 6.2.1 Detailed Description

The ASN1OEREncodeBuffer class is derived from the ASN1OERMessageBuffer base class. It contains variables and methods specific to encoding ASN.1 messages. It is used to manage the buffer into which an ASN.1 OER message is to be encoded.

### 6.2.2 Constructor & Destructor Documentation

#### 6.2.2.1 ASN1OEREncodeBuffer() [1/4]

```
ASN1OEREncodeBuffer::ASN1OEREncodeBuffer ( )  [inline]
```

This version of the ASN1OEREncodeBuffer constructor creates a dynamic memory buffer into which an OER message is encoded.

#### 6.2.2.2 ASN1OEREncodeBuffer() [2/4]

```
ASN1OEREncodeBuffer::ASN1OEREncodeBuffer (
            OSOCTET * pMsgBuf,
            size_t msgBufLen )  [inline]
```

This version of the ASN1OEREncodeBuffer constructor takes a message buffer and size argument (static encoding version).

**Parameters**

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

#### 6.2.2.3 ASN1OEREncodeBuffer() [3/4]

```
ASN1OEREncodeBuffer::ASN1OEREncodeBuffer (
            OSOCTET * pMsgBuf,
            size_t msgBufLen,
            OSRTContext * pContext )  [inline]
```

This version of the ASN1OEREncodeBuffer constructor takes a message buffer and size argument (static encoding version) as well as a pointer to an existing context object.

**Parameters**

| | |
|---|---|
| *pMsgBuf* | A pointer to a fixed-size message buffer to receive the encoded message. |
| *msgBufLen* | Size of the fixed-size message buffer. |
| *pContext* | A pointer to an OSRTContext structure created by the user. |

#### 6.2.2.4 ASN1OEREncodeBuffer() [4/4]

```
EXTOERMETHOD ASN1OEREncodeBuffer::ASN1OEREncodeBuffer (
             OSRTOutputStream & ostream )
```

This version of the ASN1OEREncodeBuffer constructor takes a reference to an output stream object (stream encoding version).

**Parameters**

| | |
|---|---|
| *ostream* | A reference to an output stream object. |

### 6.2.3 Member Function Documentation

#### 6.2.3.1 encodeBit()

```
int ASN1OEREncodeBuffer::encodeBit (
             OSBOOL value )  [inline]
```

This method writes a single encoded bit value to the output buffer or stream.

**Parameters**

| | |
|---|---|
| *value* | Boolean value of bit to be written. |

**Returns**

Status of operation: 0 = success, negative value if error occurred.

#### 6.2.3.2 encodeBits()

```
int ASN1OEREncodeBuffer::encodeBits (
             const OSOCTET * pvalue,
             size_t nbits,
             OSUINT32 bitOffset = 0 )  [inline]
```

This method writes the given number of bits from the byte array to the output buffer or stream starting from the given bit offset.

**Parameters**

| | |
|---|---|
| *pvalue* | Pointer to byte array containing data to be encoded. |
| *nbits* | Number of bits to copy from byte array to encode buffer. |
| *bitOffset* | Starting bit offset from which bits are to be copied. |

**Returns**

Status of operation: 0 = success, negative value if error occurred.

#### 6.2.3.3 getMsgCopy()

```
virtual EXTOERMETHOD OSOCTET* ASN1OEREncodeBuffer::getMsgCopy ( )  [virtual]
```

This method returns a copy of the current encoded message. Memory is allocated for the message using the 'new' operation. It is the user's responsibility to free the memory using 'delete'.

**Returns**

Pointer to copy of encoded message. It is the user's responsibility to release the memory using the 'delete' operator (i.e., delete [] ptr;)

#### 6.2.3.4 getMsgPtr()

```
virtual EXTOERMETHOD const OSOCTET* ASN1OEREncodeBuffer::getMsgPtr ( )  [virtual]
```

This method returns the internal pointer to the current encoded message.

**Returns**

Pointer to encoded message.

#### 6.2.3.5 init()

```
EXTOERMETHOD int ASN1OEREncodeBuffer::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.

#### 6.2.3.6 isA()

```
virtual OSBOOL ASN1OEREncodeBuffer::isA (
             Type 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 OEREncode. |

**Returns**

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

#### 6.2.3.7 writeBytes()

```
EXTOERMETHOD int ASN1OEREncodeBuffer::writeBytes (
             const OSOCTET * buffer,
             size_t nbytes )
```

This method is used to write the given number of bytes from the given buffer to the encode buffer or stream.

**Parameters**

| | |
|---|---|
| *buffer* | Buffer from which data should be read. |
| *nbytes* | Number of bytes to write. |

**Returns**

Completion status of operation:

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

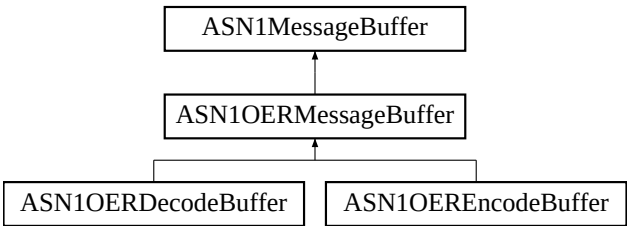
The documentation for this class was generated from the following file:

- asn1OerCppTypes.h

## 6.3 ASN1OERMessageBuffer Class Reference

`#include <asn1OerCppTypes.h>`

Inheritance diagram for ASN1OERMessageBuffer:




### Public Member Functions

- void hexDump ()
- virtual size_t getMsgLen ()
- EXTOERMETHOD int setBuffer (const OSOCTET *pMsgBuf, size_t msgBufLen)

### Protected Member Functions

- EXTOERMETHOD ASN1OERMessageBuffer (Type bufferType)
- EXTOERMETHOD ASN1OERMessageBuffer (OSRTStream &stream)
- EXTOERMETHOD ASN1OERMessageBuffer (Type bufferType, OSOCTET *pMsgBuf, size_t msgBufLen)
- EXTOERMETHOD ASN1OERMessageBuffer (Type bufferType, OSOCTET *pMsgBuf, size_t msgBufLen, OS↩ RTContext *pContext)

### 6.3.1 Detailed Description

The ASN1OERMessageBuffer class is derived from the ASN1MessageBuffer base class. It is the base class for the ASN1OEREncodeBuffer and ASN1OERDecodeBuffer derived classes. It contains variables and methods specific to encoding or decoding ASN.1 messages using the Octet Encoding Rules (OER). It is used to manage the buffer into which an ASN.1 message is to be encoded or decoded.

### 6.3.2 Constructor & Destructor Documentation

#### 6.3.2.1 ASN1OERMessageBuffer() [1/4]

```
EXTOERMETHOD ASN1OERMessageBuffer::ASN1OERMessageBuffer (
             Type bufferType )  [protected]
```

This constructor does not set a OER input source. It is used by the derived encode buffer classes. Use the getStatus() method to determine if an error has occured during initialization.

**Parameters**

| | |
|---|---|
| *bufferType* | Type of message buffer that is being created (for example, OEREncode or OERDecode). |

#### 6.3.2.2 ASN1OERMessageBuffer() [2/4]

```
EXTOERMETHOD ASN1OERMessageBuffer::ASN1OERMessageBuffer (
             OSRTStream & stream )  [protected]
```

This constructor associates a stream with a OER encode or decode buffer. It is used by the derived encode buffer classes to create a stream-based OER encoder or decoder.

**Parameters**

| | |
|---|---|
| *stream* | Stream class reference. |

#### 6.3.2.3 ASN1OERMessageBuffer() [3/4]

```
EXTOERMETHOD ASN1OERMessageBuffer::ASN1OERMessageBuffer (
             Type bufferType,
             OSOCTET * pMsgBuf,
             size_t msgBufLen )  [protected]
```

This constructor allows a memory buffer holding a binary OER message to be specified. Use the getStatus() method to determine if an error has occured during initialization.

**Parameters**

| | |
|---|---|
| *bufferType* | Type of message buffer that is being created (for example, OEREncode or OERDecode). |
| *pMsgBuf* | A pointer to a fixed size message buffer to recieve the encoded message. |
| *msgBufLen* | Size of the fixed-size message buffer. |

#### 6.3.2.4 ASN1OERMessageBuffer() [4/4]

```
EXTOERMETHOD ASN1OERMessageBuffer::ASN1OERMessageBuffer (
             Type bufferType,
             OSOCTET * pMsgBuf,
             size_t msgBufLen,
             OSRTContext * pContext )  [protected]
```

This constructor allows a memory buffer holding a binary OER message to be specified. It also allows a pre-existing context to be associated with this buffer. Use the getStatus() method to determine if an error has occured during initialization.

**Parameters**

| | |
|---|---|
| *bufferType* | Type of message buffer that is being created (for example, OEREncode or OERDecode). |
| *pMsgBuf* | A pointer to a fixed size message buffer to recieve the encoded message. |
| *msgBufLen* | Size of the fixed-size message buffer. |
| *pContext* | A pointer to an OSRTContext structure. |

### 6.3.3 Member Function Documentation

#### 6.3.3.1 getMsgLen()

```
virtual size_t ASN1OERMessageBuffer::getMsgLen ( )  [inline], [virtual]
```

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

**Parameters**

| | |
|---|---|
| - | none |

#### 6.3.3.2 hexDump()

```
void ASN1OERMessageBuffer::hexDump ( )  [inline]
```

This method outputs a hexadecimal dump of the current buffer contents to stdout.

**Parameters**

| | |
|---|---|
| - | none |

#### 6.3.3.3 setBuffer()

```
EXTOERMETHOD int ASN1OERMessageBuffer::setBuffer (
            const OSOCTET * pMsgBuf,
            size_t msgBufLen )
```

This method sets a buffer to receive the encoded message.

**Parameters**

| | |
|---|---|
| *pMsgBuf* | A pointer to a memory buffer to use to encode a message. The buffer should be declared as an array of unsigned characters (OSOCTETs). This parameter can be set to NULL to specify dynamic encoding (i.e., the encode functions will dynamically allocate a buffer for the message). |
| *msgBufLen* | The length of the memory buffer in bytes. If pMsgBuf is NULL, this parameter specifies the initial size of the dynamic buffer; if 0 - the default size will be used. |

**Returns**

Completion status of operation:

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

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

- asn1OerCppTypes.h

# Chapter 7

# File Documentation

## 7.1 asn1oer.h File Reference

```
#include "rtsrc/asn1type.h"
#include "rtsrc/asn1CharSet.h"
#include "rtxsrc/rtxBuffer.h"
```

### Macros

- #define **EXTOERMETHOD**
- #define **EXTERNOER**
- #define **EXTOERCLASS**
- #define oerDecInt8(pctxt, pvalue) rtxReadBytes(pctxt,pvalue,1)
- #define oerDecUInt8(pctxt, pvalue) rtxReadBytes(pctxt,pvalue,1)
- #define **oerDecUnrestInt** oerDecUnrestInt32
- #define **oerDecUnrestUInt** oerDecUnrestUInt32

### Functions

- int oerDecBitStr (OSCTXT *pctxt, OSOCTET *pvalue, size_t bufsiz, OSUINT32 *pnbits)
- int oerDecBitStrExt (OSCTXT *pctxt, OSOCTET *pvalue, size_t bufsiz, OSUINT32 *pnbits, OSOCTET **extdata)
- int oerDecDynBitStr (OSCTXT *pctxt, const OSOCTET **ppvalue, OSUINT32 *pnbits)
- int oerDecDynCharStr (OSCTXT *pctxt, char **ppvalue)
- int oerDecDynOctStr (OSCTXT *pctxt, OSOCTET **ppvalue, OSUINT32 *pnocts, size_t len)
- int oerDecDynOctStr64 (OSCTXT *pctxt, OSOCTET **ppvalue, OSSIZE *pnocts, size_t len)
- int oerDecInt16 (OSCTXT *pctxt, OSINT16 *pvalue)
- int oerDecInt32 (OSCTXT *pctxt, OSINT32 *pvalue)
- int oerDecInt64 (OSCTXT *pctxt, OSINT64 *pvalue)
- int oerDecUnrestInt64 (OSCTXT *pctxt, OSINT64 *pvalue)
- int oerDecLen (OSCTXT *pctxt, OSSIZE *plength)

- int oerDecLen32 (OSCTXT *pctxt, OSUINT32 *plength)
- int oerDecObjId (OSCTXT *pctxt, ASN1OBJID *pvalue)
- int oerDecSignedEnum (OSCTXT *pctxt, OSINT32 *pvalue)
- int oerDecTag (OSCTXT *pctxt, ASN1TAG *ptag)
- int oerDecUInt16 (OSCTXT *pctxt, OSUINT16 *pvalue)
- int oerDecUInt32 (OSCTXT *pctxt, OSUINT32 *pvalue)
- int oerDecUInt64 (OSCTXT *pctxt, OSUINT64 *pvalue)
- int oerDecUnrestSignedUInt64 (OSCTXT *pctxt, OSUINT64 *pvalue)
- int oerDecUnrestUInt64 (OSCTXT *pctxt, OSUINT64 *pvalue)
- int oerDecUnrestInt32 (OSCTXT *pctxt, OSINT32 *pvalue)
- int oerDecUnrestSignedUInt32 (OSCTXT *pctxt, OSUINT32 *pvalue)
- int oerDecUnrestUInt32 (OSCTXT *pctxt, OSUINT32 *pvalue)
- int oerDecBigInt (OSCTXT *pctxt, const char **ppvalue, int radix)
- int oerDecBigUInt (OSCTXT *pctxt, const char **ppvalue, int radix)
- int oerDecUnrestSize (OSCTXT *pctxt, OSSIZE *pvalue)
- int oerDecUnsignedEnum (OSCTXT *pctxt, OSUINT32 *pvalue)
- int oerEncBigInt (OSCTXT *pctxt, const char *pvalue, int radix)
- int **oerEncBigIntValue** (OSCTXT *pctxt, struct OSBigInt *pvalue)
- int oerEncBitStr (OSCTXT *pctxt, const OSOCTET *pvalue, size_t numbits)
- int oerEncBitStrExt (OSCTXT *pctxt, const OSOCTET *pvalue, size_t numbits, const OSOCTET *extdata, size_t dataSize)
- int oerEncExtElem (OSCTXT *pctxt, OSRTBuffer *pbuffer)
- int oerEncInt (OSCTXT *pctxt, OSINT64 value, size_t size)
- int oerEncUnrestInt64 (OSCTXT *pctxt, OSINT64 value)
- int oerEncUnrestSignedUInt64 (OSCTXT *pctxt, OSUINT64 value)
- int oerEncLen (OSCTXT *pctxt, size_t length)
- int oerEncObjId (OSCTXT *pctxt, const ASN1OBJID *pvalue)
- int **oerEncObjId64** (OSCTXT *pctxt, const ASN1OID64 *pvalue)
- int oerEncOpenExt (OSCTXT *pctxt, OSRTDList *pElemList)
- int oerEncRelOID64 (OSCTXT *pctxt, const ASN1OID64 *pvalue)
- int oerEncReal (OSCTXT *pctxt, OSREAL value)
- int oerEncSignedEnum (OSCTXT *pctxt, OSINT32 value)
- int oerEncTag (OSCTXT *pctxt, ASN1TAG tag)
- int oerEncUInt (OSCTXT *pctxt, OSUINT64 value, size_t size)
- int oerEncUnrestUInt64 (OSCTXT *pctxt, OSUINT64 value)
- int oerEncUnrestInt32 (OSCTXT *pctxt, OSINT32 value)
- int oerEncUnrestSignedUInt32 (OSCTXT *pctxt, OSUINT32 value)
- int oerEncUnrestUInt32 (OSCTXT *pctxt, OSUINT32 value)
- int oerEncUnrestSize (OSCTXT *pctxt, OSSIZE value)
- int oerEncUnsignedEnum (OSCTXT *pctxt, OSUINT32 value)
- int **oerEncIdent** (OSCTXT *pctxt, OSUINT64 ident)

### 7.1.1 Detailed Description

ASN.1 runtime constants, data structure definitions, and functions to support the Octet Encoding Rules (OER) as defined in the National Transportation Communications for ITS Protocol (NTCIP) 1102 standard.

## 7.2 asn1OerCppTypes.h File Reference

```
#include "rtoersrc/asn1oer.h"
#include "rtsrc/asn1CppTypes.h"
#include "rtxsrc/rtxBitEncode.h"
#include "rtxsrc/rtxHexDump.h"
```

### Classes

- class ASN1OERMessageBuffer
- class ASN1OEREncodeBuffer
- class ASN1OERDecodeBuffer

### 7.2.1 Detailed Description

OER C++ type and class definitions.

# Index