

ASN1C

ASN.1 Compiler
Version 6.0
C++ Runtime
Reference Manual

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

Copyright Notice

Copyright ©1997-2006 Objective Systems, Inc. All rights reserved.

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

Author's Contact Information

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

Contents

1	ASN1C C++ Common Runtime Classes Main Page	1
2	ASN1C C++ Common Runtime Classes Module Documentation	2
2.1	C++ Run-Time Classes	2
2.2	Control (ASN1C_) Base Classes	3
2.3	ASN.1 Type (ASN1T_) Base Classes	4
2.4	Context Management Classes	12
2.5	Date and Time Runtime Classes	13
2.6	Message Buffer Classes	14
2.7	ASN.1 Stream Classes	15
2.8	Generic Input Stream Classes	16
2.9	Generic Output Stream Classes	17
2.10	TCP/IP or UDP Socket Classes	18
2.11	Asn1NamedEventHandler	19
2.12	Run-time error status codes.	20
3	ASN1C C++ Common Runtime Classes Class Documentation	22
3.1	ASN1CBitStr Class Reference	22
3.2	ASN1CGeneralizedTime Class Reference	35
3.3	ASN1Context Class Reference	38
3.4	ASN1CSeqOfList Class Reference	39
3.5	ASN1CSeqOfListIterator Class Reference	48
3.6	ASN1CTime Class Reference	51
3.7	ASN1CType Class Reference	64
3.8	ASN1CUTCTime Class Reference	70
3.9	Asn1ErrorHandler Class Reference	73
3.10	ASN1MessageBuffer Class Reference	75
3.11	Asn1NamedEventHandler Class Reference	80
3.12	ASN1TBMPString Struct Reference	86

3.13	ASN1TDynBitStr Struct Reference	87
3.14	ASN1TDynOctStr Struct Reference	88
3.15	ASN1TGeneralizedTime Class Reference	91
3.16	Asn1TObject Struct Reference	94
3.17	ASN1TObjId Struct Reference	95
3.18	ASN1TObjId64 Struct Reference	99
3.19	ASN1TOpenType Struct Reference	100
3.20	ASN1TPDU Struct Reference	101
3.21	ASN1TPDUSeqOfList Struct Reference	102
3.22	ASN1TSeqExt Struct Reference	103
3.23	ASN1TSeqOfList Struct Reference	104
3.24	ASN1TTime Class Reference	105
3.25	ASN1TUniversalString Struct Reference	118
3.26	ASN1TUTCTime Class Reference	119
3.27	OSRTBaseType Class Reference	123
3.28	OSRTContext Class Reference	124
3.29	OSRTCtxtPtr Class Reference	129
3.30	OSRTFastString Class Reference	131
3.31	OSRTFileInputStream Class Reference	134
3.32	OSRTFileOutputStream Class Reference	136
3.33	OSRTInputStream Class Reference	138
3.34	OSRTMemoryInputStream Class Reference	144
3.35	OSRTMemoryOutputStream Class Reference	146
3.36	OSRTMessageBuffer Class Reference	148
3.37	OSRTMessageBufferIF Class Reference	153
3.38	OSRTOutputStream Class Reference	156
3.39	OSRTSocket Class Reference	160
3.40	OSRTSocketInputStream Class Reference	167
3.41	OSRTSocketOutputStream Class Reference	169
3.42	OSRTStream Class Reference	171
3.43	OSRTString Class Reference	175
3.44	OSRTStringIF Class Reference	178
3.45	OSRTUTF8String Class Reference	181
4	ASN1C C++ Common Runtime Classes File Documentation	184
4.1	ASN1CBitStr.h File Reference	184
4.2	ASN1CGeneralizedTime.h File Reference	185

4.3	ASN1Context.h File Reference	186
4.4	asn1CppEvtHndlr.h File Reference	187
4.5	asn1CppTypes.h File Reference	188
4.6	ASN1CSeqOfList.h File Reference	189
4.7	ASN1CTime.h File Reference	190
4.8	ASN1CUTCTime.h File Reference	191
4.9	asn1ErrCodes.h File Reference	192
4.10	ASN1TObjId.h File Reference	193
4.11	ASN1TOctStr.h File Reference	194
4.12	ASN1TTime.h File Reference	195
4.13	OSRTBaseType.h File Reference	196
4.14	OSRTContext.h File Reference	197
4.15	OSRTFastString.h File Reference	198
4.16	OSRTFileInputStream.h File Reference	199
4.17	OSRTFileOutputStream.h File Reference	200
4.18	OSRTInputStream.h File Reference	201
4.19	OSRTInputStreamIF.h File Reference	202
4.20	OSRTMemoryInputStream.h File Reference	203
4.21	OSRTMemoryOutputStream.h File Reference	204
4.22	OSRTMsgBuf.h File Reference	205
4.23	OSRTMsgBufIF.h File Reference	206
4.24	OSRTOutputStream.h File Reference	207
4.25	OSRTOutputStreamIF.h File Reference	208
4.26	OSRTSocket.h File Reference	209
4.27	OSRTSocketInputStream.h File Reference	210
4.28	OSRTSocketOutputStream.h File Reference	211
4.29	OSRTStream.h File Reference	212
4.30	OSRTStreamIF.h File Reference	213
4.31	OSRTString.h File Reference	214
4.32	OSRTStringIF.h File Reference	215
4.33	OSRTUTF8String.h File Reference	216

Chapter 1

ASN1C C++ Common Runtime Classes Main Page

C++ Common / XML Runtime Library Classes

The **OSRT C++ run-time classes** are wrapper classes that provide an object-oriented interface to the common C run-time library functions. The categories of classes provided are as follows:

- Context management classes manage the context structure (OSCTXT) used to keep track of the working variables required to encode or decode XML messages.
- Message buffer classes are used to manage message buffers for encoding or decoding XML messages.
- XSD type base classes are used as the base for compiler-generated C++ data structures.
- Stream classes are used to read and write messages to and from files, sockets, and memory buffers.

Chapter 2

ASN1C C++ Common Runtime Classes Module Documentation

2.1 C++ Run-Time Classes

Modules

- [Control \(ASN1C_\) Base Classes](#)
- [ASN.1 Type \(ASN1T_\) Base Classes](#)
- [Context Management Classes](#)
- [Generic Input Stream Classes](#)
- [Generic Output Stream Classes](#)
- [TCP/IP or UDP Socket Classes](#)
- [Asn1NamedEventHandler](#)
- [Message Buffer Classes](#)

2.2 Control (ASN1C_) Base Classes

2.2.1 Detailed Description

The ASN1C Control Base Classes are used as the base for compiler-generated ASN1C_ classes. These are wrapper classes that can be used to encode/decode PDU types and as helper classes for performing operations on complex data types.

Modules

- [Date and Time Runtime Classes](#)

Classes

- class [ASN1CType](#)
- class [ASN1CBitStr](#)
- class [ASN1CSeqOfListIterator](#)
- class [ASN1CSeqOfList](#)

2.3 ASN.1 Type (ASN1T_) Base Classes

2.3.1 Detailed Description

These classes are used as the base for compiler-generated ASN1T_ C++ data structures. These are enhanced versions of the C structures used for mapping ASN.1 types. The main difference is that constructors and operators have been added to the derived classes.

Modules

- [Date and Time Runtime Classes](#)

Classes

- struct [ASN1TDynBitStr](#)
- struct [ASN1TBMPString](#)
- struct [ASN1TUniversalString](#)
- struct [ASN1TOpenType](#)
- struct [Asn1TObject](#)
- struct [ASN1TObjId64](#)
- struct [ASN1TSeqExt](#)
- struct [ASN1TPDU](#)
- struct [ASN1TSeqOfList](#)
- struct [ASN1TPDUSeqOfList](#)
- struct [ASN1TObjId](#)
- struct [ASN1TDynOctStr](#)

Typedefs

- typedef [Asn1TObject](#) **ASN1TObject**

Functions

- int [operator==](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator==](#) (const [ASN1TObjId](#) &lhs, const [ASN1TObjId](#) &rhs)
- int [operator==](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator==](#) (const [ASN1TObjId](#) &lhs, const char *dotted_oid_string)
- int [operator!=](#) (const [ASN1TObjId](#) &lhs, const [ASN1TObjId](#) &rhs)
- int [operator!=](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator!=](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator!=](#) (const [ASN1TObjId](#) &lhs, const char *dotted_oid_string)
- int [operator<](#) (const [ASN1TObjId](#) &lhs, const [ASN1TObjId](#) &rhs)
- int [operator<](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator<](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator<](#) (const [ASN1TObjId](#) &lhs, const char *dotted_oid_string)
- int [operator<=](#) (const [ASN1TObjId](#) &lhs, const [ASN1TObjId](#) &rhs)
- int [operator<=](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator<=](#) (const [ASN1TObjId](#) &lhs, const char *dotted_oid_string)
- int [operator<=](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)

- int **operator>** (const [ASN1ObjId](#) &lhs, const [ASN1ObjId](#) &rhs)
- int **operator>** (const [ASN1ObjId](#) &lhs, const char *dotted_oid_string)
- int **operator>** (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int **operator>** (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int **operator>=** (const [ASN1ObjId](#) &lhs, const [ASN1ObjId](#) &rhs)
- int **operator>=** (const [ASN1ObjId](#) &lhs, const char *dotted_oid_string)
- int **operator>=** (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int **operator>=** (const ASN1OBJID &lhs, const char *dotted_oid_string)
- [ASN1ObjId](#) **operator+** (const [ASN1ObjId](#) &lhs, const [ASN1ObjId](#) &rhs)
- int **operator==** (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int **operator==** (const [ASN1DynOctStr](#) &lhs, const char *string)
- int **operator==** (const ASN1DynOctStr &lhs, const ASN1DynOctStr &rhs)
- int **operator==** (const ASN1DynOctStr &lhs, const char *string)
- int **operator!=** (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int **operator!=** (const [ASN1DynOctStr](#) &lhs, const char *string)
- int **operator!=** (const ASN1DynOctStr &lhs, const ASN1DynOctStr &rhs)
- int **operator!=** (const ASN1DynOctStr &lhs, const char *string)
- int **operator<** (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int **operator<** (const [ASN1DynOctStr](#) &lhs, const char *string)
- int **operator<** (const ASN1DynOctStr &lhs, const ASN1DynOctStr &rhs)
- int **operator<** (const ASN1DynOctStr &lhs, const char *string)
- int **operator<=** (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int **operator<=** (const [ASN1DynOctStr](#) &lhs, const char *string)
- int **operator<=** (const ASN1DynOctStr &lhs, const ASN1DynOctStr &rhs)
- int **operator<=** (const ASN1DynOctStr &lhs, const char *string)
- int **operator>** (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int **operator>** (const [ASN1DynOctStr](#) &lhs, const char *string)
- int **operator>** (const ASN1DynOctStr &lhs, const ASN1DynOctStr &rhs)
- int **operator>** (const ASN1DynOctStr &lhs, const char *string)
- int **operator>=** (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int **operator>=** (const [ASN1DynOctStr](#) &lhs, const char *string)
- int **operator>=** (const ASN1DynOctStr &lhs, const ASN1DynOctStr &rhs)
- int **operator>=** (const ASN1DynOctStr &lhs, const char *string)

2.3.2 Function Documentation

2.3.2.1 int **operator!=** (const [ASN1ObjId](#) &lhs, const char * *dotted_oid_string*)

Overloaded not equal operator. This comparison operator allows for comparison of not equality of C++ based object identifier structure and a dotted string.

Parameters:

lhs - C++ object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns:

- True if values are equal.

2.3.2.2 int operator!= (const ASN1OBJID & lhs, const char * dotted_oid_string)

Overloaded not equal operator. This comparison operator allows for comparison of not equality of C based object identifier structure and a dotted string.

Parameters:

lhs - C object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns:

- True if values are equal.

2.3.2.3 int operator!= (const ASN1OBJID & lhs, const ASN1OBJID & rhs)

Overloaded not equal operator. This comparison operator allows for comparison of not equality of C based object identifier structure and a dotted string.

Parameters:

lhs - C object identifier value.

rhs - C object identifier value

Returns:

- True if values are equal.

2.3.2.4 int operator!= (const ASN1ObjId & lhs, const ASN1ObjId & rhs)

Overloaded not equal operator. This comparison operator allows for comparison of not equality of C++ based object identifier structure and a dotted string.

Parameters:

lhs - C++ object identifier value.

rhs - C++ object identifier value

Returns:

- True if values are equal.

2.3.2.5 ASN1ObjId operator+ (const ASN1ObjId & lhs, const ASN1ObjId & rhs)

Overloaded append + operator. This operator allows two Object Identifier values to be concatenated.

Parameters:

lhs - C++ object identifier value.

rhs - C++ object identifier value.

2.3.2.6 `int operator< (const ASN1ObjId & lhs, const char * dotted_oid_string)`

Overloaded less than < operator. This comparison operator allows for comparison of less than of C++ based object identifier structure and a dotted string.

Parameters:

lhs - C++ object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns:

- True if values are equal.

2.3.2.7 `int operator< (const ASN1OBJID & lhs, const char * dotted_oid_string)`

Overloaded less than < operator. This comparison operator allows for comparison of less than of C based object identifier structure and a dotted string.

Parameters:

lhs - C object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns:

- True if values are equal.

2.3.2.8 `int operator< (const ASN1OBJID & lhs, const ASN1OBJID & rhs)`

Overloaded less than < operator. This comparison operator allows for comparison of less than of C based object identifier structure and a dotted string.

Parameters:

lhs - C object identifier value.

rhs - C object identifier value.

Returns:

- True if values are equal.

2.3.2.9 `int operator< (const ASN1ObjId & lhs, const ASN1ObjId & rhs)`

Overloaded less than < operator. This comparison operator allows for comparison of less than of C++ based object identifier structure and a dotted string.

Parameters:

lhs - C++ object identifier value.

rhs - C++ object identifier value.

Returns:

- True if values are equal.

2.3.2.10 int operator<= (const ASN1OBJID & lhs, const char * dotted_oid_string)

Overloaded less than <= operator. This comparison operator allows for comparison of less than or equal of a C based object identifier structure and a dotted string.

Parameters:

lhs - C object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns:

- True if values are equal.

2.3.2.11 int operator<= (const ASN1ObjId & lhs, const char * dotted_oid_string)

Overloaded less than <= operator. This comparison operator allows for comparison of less than of a C++ based object identifier structure and a dotted string.

Parameters:

lhs - C++ object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns:

- True if values are equal.

2.3.2.12 int operator<= (const ASN1OBJID & lhs, const ASN1OBJID & rhs)

Overloaded less than <= operator. This comparison operator allows for comparison of less than of C based object identifier structure and a dotted string.

Parameters:

lhs - C object identifier value.

rhs - C object identifier value

Returns:

- True if values are equal.

2.3.2.13 int operator<= (const ASN1ObjId & lhs, const ASN1ObjId & rhs)

Overloaded less than <= operator. This comparison operator allows for comparison of less than of C++ based object identifier structure and a dotted string.

Parameters:

lhs - C++ object identifier value.

rhs - C++ object identifier value

Returns:

- True if values are equal.

2.3.2.14 int operator==(const ASN1ObjId & lhs, const char * dotted_oid_string)

This comparison operator allows for comparison of equality of C++ based object identifier structure and a dotted string.

Parameters:

lhs - C++ object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns:

- True if values are equal.

2.3.2.15 int operator==(const ASN1OBJID & lhs, const char * dotted_oid_string)

This comparison operator allows for comparison of equality of a C-based object identifier structure and a dotted string.

Parameters:

lhs - C object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns:

- True if values are equal.

2.3.2.16 int operator==(const ASN1ObjId & lhs, const ASN1ObjId & rhs)

This comparison operator allows for comparison of equality of two C++-based object identifier structures.

Parameters:

lhs - C++ object identifier value.

rhs - C++ object identifier value.

Returns:

- True if values are equal.

2.3.2.17 int operator==(const ASN1OBJID & lhs, const ASN1OBJID & rhs)

This comparison operator allows for comparison of equality of two C-based object identifier structures.

Parameters:

lhs - C object identifier value.

rhs - C object identifier value.

Returns:

- True if values are equal.

2.3.2.18 int operator> (const ASN1OBJID & lhs, const char * dotted_oid_string)

Overloaded greater than > operator. This comparison operator allows for comparison of greater than of a C based object identifier structure and a dotted string.

Parameters:

lhs - C object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns:

- True if values are equal.

2.3.2.19 int operator> (const ASN1OBJID & lhs, const ASN1OBJID & rhs)

Overloaded greater than > operator. This comparison operator allows for comparison of greater than of C based object identifier structures.

Parameters:

lhs - C object identifier value.

rhs - C object identifier value.

Returns:

- True if values are equal.

2.3.2.20 int operator> (const ASN1ObjId & lhs, const char * dotted_oid_string)

Overloaded greater than > operator. This comparison operator allows for comparison of greater than of a C++ based object identifier structure and a dotted string.

Parameters:

lhs - C++ object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns:

- True if values are equal.

2.3.2.21 int operator> (const ASN1ObjId & lhs, const ASN1ObjId & rhs)

Overloaded greater than > operator. This comparison operator allows for comparison of greater than of C++ based object identifier structures

Parameters:

lhs - C++ object identifier value.

rhs - C++ object identifier value.

Returns:

- True if values are equal.

2.3.2.22 int operator>= (const ASN1OBJID & lhs, const char * dotted_oid_string)

Overloaded greater than equal >= operator. This comparison operator allows for comparison of greater than or equal of a C based object identifier structure and a dotted string.

Parameters:

lhs - C object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns:

- True if values are equal.

2.3.2.23 int operator>= (const ASN1OBJID & lhs, const ASN1OBJID & rhs)

Overloaded greater than equal >= operator. This comparison operator allows for comparison of greater than or equal of C based object identifier structures.

Parameters:

lhs - C object identifier value.

rhs - C object identifier value.

Returns:

- True if values are equal.

2.3.2.24 int operator>= (const ASN1ObjId & lhs, const char * dotted_oid_string)

Overloaded greater than equal >= operator. This comparison operator allows for comparison of greater than or equal of a C++ based object identifier structure and a dotted string.

Parameters:

lhs - C++ object identifier value.

dotted_oid_string - String containing OID value to compare.

Returns:

- True if values are equal.

2.3.2.25 int operator>= (const ASN1ObjId & lhs, const ASN1ObjId & rhs)

Overloaded greater than equal >= operator. This comparison operator allows for comparison of greater than or equal of C++ based object identifier structures.

Parameters:

lhs - C++ object identifier value.

rhs - C++ object identifier value.

Returns:

- True if values are equal.

2.4 Context Management Classes

2.4.1 Detailed Description

This group of classes manages an OSCTXT structure. This is the C structure use to keep track of all of the working variables required to encode or decode an ASN.1 message.

Classes

- class [ASN1Context](#)

2.5 Date and Time Runtime Classes

2.5.1 Detailed Description

The date and time classes contain methods for doing date/time calculations for the various ASN.1 time types including GeneralizedTime and UTCTime.

Classes

- class [ASN1TTime](#)
- class [ASN1TGeneralizedTime](#)
- class [ASN1TUTCTime](#)
- class [ASN1CTime](#)
- class [ASN1CGeneralizedTime](#)
- class [ASN1CUTCTime](#)

Defines

- #define **LOG_TMERR**(pctxt, stat) ((pctxt != 0) ? LOG_RTERR (pctxt, stat) : stat)

2.6 Message Buffer Classes

2.6.1 Detailed Description

These classes are used to manage message buffers. During encoding, messages are constructed within these buffers. During decoding, the messages to be decoded are held in these buffers.

Classes

- class [ASN1MessageBuffer](#)
- class [OSRTMessageBuffer](#)
- class [OSRTMessageBufferIF](#)

2.7 ASN.1 Stream Classes

2.7.1 Detailed Description

Classes that read or write ASN.1 messages to files, sockets, memory buffers, et c., are derived from this class.

Classes

- class [OSRTStream](#)
- class **OSRTStreamIF**

2.8 Generic Input Stream Classes

2.8.1 Detailed Description

The C++ interface class definitions for operations with input streams. Classes that implement this interface are used to input data from the various stream types, not to decode ASN.1 messages.

Classes

- class [OSRTInputStream](#)
- class **OSRTInputStreamIF**
- class **OSRTInputStreamPtr**

2.9 Generic Output Stream Classes

2.9.1 Detailed Description

The interface class definition for operations with output streams. Classes that implement this interface are used for writing data to the various stream types, not to encode ASN.1 messages.

Classes

- class [OSRTOutputStream](#)
- class **OSRTOutputStreamIF**
- class **OSRTOutputStreamPtr**

2.10 TCP/IP or UDP Socket Classes

2.10.1 Detailed Description

These classes provide utility methods for doing socket I/O.

Classes

- class [OSRSocket](#)

2.11 Asn1NamedEventHandler

2.11.1 Detailed Description

[Asn1NamedEventHandler](#) Classes include base classes from which user-defined error handler and event handler classes are derived.

Classes

- class [Asn1NamedEventHandler](#)
- class [Asn1ErrorHandler](#)

2.12 Run-time error status codes.

2.12.1 Detailed Description

This is a list of status codes that can be returned by the ASN1C run-time functions and generated code. In many cases, additional information and parameters for the different errors are stored in the context structure at the time the error is raised. This additional information can be output using the `rtxErrPrint` or `rtxErrLogUsingCB` run-time functions.

Defines

- `#define ASN_OK_FRAG 2`
- `#define ASN_E_BASE -100`
- `#define ASN_E_INVOBJID (ASN_E_BASE)`
- `#define ASN_E_INVLEN (ASN_E_BASE-1)`
- `#define ASN_E_BADTAG (ASN_E_BASE-2)`
- `#define ASN_E_INVBINS (ASN_E_BASE-3)`
- `#define ASN_E_INVINDEX (ASN_E_BASE-4)`
- `#define ASN_E_INVTCVAL (ASN_E_BASE-5)`
- `#define ASN_E_CONCMODF (ASN_E_BASE-6)`
- `#define ASN_E_ILLSTATE (ASN_E_BASE-7)`
- `#define ASN_E_NOTPDU (ASN_E_BASE-8)`
- `#define ASN_E_UNDEFTYP (ASN_E_BASE-9)`

2.12.2 Define Documentation

2.12.2.1 `#define ASN_E_BADTAG (ASN_E_BASE-2)`

Bad tag value. This error code is returned when a tag value is parsed with an identifier code that is too large to fit in a 32-bit integer variable.

2.12.2.2 `#define ASN_E_BASE -100`

Error base. ASN.1 specific errors start at this base number to distinguish them from common and other error types.

2.12.2.3 `#define ASN_E_CONCMODF (ASN_E_BASE-6)`

Concurrent list modification error. This error is returned from within a list iterator when it is detected that the list was modified outside the control of the iterator.

2.12.2.4 `#define ASN_E_ILLSTATE (ASN_E_BASE-7)`

Illegal state for operation. This error is returned in places where an operation is attempted but the object is not in a state that would allow the operation to be completed. One example is in a list iterator class when an attempt is made to remove a node but the node does not exist.

2.12.2.5 #define ASN_E_INVBINS (ASN_E_BASE-3)

Invalid binary string. This error code is returned when decoding XER data and a bit string value is received that contains something other than '1' or '0' characters.

2.12.2.6 #define ASN_E_INVINDEX (ASN_E_BASE-4)

Invalid table constraint index. This error code is returned when a value is provided to index into a table and the value does not match any of the defined indexes.

2.12.2.7 #define ASN_E_INVLEN (ASN_E_BASE-1)

Invalid length. This error code is returned when a length value is parsed that is not consistent with other lengths in a BER/DER message. This typically happens when an inner length within a constructed type is larger than the outer length value.

2.12.2.8 #define ASN_E_INVOBJID (ASN_E_BASE)

Invalid object identifier. This error code is returned when an object identifier is encountered that is not valid. Possible reasons for being invalid include invalid first and second arc identifiers (first must be 0, 1, or 2; second must be less than 40), not enough subidentifier values (must be 2 or more), or too many arc values (maximum number is 128).

2.12.2.9 #define ASN_E_INVTCVAL (ASN_E_BASE-5)

Invalid table constraint value. This error code is returned when a the value for an element in a table-constrained message instance does not match the value for the element defined in the table.

2.12.2.10 #define ASN_E_NOTPDU (ASN_E_BASE-8)

Encode/Decode method called for non-PDU. This error is returned when an Encode or Decode method is called on a non-PDU. Only PDUs have implementations of these methods.

2.12.2.11 #define ASN_E_UNDEFTYP (ASN_E_BASE-9)

Element type could not be resolved at run-time. This error is returned when the run-time parser modules is used (Asn1RTProd) to decode a type at run-time and the type of the element could not be resolved.

2.12.2.12 #define ASN_OK_FRAG 2

Fragment decode success status. This is returned when decoding is successful but only a fragment of the item was decoded. User should repeat the decode operation in order to fully decode message.

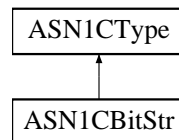
Chapter 3

ASN1C C++ Common Runtime Classes Class Documentation

3.1 ASN1CBitStr Class Reference

```
#include <ASN1CBitStr.h>
```

Inheritance diagram for ASN1CBitStr::



3.1.1 Detailed Description

ASN.1 bit string control class. The [ASN1CBitStr](#) class is derived from the [ASN1CType](#) base class. It is used as the base class for generated control classes for the ASN.1 BIT STRING type. This class provides utility methods for operating on the bit string referenced by the generated class. This class can also be used inline to operate on bits within generated BIT STRING elements in a SEQUENCE, SET, or CHOICE construct.

Public Member Functions

- [ASN1CBitStr](#) ([OSRTMessageBufferIF](#) &msgbuf, OSUINT32 nbits)
- [ASN1CBitStr](#) ([OSRTMessageBufferIF](#) &msgbuf, OSOCTET *bitStr, OSUINT32 &nbits, OSUINT32 maxNumbits_)
- [ASN1CBitStr](#) ([OSRTMessageBufferIF](#) &msgBuf, [ASN1TDynBitStr](#) &bitStr)
- [ASN1CBitStr](#) ([ASN1Context](#) &ctxt, OSUINT32 nbits)
- [ASN1CBitStr](#) ([ASN1Context](#) &ctxt, OSOCTET *bitStr, OSUINT32 &octsNumbits, OSUINT32 maxNumbits_)
- [ASN1CBitStr](#) ([ASN1Context](#) &ctxt, [ASN1TDynBitStr](#) &bitStr)
- [ASN1CBitStr](#) (const [ASN1CBitStr](#) &bitStr)
- [ASN1CBitStr](#) (const [ASN1CBitStr](#) &bitStr, OSBOOL extendable)
- int [set](#) (OSUINT32 bitIndex)

- int **set** (OSUINT32 fromIndex, OSUINT32 toIndex)
- int **change** (OSUINT32 bitIndex, OSBOOL value)
- int **clear** (OSUINT32 bitIndex)
- int **clear** (OSUINT32 fromIndex, OSUINT32 toIndex)
- void **clear** ()
- int **invert** (OSUINT32 bitIndex)
- int **invert** (OSUINT32 fromIndex, OSUINT32 toIndex)
- OSBOOL **get** (OSUINT32 bitIndex)
- OSBOOL **isSet** (OSUINT32 bitIndex)
- OSBOOL **isEmpty** ()
- int **size** () const
- OSUINT32 **length** () const
- int **cardinality** () const
- int **getBytes** (OSOCKET *pBuf, int bufSz)
- int **get** (OSUINT32 fromIndex, OSUINT32 toIndex, OSOCKET *pBuf, int bufSz)
- int **doAnd** (const OSOCKET *pOctstr, OSUINT32 octsNumbits)
- int **doAnd** (const **ASN1TDynBitStr** &bitStr)
- int **doAnd** (const **ASN1CBitStr** &bitStr)
- int **doOr** (const OSOCKET *pOctstr, OSUINT32 octsNumbits)
- int **doOr** (const **ASN1TDynBitStr** &bitStr)
- int **doOr** (const **ASN1CBitStr** &bitStr)
- int **doXor** (const OSOCKET *pOctstr, OSUINT32 octsNumbits)
- int **doXor** (const **ASN1TDynBitStr** &bitStr)
- int **doXor** (const **ASN1CBitStr** &bitStr)
- int **doAndNot** (const OSOCKET *pOctstr, OSUINT32 octsNumbits)
- int **doAndNot** (const **ASN1TDynBitStr** &bitStr)
- int **doAndNot** (const **ASN1CBitStr** &bitStr)
- int **shiftLeft** (OSUINT32 shift)
- int **shiftRight** (OSUINT32 shift)
- int **unusedBitsInLastUnit** ()
- operator **ASN1TDynBitStr** ()
- operator **ASN1TDynBitStr** * ()

Protected Member Functions

- **ASN1CBitStr** (**OSRTMessageBufferIF** &msgBuf)
- **ASN1CBitStr** (**ASN1Context** &ctxt)
- **ASN1CBitStr** (OSOCKET *pBits, OSUINT32 &numbits, OSUINT32 maxNumbits)
- **ASN1CBitStr** (**ASN1TDynBitStr** &bitStr)
- void **init** (OSOCKET *pBits, OSUINT32 &numbits, OSUINT32 maxNumbits)
- void **init** (**ASN1TDynBitStr** &bitStr)

Protected Attributes

- OSOCKET ** **mpUnits**
- OSUINT32 **mMaxNumBits**
- OSUINT32 * **mpNumBits**
- int **mUnitsUsed**
- int **mUnitsAllocated**
- OSBOOL **mDynAlloc**

3.1.2 Constructor & Destructor Documentation

3.1.2.1 ASN1CBitStr::ASN1CBitStr (OSRTMessageBufferIF & msgbuf, OSUINT32 nbits)

This constructor creates an empty bit string. If the `nbits` argument is zero, the bit string is set to be dynamic; otherwise, the capacity is set to `nbits`.

Parameters:

msgbuf - ASN.1 message buffer or stream object.

nbits - Number of bits this bit string can contain (zero if unbounded).

3.1.2.2 ASN1CBitStr::ASN1CBitStr (OSRTMessageBufferIF & msgbuf, OSOCTET * bitStr, OSUINT32 & numbits, OSUINT32 maxNumbits_)

This constructor creates a bit string from an array of bits. The constructor does not copy the bit string data, it just references the given data variables. All operations on the bit string cause the referenced items to be updated directly.

Parameters:

msgbuf - ASN.1 message buffer or stream object.

bitStr - Pointer to static byte array

numbits - Reference to length of bit string (in bits)

maxNumbits_ - sets maximum length in bits

3.1.3 Member Function Documentation

3.1.3.1 int ASN1CBitStr::set (OSUINT32 bitIndex)

This version of the set method sets the given bit in the target string.

Parameters:

bitIndex Relative index of the bit to set in the string. The bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from the left to right (MS to LS).

Returns:

Completion status of operation:

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

3.1.3.2 int ASN1CBitStr::set (OSUINT32 fromIndex, OSUINT32 toIndex)

This version of the set method sets the bits from the specified `fromIndex` (inclusive) to the specified `toIndex` (exclusive) to one.

Parameters:

fromIndex Relative start index (inclusive) of bits in the string. The bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from the left to right (MS to LS).

toIndex Relative end index (exclusive) of bits in the string. The bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from the left to right (MS to LS).

Returns:

Completion status of operation:

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

3.1.3.3 int ASN1CBitStr::change (OSUINT32 *bitIndex*, OSBOOL *value*) [inline]

Changes the bit at the specified index to the specified value.

Parameters:

bitIndex Relative index of bit to set in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).

value Boolean value to which the bit is to be set.

Returns:

Completion status of operation: 0 - if succeed

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

3.1.3.4 int ASN1CBitStr::clear (OSUINT32 *bitIndex*)

This version of the clear method sets the given bit in the target string to zero.

Parameters:

bitIndex Relative index of bit in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).

Returns:

Completion status of operation:

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

3.1.3.5 int ASN1CBitStr::clear (OSUINT32 *fromIndex*, OSUINT32 *toIndex*)

This version of the clear method sets the bits from the specified *fromIndex* (inclusive) to the specified *toIndex* (exclusive) to zero.

Parameters:

fromIndex Relative start index (inclusive) of bits in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).

toIndex Relative end index (exclusive) of bits in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).

Returns:

Completion status of operation:

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

3.1.3.6 void ASN1CBitStr::clear ()

This version of the clear method sets all bits in the bit string to zero.

Parameters:

- none

Returns:

Completion status of operation:

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

3.1.3.7 int ASN1CBitStr::invert (OSUINT32 bitIndex)

This version of the invert method inverts the given bit in the target string.

If the bit in the bit string is a zero, it will be set to 1; if the bit is a one, it will be set to 0.

Parameters:

bitIndex Relative index of bit in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).

Returns:

Completion status of operation:

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

3.1.3.8 int ASN1CBitStr::invert (OSUINT32 fromIndex, OSUINT32 toIndex)

This version inverts the bits from the specified fromIndex (inclusive) to the specified toIndex (exclusive).

If the bit in the bit string is a zero, it will be set to 1; if the bit is a one, it will be set to 0.

Parameters:

fromIndex Relative start index (inclusive) of bits in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).

toIndex Relative end index (exclusive) of bits in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).

Returns:

Completion status of operation:

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

3.1.3.9 OSBOOL ASN1CBitStr::get (OSUINT32 bitIndex)

This method returns the value of the bit with the specified index.

Parameters:

bitIndex Relative index of bit in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).

Returns:

Completion status of operation:

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

3.1.3.10 OSBOOL ASN1CBitStr::isSet (OSUINT32 bitIndex) [inline]

This method is the same as [ASN1CBitStr::get](#).

See also:

[get](#) (OSUINT32 bitIndex)

3.1.3.11 OSBOOL ASN1CBitStr::isEmpty () [inline]

This method returns TRUE if this bit string contains no bits that are set to 1.

Parameters:

- none

Returns:

TRUE, if the bit string contains no bits that are set to 1.

3.1.3.12 int ASN1CBitStr::size () const

This method returns the number of bytes of space actually in use by this bit string to represent bit values.

Parameters:

- none

Returns:

Number of bytes of space actually in use by this bit string to represent bit values.

3.1.3.13 OSUINT32 ASN1CBitStr::length () const

This method Calculates the "logical size" of the bith string.

The "logical size" is calculated by noting the index of the highest set bit in the bit string plus one. Zero will be returned if the bit string contains no set bits. The highest bit in the bit string is the LS bit in the last octet set to 1.

Parameters:

- none

Returns:

Returns the "logical size" of this bit string.

3.1.3.14 int ASN1CBitStr::cardinality () const

This method calculates the cardinality of the target bit string.

Cardinality of the bit string is the number of bits set to 1.

Parameters:

- none

Returns:

The number of bytes of space actually in use by this bit string to represent the bit values.

3.1.3.15 int ASN1CBitStr::getBytes (OSOCKET * pBuf, int bufSz)

This method copies the bit string to the given buffer.

Parameters:

pBuf Pointer to the destination buffer where bytes will be copied.

bufSz Size of the destination buffer. If the size of the buffer is not large enough to receive the entire bit string, a negative status value (RTERR_BUFOVFLOW) will be returned.

Returns:

Completion status of operation:

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

3.1.3.16 int ASN1CBitStr::get (OSUINT32 fromIndex, OSUINT32 toIndex, OSOCKET * pBuf, int bufSz)

This version of the get method copies the bit string composed of bits from this bit string from the specified fromIndex (inclusive) to the specified toIndex (exclusive) into the given buffer.

Parameters:

fromIndex Relative start index (inclusive) of bits in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).

toIndex Relative end index (exclusive) of bits in string. Bit index 0 refers to the MS bit (bit 8) in the first octet. The index values then progress from left to right (MS to LS bits).

pBuf Pointer to destination buffer where bytes will be copied.

bufSz Size of the destination buffer. If the size of the buffer is not large enough to receive the entire bit string, a negative status value (RTERR_BUFOVFLOW) will be returned.

Returns:

Completion status of operation:

- 0 (0) = success,
- RTERR_OUTOFBND index value is out of bounds
- RTERR_RANGERR fromIndex > toIndex
- other error codes (see asn1type.h).

3.1.3.17 int ASN1CBitStr::doAnd (const OSOCTET * pOctstr, OSUINT32 octsNumbits)

This method provides the logical AND of the target bit string with the argument bit string.

Parameters:

pOctstr A pointer to octets of another bit string for performing logical operation.

octsNumbits A number of bits in arguent bit string.

Returns:

Completion status of operation:

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

3.1.3.18 int ASN1CBitStr::doAnd (const ASN1TDynBitStr & bitStr) [inline]

This method performs a logical AND of the target bit string with the argument bit string.

Parameters:

bitStr A reference t another bit string represented by [ASN1TDynBitStr](#) type for performing logical operation.

Returns:

Completion status of operation:

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

3.1.3.19 int ASN1CBitStr::doAnd (const ASN1CBitStr & bitStr) [inline]

This method performs a logical AND of the target bit string with the argument bit string.

Parameters:

bitStr A reference to another bit string represented by [ASN1CBitStr](#) type for performing logical operation.

Returns:

Completion status of operation:

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

3.1.3.20 int ASN1CBitStr::doOr (const OSOCTET * pOctstr, OSUIN32 octsNumbits)

This method performs a logical OR of the target bit string with the argument bit string.

Parameters:

pOctstr A pointer to octets of another bit string for performing logical operation.

octsNumbits A number of bits in argument bit string.

Returns:

Completion status of operation:

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

3.1.3.21 int ASN1CBitStr::doOr (const ASN1TDynBitStr & bitStr) [inline]

This method performs a logical OR of the target bit string with the argument bit string.

Parameters:

bitStr A reference to another bit string represented by [ASN1TDynBitStr](#) type for performing logical operation.

Returns:

Completion status of operation:

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

3.1.3.22 int ASN1CBitStr::doOr (const ASN1CBitStr & bitStr) [inline]

This method performs a logical OR of the target bit string with the argument bit string.

Parameters:

bitStr A reference to another bit string represented by [ASN1CBitStr](#) type for performing logical operation.

Returns:

Completion status of operation:

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

3.1.3.23 int ASN1CBitStr::doXor (const OSOCTET * *pOctstr*, OSUINT32 *octsNumbits*)

This method provides the logical XOR of the target bit string with the argument bit string.

Parameters:

pOctstr A pointer to octets of another bit string for performing logical operation.

octsNumbits A number of bits in arguent bit string.

Returns:

Completion status of operation:

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

3.1.3.24 int ASN1CBitStr::doXor (const ASN1TDynBitStr & *bitStr*) [inline]

This method performs a logical XOR of the target bit string with the argument bit string.

Parameters:

bitStr A reference t another bit string represented by [ASN1TDynBitStr](#) type for performing logical operation.

Returns:

Completion status of operation:

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

3.1.3.25 int ASN1CBitStr::doXor (const ASN1CBitStr & *bitStr*) [inline]

This method performs a logical OR of the target bit string with the argument bit string.

Parameters:

bitStr A reference to another bit string represented by [ASN1CBitStr](#) type for performing logical operation.

Returns:

Completion status of operation:

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

3.1.3.26 `int ASN1CBitStr::doAndNot (const OSOCTET * pOctstr, OSUINT32 octsNumbits)`

This method performs a logical ANDNOT of the target bit string with the argument bit string.

Logical ANDNOT clears all of the bits in this bit string whose corresponding bit is set in the specified bit string.

Parameters:

pOctstr A pointer to octets of another bit string for performing logical operation.

octsNumbits A number of bits in argument bit string.

Returns:

Completion status of operation:

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

3.1.3.27 `int ASN1CBitStr::doAndNot (const ASN1TDynBitStr & bitStr) [inline]`

This method performs a logical ANDNOT of the target bit string with the argument bit string.

Logical ANDNOT clears all of the bits in this bit string whose corresponding bit is set in the specified bit string.

Parameters:

bitStr A reference to another bit string represented by `ASN1TDynBitStr` type for performing logical operation.

Returns:

Completion status of operation:

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

3.1.3.28 `int ASN1CBitStr::doAndNot (const ASN1CBitStr & bitStr) [inline]`

This method performs a logical ANDNOT of the target bit string with the argument bit string.

Logical ANDNOT clears all of the bits in this bit string whose corresponding bit is set in the specified bit string.

Parameters:

bitStr A reference to another bit string represented by `ASN1CBitStr` type for performing logical operation.

Returns:

Completion status of operation:

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

3.1.3.29 `int ASN1CBitStr::shiftLeft (OSUINT32 shift)`

This method shifts all bits to the left by the number of specified in the shift operand.

If the bit string can dynamically grow, then the length of the bit string will be decreased by shift bits. Otherwise, bits that are shifted into the bitstring are filled with zeros from the right. Most left bits are lost.

Parameters:

shift Number of bits to be shifted.

Returns:

Completion status of operation:

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

3.1.3.30 `int ASN1CBitStr::shiftRight (OSUINT32 shift)`

This method shifts all bits to the right by the number of specified in the shift operand.

If the bit string can dynamically grow, then the length of the bit string will be decreased by shift bits. Otherwise, bits that are shifted into the bitstring are filled with zeros from the left. Most right bits are lost.

Parameters:

shift Number of bits to be shifted.

Returns:

Completion status of operation:

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

3.1.3.31 `int ASN1CBitStr::unusedBitsInLastUnit ()`

This method returns the number of unused bits in the last octet.

Parameters:

- none

Returns:

Number of bits in the last octet. It equals to `length() % 8`.

3.1.3.32 `ASN1CBitStr::operator ASN1TDynBitStr ()`

This method returns a filled ANSDITDynBitStr variable.

Memory is not allocated when calling this method; only a pointer is assigned. Thus, the `ASN1TDynBitStr` variable is only valid while this `ASN1CBitStr` is in scope.

Parameters:

- none

Returns:

Filled [ASN1TDynBitStr](#).

3.1.3.33 ASN1CBitStr::operator ASN1TDynBitStr * ()

This method returns a pointer to the filled ANSDITDynBitStr variable.

Memory for the ASN1DynBitStr variable is allocated using memory memAlloc and bits are copied into it.

Parameters:

- none

Returns:

Pointer to a filled [ASN1TDynBitStr](#).

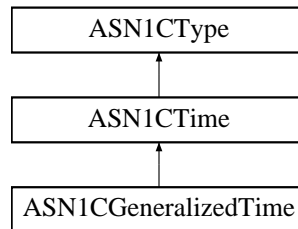
The documentation for this class was generated from the following file:

- [ASN1CBitStr.h](#)

3.2 ASN1CGeneralizedTime Class Reference

```
#include <ASN1CGeneralizedTime.h>
```

Inheritance diagram for ASN1CGeneralizedTime::



3.2.1 Detailed Description

ASN.1 GeneralizedTime control class. The [ASN1CGeneralizedTime](#) class is derived from the [ASN1CTime](#) base class. It is used as the base class for generated control classes for the ASN.1 Generalized Time ([UNIVERSAL 24] IMPLICIT VisibleString) type. This class provides utility methods for operating on the time information referenced by the generated class. This class can also be used inline to operate on the times within generated time string elements in a SEQUENCE, SET, or CHOICE construct. The time string generally is encoded according to ISO 8601 format with some exceptions (see X.680).

Public Member Functions

- [ASN1CGeneralizedTime](#) ([OSRTMessageBufferIF](#) &msgBuf, char *&buf, int bufSize, OSBOOL useDerRules=FALSE)
- [ASN1CGeneralizedTime](#) ([OSRTMessageBufferIF](#) &msgBuf, [ASN1GeneralizedTime](#) &buf, OSBOOL useDerRules=FALSE)
- [ASN1CGeneralizedTime](#) ([ASN1Context](#) &ctxt, char *&buf, int bufSize, OSBOOL useDerRules=FALSE)
- [ASN1CGeneralizedTime](#) ([ASN1Context](#) &ctxt, [ASN1GeneralizedTime](#) &buf, OSBOOL useDerRules=FALSE)
- [ASN1CGeneralizedTime](#) (const [ASN1CGeneralizedTime](#) &original)
- int [getCentury](#) ()
- int [setCentury](#) (int century)
- int [setTime](#) (time_t time, OSBOOL diffTime)
- const [ASN1CGeneralizedTime](#) & **operator=** (const [ASN1CGeneralizedTime](#) &tm)

Protected Member Functions

- virtual [ASN1TTime](#) & [getTimeObj](#) ()
- virtual const [ASN1TTime](#) & [getTimeObj](#) () const
- [ASN1CGeneralizedTime](#) (char *&buf, int bufSize, OSBOOL useDerRules=FALSE)
- [ASN1CGeneralizedTime](#) ([ASN1GeneralizedTime](#) &buf, OSBOOL useDerRules=FALSE)
- int [compileString](#) ()

Protected Attributes

- [ASN1TGeneralizedTime](#) [timeObj](#)

3.2.2 Constructor & Destructor Documentation

3.2.2.1 ASN1CGeneralizedTime::ASN1CGeneralizedTime (OSRTMessageBufferIF & msgBuf, char *& buf, int bufSize, OSBOOL useDerRules = FALSE)

This constructor creates a time string from a buffer. It does not deep-copy the data, it just assigns the passed array to an internal reference variable. The object will then directly operate on the given data variable.

Parameters:

msgBuf Reference to an OSRTMessage buffer derived object (for example, an ASN1BEREncodeBuffer).

buf A reference pointer to the time string buffer.

bufSize The size of the passed buffer, in bytes.

useDerRules An OSBOOL value.

3.2.2.2 ASN1CGeneralizedTime::ASN1CGeneralizedTime (OSRTMessageBufferIF & msgBuf, ASN1GeneralizedTime & buf, OSBOOL useDerRules = FALSE)

This constructor creates a time string using the ASN1GeneralizedTime argument. The constructor does not deep-copy the variable, it assigns a reference to it to an internal variable. The object will then directly operate on the given data variable. This form of the constructor is used with a compiler-generated time string variable.

Parameters:

msgBuf Reference to an OSRTMessage buffer derived object (for example, an ASN1BEREncodeBuffer).

buf A reference pointer to the time string buffer.

useDerRules An OSBOOL value.

3.2.3 Member Function Documentation

3.2.3.1 int ASN1CGeneralizedTime::getCentury ()

This method returns the centry part (first two digits) of the year component of the time value.

Parameters:

- none

Returns:

Century part (first two digits) of the year component is returned if the operation is successful. If the operation fails, one of the negative status codes is returned.

3.2.3.2 int ASN1CGeneralizedTime::setCentury (int century)

This method sets the centry part (first two digits) of the year component of the time value.

Parameters:

century Century part (first two digits) of the year component.

Returns:

Completion status of operation:

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

3.2.3.3 int ASN1CGeneralizedTime::setTime (time_t *time*, OSBOOL *diffTime*) [virtual]

This converts the value of the C built-in type `time_t` to a time string.

The value is the number of seconds from January 1, 1970. Note that the action of this method may differ for different inherited [ASN1CTime](#) Classes.

Parameters:

time The time value, expressed as a number of seconds from January 1, 1970.

diffTime TRUE means the difference between local time and UTC time will be calculated; in other case, only local time will be stored.

Returns:

Completion status of operation:

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

Implements [ASN1CTime](#).

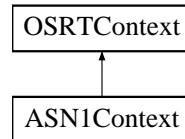
The documentation for this class was generated from the following file:

- [ASN1CGeneralizedTime.h](#)

3.3 ASN1Context Class Reference

```
#include <ASN1Context.h>
```

Inheritance diagram for ASN1Context::



3.3.1 Detailed Description

Reference counted ASN.1 context class. This keeps track of all encode/decode function variables between function invocations. It is reference counted to allow a message buffer and type class to share access to it.

Public Member Functions

- [ASN1Context \(\)](#)
- virtual int [setRunTimeKey](#) (const OSOCTET *key, size_t keylen)
- OSCTXT * [GetPtr](#) ()
- void [PrintErrorInfo](#) ()

3.3.2 Constructor & Destructor Documentation

3.3.2.1 ASN1Context::ASN1Context ()

The default constructor initializes the mCtxt member variable for ASN.1 encoding/decoding.

3.3.3 Member Function Documentation

3.3.3.1 virtual int ASN1Context::setRunTimeKey (const OSOCTET * key, size_t keylen) [virtual]

This method sets run-time key to the context. This method does nothing for unlimited redistribution libraries.

Parameters:

key - array of octets with the key

keylen - number of octets in key array.

Returns:

Completion status of operation:

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

Reimplemented from [OSRTContext](#).

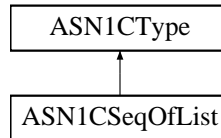
The documentation for this class was generated from the following file:

- [ASN1Context.h](#)

3.4 ASN1CSeqOfList Class Reference

```
#include <ASN1CSeqOfList.h>
```

Inheritance diagram for ASN1CSeqOfList::



3.4.1 Detailed Description

Doubly-linked list implementation. This class provides all functionality necessary for linked list operations. It is the base class for ASN1C compiler-generated ASN1C_ control classes for SEQUENCE OF and SET OF PDU types.

Public Member Functions

- [ASN1CSeqOfList](#) ([OSRTMessageBufferIF](#) &msgBuf, [OSRTDList](#) &lst, [OSBOOL](#) initBeforeUse=TRUE)
- [ASN1CSeqOfList](#) ([OSRTMessageBufferIF](#) &msgBuf)
- [ASN1CSeqOfList](#) ([ASN1CType](#) &ccobj)
- [ASN1CSeqOfList](#) ([OSRTMessageBufferIF](#) &msgBuf, [ASN1TSeqOfList](#) &lst)
- [ASN1CSeqOfList](#) ([ASN1CType](#) &ccobj, [ASN1TSeqOfList](#) &lst)
- [ASN1CSeqOfList](#) ([OSRTMessageBufferIF](#) &msgBuf, [ASN1TPDUSeqOfList](#) &lst)
- [ASN1CSeqOfList](#) ([OSRTContext](#) &ctxt, [OSRTDList](#) &lst, [OSBOOL](#) initBeforeUse=TRUE)
- [ASN1CSeqOfList](#) ([OSRTContext](#) &ctxt)
- [ASN1CSeqOfList](#) ([OSRTContext](#) &ctxt, [ASN1TSeqOfList](#) &lst)
- [ASN1CSeqOfList](#) ([OSRTContext](#) &ctxt, [ASN1TPDUSeqOfList](#) &lst)
- void [append](#) (void *data)
- void [appendArray](#) (const void *data, int numElems, int elemSize)
- void [appendArrayCopy](#) (const void *data, int numElems, int elemSize)
- void [init](#) ()
- void [insert](#) (int index, void *data)
- void [remove](#) (int index)
- void [remove](#) (void *data)
- void [removeFirst](#) ()
- void [removeLast](#) ()
- int [indexOf](#) (void *data) const
- [OSBOOL](#) [contains](#) (void *data) const
- void * [getFirst](#) ()
- void * [getLast](#) ()
- void * [get](#) (int index) const
- void * [set](#) (int index, void *data)
- void [clear](#) ()
- void [OSCRTLFREE](#) ()
- [OSBOOL](#) [isEmpty](#) () const
- int [size](#) () const
- [ASN1CSeqOfListIterator](#) * [iterator](#) ()
- [ASN1CSeqOfListIterator](#) * [iteratorFromLast](#) ()

- [ASN1CSeqOfListIterator](#) * [iteratorFrom](#) (void *data)
- void * [toArray](#) (int elemSize)
- void * [toArray](#) (void *pArray, int elemSize, int allocatedElems)
- void * [operator\[\]](#) (int index) const
- **operator OSRTDList** * ()

Protected Member Functions

- **ASN1CSeqOfList** (OSRTDList &lst)
- **ASN1CSeqOfList** ([ASN1TSeqOfList](#) &lst)
- **ASN1CSeqOfList** ([ASN1TPDUSeqOfList](#) &lst)
- void **remove** (OSRTDListNode *node)
- void **insertBefore** (void *data, OSRTDListNode *node)
- void **insertAfter** (void *data, OSRTDListNode *node)

Protected Attributes

- OSRTDList * **pList**
- volatile int **modCount**
- OSBOOL **wasAssigned**

3.4.2 Constructor & Destructor Documentation

3.4.2.1 ASN1CSeqOfList::ASN1CSeqOfList ([OSRTMessageBufferIF](#) & *msgBuf*, OSRTDList & *lst*, OSBOOL *initBeforeUse* = TRUE)

This constructor creates a linked list using the OSRTDList argument. The constructor does not deep-copy the variable; it assigns a reference to it to an external variable.

The object will then directly operate on the given list variable.

Parameters:

msgBuf Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).

lst Reference to a linked list structure.

initBeforeUse Set to TRUE if the passed linked list needs to be initialized (rtxDListInit to be called).

3.4.2.2 ASN1CSeqOfList::ASN1CSeqOfList ([OSRTMessageBufferIF](#) & *msgBuf*)

This constructor creates an empty linked list.

Parameters:

msgBuf Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).

3.4.2.3 ASN1CSeqOfList::ASN1CSeqOfList (ASN1CType & ccobj)

This constructor creates an empty linked list.

Parameters:

ccobj Reference to a control class object (for example, any generated ASN1C_ class object).

3.4.2.4 ASN1CSeqOfList::ASN1CSeqOfList (OSRTMessageBufferIF & msgBuf, ASN1TSeqOfList & lst)

This constructor creates a linked list using the [ASN1TSeqOfList](#) (holder of OSRTDList) argument.

The construction does not deep-copy the variable, it assigns a reference to it to an internal variable. The object will then directly operate on the given list variable. This constructor is used with a compiler-generated linked list variable.

Parameters:

msgBuf Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).

lst Reference to a linked list holder.

3.4.2.5 ASN1CSeqOfList::ASN1CSeqOfList (ASN1CType & ccobj, ASN1TSeqOfList & lst)

This constructor creates a linked list using the [ASN1TSeqOfList](#) (holder of OSRTDList) argument.

The construction does not deep-copy the variable, it assigns a reference to it to an internal variable. The object will then directly operate on the given list variable. This constructor is used with a compiler-generated linked list variable.

Parameters:

ccobj Reference to a control class object (for example, any generated ASN1C_ class object).

lst Reference to a linked list holder.

3.4.2.6 ASN1CSeqOfList::ASN1CSeqOfList (OSRTMessageBufferIF & msgBuf, ASN1TPDUSeqOfList & lst)

This constructor creates a linked list using the [ASN1TPDUSeqOfList](#) argument.

The construction does not deep-copy the variable, it assigns a reference to it to an internal variable. The object will then directly operate on the given list variable. This constructor is used with a compiler-generated linked list variable.

Parameters:

msgBuf Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).

lst Reference to a linked list holder.

3.4.3 Member Function Documentation

3.4.3.1 void ASN1CSeqOfList::append (void * data)

This method appends an item to the linked list.

This item is represented by a void pointer that can point to an object of any type. The `rtxMemAlloc` function is used to allocate memory for the list node structure, therefore, all internal list memory will be released whenever `rtxMemFree` is called.

Parameters:

data Pointer to a data item to be appended to the list.

Returns:

- none

3.4.3.2 void ASN1CSeqOfList::appendArray (const void * data, int numElems, int elemSize)

This method appends array items' pointers to a doubly linked list.

The rtxMemAlloc function is used to allocate memory for the list node structure, therefore all internal list memory will be released whenever the rtxMemFree is called. The data is not copied; it is just assigned to the node.

Parameters:

data Pointer to source array to be appended to the list.

numElems The number of elements in the source array.

elemSize The size of one element in the array. Use the *sizeof()* operator to pass this parameter.

Returns:

- none

3.4.3.3 void ASN1CSeqOfList::appendArrayCopy (const void * data, int numElems, int elemSize)

This method appends array items into a doubly linked list.

The rtxMemAlloc function is used to allocate memory for the list node structure; therefore all internal list memory will be released whenever rtxMemFree is called. The data will be copied; the memory will be allocated using rtxMemAlloc.

Parameters:

data Pointer to source array to be appended to the list.

numElems The number of elements in the source array.

elemSize The size of one element in the array. Use the *sizeof()* operator to pass this parameter.

Returns:

- none

3.4.3.4 void ASN1CSeqOfList::init () [inline]

This method initializes the linked list structure.

3.4.3.5 void ASN1CSeqOfList::insert (int index, void * data)

This method inserts an item into the linked list structure.

The item is represented by a void pointer that can point to an object of any type. The rtxMemAlloc function is used to allocate memory for the list node structure. All internal list memory will be released when the rtxMemFree function is called.

Parameters:

index Index at which the specified item is to be inserted.

data Pointer to data item to be appended to the list.

Returns:

- none

3.4.3.6 void ASN1CSeqOfList::remove (int *index*)

This method removed a node at the specified index from the linked list structure.

The rtxMemAlloc function was used to allocate the memory for the list node structure, therefore, all internal list memory will be released whenever the rtxMemFree is called.

Parameters:

index Index of the item to be removed.

Returns:

- none

3.4.3.7 void ASN1CSeqOfList::remove (void * *data*)

This method removes the first occurrence of the node with specified data from the linked list structure.

The rtxMemAlloc function was used to allocate the memory for the list node structure, therefore, all internal list memory will be released whenever the rtxMemFree function is called.

Parameters:

data - Pointer to the data item to be appended to the list.

3.4.3.8 void ASN1CSeqOfList::removeFirst () [inline]

This method removes the first node (head) from the linked list structure.

Parameters:

- none

Returns:

- none

3.4.3.9 void ASN1CSeqOfList::removeLast () [inline]

This method removes the last node (tail) from the linked list structure.

Parameters:

- none

Returns:

- none

3.4.3.10 int ASN1CSeqOfList::indexOf (void * *data*) const

This method returns the index in this list of the first occurrence of the specified item, or -1 if the list does not contain the time.

Parameters:

data - Pointer to data item to searched.

Returns:

The index in this list of the first occurrence of the specified item, or -1 if the list does not contain the item.

3.4.3.11 OSBOOL ASN1CSeqOfList::contains (void * *data*) const [inline]

This method returns TRUE if this list contains the specified pointer. Note that a match is not done on the *contents* of each data item (i.e. what is pointed at by the pointer), only the pointer values.

Parameters:

data - Pointer to data item.

Returns:

TRUE if this pointer value found in the list.

3.4.3.12 void* ASN1CSeqOfList::getFirst ()

This method returns the first item from the list or null if there are no elements in the list.

Returns:

The first item of the list.

3.4.3.13 void* ASN1CSeqOfList::getLast ()

This method returns the last item from the list or null if there are no elements in the list.

Returns:

The last item in the list.

3.4.3.14 void* ASN1CSeqOfList::get (int *index*) const

This method returns the item at the specified position in the list.

Parameters:

index Index of the item to be returned.

Returns:

The item at the specified index in the list.

3.4.3.15 void* ASN1CSeqOfList::set (int *index*, void * *data*)

This method replaces the item at the specified index in this list with the specified item.

Parameters:

index The index of the item to be replaced.

data The item to be stored at the specified index.

Returns:

The item previously at the specified position.

3.4.3.16 void ASN1CSeqOfList::clear ()

This method removes all items from the list.

3.4.3.17 void ASN1CSeqOfList::OSCRTLFREE ()

This method removes all items from the list and frees the associated memory.

3.4.3.18 OSBOOL ASN1CSeqOfList::isEmpty () const

This method returns TRUE if the list is empty.

Returns:

TRUE if this list is empty.

3.4.3.19 int ASN1CSeqOfList::size () const

This method returns the number of nodes in the list.

Returns:

The number of items in this list.

3.4.3.20 [ASN1CSeqOfListIterator*](#) ASN1CSeqOfList::iterator ()

This method returns an iterator over the elements in the linked list in the sequence from the first to the last.

Returns:

The iterator over this linked list.

3.4.3.21 [ASN1CSeqOfListIterator*](#) ASN1CSeqOfList::iteratorFromLast ()

This method creates a reverse iterator over the elements in this linked list in the sequence from last to first.

Parameters:

- none

Returns:

The reverse iterator over this linked list.

3.4.3.22 [ASN1CSeqOfListIterator*](#) ASN1CSeqOfList::iteratorFrom (void * *data*)

This method runs an iterator over the elements in this linked list starting from the specified item in the list.

Parameters:

data The item of the list to be iterated first.

Returns:

The iterator over this linked list.

3.4.3.23 void* ASN1CSeqOfList::toArray (int *elemSize*)

This method converts the linked list into a new array.

The `rtxMemAlloc` function is used to allocate memory for an array.

Parameters:

elemSize The size of one element in the array. Use the `sizeof()` operator to pass this parameter.

Returns:

The point to converted array.

3.4.3.24 void* ASN1CSeqOfList::toArray (void * *pArray*, int *elemSize*, int *allocatedElems*)

This method converts the linked list into an array.

The `rtxMemAlloc` function is used to allocate memory for the array if the capacity of the specified array is exceeded.

Parameters:

pArray Pointer to destination array.

elemSize The size of one element in the array. Use the sizeof() operator to pass this parameter.

allocatedElems The number of elements already allocated in the array. If this number is less than the number of nodes in the list, then a new array is allocated and returned. Memory is allocated using rtxMemAlloc function.

Returns:

The pointer to the converted array.

3.4.3.25]

```
void* ASN1CSeqOfList::operator[] (int index) const [inline]
```

This method is the overloaded operator[].

It returns the item at the specified position in the list.

See also:

[get](#) (int index)

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

- [ASN1CSeqOfList.h](#)

3.5 ASN1CSeqOfListIterator Class Reference

```
#include <ASN1CSeqOfList.h>
```

3.5.1 Detailed Description

Linked list iterator class. The [ASN1CSeqOfListIterator](#) class is an iterator for linked lists (represented by [ASN1CSeqOfList](#)) that allows the programmer to traverse the list in either direction and modify the list during iteration. The iterator is fail-fast. This means the list is structurally modified at any time after the [ASN1CSeqOfListIterator](#) class is created, in any way except through the iterator's own remove or insert methods, the iterator's methods next and prev methods will return NULL. The remove, set and insert methods will return the RTERR_CONCMODF error code.

Public Member Functions

- OSBOOL [hasNext](#) ()
- OSBOOL [hasPrev](#) ()
- void * [next](#) ()
- void * [prev](#) ()
- int [remove](#) ()
- int [set](#) (void *data)
- int [insert](#) (void *data)
- int [getState](#) ()

Protected Member Functions

- [ASN1CSeqOfListIterator](#) ([ASN1CSeqOfList](#) *list)
- [ASN1CSeqOfListIterator](#) ([ASN1CSeqOfList](#) *list, OSRTDListNode *startNode)
- void * **operator new** (size_t, void *data)
- void **operator delete** (void *, void *)
- void **operator delete** (void *, size_t)

Protected Attributes

- [ASN1CSeqOfList](#) * **pSeqList**
- OSRTDListNode * **nextNode**
- OSRTDListNode * **lastNode**
- volatile int **expectedModCount**
- int **stat**

3.5.2 Member Function Documentation

3.5.2.1 OSBOOL ASN1CSeqOfListIterator::hasNext () [inline]

This method returns TRUE if this iterator has more elements when traversing the list in the forward direction.

In other words, the method returns TRUE if the `next` method would return an element rather than returning a null value.

Returns:

TRUE if next would return an element rather than returning a null value.

3.5.2.2 OSBOOL ASN1CSeqOfListIterator::hasPrev () [inline]

This method returns TRUE if this iterator has more elements when traversing the list in the reverse direction. In other words, this method will return TRUE if prev would return an element rather than returning a null value.

Returns:

TRUE if next would return an element rather than returning a null value.

3.5.2.3 void* ASN1CSeqOfListIterator::next ()

This method returns the next element in the list.

This method may be called repeatedly to iterate through the list or intermixed with calls to prev to go back and forth.

Returns:

The next element in the list. A null value will be returned if the iteration is not successful.

3.5.2.4 void* ASN1CSeqOfListIterator::prev ()

This method returns the previous element in the list.

This method may be called repeatedly to iterate through the list or intermixed with calls to next to go back and forth.

Parameters:

- none

Returns:

The previous element in the list. A null value will be returned if the iteration is not successful.

3.5.2.5 int ASN1CSeqOfListIterator::remove ()

This method removes from the list the last element that was returned by the next or prev methods.

This call can only be made once per call to the next or prev methods.

Parameters:

- none

Returns:

Completion status of operation:

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

3.5.2.6 int ASN1CSeqOfListIterator::set (void * data)

This method replaces the last element returned by the next or prev methods with the specified element.

This call can be made only if neither remove nor insert methods have been called after the last call to next or prev methods.

Parameters:

data The element that replaces the last element returned by the next or prev methods

Returns:

Completion status of operation:

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

3.5.2.7 int ASN1CSeqOfListIterator::insert (void * data)

This method inserts the specified element into the list.

The element is inserted immediately before the next element that would be returned by the next method, if any, and after the next element would be returned by the prev method, if any. If the list contains no elements, the new element becomes the sole element in the list. The new element is inserted before the implicit cursor: a subsequent call to next would be unaffected, and a subsequent call to prev would return the new element.

Parameters:

data The element to be inserted

Returns:

Completion status of operation:

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

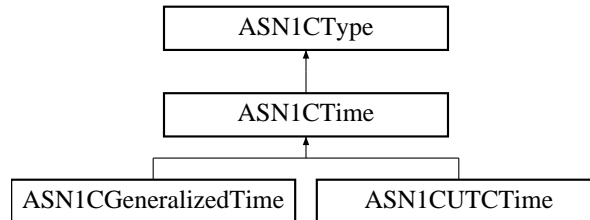
The documentation for this class was generated from the following file:

- [ASN1CSeqOfList.h](#)

3.6 ASN1CTime Class Reference

```
#include <ASN1CTime.h>
```

Inheritance diagram for ASN1CTime::



3.6.1 Detailed Description

ASN.1 Time control base class. The [ASN1CTime](#) class is derived from the [ASN1CType](#) base class. It is used as the abstract base class for generated control classes for the ASN.1 Generalized Time ([UNIVERSAL 24] IMPLICIT VisibleString) types and Universal Time ([UNIVERSAL 23] IMPLICIT VisibleString) types. This class provides utility methods for operating on the time information referenced by the generated class. This class can also be used inline to operate on the times within generated time string elements in a SEQUENCE, SET, or CHOICE construct. The time string are generally formatted according to ISO 8601 format with some exceptions (X.680).

Public Types

- enum {
 January = 1, **Jan** = 1, **February** = 2, **Feb** = 2,
 March = 3, **Mar** = 3, **April** = 4, **Apr** = 4,
 May = 5, **June** = 6, **Jun** = 6, **July** = 7,
 Jul = 7, **August** = 8, **Aug** = 8, **September** = 9,
 Sep = 9, **October** = 10, **Oct** = 10, **November** = 11,
 Nov = 11, **December** = 12, **Dec** = 12 }

Public Member Functions

- [ASN1CTime](#) ([OSRTMessageBufferIF](#) &msgBuf, char *&buf, int bufSize, OSBOOL useDerRules)
- [ASN1CTime](#) ([OSRTMessageBufferIF](#) &msgBuf, ASN1VisibleString &buf, OSBOOL useDerRules)
- [ASN1CTime](#) ([ASN1Context](#) &ctxt, char *&buf, int bufSize, OSBOOL useDerRules)
- [ASN1CTime](#) ([ASN1Context](#) &ctxt, ASN1VisibleString &buf, OSBOOL useDerRules)
- [ASN1CTime](#) (const [ASN1CTime](#) &original)
- virtual int [getYear](#) ()
- virtual int [getMonth](#) ()
- virtual int [getDay](#) ()
- virtual int [getHour](#) ()
- virtual int [getMinute](#) ()
- virtual int [getSecond](#) ()
- virtual int [getFraction](#) ()
- virtual double [getFractionAsDouble](#) ()

- virtual int `getFractionStr` (char *const pBuf, size_t bufSize)
- virtual int `getFractionLen` ()
- virtual int `getDiffHour` ()
- virtual int `getDiffMinute` ()
- virtual int `getDiff` ()
- virtual OSBOOL `getUTC` ()
- virtual time_t `getTime` ()
- void `setDER` (OSBOOL bvalue)
- virtual int `setUTC` (OSBOOL utc)
- virtual int `setYear` (int year_)
- virtual int `setMonth` (int month_)
- virtual int `setDay` (int day_)
- virtual int `setHour` (int hour_)
- virtual int `setMinute` (int minute_)
- virtual int `setSecond` (int second_)
- virtual int `setFraction` (int fraction, int fracLen=-1)
- virtual int `setFraction` (double frac, int fracLen)
- virtual int `setFraction` (char const *frac)
- virtual int `setTime` (time_t time, OSBOOL diffTime)=0
- virtual int `setDiffHour` (int dhour)
- virtual int `setDiff` (int dhour, int dminute)
- virtual int `setDiff` (int inMinutes)
- virtual int `parseString` (const char *string)
- virtual void `clear` ()
- virtual int `equals` (ASN1Time &)
- size_t `getTimeStringLength` ()
- const char * `getTimeString` (char *pbuf, size_t bufsize)
- const ASN1Time & `operator=` (const ASN1Time &)
- virtual OSBOOL `operator==` (ASN1Time &)
- virtual OSBOOL `operator!=` (ASN1Time &)
- virtual OSBOOL `operator>` (ASN1Time &)
- virtual OSBOOL `operator<` (ASN1Time &)
- virtual OSBOOL `operator>=` (ASN1Time &)
- virtual OSBOOL `operator<=` (ASN1Time &)

Protected Member Functions

- void `checkCapacity` ()
- char *& `getTimeStringPtr` ()
- virtual ASN1Time & `getTimeObj` ()=0
- virtual const ASN1Time & `getTimeObj` () const=0
- ASN1Time (char *&buf, int bufSize, OSBOOL useDerRules)
- ASN1Time (ASN1VisibleString &buf, OSBOOL useDerRules)
- virtual int `compileString` ()=0

Protected Attributes

- OSBOOL `parsed`
- OSBOOL `derRules`
- char *& `timeStr`
- int `strSize`

3.6.2 Constructor & Destructor Documentation

3.6.2.1 ASN1CTime::ASN1CTime (OSRTMessageBufferIF & msgBuf, char *& buf, int bufSize, OSBOOL useDerRules)

This constructor creates a time string from buffer.

It does not deep-copy the data; it just assigns the passed array to an internal reference variable. The object will then directly operate on the given data variable.

Parameters:

msgBuf Reference to an OSRTMessage buffer derived object (for example, ASNBEREncodeBuffer).

buf Reference to a pointer to a time string buffer.

bufSize Size of buffer in bytes.

useDerRules Use the Distinguished Encoding Rules (DER) to operate on this time value.

3.6.2.2 ASN1CTime::ASN1CTime (OSRTMessageBufferIF & msgBuf, ASN1VisibleString & buf, OSBOOL useDerRules)

This constructor creates a time string from an ASN1VisibleString object.

It does not deep-copy the data; it just assigns the passed object to an internal reference variable. The object will then directly operate on the given data variable.

Parameters:

msgBuf Reference to an OSRTMessage buffer derived object (for example, ASNBEREncodeBuffer).

buf Reference to a visible string object to hold the time data.

useDerRules Use the Distinguished Encoding Rules (DER) to operate on this time value.

3.6.2.3 ASN1CTime::ASN1CTime (const ASN1CTime & original)

The copy constructor. This does not deep-copy the original value. Instead, it assigns references to the internal components.

Parameters:

original The original time string object value.

3.6.3 Member Function Documentation

3.6.3.1 virtual int ASN1CTime::getYear () [virtual]

This method returns the year component of the time value.

Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters:

- none

Returns:

Year component (full 4 digits) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.6.3.2 virtual int ASN1CTime::getMonth () [virtual]

This method returns the month number component of the time value.

The number of January is 1, February 2, ... up to December 12. You may also use enumerated values for decoded months: `ASN1CTime::January`, `ASN1CTime::February`, etc. Also short aliases for months can be used: `ASN1CTime::Jan`, `ASN1CTime::Feb`, etc. Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters:

- none

Returns:

Month component (1 - 12) is returned if operation is successful. If the operation fails, a negative value is returned.

3.6.3.3 virtual int ASN1CTime::getDay () [virtual]

This method returns the day of month number component of the time value.

The number of the first day in the month is 1; the number of the last day may be in the interval from 28 to 31. Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters:

- none

Returns:

Day of month component (1 - 31) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.6.3.4 virtual int ASN1CTime::getHour () [virtual]

This method returns the hour component of the time value.

As the ISO 8601 is based on the 24-hour timekeeping system, hours are represented by two-digit values from 00 to 23. Note that the return value may differ from different inherited [ASN1CTime](#) classes.

Parameters:

- none

Returns:

Hour component (0 - 23) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.6.3.5 `virtual int ASN1CTime::getMinute () [virtual]`

This method returns the minute component of the time value.

Minutes are represented by the two digits from 00 to 59. Note that the return value may differ from different inherited [ASN1CTime](#) classes.

Parameters:

- none

Returns:

Minute component (0 - 59) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.6.3.6 `virtual int ASN1CTime::getSecond () [virtual]`

This method returns the second component of the time value.

Seconds are represented by two digits from 00 to 59. Note that the return value may differ from different inherited [ASN1CTime](#) classes.

Parameters:

- none

Returns:

Second component (0 - 59) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.6.3.7 `virtual int ASN1CTime::getFraction () [virtual]`

This method returns the second's decimal component of the time value.

Second's decimal fraction is represented by one or more digits from 0 to 9. Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters:

- none

Returns:

Second's decimal fraction component (0 - 9) is returned if operation is successful. If the operation fails, a negative value is returned.

Reimplemented in [ASN1CUTCTime](#).

3.6.3.8 `virtual double ASN1CTime::getFractionAsDouble () [virtual]`

This method returns the second's decimal component of the time value. Second's fraction will be represented as double value more than 0 and less than 1.

Second's decimal fraction is represented by one or more digits from 0 to 9.

Returns:

Second's decimal fraction component is returned if operation is successful. If the operation fails, a negative value is returned.

3.6.3.9 virtual int ASN1CTime::getFractionStr (char *const pBuf, size_t bufSize) [virtual]

This method returns the second's decimal component of the time value. Second's fraction will be represented as string w/o integer part and decimal point.

Returns:

Length of the fraction string returned in pBuf, if operation is successful. If the operation fails, a negative value is returned.

3.6.3.10 virtual int ASN1CTime::getFractionLen () [virtual]

This method returns the number of digits in second's decimal component of the time value.

Returns:

Second's decimal fraction's length is returned if operation is successful. If the operation fails, a negative value is returned.

3.6.3.11 virtual int ASN1CTime::getDiffHour () [virtual]

This method returns the hour component of the difference between the time zone of the object and the Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters:

- none

Returns:

The negative or positive hour component of the difference between the time zone of the object and the UTC time (-12 - +12) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.6.3.12 virtual int ASN1CTime::getDiffMinute () [virtual]

This method returns the minute component of the difference between the time zone of the object and the Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters:

- none

Returns:

The negative or positive minute component of the difference between the time zone of the object and the UTC time (-59 - +59) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.6.3.13 virtual int ASN1CTime::getDiff () [virtual]

This method returns the difference between the time zone of the object and the Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters:

- none

Returns:

The negative or positive minute component of the difference between the time zone of the object and the UTC time (-12*60 - +12*60) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.6.3.14 virtual OSBOOL ASN1CTime::getUTC () [virtual]

This method returns the UTC flag state.

If the UTC flag is TRUE, then the time is a UTC time and symbol Z is added at the end of the time string. Otherwise, it is local time.

Parameters:

- none

Returns:

UTC flag state is returned.

3.6.3.15 virtual time_t ASN1CTime::getTime () [virtual]

This method converts the time string to a value of the built-in C type time_t.

The value is the number of seconds from January 1, 1970. If the time is represented as UTC time plus or minus a time difference, then the resulting value will be recalculated as local time. For example, if the time string is "19991208120000+0930", then this string will be converted to "19991208213000" and then converted to a time_t value. Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters:

- none

Returns:

The time value, expressed as a number of seconds from January 1, 1970. If the operation fails, a negative value is returned.

3.6.3.16 void ASN1CTime::setDER (OSBOOL *bvalue*) [inline]

This method sets the 'use DER' flag which enforces the DER rules when time strings are constructed or parsed.

3.6.3.17 virtual int ASN1CTime::setUTC (OSBOOL *utc*) [virtual]

This method sets the UTC flag state.

If the UTC flag is TRUE, then the time is a UTC time and symbol 'Z' is added to the end of the string. Otherwise, it is a local time.

Parameters:

utc UTC flag state.

Returns:

Completion status of operation:

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

3.6.3.18 virtual int ASN1CTime::setYear (int *year_*) [virtual]

This method sets the year component of the time value.

Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters:

year_ Year component (full 4 digits).

Returns:

Completion status of operation:

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

3.6.3.19 virtual int ASN1CTime::setMonth (int *month_*) [virtual]

This method sets the month number component of the time value.

The number of January is 1, February 2, ..., through December 12. You may use enumerated values for months encoding: ASN1CTime::January, ASN1CTime::February, etc. Also you can use short aliases for months: ASN1CTime::Jan, ASN1CTime::Feb, etc. Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters:

month_ Month component (1 - 12).

Returns:

Completion status of operation:

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

3.6.3.20 virtual int ASN1CTime::setDay (int *day_*) [virtual]

This method sets the day of month number component of the time value.

The number of the first day in the month is 1; the number of the last day in the month may be in the interval from 28 to 31. Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters:

day_ Day of month component (1 - 31).

Returns:

Completion status of operation:

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

3.6.3.21 virtual int ASN1CTime::setHour (int *hour_*) [virtual]

This method sets the hour component of the time value.

As the ISO 8601 is based on the 24-hour timekeeping system, hours are represented by two digits from 00 to 23. Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters:

hour_ Hour component (0 - 23).

Returns:

Completion status of operation:

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

3.6.3.22 virtual int ASN1CTime::setMinute (int *minute_*) [virtual]

This method sets the minute component of the time value.

Minutes are represented by two digits from 00 to 59. Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters:

minute_ Minute component (0 - 59).

Returns:

Completion status of operation:

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

3.6.3.23 virtual int ASN1CTime::setSecond (int *second_*) [virtual]

This method sets the second component of the time value.

Seconds are represented by two digits from 00 to 59. Note that the action of this method may differ from different inherited [ASN1CTime](#) classes.

Parameters:

second_ Second component (0 - 59).

Returns:

Completion status of operation:

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

3.6.3.24 virtual int ASN1CTime::setFraction (int *fraction*, int *fracLen* = -1) [virtual]

This method sets the second's decimal fraction component of the time value.

Second's decimal fraction is represented by one or more digits from 0 to 9. Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters:

fraction Second's decimal fraction component (0 - 9).

fracLen Optional parameter specifies number of digits in second's fraction.

Returns:

Completion status of operation:

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

3.6.3.25 virtual int ASN1CTime::setFraction (double *frac*, int *fracLen*) [virtual]

This method sets the second's decimal fraction component of the time value. Double value must be greater or equal 0 and less than 1.

Parameters:

frac Second's decimal fraction component.

fracLen Specifies number of digits in second's fraction.

Returns:

Completion status of operation:

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

3.6.3.26 `virtual int ASN1CTime::setFraction (char const * frac) [virtual]`

This method sets the second's decimal fraction component of the time value. Double value must be greater or equal 0 and less than 1.

Parameters:

frac Second's decimal fraction component.

Returns:

Completion status of operation:

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

3.6.3.27 `virtual int ASN1CTime::setTime (time_t time, OSBOOL diffTime) [pure virtual]`

This converts the value of the C built-in type `time_t` to a time string.

The value is the number of seconds from January 1, 1970. Note that the action of this method may differ for different inherited [ASN1CTime](#) Classes.

Parameters:

time The time value, expressed as a number of seconds from January 1, 1970.

diffTime TRUE means the difference between local time and UTC time will be calculated; in other case, only local time will be stored.

Returns:

Completion status of operation:

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

Implemented in [ASN1CGeneralizedTime](#), and [ASN1CUTCTime](#).

3.6.3.28 `virtual int ASN1CTime::setDiffHour (int dhour) [virtual]`

This method sets the hour component of the difference between the time zone of the object and the Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the action of this method may differ from different inherited [ASN1CTime](#) classes.

Parameters:

dhour The negative or positive hour component of the difference between the time zone of the object and the UTC time (-12 - +12) is returned if the operation is successful. If the operation fails, a negative value is returned.

Returns:

Completion status of operation:

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

3.6.3.29 virtual int ASN1CTime::setDiff (int *dhour*, int *dminute*) [virtual]

This method sets the hours and the minute components of the difference between the time zone of the object and Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters:

dhour The negative or positive hour component of the difference between the time zone of the object and the UTC time (-12 - +12).

dminute The negative or positive minute component of the difference between the time zone of the object and the UTC time (-59 - +59).

Returns:

Completion status of operation:

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

3.6.3.30 virtual int ASN1CTime::setDiff (int *inMinutes*) [virtual]

This method sets the difference between the time zone of the object and Coordinated Universal Time (UTC), in minutes.

The UTC time is the sum of the local time and a positive or negative time difference. Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters:

inMinutes The negative or positive difference, in minutes, between the time zone of the object and the UTC time (-12*60 - +12*60) is returned if the operation is successful.

Returns:

Completion status of operation:

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

3.6.3.31 virtual int ASN1CTime::parseString (const char * *string*) [virtual]

This method parses the given time string.

The string is expected to be in the ASN.1 value notation format for the given ASN.1 time string type. Note that the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters:

string The time string value to be parsed.

Returns:

Completion status of operation:

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

3.6.3.32 virtual void ASN1CTime::clear () [virtual]

This method clears the time string.

Note the action of this method may differ for different inherited [ASN1CTime](#) classes.

Parameters:

- none

Returns:

- none

3.6.3.33 virtual int ASN1CTime::equals ([ASN1CTime &](#)) [virtual]

This method compares times.

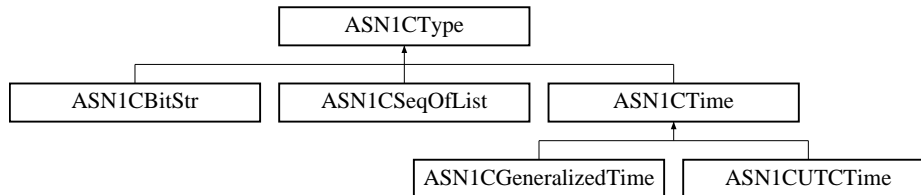
The documentation for this class was generated from the following file:

- [ASN1CTime.h](#)

3.7 ASN1CType Class Reference

```
#include <asn1CppType.h>
```

Inheritance diagram for ASN1CType::



3.7.1 Detailed Description

ASN1C control class base class. This is the main base class for all generated ASN1C_<name> control classes. It holds a variable of a generated data type as well as the associated message buffer or stream class to which a message will be encoded or from which a message will be decoded.

Public Member Functions

- [ASN1CType](#) ([OSRTMessageBufferIF](#) &msgBuf)
- [ASN1CType](#) (const [ASN1CType](#) &orig)
- virtual [~ASN1CType](#) ()
- void [append](#) ([OSRTDList](#) &l1ist, void *pdata)
- [OSRTCtxtPtr](#) [getContext](#) ()
- [OSCTXT](#) * [getCtxtPtr](#) ()
- int [getStatus](#) () const
- void * [memAlloc](#) (size_t numocts)
- void [memFreeAll](#) ()
- void * [memRealloc](#) (void *ptr, size_t numocts)
- void [memReset](#) ()
- void [memFreePtr](#) (void *ptr)
- void [printErrorInfo](#) ()
- void [resetError](#) ()
- OSBOOL [setDiag](#) (OSBOOL value)
- virtual int [Encode](#) ()
- virtual int [Decode](#) ()
- virtual int [EncodeTo](#) ([OSRTMessageBufferIF](#) &msgBuf)
- virtual int [DecodeFrom](#) ([OSRTMessageBufferIF](#) &msgBuf)

Protected Member Functions

- [ASN1CType](#) ()
- [ASN1CType](#) ([OSRTContext](#) &ctxt)
- int [setMsgBuf](#) ([OSRTMessageBufferIF](#) &msgBuf, OSBOOL initBuf=FALSE)
- int [setRunTimeKey](#) (const OSOCTET *key, size_t keylen)

Protected Attributes

- OSRTCtxtPtr mpContext
- OSRTMessageBufferIF * mpMsgBuf

3.7.2 Constructor & Destructor Documentation

3.7.2.1 ASN1Type::ASN1Type () [protected]

The default constructor sets the message pointer member variable to NULL and creates a new context object.

3.7.2.2 ASN1Type::ASN1Type (OSRTContext & ctxt) [protected]

This constructor sets the message pointer member variable to NULL and initializes the context object to point at the given context value.

Parameters:

ctxt - Reference to a context object.

3.7.2.3 ASN1Type::ASN1Type (OSRTMessageBufferIF & msgBuf)

This constructor sets the internal message buffer pointer to point at the given message buffer or stream object. The context is set to point at the context contained within the message buffer object. Thus, the message buffer and control class object share the context. It will not be released until both objects are destroyed.

Parameters:

msgBuf - Reference to a message buffer or stream object.

3.7.2.4 ASN1Type::ASN1Type (const ASN1Type & orig)

The copy constructor sets the internal message buffer pointer and context to point at the message buffer and context from the original ASN1Type object.

Parameters:

orig - Reference to a message buffer or stream object.

3.7.2.5 virtual ASN1Type::~~ASN1Type () [inline, virtual]

The virtual destructor does nothing. It is overridden by derived versions of this class.

3.7.3 Member Function Documentation

3.7.3.1 int ASN1Type::setRunTimeKey (const OSOCTET * key, size_t keylen) [protected]

This method sets run-time key to the context. This method does nothing for unlimited redistribution libraries.

Parameters:

key - array of octets with the key

keylen - number of octets in key array.

Returns:

Completion status of operation:

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

3.7.3.2 void ASN1Type::append (OSRTDList & llist, void *pdata) [inline]

The append method can be used to append an element to any linked list structure contained within the generated type.

Parameters:

llist Linked list structure.

pdata Data record to be appended. Note that the pointer value is appended. The data is not copied.

3.7.3.3 OSRTxtPtr ASN1Type::getContext () [inline]

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

Returns:

Context smart pointer object.

3.7.3.4 OSCTXT* ASN1Type::getCtxtPtr () [inline]

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

3.7.3.5 int ASN1Type::getStatus () const [inline]

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.

3.7.3.6 void* ASN1Type::memAlloc (size_t numocts) [inline]

The memAlloc method allocates memory using the C runtime memory management functions. The memory is tracked in the underlying context structure. When both this ASN1Type derived control class object and the message buffer object are destroyed, this memory will be freed.

Parameters:

numocts Number of bytes of memory to allocate

Returns:

Void pointer to allocated memory or NULL if insufficient memory was available to fulfill the request.

3.7.3.7 void ASN1Type::memFreeAll () [inline]

The memFreeAll method will free all memory currently tracked within the context. This includes all memory allocated with the memAlloc method as well as any memory allocated using the C `rtxMemAlloc` function with the context returned by the `getCtxtPtr` method.

3.7.3.8 void* ASN1Type::memRealloc (void * ptr, size_t numocts) [inline]

The memRealloc method reallocates memory using the C runtime memory management functions.

Parameters:

ptr Original pointer containing dynamic memory to be resized.

numocts Number of bytes of memory to allocate

Returns:

Reallocated memory pointer

3.7.3.9 void ASN1Type::memReset () [inline]

The memReset method resets dynamic memory using the C runtime memory management functions.

3.7.3.10 void ASN1Type::memFreePtr (void * ptr) [inline]

The memFreePtr method frees the memory at a specific location. This memory must have been allocated using the memAlloc method described earlier.

Parameters:

ptr - Pointer to a block of memory allocated with `memAlloc`

3.7.3.11 void ASN1Type::printErrorInfo () [inline]

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

3.7.3.12 void ASN1CType::resetError () [inline]

This method resets error status and stack trace. This method should be used to continue operations after recovering from the error.

3.7.3.13 OSBOOL ASN1CType::setDiag (OSBOOL *value*) [inline]

This method turns diagnostic tracing on or off.

Parameters:

value Boolean value; TRUE = turn tracing on.

Returns:

Previous state.

3.7.3.14 virtual int ASN1CType::Encode () [virtual]

The `Encode` method encodes an ASN.1 message using the encoding rules specified by the derived message buffer object.

Returns:

Completion status of operation:

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

3.7.3.15 virtual int ASN1CType::Decode () [virtual]

The `Decode` method decodes the ASN.1 message described by the encapsulated message buffer object.

Returns:

Completion status of operation:

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

3.7.3.16 virtual int ASN1CType::EncodeTo (OSRTMessageBufferIF & *msgBuf*) [inline, virtual]

The `EncodeTo` method encodes an ASN.1 message into the given message buffer or stream argument.

Parameters:

msgBuf Message buffer or stream to which the message is to be encoded.

Returns:

Completion status of operation:

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

3.7.3.17 `virtual int ASN1Type::DecodeFrom (OSRTMessageBufferIF & msgBuf)` [inline, virtual]

The `DecodeFrom` method decodes an ASN.1 message from the given message buffer or stream argument.

Parameters:

msgBuf Message buffer or stream containing message to decode.

Returns:

Completion status of operation:

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

3.7.4 Member Data Documentation

3.7.4.1 `OSRTCtxPtr ASN1Type::mpContext` [protected]

The `mpContext` member variable holds a reference-counted C runtime variable. This context is used in calls to all C run-time functions. The context pointed at by this smart-pointer object is shared with the message buffer object contained within this class.

3.7.4.2 `OSRTMessageBufferIF* ASN1Type::mpMsgBuf` [protected]

The `mpMsgBuf` member variable is a pointer to a derived message buffer or stream class that will manage the ASN.1 message being encoded or decoded.

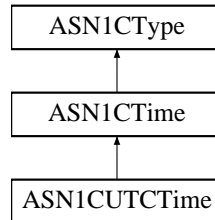
The documentation for this class was generated from the following file:

- [asn1CppType.h](#)

3.8 ASN1CUTCTime Class Reference

```
#include <ASN1CUTCTime.h>
```

Inheritance diagram for ASN1CUTCTime::



3.8.1 Detailed Description

ASN.1 UTCTime control class. The ASN1CUTCTime class is derived from the [ASN1CTime](#) base class. It used as the bass class for generated control classes for the ASN.1 Universal Time ([UNIVERSAL 23] IMPLICIT VisibleString) type. This class provides utility methods for operating on the time information referenced by the generated class. This class can also be used inline to operate on the time within generated time string elements in a SEQUENCE, SET, or CHOICE construct. The string generally is encoded according to ISO 8601 format with some exceptions (see X.680).

Public Member Functions

- [ASN1CUTCTime](#) ([OSRTMessageBufferIF](#) &msgBuf, char *&buf, int bufSize, OSBOOL useDerRules=FALSE)
- [ASN1CUTCTime](#) ([OSRTMessageBufferIF](#) &msgBuf, [ASN1UTCTime](#) &buf, OSBOOL useDerRules=FALSE)
- [ASN1CUTCTime](#) ([ASN1Context](#) &ctxt, char *&buf, int bufSize, OSBOOL useDerRules=FALSE)
- [ASN1CUTCTime](#) ([ASN1Context](#) &ctxt, [ASN1UTCTime](#) &buf, OSBOOL useDerRules=FALSE)
- [ASN1CUTCTime](#) (const [ASN1CUTCTime](#) &original)
- int [setTime](#) (time_t time, OSBOOL diffTime)
- const [ASN1CUTCTime](#) & **operator=** (const [ASN1CUTCTime](#) &tm)

Protected Member Functions

- virtual [ASN1TTime](#) & **getTimeObj** ()
- virtual const [ASN1TTime](#) & **getTimeObj** () const
- [ASN1CUTCTime](#) (char *&buf, int bufSize, OSBOOL useDerRules=FALSE)
- [ASN1CUTCTime](#) ([ASN1UTCTime](#) &buf, OSBOOL useDerRules=FALSE)
- int **compileString** ()
- int [getFraction](#) ()
- int **setFraction** (int fraction)

Protected Attributes

- [ASN1TUTCTime](#) **timeObj**

3.8.2 Constructor & Destructor Documentation

3.8.2.1 ASN1CUTCTime::ASN1CUTCTime (OSRTMessageBufferIF & msgBuf, char *& buf, int bufSize, OSBOOL useDerRules = FALSE)

This constructor creates a time string from a buffer.

It does not deep-copy the data, it just assigns the passed array to an internal reference variable. The object will then directly operate on the given data variable.

Parameters:

msgBuf Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).

buf Reference to a pointer to a time string buffer.

bufSize Size of passed buffer, in bytes.

useDerRules Use the Distinguished Encoding Rules to encode or decode the value,

3.8.2.2 ASN1CUTCTime::ASN1CUTCTime (OSRTMessageBufferIF & msgBuf, ASN1UTCTime & buf, OSBOOL useDerRules = FALSE)

This constructor creates a time string using the ASN1UTCTime argument. The constructor does not deep-copy the variable, it assigns a reference to it to an internal variable. The object will then directly operate on the given data variable. This form of the constructor is used with a compiler-generated time string variable.

Parameters:

msgBuf Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).

buf Reference to a time string structure.

useDerRules Use the Distinguished Encoding Rules to encode or decode the value,

3.8.3 Member Function Documentation

3.8.3.1 int ASN1CUTCTime::setTime (time_t time, OSBOOL diffTime) [virtual]

Converts time_t to time string.

Parameters:

time time to convert,

diffTime TRUE means the difference between local time and UTC will be calculated; in other case only local time will be stored.

Returns:

Completion status of operation:

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

Implements [ASN1CTime](#).

3.8.3.2 `int ASN1CUTCTime::getFraction ()` [protected, virtual]

This method returns the second's decimal component of the time value.

Second's decimal fraction is represented by one or more digits from 0 to 9. Note that the return value may differ for different inherited [ASN1CTime](#) classes.

Parameters:

- none

Returns:

Second's decimal fraction component (0 - 9) is returned if operation is successful. If the operation fails, a negative value is returned.

Reimplemented from [ASN1CTime](#).

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

- [ASN1CUTCTime.h](#)

3.9 Asn1ErrorHandler Class Reference

```
#include <asn1CppEvtHndlr.h>
```

3.9.1 Detailed Description

Error handler base class. This is the base class from which user-defined error classes are derived. These classes can be used to provide fault-tolerance when parsing a message. The normal decoder behavior is to stop decoding when it encounters an error. An error handler can be used to ignore or take corrective action that will allow the decoding process to continue.

Public Member Functions

- virtual int [error](#) (OSCTXT *pCtxt, ASN1CCB *pCCB, int stat)=0

Static Public Member Functions

- static int [invoke](#) (OSCTXT *pCtxt, ASN1CCB *pCCB, int stat)
- static int [invoke](#) (OSCTXT *pCtxt, OSOCTET *ptr, int len, int stat)
- static void [setErrorHandler](#) (OSCTXT *pCtxt, [Asn1ErrorHandler](#) *pHandler)

3.9.2 Member Function Documentation

3.9.2.1 virtual int Asn1ErrorHandler::error (OSCTXT * *pCtxt*, ASN1CCB * *pCCB*, int *stat*) [pure virtual]

The error handler callback method. This is the method that the user must override to provide customized error handling.

Parameters:

- pCtxt* - Pointer to a context block structure.
- pCCB* - Pointer to a context control block structure.
- stat* - The error status that caused the handler to be invoked.

Returns:

- Corrected status. Set to 0 to cause decoding to continue or to a negative status code (most likely *stat*) to cause decoding to terminate.

3.9.2.2 static void Asn1ErrorHandler::setErrorHandler (OSCTXT * *pCtxt*, [Asn1ErrorHandler](#) * *pHandler*) [static]

This static method is called to set the error handler within the context structure. Note that unlike event handlers, only a single error handling object can be specified. This must be called by the user to specify the error handling object prior to execution of the main decode function..

Parameters:

- pCtxt* - Pointer to a context block structure.

pHandler - Pointer to error handler object to register.

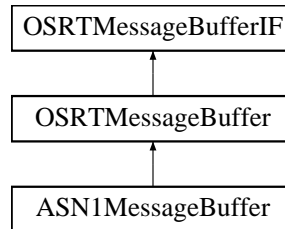
The documentation for this class was generated from the following file:

- [asn1CppEvtHndlr.h](#)

3.10 ASN1MessageBuffer Class Reference

```
#include <asn1CppTypes.h>
```

Inheritance diagram for ASN1MessageBuffer::



3.10.1 Detailed Description

Abstract ASN.1 message buffer base class. This class is used to manage a message buffer containing an ASN.1 message. For encoding, this is the buffer the message is being built in. For decoding, it is a message that was read into memory to be decoded. Further classes are derived from this to handle encoding and decoding of messages for different encoding rules types.

Public Member Functions

- virtual `~ASN1MessageBuffer ()`
- virtual void `addEventHandler (Asn1NamedEventHandler *pEventHandler)`
- virtual `ASN1BMPString * CStringToBMPString (const char *cstring, ASN1BMPString *pBMPString, Asn116BitCharSet *pCharSet=0)`
- virtual void `* getAppInfo ()`
- virtual int `initBuffer (OSRTMEMBUF &membuf)`
- virtual int `initBuffer (OSUNICHAR *unistr)`
- virtual int `initBuffer (const OSUTF8CHAR *str)`
- virtual `OSBOOL isA (Type bufferType)`
- virtual void `removeEventHandler (Asn1NamedEventHandler *pEventHandler)`
- virtual void `resetErrorInfo ()`
- virtual void `setAppInfo (void *pAppInfo)`
- virtual void `setErrorHandler (Asn1ErrorHandler *pErrorHandler)`
- int `setRunTimeKey (const OSOCTET *key, size_t keylen)`
- `OSOCTET * GetMsgCopy ()`
- `const OSOCTET * GetMsgPtr ()`
- void `PrintErrorInfo ()`

Protected Member Functions

- `ASN1MessageBuffer (Type bufferType)`
- `ASN1MessageBuffer (Type bufferType, OSRTContext *pContext)`
- virtual int `setStatus (int stat)`

3.10.2 Constructor & Destructor Documentation

3.10.2.1 ASN1MessageBuffer::ASN1MessageBuffer (Type *bufferType*) [protected]

The protected constructor creates a new context and sets the buffer class type.

Parameters:

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

3.10.2.2 virtual ASN1MessageBuffer::~~ASN1MessageBuffer () [inline, virtual]

The virtual destructor does nothing. It is overridden by derived versions of this class.

3.10.3 Member Function Documentation

3.10.3.1 virtual int ASN1MessageBuffer::setStatus (int *stat*) [inline, protected, virtual]

This method sets error status to the context.

Returns:

Error status value being set.

3.10.3.2 virtual void ASN1MessageBuffer::addEventHandler ([Asn1NamedEventHandler](#) * *pEventHandler*) [inline, virtual]

The addEventHandler method is used to register a user-defined named event handler. Methods from within this handler will be invoked when this message buffer is used in the decoding of a message.

Parameters:

pEventHandler - Pointer to named event handler object to register.

3.10.3.3 virtual ASN1BMPString* ASN1MessageBuffer::CStringToBMPString (const char * *cstring*, ASN1BMPString * *pBMPString*, Asn116BitCharSet * *pCharSet* = 0) [virtual]

The CStringToBMPString method is a utility function for converting a null-terminated Ascii string into a BMP string. A BMP string contains 16-bit Unicode characters.

Parameters:

cstring - Null-terminated character string to convert

pBMPString - Pointer to BMP string target variable

pCharSet - Pointer to permitted alphabet character set. If provided, index to character within this set is returned.

3.10.3.4 virtual void* ASN1MessageBuffer::getAppInfo () [inline, virtual]

Returns a pointer to application-specific information block

Reimplemented from [OSRTMessageBuffer](#).

3.10.3.5 **virtual int ASN1MessageBuffer::initBuffer (OSRTMEMBUF & *membuf*)** [virtual]

This version of the overloaded `initBuffer` method initializes the message buffer to point at the memory contained within the referenced `OSRTMEMBUF` object.

Parameters:

membuf `OSRTMEMBUF` memory buffer class object reference.

Returns:

Completion status of operation:

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

3.10.3.6 **virtual int ASN1MessageBuffer::initBuffer (OSUNICHAR * *unistr*)** [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:

unistr Pointer to a Unicode character string.

Returns:

Completion status of operation:

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

3.10.3.7 **virtual int ASN1MessageBuffer::initBuffer (const OSUTF8CHAR * *str*)** [inline, 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.

3.10.3.8 **virtual OSBOOL ASN1MessageBuffer::isA (Type *bufferType*)** [inline, virtual]

This method checks the type of the message buffer.

Parameters:

bufferType Enumerated identifier specifying a derived class. 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.

3.10.3.9 virtual void ASN1MessageBuffer::removeEventHandler (Asn1NamedEventHandler * pEventHandler) [inline, virtual]

The removeEventHandler method is used to de-register a used-defined named event handler.

Parameters:

pEventHandler - Pointer to named event handler object to de-register.

3.10.3.10 virtual void ASN1MessageBuffer::resetErrorInfo () [inline, virtual]

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

Reimplemented from [OSRTMessageBuffer](#).

3.10.3.11 virtual void ASN1MessageBuffer::setAppInfo (void *pAppInfo) [inline, virtual]

Sets the application-specific information block.

Reimplemented from [OSRTMessageBuffer](#).

3.10.3.12 virtual void ASN1MessageBuffer::setErrorHandler (Asn1ErrorHandler * pErrorHandler) [inline, virtual]

The setErrorHandler method is used to register a used-defined error handler. Methods from within this handler will be invoked when an error occurs in decoding a message using this message buffer object.

Parameters:

pErrorHandler - Pointer to error handler object to register.

3.10.3.13 int ASN1MessageBuffer::setRunTimeKey (const OSOCTET * key, size_t keylen)

This method sets run-time key to the context. This method does nothing for unlimited redistribution libraries.

Parameters:

key - array of octets with the key

keylen - number of octets in key array.

Returns:

Completion status of operation:

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

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

- [asn1CppType.h](#)

3.11 Asn1NamedEventHandler Class Reference

```
#include <asn1CppEvtHndlr.h>
```

3.11.1 Detailed Description

Named event handler base class. This is the base class from which user-defined event handler classes are derived. These classes can be used to handle events during the parsing of an ASN.1 message. The event callback methods that can be implemented are `startElement`, `endElement`, and `contents` methods.

Public Member Functions

- virtual void `startElement` (const char *name, int index)=0
- virtual void `endElement` (const char *name, int index)=0
- virtual void `boolValue` (OSBOOL value)=0
- virtual void `intValue` (OSINT32 value)=0
- virtual void `uintValue` (OSUINT32 value)=0
- virtual void `int64Value` (OSINT64 value)
- virtual void `uint64Value` (OSUINT64 value)
- virtual void `bitStrValue` (OSUINT32 numbits, const OSOCTET *data)=0
- virtual void `octStrValue` (OSUINT32 numocts, const OSOCTET *data)=0
- virtual void `charStrValue` (const char *value)=0
- virtual void `charStrValue` (OSUINT32 nchars, OSUNICHAR *data)=0
- virtual void `charStrValue` (OSUINT32 nchars, OS32BITCHAR *data)=0
- virtual void `nullValue` ()=0
- virtual void `oidValue` (OSUINT32 numSubIds, OSUINT32 *pSubIds)=0
- virtual void `realValue` (double value)=0
- virtual void `enumValue` (OSUINT32 value, const OSUTF8CHAR *text)=0
- virtual void `openTypeValue` (OSUINT32 numocts, const OSOCTET *data)=0

Static Public Member Functions

- static void `addEventHandler` (OSCTXT *pCtxt, [Asn1NamedEventHandler](#) *pHandler)
- static void `removeEventHandler` (OSCTXT *pCtxt, [Asn1NamedEventHandler](#) *pHandler)
- static void `invokeStartElement` (OSCTXT *pCtxt, const char *name, int index)
- static void `invokeEndElement` (OSCTXT *pCtxt, const char *name, int index)
- static void `invokeBoolValue` (OSCTXT *pCtxt, OSBOOL value)
- static void `invokeIntValue` (OSCTXT *pCtxt, OSINT32 value)
- static void `invokeUIntValue` (OSCTXT *pCtxt, OSUINT32 value)
- static void `invokeInt64Value` (OSCTXT *pCtxt, OSINT64 value)
- static void `invokeUInt64Value` (OSCTXT *pCtxt, OSUINT64 value)
- static void `invokeBitStrValue` (OSCTXT *pCtxt, OSUINT32 numbits, const OSOCTET *data)
- static void `invokeOctStrValue` (OSCTXT *pCtxt, OSUINT32 numocts, const OSOCTET *data)
- static void `invokeCharStrValue` (OSCTXT *pCtxt, const char *value)
- static void `invokeCharStrValue` (OSCTXT *pCtxt, OSUINT32 nchars, OSUNICHAR *data)
- static void `invokeCharStrValue` (OSCTXT *pCtxt, OSUINT32 nchars, OS32BITCHAR *data)
- static void `invokeNullValue` (OSCTXT *pCtxt)
- static void `invokeOidValue` (OSCTXT *pCtxt, OSUINT32 numSubIds, OSUINT32 *pSubIds)
- static void `invokeRealValue` (OSCTXT *pCtxt, double value)
- static void `invokeEnumValue` (OSCTXT *pCtxt, OSUINT32 value, const OSUTF8CHAR *text)
- static void `invokeOpenTypeValue` (OSCTXT *pCtxt, OSUINT32 numocts, const OSOCTET *data)

3.11.2 Member Function Documentation

3.11.2.1 `virtual void Asn1NamedEventHandler::startElement (const char * name, int index)` [pure virtual]

This method is invoked from within a decode function when an element of a SEQUENCE, SET, SEQUENCE OF, SET OF, or CHOICE construct is parsed.

Parameters:

name For SEQUENCE, SET, or CHOICE, this is the name of the element as defined in the ASN.1 definition. For SEQUENCE OF or SET OF, this is set to the name "element".

index For SEQUENCE, SET, or CHOICE, this is not used and is set to the value -1. For SEQUENCE OF or SET OF, this contains the zero-based index of the element in the conceptual array associated with the construct.

Returns:

- none

3.11.2.2 `virtual void Asn1NamedEventHandler::endElement (const char * name, int index)` [pure virtual]

This method is invoked from within a decode function when parsing is complete on an element of a SEQUENCE, SET, SEQUENCE OF, SET OF, or CHOICE construct.

Parameters:

name For SEQUENCE, SET, or CHOICE, this is the name of the element as defined in the ASN.1 definition. For SEQUENCE OF or SET OF, this is set to the name "element".

index For SEQUENCE, SET, or CHOICE, this is not used and is set to the value -1. For SEQUENCE OF or SET OF, this contains the zero-based index of the element in the conceptual array associated with the construct.

Returns:

- none

3.11.2.3 `virtual void Asn1NamedEventHandler::boolValue (OSBOOL value)` [pure virtual]

This method is invoked from within a decode function when a value of the BOOLEAN ASN.1 type is parsed.

Parameters:

value Parsed value.

Returns:

- none

3.11.2.4 **virtual void Asn1NamedEventHandler::intValue (OSINT32 *value*)** [pure virtual]

This method is invoked from within a decode function when a value of the INTEGER ASN.1 type is parsed.

Parameters:

value Parsed value.

Returns:

- none

3.11.2.5 **virtual void Asn1NamedEventHandler::uIntValue (OSUINT32 *value*)** [pure virtual]

This method is invoked from within a decode function when a value of the INTEGER ASN.1 type is parsed. In this case, constraints on the integer value forced the use of unsigned integer C type to represent the value.

Parameters:

value Parsed value.

Returns:

- none

3.11.2.6 **virtual void Asn1NamedEventHandler::int64Value (OSINT64 *value*)** [inline, virtual]

This method is invoked from within a decode function when a value of the 64-bit INTEGER ASN.1 type is parsed.

Parameters:

value Parsed value.

Returns:

- none

3.11.2.7 **virtual void Asn1NamedEventHandler::uInt64Value (OSUINT64 *value*)** [inline, virtual]

This method is invoked from within a decode function when a value of the 64-bit INTEGER ASN.1 type is parsed.

Parameters:

value Parsed value.

Returns:

- none

3.11.2.8 virtual void Asn1NamedEventHandler::bitStrValue (OSUINT32 *numbits*, const OSOCTET * *data*)
[pure virtual]

This method is invoked from within a decode function when a value of the BIT STRING ASN.1 type is parsed.

Parameters:

- numbits* - Number of bits in the parsed value.
- data* - Pointer to a byte array that contains the bit string data.

Returns:

- none

3.11.2.9 virtual void Asn1NamedEventHandler::octStrValue (OSUINT32 *numocts*, const OSOCTET * *data*)
[pure virtual]

This method is invoked from within a decode function when a value of one of the OCTET STRING ASN.1 type is parsed.

Parameters:

- numocts* Number of octets in the parsed value.
- data* Pointer to byte array containing the octet string data.

Returns:

- none

3.11.2.10 virtual void Asn1NamedEventHandler::charStrValue (const char * *value*) [pure virtual]

This method is invoked from within a decode function when a value of one of the 8-bit ASN.1 character string types is parsed.

Parameters:

- value* Null terminated character string value.

Returns:

- none

3.11.2.11 virtual void Asn1NamedEventHandler::charStrValue (OSUINT32 *nchars*, OSUNICHAR * *data*)
[pure virtual]

This method is invoked from within a decode function when a value of one of the 16-bit ASN.1 character string types is parsed.

This is used for the ASN.1 BmpString type.

Parameters:

- nchars* Number of characters in the parsed value.

data Pointer to an array containing 16-bit values. These are represented using unsigned short integer values.

Returns:

- none

3.11.2.12 virtual void Asn1NamedEventHandler::charStrValue (OSUINT32 *nchars*, OS32BITCHAR * *data*)
[pure virtual]

This method is invoked from within a decode function when a value of one of the 32-bit ASN.1 character string types is parsed.

This is used for the ASN.1 UniversalString type.

Parameters:

nchars Number of characters in the parsed value.

data Pointer to an array containing 32-bit values. Each 32-bit integer value is a universal character.

Returns:

- none

3.11.2.13 virtual void Asn1NamedEventHandler::nullValue () [pure virtual]

This method is invoked from within a decode function when a value of the NULL ASN.1 type is parsed.

Parameters:

- none

Returns:

- none

3.11.2.14 virtual void Asn1NamedEventHandler::oidValue (OSUINT32 *numSubIds*, OSUINT32 * *pSubIds*)
[pure virtual]

This method is invoked from within a decode function when a value of the OBJECT IDENTIFIER ASN.1 type is parsed.

Parameters:

numSubIds Number of subidentifiers in the object identifier.

pSubIds Pointer to array containing the subidentifier values.

Returns:

-none

3.11.2.15 **virtual void Asn1NamedEventHandler::realValue (double *value*)** [pure virtual]

This method is invoked from within a decode function when a value the REAL ASN.1 type is parsed.

Parameters:

value Parsed value.

Returns:

- none

3.11.2.16 **virtual void Asn1NamedEventHandler::enumValue (OSUINT32 *value*, const OSUTF8CHAR **text*)** [pure virtual]

This method is invoked from within a decode function when a value of the ENUMERATED ASN.1 type is parsed.

Parameters:

value - Parsed enumerated value

text - Textual value of enumerated identifier

Returns:

- none

3.11.2.17 **virtual void Asn1NamedEventHandler::openTypeValue (OSUINT32 *numocts*, const OSOCTET **data*)** [pure virtual]

This value is invoked from within a decode function when an ASN.1 open type is parsed.

Parameters:

numocts Number of octets in the parsed value.

data Pointer to byet array contain in tencoded ASN.1 value.

Returns:

- none

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

- [asn1CppEvtHndlr.h](#)

3.12 ASN1TBMPString Struct Reference

```
#include <asn1CppType.h>
```

3.12.1 Detailed Description

BMPString. This is the base class for generated C++ data type classes for BMPString values.

Public Member Functions

- [ASN1TBMPString \(\)](#)

3.12.2 Constructor & Destructor Documentation

3.12.2.1 ASN1TBMPString::ASN1TBMPString () [inline]

The default constructor creates an empty BMPString value.

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

- [asn1CppType.h](#)

3.13 ASN1TDynBitStr Struct Reference

```
#include <asn1CppTypes.h>
```

3.13.1 Detailed Description

Dynamic bit string. This is the base class for generated C++ data type classes for unsized BIT STRING's.

Public Member Functions

- [ASN1TDynBitStr \(\)](#)
- [ASN1TDynBitStr \(OSUINT32 _numbits, const OSOCTET *_data\)](#)
- [ASN1TDynBitStr \(ASN1DynBitStr &_bs\)](#)

3.13.2 Constructor & Destructor Documentation

3.13.2.1 ASN1TDynBitStr::ASN1TDynBitStr () [inline]

The default constructor creates an empty bit string.

3.13.2.2 ASN1TDynBitStr::ASN1TDynBitStr (OSUINT32 *_numbits*, const OSOCTET * *_data*) [inline]

This constructor initializes the bit string to contain the given data values.

Parameters:

_numbits - Number of bits in the bit string.

_data - The binary bit data values.

3.13.2.3 ASN1TDynBitStr::ASN1TDynBitStr (ASN1DynBitStr & *_bs*) [inline]

This constructor initializes the bit string to contain the given data values.

Parameters:

_bs - C bit string structure.

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

- [asn1CppTypes.h](#)

3.14 ASN1TDynOctStr Struct Reference

```
#include <ASN1TOctStr.h>
```

3.14.1 Detailed Description

Dynamic octet string. This is the base class for generated C++ data type classes for unsized OCTET string's.

Public Member Functions

- [ASN1TDynOctStr \(\)](#)
- [ASN1TDynOctStr \(OSUINT32 _numocts, const OSOCTET * _data\)](#)
- [ASN1TDynOctStr \(ASN1TDynOctStr &_os\)](#)
- [ASN1TDynOctStr \(const char *cstring\)](#)
- [ASN1TDynOctStr & operator= \(const char *cstring\)](#)
- [ASN1TDynOctStr & operator= \(const ASN1TDynOctStr &octet\)](#)
- [const char * toString \(OSCTXT *pctxt\) const](#)
- [const char * toHexString \(OSCTXT *pctxt\) const](#)
- [int nCompare \(OSUINT32 n, const ASN1TDynOctStr &o\) const](#)

3.14.2 Constructor & Destructor Documentation

3.14.2.1 ASN1TDynOctStr::ASN1TDynOctStr () [inline]

The default constructor creates an empty octet string.

3.14.2.2 ASN1TDynOctStr::ASN1TDynOctStr (OSUINT32 _numocts, const OSOCTET * _data) [inline]

This constructor initializes the octet string to contain the given data values.

Parameters:

_numocts - Number of octet in the octet string.

_data - The binary octet data values.

3.14.2.3 ASN1TDynOctStr::ASN1TDynOctStr (ASN1TDynOctStr &_os) [inline]

This constructor initializes the octet string to contain the given data values.

Parameters:

_os - C octet string structure.

3.14.2.4 ASN1TDynOctStr::ASN1TDynOctStr (const char * *cstring*) [inline]

This constructor initializes the octet string to contain the given data values. In this case, it is initialized the string to contain the characters in a null-terminated C character string.

Parameters:

cstring - C null-terminated string.

3.14.3 Member Function Documentation

3.14.3.1 ASN1TDynOctStr& ASN1TDynOctStr::operator= (const char * *cstring*) [inline]

This assignment operator sets the octet string to contain the characters in a null-terminated C character string. For example, `myOctStr = "a char string";`

Parameters:

cstring - C null-terminated string.

3.14.3.2 ASN1TDynOctStr& ASN1TDynOctStr::operator= (const ASN1TDynOctStr & *octet*)

This assignment operator sets the octet string to contain the characters from the given C++ octet string object.

Parameters:

octet - Octet string object reference

3.14.3.3 const char* ASN1TDynOctStr::toString (OSCTXT * *pctxt*) const

This method converts the binary octet string to a human-readable representation. The string is first checked to see if it contains all printable characters. If this is the case, the characters in the string are returned; otherwise, the string contents are converted into a hexadecimal character string.

Parameters:

pctxt - Pointer to a context structure.

3.14.3.4 const char* ASN1TDynOctStr::toHexString (OSCTXT * *pctxt*) const

This method converts the binary octet string to a hexadecimal string representation.

Parameters:

pctxt - Pointer to a context structure.

3.14.3.5 `int ASN1TDynOctStr::nCompare (OSUIN32 n, const ASN1TDynOctStr & o) const`

This method compares the first *n* octets of this octet string with the given octet string.

Parameters:

- n* - Number of octets to compare
- o* - Octet string for comparison

Returns:

- 0 if strings are equal, -1 if this octet string is less than the given string, +1 if this string > given string.

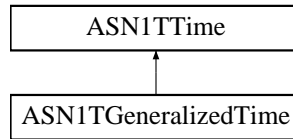
The documentation for this struct was generated from the following file:

- [ASN1TOctStr.h](#)

3.15 ASN1TGeneralizedTime Class Reference

```
#include <ASN1TTime.h>
```

Inheritance diagram for ASN1TGeneralizedTime::



3.15.1 Detailed Description

ASN.1 GeneralizedTime utility class. The [ASN1TGeneralizedTime](#) class is derived from the [ASN1TTime](#) base class.

Public Member Functions

- [ASN1TGeneralizedTime](#) ()
- [ASN1TGeneralizedTime](#) (const char *buf, OSBOOL useDerRules=FALSE)
- [ASN1TGeneralizedTime](#) (OSBOOL useDerRules)
- [ASN1TGeneralizedTime](#) (const [ASN1TGeneralizedTime](#) &original)
- int [getCentury](#) () const
- int [setCentury](#) (int century)
- int [setTime](#) (time_t time, OSBOOL diffTime)
- int [parseString](#) (const char *string)
- const [ASN1TGeneralizedTime](#) & **operator=** (const [ASN1TGeneralizedTime](#) &tm)
- int [compileString](#) (char *pbuf, size_t bufsize) const

3.15.2 Constructor & Destructor Documentation

3.15.2.1 ASN1TGeneralizedTime::ASN1TGeneralizedTime () [inline]

A default constructor.

3.15.2.2 ASN1TGeneralizedTime::ASN1TGeneralizedTime (const char * buf, OSBOOL useDerRules = FALSE)

This constructor creates a time object using the specified time string.

Parameters:

- buf* A pointer to the time string to be parsed.
- useDerRules* An OSBOOL value.

3.15.2.3 ASNITGeneralizedTime::ASNITGeneralizedTime (OSBOOL *useDerRules*) [inline]

This constructor creates an empty time object.

Parameters:

useDerRules An OSBOOL value.

3.15.2.4 ASNITGeneralizedTime::ASNITGeneralizedTime (const ASNITGeneralizedTime & *original*) [inline]

A copy constructor.

3.15.3 Member Function Documentation

3.15.3.1 int ASNITGeneralizedTime::getCentury () const

This method returns the centry part (first two digits) of the year component of the time value.

Returns:

Century part (first two digits) of the year component is returned if the operation is successful. If the operation fails, one of the negative status codes is returned.

3.15.3.2 int ASNITGeneralizedTime::setCentury (int *century*)

This method sets the centry part (first two digits) of the year component of the time value.

Parameters:

century Century part (first two digits) of the year component.

Returns:

Completion status of operation:

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

3.15.3.3 int ASNITGeneralizedTime::setTime (time_t *time*, OSBOOL *diffTime*) [virtual]

This converts the value of the C built-in type `time_t` to a time string.

The value is the number of seconds from January 1, 1970.

Parameters:

time The time value, expressed as a number of seconds from January 1, 1970.

diffTime TRUE means the difference between local time and UTC time will be calculated; in other case, only local time will be stored.

Returns:

Completion status of operation:

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

Implements [ASN1Time](#).

3.15.3.4 int ASN1TimeGeneralizedTime::parseString (const char * *string*) [virtual]

Parses sting.

Returns:

Completion status of operation:

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

Implements [ASN1Time](#).

3.15.3.5 int ASN1TimeGeneralizedTime::compileString (char * *pbuf*, size_t *bufsize*) const [virtual]

Compiles new time string accoring X.680 and ISO 8601. Returns 0, if succeed, or error code, if error.

Parameters:

pbuf A pointer to destination buffer.

bufsize A size of destination buffer.

Returns:

Completion status of operation:

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

Implements [ASN1Time](#).

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

- [ASN1Time.h](#)

3.16 Asn1TObject Struct Reference

```
#include <asn1CppType.h>
```

3.16.1 Detailed Description

Open type with table constraint. This is the base class for generated C++ data type classes for open type values with table constraints. It is only used when the `-tables` compiler command line option is specified.

Public Member Functions

- [Asn1TObject \(\)](#)

3.16.2 Constructor & Destructor Documentation

3.16.2.1 `Asn1TObject::Asn1TObject ()` [inline]

The default constructor creates an empty object value.

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

- [asn1CppType.h](#)

3.17 ASN1ObjId Struct Reference

```
#include <ASN1ObjId.h>
```

3.17.1 Detailed Description

Object identifier. This is the base class for generated C++ data type classes for object identifier values.

Public Member Functions

- [ASN1ObjId \(\)](#)
- virtual [~ASN1ObjId \(\)](#)
- [ASN1ObjId \(OSOCKET _numids, const OSUINTEGER32 *_subids\)](#)
- [ASN1ObjId \(const ASN1OBJID &oid\)](#)
- [ASN1ObjId \(const ASN1ObjId &oid\)](#)
- [ASN1ObjId \(const char *dotted_oid_string\)](#)
- [ASN1ObjId & operator= \(const char *dotted_oid_string\)](#)
- void [operator= \(const ASN1OBJID &rhs\)](#)
- void [operator= \(const ASN1ObjId &rhs\)](#)
- [ASN1ObjId & operator+= \(const char *dotted_oid_string\)](#)
- [ASN1ObjId & operator+= \(const OSUINTEGER32 i\)](#)
- [ASN1ObjId & operator+= \(const ASN1ObjId &o\)](#)
- const char * [toString \(OSCTXT *pctx\) const](#)
- void [set_data \(const OSUINTEGER32 *raw_oid, OSUINTEGER32 oid_len\)](#)
- int [nCompare \(const OSUINTEGER32 n, const ASN1ObjId &o\) const](#)
- int [RnCompare \(const OSUINTEGER32 n, const ASN1ObjId &o\) const](#)
- void [trim \(const OSUINTEGER32 n\)](#)

3.17.2 Constructor & Destructor Documentation

3.17.2.1 ASN1ObjId::ASN1ObjId () [inline]

The default constructor creates an empty object identifier value.

3.17.2.2 virtual ASN1ObjId::~~ASN1ObjId () [virtual]

The Virtual Destructor

3.17.2.3 ASN1ObjId::ASN1ObjId (OSOCKET _numids, const OSUINTEGER32 *_subids)

This constructor initializes the object identifier to contain the given data values.

Parameters:

_numids - Number of subidentifiers in the OID.

_subids - Array of subidentifier values.

3.17.2.4 ASN1ObjId::ASN1ObjId (const ASN1OBJID & oid)

This constructor initializes the object identifier to contain the given data values. This can be used to set the value to a compiler-generated OID value.

Parameters:

oid - C object identifier value.

3.17.2.5 ASN1ObjId::ASN1ObjId (const ASN1ObjId & oid)

The copy constructor.

Parameters:

oid - C++ object identifier value.

3.17.2.6 ASN1ObjId::ASN1ObjId (const char * dotted_oid_string)

Construct an OID from a dotted string.

Parameters:

dotted_oid_string - for example "1.3.1.6.1.10"

3.17.3 Member Function Documentation

3.17.3.1 ASN1ObjId& ASN1ObjId::operator= (const char * dotted_oid_string)

Assignment from a string.

Parameters:

dotted_oid_string - New value (for example "1.3.6.1.6.0");

3.17.3.2 void ASN1ObjId::operator= (const ASN1OBJID & rhs)

This assignment operator sets the object identifier to contain the OID in the given C structure. This can be used to set the value to a compiler-generated OID value.

Parameters:

rhs - C object identifier value.

3.17.3.3 void ASN1ObjId::operator= (const ASN1ObjId & rhs)

This assignment operator sets the object identifier to contain the OID in the given C++ structure.

Parameters:

rhs - C++ object identifier value.

3.17.3.4 `ASN1ObjId& ASN1ObjId::operator+= (const char * dotted_oid_string)`

Overloaded += operator. This operator allows subidentifiers in the form of a dotted OID string ("n.n.n") to be appended to an existing OID object.

Parameters:

dotted_oid_string - C++ object identifier value.

Returns:

- True if values are equal.

3.17.3.5 `ASN1ObjId& ASN1ObjId::operator+= (const OSUINT32 i)`

Overloaded += operator. This operator allows a single subidentifier in the form of an integer value to be appended to an existing OID object.

Parameters:

i - Subidentifier to append.

Returns:

- True if values are equal.

3.17.3.6 `ASN1ObjId& ASN1ObjId::operator+= (const ASN1ObjId & o)`

Overloaded += operator. This operator allows one object identifier to be appended to another object identifier.

Parameters:

o - C++ object identifier value.

Returns:

- True if values are equal.

3.17.3.7 `const char* ASN1ObjId::toString (OSCTXT * pctx) const`

Get a printable ASCII string of a part of the value.

Parameters:

pctx - Pointer to a context structure.

Returns:

- Dotted OID string (for example "3.6.1.6")

3.17.3.8 void ASN1ObjId::set_data (const OSUINT32 * raw_oid, OSUINT32 oid_len)

Sets the data of an object identifier using a pointer and a length.

Parameters:

raw_oid - Pointer to an array of subidentifier values.

oid_len - Number of subids in the array,

3.17.3.9 int ASN1ObjId::nCompare (const OSUINT32 n, const ASN1ObjId & o) const

Compare the first n sub-ids(left to right) of two object identifiers.

Parameters:

n - Number of subid values to compare.

o - OID to compare this OID with.

Returns:

- 0 if OID's are equal, -1 if this OID less than given OID, +1 if this OID > given OID.

3.17.3.10 int ASN1ObjId::RnCompare (const OSUINT32 n, const ASN1ObjId & o) const

Compare the last n sub-ids(right to left) of two object identifiers.

Parameters:

n - Number of subid values to compare.

o - OID to compare this OID with.

Returns:

- 0 if OID's are equal, -1 if this OID less than given OID, +1 if this OID > given OID.

3.17.3.11 void ASN1ObjId::trim (const OSUINT32 n)

Trim the given number of rightmost sub elements from this OID.

Parameters:

n - number of subids to trim from OID

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

- [ASN1ObjId.h](#)

3.18 ASN1ObjId64 Struct Reference

```
#include <asn1CppTypes.h>
```

3.18.1 Detailed Description

Object identifier with 64-bit arcs. This class is identical to the [ASN1ObjId](#) class except it allows 64-bit integers to be used for the arc (i.e. subidentifier) values.

Public Member Functions

- **ASN1ObjId64** (OSOCKET _numids, const OSINT64 *_subids)
- **ASN1ObjId64** (const ASN1OID64 &oid)
- **ASN1ObjId64** (const [ASN1ObjId64](#) &oid)
- void **operator=** (const ASN1OID64 &rhs)
- void **operator=** (const [ASN1ObjId64](#) &rhs)

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

- [asn1CppTypes.h](#)

3.19 ASN1OpenType Struct Reference

```
#include <asn1CppType.h>
```

3.19.1 Detailed Description

Open type. This is the base class for generated C++ data type classes for open type values.

Public Member Functions

- [ASN1OpenType \(\)](#)

3.19.2 Constructor & Destructor Documentation

3.19.2.1 ASN1OpenType::ASN1OpenType () [inline]

The default constructor creates an empty open type value.

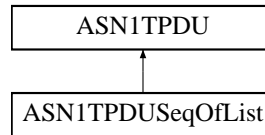
The documentation for this struct was generated from the following file:

- [asn1CppType.h](#)

3.20 ASN1TPDU Struct Reference

```
#include <asn1CppTypes.h>
```

Inheritance diagram for ASN1TPDU::



3.20.1 Detailed Description

Base class for PDU types. This class is used as the base class for all compiler-generated PDU types.

Public Member Functions

- void [setContext](#) ([OSRTContext](#) *ctxt)

Protected Attributes

- [OSRTCtxtPtr](#) [mpContext](#)

3.20.2 Member Function Documentation

3.20.2.1 void ASN1TPDU::setContext ([OSRTContext](#) * ctxt) [inline]

The setContext method allows the context member variable to be set. It is invoked in compiler-generated control class decode and copy methods. This method is invoked to prevent memory freeing of decoded or copied data after a control class or message buffer object goes out of scope. Also, if context is set to [ASN1TPDU](#) then generated destructor of inherited ASN1T_<type> class will invoke generated free routines. Note, it is not obligatory to call generated free routines unless a series of messages is being decoded or control class and message buffer objects go out of scope somewhere. The destructor of the control class or message buffer class will free all dynamically allocated memory. Thus, if performance is a main issue, "setContext (NULL)" may be called after Decode method call. In this case destructor of ASN1T_<type> will do nothing.

Parameters:

ctxt A pointer to reference counted ASN.1 context class instance.

3.20.3 Member Data Documentation

3.20.3.1 [OSRTCtxtPtr](#) [ASN1TPDU::mpContext](#) [protected]

The mpContext member variable holds a smart-pointer to the current context variable. This ensures an instance of this PDU type will persist if the control class and message buffer classes used to decode or copy the message are destroyed.

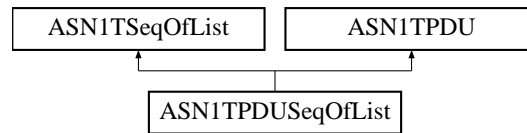
The documentation for this struct was generated from the following file:

- [asn1CppTypes.h](#)

3.21 ASN1TPDUSeqOfList Struct Reference

```
#include <asn1CppTypes.h>
```

Inheritance diagram for ASN1TPDUSeqOfList::



3.21.1 Detailed Description

SEQUENCE OF element holder (PDU). This class is used as the base class for compiler-generated SEQUENCE OF linked-list types. In this case, the type has also been determined to be a PDU.

Public Member Functions

- [ASN1TPDUSeqOfList \(\)](#)

3.21.2 Constructor & Destructor Documentation

3.21.2.1 ASN1TPDUSeqOfList::ASN1TPDUSeqOfList () [inline]

The default constructor creates an empty list.

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

- [asn1CppTypes.h](#)

3.22 ASN1TSeqExt Struct Reference

```
#include <asn1CppTypes.h>
```

3.22.1 Detailed Description

SEQUENCE or SET extension element holder. This is used for the /c extElem1 open extension element in extensible SEQUENCE or SET constructs.

Public Member Functions

- [ASN1TSeqExt \(\)](#)

3.22.2 Constructor & Destructor Documentation

3.22.2.1 ASN1TSeqExt::ASN1TSeqExt () [inline]

The default constructor creates an empty open extension element.

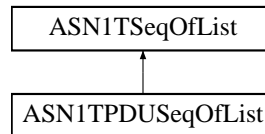
The documentation for this struct was generated from the following file:

- [asn1CppTypes.h](#)

3.23 ASN1TSeqOfList Struct Reference

```
#include <asn1CppTypes.h>
```

Inheritance diagram for ASN1TSeqOfList::



3.23.1 Detailed Description

SEQUENCE OF element holder. This class is used as the base class for compiler-generated SEQUENCE OF linked-list types.

Public Member Functions

- [ASN1TSeqOfList \(\)](#)

3.23.2 Constructor & Destructor Documentation

3.23.2.1 ASN1TSeqOfList::ASN1TSeqOfList () [inline]

The default constructor creates an empty list.

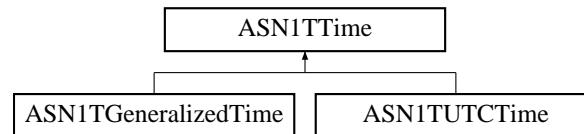
The documentation for this struct was generated from the following file:

- [asn1CppTypes.h](#)

3.24 ASN1Time Class Reference

```
#include <ASN1Time.h>
```

Inheritance diagram for ASN1Time::



3.24.1 Detailed Description

ASN.1 Time utility base class.

Public Types

- enum {
January = 1, **Jan** = 1, **February** = 2, **Feb** = 2,
March = 3, **Mar** = 3, **April** = 4, **Apr** = 4,
May = 5, **June** = 6, **Jun** = 6, **July** = 7,
Jul = 7, **August** = 8, **Aug** = 8, **September** = 9,
Sep = 9, **October** = 10, **Oct** = 10, **November** = 11,
Nov = 11, **December** = 12, **Dec** = 12 }

Public Member Functions

- [ASN1Time](#) (OSBOOL useDerRules)
- [ASN1Time](#) (const [ASN1Time](#) &original)
- virtual [~ASN1Time](#) ()
- virtual int [getYear](#) () const
- virtual int [getMonth](#) () const
- virtual int [getDay](#) () const
- virtual int [getHour](#) () const
- virtual int [getMinute](#) () const
- virtual int [getSecond](#) () const
- virtual int [getFraction](#) () const
- virtual double [getFractionAsDouble](#) () const
- virtual int [getFractionStr](#) (char *const pBuf, size_t bufSize) const
- virtual int [getFractionLen](#) () const
- virtual int [getDiffHour](#) () const
- virtual int [getDiffMinute](#) () const
- virtual int [getDiff](#) () const
- virtual OSBOOL [getUTC](#) () const
- virtual time_t [getTime](#) () const
- void [setDER](#) (OSBOOL bvalue)
- virtual int [setUTC](#) (OSBOOL utc)

- virtual int [setYear](#) (int year_)
- virtual int [setMonth](#) (int month_)
- virtual int [setDay](#) (int day_)
- virtual int [setHour](#) (int hour_)
- virtual int [setMinute](#) (int minute_)
- virtual int [setSecond](#) (int second_)
- virtual int [setFraction](#) (int fraction, int fracLen=-1)
- virtual int [setFraction](#) (double frac, int fracLen)
- virtual int [setFraction](#) (char const *frac)
- virtual int [setTime](#) (time_t time, OSBOOL diffTime)=0
- virtual int [setDiffHour](#) (int dhour)
- virtual int [setDiff](#) (int dhour, int dminute)
- virtual int [setDiff](#) (int inMinutes)
- virtual int [parseString](#) (const char *string)=0
- virtual void [clear](#) ()
- virtual int [compileString](#) (char *pbuf, size_t bufsize) const=0
- virtual int [equals](#) (const [ASN1TTime](#) &) const
- const char * [toString](#) (char *pbuf, size_t bufsize) const
- const char * [toString](#) (OSCTXT *pctx) const
- const char * [toString](#) () const
- const [ASN1TTime](#) & [operator=](#) (const [ASN1TTime](#) &)
- virtual OSBOOL [operator==](#) (const [ASN1TTime](#) &) const
- virtual OSBOOL [operator!=](#) (const [ASN1TTime](#) &) const
- virtual OSBOOL [operator>](#) (const [ASN1TTime](#) &) const
- virtual OSBOOL [operator<](#) (const [ASN1TTime](#) &) const
- virtual OSBOOL [operator>=](#) (const [ASN1TTime](#) &) const
- virtual OSBOOL [operator<=](#) (const [ASN1TTime](#) &) const

Public Attributes

- short **mYear**
- short **mMonth**
- short **mDay**
- short **mHour**
- short **mMinute**
- short **mSecond**
- short **mDiffHour**
- short **mDiffMin**
- int **mSecFraction**
- int **mSecFracLen**
- int **mStatus**
- OSBOOL **mbUtcFlag**
- OSBOOL **mbDerRules**

Static Public Attributes

- static short **mDaysInMonth** []

Protected Member Functions

- void **privateInit** ()
- int **getDaysNum** () const
- long **getMillisNum** () const

Static Protected Member Functions

- static int **checkDate** (int day, int month, int year)
- static void **addMilliseconds** (int deltaMs, short &year, short &month, short &day, short &hour, short &minute, short &second, int &secFraction, int secFracLen)
- static void **addDays** (int deltaDays, short &year, short &month, short &day)

3.24.2 Constructor & Destructor Documentation

3.24.2.1 ASN1TTime::ASN1TTime (OSBOOL *useDerRules*)

This constructor creates an empty time class.

Parameters:

useDerRules Use the Distinguished Encoding Rules (DER) to operate on this time value.

3.24.2.2 ASN1TTime::ASN1TTime (const ASN1TTime & *original*)

The copy constructor.

Parameters:

original The original time string object value.

3.24.2.3 virtual ASN1TTime::~ASN1TTime () [virtual]

The destructor.

3.24.3 Member Function Documentation

3.24.3.1 virtual int ASN1TTime::getYear () const [virtual]

This method returns the year component of the time value.

Note that the return value may differ for different inherited [ASN1TTime](#) classes.

Returns:

Year component (full 4 digits) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.24.3.2 `virtual int ASN1TTime::getMonth () const` [virtual]

This method returns the month number component of the time value.

The number of January is 1, February 2, ... up to December 12. You may also use enumerated values for decoded months: `ASN1TTime::January`, `ASN1TTime::February`, etc. Also short aliases for months can be used: `ASN1TTime::Jan`, `ASN1TTime::Feb`, etc. Note that the return value may differ for different inherited [ASN1TTime](#) classes.

Returns:

Month component (1 - 12) is returned if operation is successful. If the operation fails, a negative value is returned.

3.24.3.3 `virtual int ASN1TTime::getDay () const` [virtual]

This method returns the day of month number component of the time value.

The number of the first day in the month is 1; the number of the last day may be in the interval from 28 to 31. Note that the return value may differ for different inherited [ASN1TTime](#) classes.

Returns:

Day of month component (1 - 31) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.24.3.4 `virtual int ASN1TTime::getHour () const` [virtual]

This method returns the hour component of the time value.

As the ISO 8601 is based on the 24-hour timekeeping system, hours are represented by two-digit values from 00 to 23. Note that the return value may differ from different inherited [ASN1TTime](#) classes.

Returns:

Hour component (0 - 23) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.24.3.5 `virtual int ASN1TTime::getMinute () const` [virtual]

This method returns the minute component of the time value.

Minutes are represented by the two digits from 00 to 59. Note that the return value may differ from different inherited [ASN1TTime](#) classes.

Returns:

Minute component (0 - 59) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.24.3.6 `virtual int ASN1TTime::getSecond () const` [virtual]

This method returns the second component of the time value.

Seconds are represented by two digits from 00 to 59. Note that the return value may differ from different inherited [ASN1TTime](#) classes.

Returns:

Second component (0 - 59) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.24.3.7 `virtual int ASN1TTime::getFraction () const` [virtual]

This method returns the second's decimal component of the time value.

Second's decimal fraction is represented by one or more digits from 0 to 9.

Returns:

Second's decimal fraction component is returned if operation is successful. If the operation fails, a negative value is returned.

Reimplemented in [ASN1TUTCTime](#).

3.24.3.8 `virtual double ASN1TTime::getFractionAsDouble () const` [virtual]

This method returns the second's decimal component of the time value. Second's fraction will be represented as double value more than 0 and less than 1.

Second's decimal fraction is represented by one or more digits from 0 to 9.

Returns:

Second's decimal fraction component is returned if operation is successful. If the operation fails, a negative value is returned.

3.24.3.9 `virtual int ASN1TTime::getFractionStr (char *const pBuf, size_t bufSize) const` [virtual]

This method returns the second's decimal component of the time value. Second's fraction will be represented as string w/o integer part and decimal point.

Returns:

Length of the fraction string returned in pBuf, if operation is successful. If the operation fails, a negative value is returned.

3.24.3.10 `virtual int ASN1TTime::getFractionLen () const` [virtual]

This method returns the number of digits in second's decimal component of the time value.

Returns:

Second's decimal fraction's length is returned if operation is successful. If the operation fails, a negative value is returned.

3.24.3.11 **virtual int ASN1TTime::getDiffHour () const** [virtual]

This method returns the hour component of the difference between the time zone of the object and the Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the return value may differ for different inherited [ASN1TTime](#) classes.

Returns:

The negative or positive hour component of the difference between the time zone of the object and the UTC time (-12 - +12) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.24.3.12 **virtual int ASN1TTime::getDiffMinute () const** [virtual]

This method returns the minute component of the difference between the time zone of the object and the Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the return value may differ for different inherited [ASN1TTime](#) classes.

Returns:

The negative or positive minute component of the difference between the time zone of the object and the UTC time (-59 - +59) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.24.3.13 **virtual int ASN1TTime::getDiff () const** [virtual]

This method returns the difference between the time zone of the object and the Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the return value may differ for different inherited [ASN1TTime](#) classes.

Returns:

The negative or positive minute component of the difference between the time zone of the object and the UTC time (-12*60 - +12*60) is returned if the operation is successful. If the operation fails, a negative value is returned.

3.24.3.14 **virtual OSBOOL ASN1TTime::getUTC () const** [virtual]

This method returns the UTC flag state.

If the UTC flag is TRUE, then the time is a UTC time and symbol Z is added at the end of the time string. Otherwise, it is local time.

Returns:

UTC flag state is returned.

3.24.3.15 `virtual time_t ASN1Time::getTime () const` [virtual]

This method converts the time string to a value of the built-in C type `time_t`.

The value is the number of seconds from January 1, 1970. If the time is represented as UTC time plus or minus a time difference, then the resulting value will be recalculated as local time. For example, if the time string is "19991208120000+0930", then this string will be converted to "19991208213000" and then converted to a `time_t` value. Note that the return value may differ for different inherited [ASN1Time](#) classes.

Returns:

The time value, expressed as a number of seconds from January 1, 1970. If the operation fails, a negative value is returned.

3.24.3.16 `void ASN1Time::setDER (OSBOOL bvalue)` [inline]

This method sets the 'use DER' flag which enforces the DER rules when time strings are constructed or parsed.

3.24.3.17 `virtual int ASN1Time::setUTC (OSBOOL utc)` [virtual]

This method sets the UTC flag state.

If the UTC flag is TRUE, then the time is a UTC time and symbol 'Z' is added to the end of the string. Otherwise, it is a local time.

Parameters:

utc UTC flag state.

Returns:

Completion status of operation:

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

Reimplemented in [ASN1TUTCTime](#).

3.24.3.18 `virtual int ASN1Time::setYear (int year_)` [virtual]

This method sets the year component of the time value.

Note that the action of this method may differ for different inherited [ASN1Time](#) classes.

Parameters:

year_ Year component (full 4 digits).

Returns:

Completion status of operation:

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

Reimplemented in [ASN1TUTCTime](#).

3.24.3.19 **virtual int ASN1TTime::setMonth (int month_)** [virtual]

This method sets the month number component of the time value.

The number of January is 1, February 2, ..., through December 12. You may use enumerated values for months encoding: ASN1TTime::January, ASN1TTime::February, etc. Also you can use short aliases for months: ASN1TTime::Jan, ASN1TTime::Feb, etc. Note that the action of this method may differ for different inherited [ASN1TTime](#) classes.

Parameters:

month_ Month component (1 - 12).

Returns:

Completion status of operation:

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

3.24.3.20 **virtual int ASN1TTime::setDay (int day_)** [virtual]

This method sets the day of month number component of the time value.

The number of the first day in the month is 1; the number of the last day in the month may be in the interval from 28 to 31. Note that the action of this method may differ for different inherited [ASN1TTime](#) classes.

Parameters:

day_ Day of month component (1 - 31).

Returns:

Completion status of operation:

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

3.24.3.21 **virtual int ASN1TTime::setHour (int hour_)** [virtual]

This method sets the hour component of the time value.

As the ISO 8601 is based on the 24-hour timekeeping system, hours are represented by two digits from 00 to 23. Note that the action of this method may differ for different inherited [ASN1TTime](#) classes.

Parameters:

hour_ Hour component (0 - 23).

Returns:

Completion status of operation:

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

3.24.3.22 virtual int ASN1Time::setMinute (int *minute_*) [virtual]

This method sets the minute component of the time value.

Minutes are represented by two digits from 00 to 59. Note that the action of this method may differ for different inherited [ASN1Time](#) classes.

Parameters:

minute_ Minute component (0 - 59).

Returns:

Completion status of operation:

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

3.24.3.23 virtual int ASN1Time::setSecond (int *second_*) [virtual]

This method sets the second component of the time value.

Seconds are represented by two digits from 00 to 59. Note that the action of this method may differ from different inherited [ASN1Time](#) classes.

Parameters:

second_ Second component (0 - 59).

Returns:

Completion status of operation:

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

3.24.3.24 virtual int ASN1Time::setFraction (int *fraction*, int *fracLen* = -1) [virtual]

This method sets the second's decimal fraction component of the time value.

Second's decimal fraction is represented by one or more digits from 0 to 9. Note that the action of this method may differ for different inherited [ASN1Time](#) classes.

Parameters:

fraction Second's decimal fraction component (0 - 9).

fracLen Optional parameter specifies number of digits in second's fraction.

Returns:

Completion status of operation:

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

Reimplemented in [ASN1TUTCTime](#).

3.24.3.25 `virtual int ASN1TTime::setFraction (double frac, int fracLen)` [virtual]

This method sets the second's decimal fraction component of the time value. Double value must be greater or equal 0 and less than 1.

Parameters:

- frac* Second's decimal fraction component.
- fracLen* Specifies number of digits in second's fraction.

Returns:

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

3.24.3.26 `virtual int ASN1TTime::setFraction (char const *frac)` [virtual]

This method sets the second's decimal fraction component of the time value. Double value must be greater or equal 0 and less than 1.

Parameters:

- frac* Second's decimal fraction component.

Returns:

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

3.24.3.27 `virtual int ASN1TTime::setTime (time_t time, OSBOOL diffTime)` [pure virtual]

This converts the value of the C built-in type `time_t` to a time string.

The value is the number of seconds from January 1, 1970. Note that the action of this method may differ for different inherited [ASN1TTime](#) Classes.

Parameters:

- time* The time value, expressed as a number of seconds from January 1, 1970.
- diffTime* TRUE means the difference between local time and UTC time will be calculated; in other case, only local time will be stored.

Returns:

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

Implemented in [ASNITGeneralizedTime](#), and [ASNITUTCTime](#).

3.24.3.28 **virtual int ASN1TTime::setDiffHour (int *dhour*)** [virtual]

This method sets the hour component of the difference between the time zone of the object and the Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the action of this method may differ from different inherited [ASN1TTime](#) classes.

Parameters:

dhour The negative or positive hour component of the difference between the time zone of the object and the UTC time (-12 - +12) is returned if the operation is successful. If the operation fails, a negative value is returned.

Returns:

Completion status of operation:

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

3.24.3.29 **virtual int ASN1TTime::setDiff (int *dhour*, int *dminute*)** [virtual]

This method sets the hours and the minute components of the difference between the time zone of the object and Coordinated Universal Time (UTC).

The UTC time is the sum of the local time and a positive or negative time difference. Note that the action of this method may differ for different inherited [ASN1TTime](#) classes.

Parameters:

dhour The negative or positive hour component of the difference between the time zone of the object and the UTC time (-12 - +12).

dminute The negative or positive minute component of the difference between the time zone of the object and the UTC time (-59 - +59).

Returns:

Completion status of operation:

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

3.24.3.30 **virtual int ASN1TTime::setDiff (int *inMinutes*)** [virtual]

This method sets the difference between the time zone of the object and Coordinated Universal Time (UTC), in minutes.

The UTC time is the sum of the local time and a positive or negative time difference. Note that the action of this method may differ for different inherited [ASN1TTime](#) classes.

Parameters:

inMinutes The negative or positive difference, in minutes, between the time zone of the object and the UTC time (-12*60 - +12*60) is returned if the operation is successful.

Returns:

Completion status of operation:

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

3.24.3.31 virtual int ASN1TTime::parseString (const char *string) [pure virtual]

This method parses the given time string.

The string is expected to be in the ASN.1 value notation format for the given ASN.1 time string type. Note that the action of this method may differ for different inherited [ASN1TTime](#) classes.

Parameters:

string The time string value to be parsed.

Returns:

Completion status of operation:

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

Implemented in [ASN1TGeneralizedTime](#), and [ASN1TUTCTime](#).

3.24.3.32 virtual void ASN1TTime::clear () [virtual]

This method clears the time object.

Note the action of this method may differ for different inherited [ASN1TTime](#) classes.

Reimplemented in [ASN1TUTCTime](#).

3.24.3.33 virtual int ASN1TTime::compileString (char *pbuf, size_t bufsize) const [pure virtual]

Compiles new time string according X.680 and ISO 8601. Returns 0, if succeed, or error code, if error.

Parameters:

pbuf A pointer to destination buffer.

bufsize A size of destination buffer.

Returns:

Completion status of operation:

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

Implemented in [ASN1TGeneralizedTime](#), and [ASN1TUTCTime](#).

3.24.3.34 virtual int ASN1TTime::equals (const ASN1TTime &) const [virtual]

This method compares times.

3.24.3.35 `const char* ASN1TTime::toString (char * pbuf, size_t bufsize) const`

Get a printable ASCII string of the time value into the specified buffer.

Parameters:

pbuf Pointer to a destination buffer.

bufsize Size of destination buffer.

Returns:

Compiled time string. NULL, if error occurs.

3.24.3.36 `const char* ASN1TTime::toString (OSCTXT * pctxt) const`

Get a printable ASCII string of the time value.

Parameters:

pctxt Pointer to a context structure.

Returns:

Compiled time string. NULL, if error occurs.

3.24.3.37 `const char* ASN1TTime::toString () const`

Get a printable ASCII string of the time value. Memory will be allocated using `new[]` operator. User is responsible to free it using `delete[]`.

Returns:

Compiled time string. NULL, if error occurs.

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

- [ASN1TTime.h](#)

3.25 ASN1TUniversalString Struct Reference

```
#include <asn1CppType.h>
```

3.25.1 Detailed Description

UniversalString. This is the base class for generated C++ data type classes for UniversalString values.

Public Member Functions

- [ASN1TUniversalString \(\)](#)

3.25.2 Constructor & Destructor Documentation

3.25.2.1 ASN1TUniversalString::ASN1TUniversalString () [inline]

The default constructor creates an empty UniversalString value.

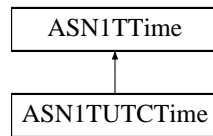
The documentation for this struct was generated from the following file:

- [asn1CppType.h](#)

3.26 ASN1TUTCTime Class Reference

```
#include <ASN1TTime.h>
```

Inheritance diagram for ASN1TUTCTime::



3.26.1 Detailed Description

ASN.1 UTCTime utility class. The ASN1TUTCTime class is derived from the [ASN1TTime](#) base class.

Public Member Functions

- [ASN1TUTCTime](#) ()
- [ASN1TUTCTime](#) (const char *timeStr, OSBOOL useDerRules=FALSE)
- [ASN1TUTCTime](#) (OSBOOL useDerRules)
- [ASN1TUTCTime](#) (const [ASN1TUTCTime](#) &original)
- int [setYear](#) (int year_)
- int [setTime](#) (time_t time, OSBOOL diffTime)
- int [setUTC](#) (OSBOOL utc)
- void [clear](#) ()
- int [compileString](#) (char *pbuf, size_t bufsize) const
- int [parseString](#) (const char *string)
- const [ASN1TUTCTime](#) & **operator=** (const [ASN1TUTCTime](#) &tm)

Protected Member Functions

- int [getFraction](#) () const
- int [setFraction](#) (int fraction, int fracLen=-1)

3.26.2 Constructor & Destructor Documentation

3.26.2.1 ASN1TUTCTime::ASN1TUTCTime ()

A default constructor.

3.26.2.2 ASN1TUTCTime::ASN1TUTCTime (const char * *timeStr*, OSBOOL *useDerRules* = FALSE)

This constructor creates a time object using the specified time string.

Parameters:

timeStr A pointer to the time string to be parsed.

useDerRules Create object using DER rules.

3.26.2.3 ASN1TUTCTime::ASN1TUTCTime (OSBOOL *useDerRules*)

This constructor creates an empty time object.

Parameters:

useDerRules An OSBOOL value.

3.26.2.4 ASN1TUTCTime::ASN1TUTCTime (const ASN1TUTCTime & *original*) [inline]

A copy constructor.

3.26.3 Member Function Documentation

3.26.3.1 int ASN1TUTCTime::setYear (int *year_*) [virtual]

This method sets the year component of the time value.

The year parameter can be either the two last digits of the year (00 - 99) or the full four digits (0 - 9999). Note: the `getYear` method returns the year in the full four digits, independent of the format of the year parameter used in this method.

Parameters:

year_ Year component (full four digits or only last two digits).

Returns:

Completion status of operation:

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

Reimplemented from [ASN1TTime](#).

3.26.3.2 int ASN1TUTCTime::setTime (time_t *time*, OSBOOL *diffTime*) [virtual]

Converts `time_t` to time string.

Parameters:

time time to convert,

diffTime TRUE means the difference between local time and UTC will be calculated; in other case only local time will be stored.

Returns:

Completion status of operation:

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

Implements [ASN1TTime](#).

3.26.3.3 `int ASN1TUTCTime::setUTC (OSBOOL utc) [virtual]`

This method sets the UTC flag state.

If the UTC flag is TRUE, then the time is a UTC time and symbol 'Z' is added to the end of the string. Otherwise, it is a local time.

Parameters:

utc UTC flag state.

Returns:

Completion status of operation:

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

Reimplemented from [ASN1TTime](#).

3.26.3.4 `void ASN1TUTCTime::clear () [virtual]`

Clears out the time object.

Reimplemented from [ASN1TTime](#).

3.26.3.5 `int ASN1TUTCTime::compileString (char * pbuf, size_t bufsize) const [virtual]`

Compiles new time string according X.680 and ISO 8601. Returns 0, if succeed, or error code, if error.

Parameters:

pbuf A pointer to destination buffer.

bufsize A size of destination buffer.

Returns:

Completion status of operation:

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

Implements [ASN1TTime](#).

3.26.3.6 `int ASN1TUTCTime::parseString (const char * string) [virtual]`

Parses the given time string. The string is assumed to be in standard UTC time format.

Parameters:

string UTC time string to be parsed.

Returns:

Completion status of operation:

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

Implements [ASN1TTime](#).

3.26.3.7 `int ASN1TUTCTime::getFraction () const` [protected, virtual]

This method returns the second's decimal component of the time value.

Second's decimal fraction is represented by one or more digits from 0 to 9.

Returns:

Second's decimal fraction component is returned if operation is successful. If the operation fails, a negative value is returned.

Reimplemented from [ASN1TTime](#).

3.26.3.8 `int ASN1TUTCTime::setFraction (int fraction, int fracLen = -1)` [protected, virtual]

This method sets the second's decimal fraction component of the time value.

Second's decimal fraction is represented by one or more digits from 0 to 9. Note that the action of this method may differ for different inherited [ASN1TTime](#) classes.

Parameters:

fraction Second's decimal fraction component (0 - 9).

fracLen Optional parameter specifies number of digits in second's fraction.

Returns:

Completion status of operation:

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

Reimplemented from [ASN1TTime](#).

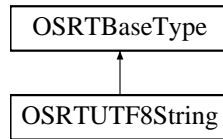
The documentation for this class was generated from the following file:

- [ASN1TTime.h](#)

3.27 OSRTBaseType Class Reference

```
#include <OSRTBaseType.h>
```

Inheritance diagram for OSRTBaseType::



3.27.1 Detailed Description

C++ structured type base class. This is the base class for all generated structured types. It defines special new and delete operators to allow memory for the structure to be allocated using the built-in memory management algorithm.

Public Member Functions

- **OSRTBaseType** (Ownership own)
- void [setOwnMemory](#) (Ownership value=Own)
- void * **operator new** (size_t size)
- void **operator delete** (void *ptr)
- virtual [OSRTBaseType * clone](#) () const

Protected Types

- enum Ownership { Own, NotOwn }

Protected Attributes

- enum OSRTBaseType::Ownership **mOwnMemory**

3.27.2 Member Function Documentation

3.27.2.1 void OSRTBaseType::setOwnMemory (Ownership value = Own) [inline]

This method transfers ownership of the string memory to the class instance. The memory will be deleted when the instance is deleted or goes out of scope.

Parameters:

bvalue - Boolean value.

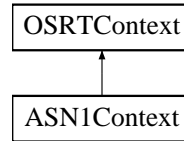
The documentation for this class was generated from the following file:

- [OSRTBaseType.h](#)

3.28 OSRContext Class Reference

```
#include <OSRContext.h>
```

Inheritance diagram for OSRContext::



3.28.1 Detailed Description

Reference counted context class. This keeps track of all encode/decode function variables between function invocations. It is reference counted to allow a message buffer and type class to share access to it.

Public Member Functions

- [OSRContext \(\)](#)
- virtual [~OSRContext \(\)](#)
- OSCTXT * [getPtr \(\)](#)
- const OSCTXT * [getPtr \(\) const](#)
- OSUINT32 [getRefCount \(\)](#)
- int [getStatus \(\) const](#)
- OSBOOL [isInitialized \(\)](#)
- void [_ref \(\)](#)
- void [_unref \(\)](#)
- char * [getErrorInfo \(\)](#)
- char * [getErrorInfo \(size_t *pBufSize\)](#)
- char * [getErrorInfo \(char *pBuf, size_t &bufSize\)](#)
- void * [memAlloc \(size_t numocts\)](#)
- void [memFreeAll \(\)](#)
- void [memFreePtr \(void *ptr\)](#)
- void * [memRealloc \(void *ptr, size_t numocts\)](#)
- void [memReset \(\)](#)
- void [printErrorInfo \(\)](#)
- void [resetErrorInfo \(\)](#)
- OSBOOL [setDiag \(OSBOOL value=TRUE\)](#)
- virtual int [setRunTimeKey \(const OSOCTET *key, size_t keylen\)](#)
- int [setStatus \(int stat\)](#)

Protected Attributes

- OSCTXT [mCtxt](#)
- OSUINT32 [mCount](#)
- OSBOOL [mbInitialized](#)
- int [mStatus](#)

3.28.2 Constructor & Destructor Documentation

3.28.2.1 OSRTContext::OSRTContext ()

The default constructor initializes the mCtxt member variable and sets the reference count variable (mCount) to zero.

3.28.2.2 virtual OSRTContext::~~OSRTContext () [virtual]

The destructor frees all memory held by the context.

3.28.3 Member Function Documentation

3.28.3.1 OSCTXT* OSRTContext::getPtr () [inline]

The getPtr method returns a pointer to the mCtxt member variable. A user can use this function to get the the context pointer variable for use in a C runtime function call.

3.28.3.2 OSUINT32 OSRTContext::getRefCount ()

The getRefCount method returns the current reference count.

3.28.3.3 int OSRTContext::getStatus () const [inline]

The getStatus method returns the runtime status code value.

Returns:

Runtime status code:

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

3.28.3.4 OSBOOL OSRTContext::isInitialized () [inline]

Returns TRUE, if initialized correctly, FALSE otherwise.

Returns:

TRUE, if initialized correctly, FALSE otherwise.

3.28.3.5 void OSRTContext::_ref ()

The _ref method increases the reference count by one.

3.28.3.6 void OSRTContext::_unref ()

The _unref method decreases the reference count by one.

3.28.3.7 char* OSRTContext::getErrorInfo ()

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

Returns:

A pointer to a newly allocated buffer with error text.

3.28.3.8 char* OSRTContext::getErrorInfo (size_t * pBufSize)

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

Parameters:

pBufSize A pointer to buffer size. It will receive the size of allocated dynamic buffer.

Returns:

A pointer to a newly allocated buffer with error text.

3.28.3.9 char* OSRTContext::getErrorInfo (char * pBuf, size_t & bufSize)

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

Parameters:

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

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

Returns:

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

3.28.3.10 void* OSRTContext::memAlloc (size_t numocts) [inline]

The memAlloc method allocates memory using the C runtime memory management functions. The memory is tracked in the underlying context structure. When both this OSXSDGlobalElement derived control class object and the message buffer object are destroyed, this memory will be freed.

Parameters:

numocts - Number of bytes of memory to allocate

3.28.3.11 void OSRTContext::memFreeAll () [inline]

The memFreeAll method will free all memory currently tracked within the context. This includes all memory allocated with the memAlloc method as well as any memory allocated using the C `rtxMemAlloc` function with the context returned by the `getCtxtPtr` method.

3.28.3.12 void OSRTContext::memFreePtr (void * ptr) [inline]

The memFreePtr method frees the memory at a specific location. This memory must have been allocated using the memAlloc method described earlier.

Parameters:

ptr - Pointer to a block of memory allocated with memAlloc

3.28.3.13 void* OSRTContext::memRealloc (void * ptr, size_t numocts) [inline]

The memRealloc method reallocates memory using the C runtime memory management functions.

Parameters:

ptr - Original pointer containing dynamic memory to be resized.

numocts - Number of bytes of memory to allocate

Returns:

Reallocated memory pointer

3.28.3.14 void OSRTContext::memReset () [inline]

The memReset method resets dynamic memory using the C runtime memory management functions.

3.28.3.15 void OSRTContext::printErrorInfo () [inline]

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

3.28.3.16 void OSRTContext::resetErrorInfo () [inline]

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

3.28.3.17 OSBOOL OSRTContext::setDiag (OSBOOL value = TRUE) [inline]

The setDiag method will turn diagnostic tracing on or off.

Parameters:

value - Boolean value (default = TRUE = on)

Returns:

- Previous state of the diagnostics enabled boolean

3.28.3.18 virtual int OSRTContext::setRunTimeKey (const OSOCTET * *key*, size_t *keylen*) [virtual]

This method sets run-time key to the context. This method does nothing for unlimited redistribution libraries.

Parameters:

key - array of octets with the key

keylen - number of octets in key array.

Returns:

Completion status of operation:

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

Reimplemented in [ASN1Context](#).

3.28.3.19 int OSRTContext::setStatus (int *stat*) [inline]

This method sets error status in the context.

Parameters:

stat Status value.

Returns:

Error status value being set.

3.28.4 Member Data Documentation

3.28.4.1 OSCTXT [OSRTContext::mCtxt](#) [protected]

The mCtxt member variable is a standard C runtime context variable used in most C runtime function calls.

3.28.4.2 OSUINT32 [OSRTContext::mCount](#) [protected]

The mCount member variable holds the reference count of this context.

3.28.4.3 OSBOOL [OSRTContext::mbInitialized](#) [protected]

TRUE, if initialized correctly, FALSE otherwise

3.28.4.4 int [OSRTContext::mStatus](#) [protected]

The mStatus variable holds the return status from C run-time function calls. The getStatus method will either return this status or the last status on the context error list.

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

- [OSRTContext.h](#)

3.29 OSRTCtxtPtr Class Reference

```
#include <OSRTCtxtPtr.h>
```

3.29.1 Detailed Description

Context reference counted pointer class. This class allows a context object to automatically be released when its reference count goes to zero. It is very similar to the standard C++ library `auto_ptr` smart pointer class but only works with an `OSRTCtxt` object.

Public Member Functions

- `OSRTCtxtPtr (OSRTCtxt *rf=0)`
- `OSRTCtxtPtr (const OSRTCtxtPtr &o)`
- `virtual ~OSRTCtxtPtr ()`
- `OSRTCtxtPtr & operator= (const OSRTCtxtPtr &rf)`
- `OSRTCtxtPtr & operator= (OSRTCtxt *rf)`
- `operator OSRTCtxt * ()`
- `operator const OSRTCtxt * () const`
- `OSRTCtxt * operator → ()`
- `const OSRTCtxt * operator → () const`
- `OSBOOL operator== (const OSRTCtxt *o) const`
- `OSBOOL isNull () const`
- `OSCTXT * getCtxtPtr ()`

Protected Attributes

- `OSRTCtxt * mPointer`

3.29.2 Constructor & Destructor Documentation

3.29.2.1 OSRTCtxtPtr::OSRTCtxtPtr (OSRTCtxt * rf = 0) [inline]

This constructor set the internal context pointer to the given value and, if it is non-zero, increases the reference count by one.

Parameters:

rf - Pointer to `OSRTCtxt` object

3.29.2.2 OSRTCtxtPtr::OSRTCtxtPtr (const OSRTCtxtPtr & o) [inline]

The copy constructor copies the pointer from the source pointer object and, if it is non-zero, increases the reference count by one.

Parameters:

o - Reference to `OSRTCtxtPtr` object to be copied

3.29.2.3 virtual OSRTCtxtPtr::~~OSRTCtxtPtr () [inline, virtual]

The destructor decrements the reference counter to the internal context pointer object. The context object will delete itself if its reference count goes to zero.

3.29.3 Member Function Documentation

3.29.3.1 OSRTCtxtPtr& OSRTCtxtPtr::operator= (const OSRTCtxtPtr & rf) [inline]

This assignment operator assigns this [OSRTCtxtPtr](#) to another. The reference count of the context object managed by this object is first decremented. Then the new pointer is assigned and that object's reference count is incremented.

Parameters:

rf - Pointer to [OSRTCtxtPtr](#) smart-pointer object

3.29.3.2 OSRTCtxtPtr& OSRTCtxtPtr::operator= (OSRTContext * rf) [inline]

This assignment operator assigns does a direct assignment of an [OSRTContext](#) object to this [OSRTCtxtPtr](#) object.

3.29.3.3 OSRTCtxtPtr::operator OSRTContext * () [inline]

The 'OSRTContext*' operator returns the context object pointer.

3.29.3.4 OSRTContext* OSRTCtxtPtr::operator → () [inline]

The '->' operator returns the context object pointer.

3.29.3.5 OSBOOL OSRTCtxtPtr::operator== (const OSRTContext * o) const [inline]

The '==' operator compares two [OSRTContext](#) pointer values.

3.29.3.6 OSBOOL OSRTCtxtPtr::isNull () const [inline]

The isNull method returns TRUE if the underlying context pointer is NULL.

3.29.3.7 OSCTXT* OSRTCtxtPtr::getCtxtPtr () [inline]

This method returns the standard context pointer used in C function calls.

3.29.4 Member Data Documentation

3.29.4.1 OSRTContext* OSRTCtxtPtr::mPointer [protected]

The mPointer member variable is a pointer to a reference-counted ASN.1 context wrapper class object.

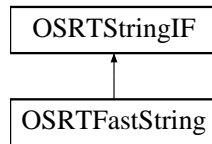
The documentation for this class was generated from the following file:

- [OSRTContext.h](#)

3.30 OSRTFastString Class Reference

```
#include <OSRTFastString.h>
```

Inheritance diagram for OSRTFastString::



3.30.1 Detailed Description

C++ fast string class definition. This can be used to hold standard ASCII or UTF-8 strings. This string class implementations directly assigns any assigned pointers to internal member variables. It does no memory management.

Public Member Functions

- [OSRTFastString](#) ()
- [OSRTFastString](#) (const char *strval)
- [OSRTFastString](#) (const OSUTF8CHAR *strval)
- [OSRTFastString](#) (const [OSRTFastString](#) &str)
- virtual [~OSRTFastString](#) ()
- virtual [OSRTStringIF](#) * [clone](#) ()
- virtual const char * [getValue](#) () const
- virtual const OSUTF8CHAR * [getUTF8Value](#) () const
- virtual void [print](#) (const char *name)
- virtual void [setValue](#) (const char *str)
- virtual void [setValue](#) (const OSUTF8CHAR *str)
- [OSRTFastString](#) & [operator=](#) (const [OSRTFastString](#) &original)

Protected Attributes

- const OSUTF8CHAR * [mValue](#)

3.30.2 Constructor & Destructor Documentation

3.30.2.1 OSRTFastString::OSRTFastString ()

The default constructor sets the internal string member variable pointer to null.

3.30.2.2 OSRTFastString::OSRTFastString (const char * strval)

This constructor initializes the string to contain the given standard ASCII string value.

Parameters:

strval - Null-terminated C string value

3.30.2.3 OSRTFastString::OSRTFastString (const OSUTF8CHAR * *strval*)

This constructor initializes the string to contain the given UTF-8 string value.

Parameters:

strval - Null-terminated C string value

3.30.2.4 OSRTFastString::OSRTFastString (const OSRTFastString & *str*)

Copy constructor. String data is not copied; the pointer is simply assigned to the target class member variable.

Parameters:

str - C++ string object to be copied.

3.30.2.5 virtual OSRTFastString::~OSRTFastString () [virtual]

The destructor does nothing.

3.30.3 Member Function Documentation

3.30.3.1 virtual OSRTStringIF* OSRTFastString::clone () [inline, virtual]

This method creates a copy of the given string object.

Implements [OSRTStringIF](#).

3.30.3.2 virtual const char* OSRTFastString::getValue () const [inline, virtual]

This method returns the pointer to UTF-8 null terminated string as a standard ASCII string.

Implements [OSRTStringIF](#).

3.30.3.3 virtual const OSUTF8CHAR* OSRTFastString::getUTF8Value () const [inline, virtual]

This method returns the pointer to UTF-8 null terminated string as a UTF-8 string.

Implements [OSRTStringIF](#).

3.30.3.4 virtual void OSRTFastString::print (const char * *name*) [inline, virtual]

This method prints the string value to standard output.

Parameters:

name - Name of generated string variable.

Implements [OSRTStringIF](#).

3.30.3.5 virtual void OSRTFastString::setValue (const char * *str*) [virtual]

This method sets the string value to the given string.

Parameters:

str - C null-terminated string.

Implements [OSRTStringIF](#).

3.30.3.6 virtual void OSRTFastString::setValue (const OSUTF8CHAR * *str*) [virtual]

This method sets the string value to the given UTF-8 string value.

Parameters:

utf8str - C null-terminated UTF-8 string.

Implements [OSRTStringIF](#).

3.30.3.7 OSRTFastString& OSRTFastString::operator= (const OSRTFastString & *original*)

Assignment operator.

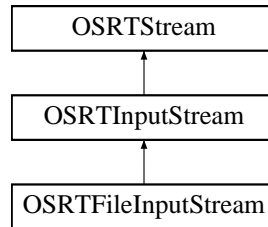
The documentation for this class was generated from the following file:

- [OSRTFastString.h](#)

3.31 OSRTFileInputStream Class Reference

```
#include <OSRTFileInputStream.h>
```

Inheritance diagram for OSRTFileInputStream::



3.31.1 Detailed Description

Generic file input stream. This class opens an existing file for input in binary mode and reads data from it.

Public Member Functions

- [OSRTFileInputStream](#) (const char *pFilename)
- [OSRTFileInputStream](#) ([OSRTContext](#) *pContext, const char *pFilename)
- [OSRTFileInputStream](#) (FILE *file)
- [OSRTFileInputStream](#) ([OSRTContext](#) *pContext, FILE *file)

3.31.2 Constructor & Destructor Documentation

3.31.2.1 OSRTFileInputStream::OSRTFileInputStream (const char * pFilename)

Creates and initializes a file input stream using the name of file.

Parameters:

pFilename Name of file.

See also:

rtxStreamFileOpen

3.31.2.2 OSRTFileInputStream::OSRTFileInputStream ([OSRTContext](#) * pContext, const char * pFilename)

Creates and initializes a file input stream using the name of file.

Parameters:

pContext Pointer to a context to use.

pFilename Name of file.

See also:

rtxStreamFileOpen

3.31.2.3 OSRTFileInputStream::OSRTFileInputStream (FILE * *file*)

Initializes the file input stream using the opened FILE structure descriptor.

Parameters:

file Pointer to FILE structure.

See also:

rtxStreamFileAttach

3.31.2.4 OSRTFileInputStream::OSRTFileInputStream (OSRTContext * *pContext*, FILE * *file*)

Initializes the file input stream using the opened FILE structure descriptor.

Parameters:

pContext Pointer to a context to use.

file Pointer to FILE structure.

See also:

rtxStreamFileAttach

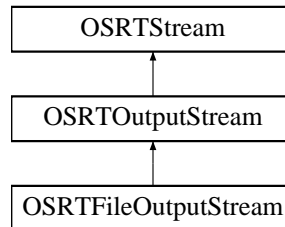
The documentation for this class was generated from the following file:

- [OSRTFileInputStream.h](#)

3.32 OSRTFileOutputStream Class Reference

```
#include <OSRTFileOutputStream.h>
```

Inheritance diagram for OSRTFileOutputStream::



3.32.1 Detailed Description

Generic file output stream. This class opens an existing file for output in binary mode and reads data from it.

Public Member Functions

- [OSRTFileOutputStream](#) (const char *pFilename)
- [OSRTFileOutputStream](#) (OSRTContext *pContext, const char *pFilename)
- [OSRTFileOutputStream](#) (FILE *file)
- [OSRTFileOutputStream](#) (OSRTContext *pContext, FILE *file)

3.32.2 Constructor & Destructor Documentation

3.32.2.1 OSRTFileOutputStream::OSRTFileOutputStream (const char * pFilename)

Creates and initializes a file output stream using the name of file.

Parameters:

pFilename Name of file.

Exceptions:

OSStreamException Stream create or initialize failed.

See also:

rtxStreamFileOpen

3.32.2.2 OSRTFileOutputStream::OSRTFileOutputStream (OSRTContext * pContext, const char * pFilename)

Creates and initializes a file output stream using the name of file.

Parameters:

pContext Pointer to a context to use.

pFilename Name of file.

Exceptions:

OSStreamException Stream create or initialize failed.

See also:

rtxStreamFileOpen

3.32.2.3 OSRTFileOutputStream::OSRTFileOutputStream (FILE * *file*)

Initializes the file output stream using the opened FILE structure descriptor.

Parameters:

file Pointer to FILE structure.

Exceptions:

OSStreamException Stream create or initialize failed.

See also:

rtxStreamFileAttach

3.32.2.4 OSRTFileOutputStream::OSRTFileOutputStream (OSRTContext * *pContext*, FILE * *file*)

Initializes the file output stream using the opened FILE structure descriptor.

Parameters:

pContext Pointer to a context to use.

file Pointer to FILE structure.

Exceptions:

OSStreamException Stream create or initialize failed.

See also:

rtxStreamFileAttach

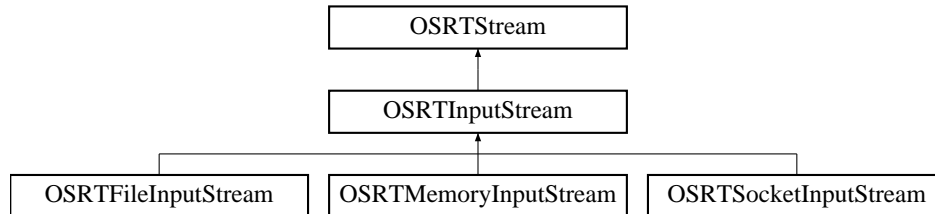
The documentation for this class was generated from the following file:

- [OSRTFileOutputStream.h](#)

3.33 OSRTInputStream Class Reference

```
#include <OSRTInputStream.h>
```

Inheritance diagram for OSRTInputStream::



3.33.1 Detailed Description

This is the base class for input streams. These streams are buffered (I/O is stored in memory prior to being written) to provide higher performance.

Public Member Functions

- [OSRTInputStream \(\)](#)
- [OSRTInputStream \(OSRTContext *mpContext, OSBOOL attachStream=FALSE\)](#)
- [virtual ~OSRTInputStream \(\)](#)
- [virtual int close \(\)](#)
- [virtual size_t currentPos \(\)](#)
- [virtual int flush \(\)](#)
- [virtual OSRTCtxPtr getContext \(\)](#)
- [virtual OSCTXT * getCtxPtr \(\)](#)
- [virtual char * getErrorInfo \(\)](#)
- [virtual char * getErrorInfo \(char *pBuf, size_t &bufSize\)](#)
- [virtual int getStatus \(\) const](#)
- [virtual OSBOOL isOpened \(\)](#)
- [virtual OSBOOL markSupported \(\)](#)
- [virtual int mark \(size_t readAheadLimit\)](#)
- [void printErrorInfo \(\)](#)
- [void resetErrorInfo \(\)](#)
- [virtual long read \(OSOCKET *pDestBuf, size_t maxToRead\)](#)
- [virtual long readBlocking \(OSOCKET *pDestBuf, size_t toReadBytes\)](#)
- [virtual int reset \(\)](#)
- [virtual int skip \(size_t n\)](#)

3.33.2 Constructor & Destructor Documentation

3.33.2.1 OSRTInputStream::OSRTInputStream ()

The default constructor. It initializes a buffered stream. A buffered stream maintains data in memory before reading or writing to the device. This generally provides better performance than an unbuffered stream.

Exceptions:

OSRTStreamException Stream create or initialize failed.

3.33.2.2 virtual OSRTInputStream::~~OSRTInputStream () [virtual]

Virtual destructor. Closes the stream if it was opened.

3.33.3 Member Function Documentation**3.33.3.1 virtual int OSRTInputStream::close () [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, rtxStreamBufClose

Reimplemented from [OSRTStream](#).

3.33.3.2 virtual size_t OSRTInputStream::currentPos () [virtual]

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

Returns:

The number of octets already read from the stream.

3.33.3.3 virtual int OSRTInputStream::flush () [virtual]

Flushes the buffered data to the stream.

Returns:

Completion status of operation:

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

See also:

rtxStreamFlush, rtxStreamBufFlush

Reimplemented from [OSRTStream](#).

3.33.3.4 virtual [OSRTCtxtPtr](#) OSRTInputStream::getContext () [inline, virtual]

This method returns a pointer to the underlying [OSRTContext](#) object.

Returns:

A reference-counted pointer to an [OSRTContext](#) object. The [OSRTContext](#) object will not be released until all referenced-counted pointer variables go out of scope. This allows safe sharing of the context between different run-time classes.

Reimplemented from [OSRTStream](#).

3.33.3.5 virtual [OSCTXT*](#) OSRTInputStream::getCtxtPtr () [inline, virtual]

This method returns a pointer to the underlying OSCTXT object. This is the structure used in calls to low-level C encode/decode functions.

Returns:

Pointer to a context (OSCTXT) structure.

Reimplemented from [OSRTStream](#).

3.33.3.6 virtual [char*](#) OSRTInputStream::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.

Reimplemented from [OSRTStream](#).

3.33.3.7 virtual [char*](#) OSRTInputStream::getErrorInfo ([char * pBuf](#), [size_t & bufSize](#)) [inline, virtual]

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

Parameters:

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

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

Returns:

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

Reimplemented from [OSRTStream](#).

3.33.3.8 `virtual int OSRTInputStream::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.

Returns:

Runtime status code:

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

Reimplemented from [OSRTStream](#).

3.33.3.9 `virtual OSBOOL OSRTInputStream::isOpened ()` [virtual]

Checks, is the stream opened or not.

Returns:

s TRUE, if the stream is opened, FALSE otherwise.

See also:

`rtxStreamIsOpened`

Reimplemented from [OSRTStream](#).

3.33.3.10 `virtual OSBOOL OSRTInputStream::markSupported ()` [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`

3.33.3.11 `virtual int OSRTInputStream::mark (size_t readAheadLimit)` [virtual]

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

Parameters:

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

Returns:

Completion status of operation:

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

See also:

rtxStreamMark, rtxStreamReset

3.33.3.12 void OSRTInputStream::printErrorInfo () [inline]

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

Reimplemented from [OSRTStream](#).

3.33.3.13 void OSRTInputStream::resetErrorInfo () [inline]

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

Reimplemented from [OSRTStream](#).

3.33.3.14 virtual long OSRTInputStream::read (OSOCKET * *pDestBuf*, size_t *maxToRead*) [virtual]

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

Parameters:

pDestBuf Pointer to a buffer to receive a data.

maxToRead Size of the buffer.

See also:

rtxStreamRead

3.33.3.15 virtual long OSRTInputStream::readBlocking (OSOCKET * *pDestBuf*, size_t *toReadBytes*) [virtual]

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

Parameters:

pDestBuf Pointer to a buffer to receive a data.

toReadBytes Number of bytes to be read.

See also:

rtxStreamRead

3.33.3.16 `virtual int OSRTInputStream::reset ()` [virtual]

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 = success,
- negative return value is error.

See also:

`rtxStreamMark`, `rtxStreamReset`

3.33.3.17 `virtual int OSRTInputStream::skip (size_t n)` [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`

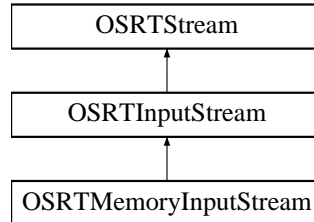
The documentation for this class was generated from the following file:

- [OSRTInputStream.h](#)

3.34 OSRTMemoryInputStream Class Reference

```
#include <OSRTMemoryInputStream.h>
```

Inheritance diagram for OSRTMemoryInputStream::



3.34.1 Detailed Description

Generic memory input stream. This class opens an existing file for input in binary mode and reads data from it.

Public Member Functions

- [OSRTMemoryInputStream](#) (OSOCKET *pMemBuf, size_t bufSize)
- [OSRTMemoryInputStream](#) (OSRTContext *pContext, OSOCKET *pMemBuf, size_t bufSize)

3.34.2 Constructor & Destructor Documentation

3.34.2.1 OSRTMemoryInputStream::OSRTMemoryInputStream (OSOCKET * *pMemBuf*, size_t *bufSize*)

Initializes the memory input stream using the specified memory buffer.

Parameters:

pMemBuf The pointer to the buffer.

bufSize The size of the buffer.

See also:

rtxStreamMemoryAttach

3.34.2.2 OSRTMemoryInputStream::OSRTMemoryInputStream (OSRTContext * *pContext*, OSOCKET * *pMemBuf*, size_t *bufSize*)

Initializes the memory input stream using the specified memory buffer.

Parameters:

pContext Pointer to a context to use.

pMemBuf The pointer to the buffer.

bufSize The size of the buffer.

See also:

`rtxStreamMemoryAttach`

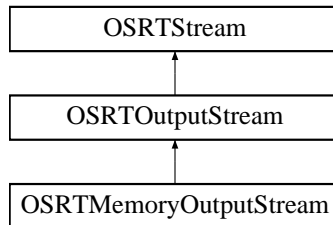
The documentation for this class was generated from the following file:

- [OSRTMemoryInputStream.h](#)

3.35 OSRTMemoryOutputStream Class Reference

```
#include <OSRTMemoryOutputStream.h>
```

Inheritance diagram for OSRTMemoryOutputStream::



3.35.1 Detailed Description

Generic memory output stream. This class opens an existing file for output in binary mode and reads data from it.

Public Member Functions

- [OSRTMemoryOutputStream](#) (OSOCKET *pMemBuf, size_t bufSize)
- [OSRTMemoryOutputStream](#) (OSRTContext *pContext, OSOCKET *pMemBuf, size_t bufSize)

3.35.2 Constructor & Destructor Documentation

3.35.2.1 OSRTMemoryOutputStream::OSRTMemoryOutputStream (OSOCKET * pMemBuf, size_t bufSize)

Initializes the memory output stream using the specified memory buffer.

Parameters:

- pMemBuf* The pointer to the buffer.
bufSize The size of the buffer.

Exceptions:

OSCStreamException stream can't be created or initialized.

See also:

rtxStreamMemoryAttach

3.35.2.2 OSRTMemoryOutputStream::OSRTMemoryOutputStream (OSRTContext * pContext, OSOCKET * pMemBuf, size_t bufSize)

Initializes the memory output stream using the specified memory buffer.

Parameters:

pContext Pointer to a context to use.

pMemBuf The pointer to the buffer.

bufSize The size of the buffer.

Exceptions:

OSCStreamException stream can't be created or initialized.

See also:

rtxStreamMemoryAttach

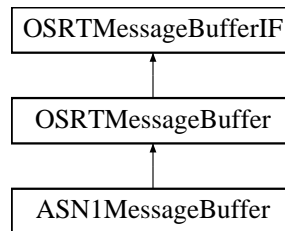
The documentation for this class was generated from the following file:

- [OSRTMemoryOutputStream.h](#)

3.36 OSRTMessageBuffer Class Reference

```
#include <OSRTMsgBuf.h>
```

Inheritance diagram for OSRTMessageBuffer::



3.36.1 Detailed Description

Abstract message buffer base class. This class is used to manage an encode or decode message buffer. For encoding, this is the buffer into which the message is being built. For decoding, it describes a message that was read into memory to be decoded. Further classes are derived from this to handle encoding and decoding of messages for different encoding rules types.

Public Member Functions

- virtual `~OSRTMessageBuffer ()`
- virtual `void * getAppInfo ()`
- virtual `size_t getByteIndex ()`
- virtual `OSRTCtxtPtr getContext ()`
- virtual `OSCTXT * getCtxtPtr ()`
- virtual `char * getErrorInfo ()`
- virtual `char * getErrorInfo (char *pBuf, size_t &bufSize)`
- virtual `OSOCTET * getMsgCopy ()`
- virtual `const OSOCTET * getMsgPtr ()`
- `int getStatus () const`
- virtual `int init ()`
- virtual `int initBuffer (OSOCTET *pMsgBuf, size_t msgBufLen)`
- virtual `void printErrorInfo ()`
- virtual `void resetErrorInfo ()`
- virtual `void setAppInfo (void *pAppInfo)`
- virtual `void setDiag (OSBOOL value=TRUE)`

Protected Member Functions

- `OSRTMessageBuffer (Type bufferType, OSRTCtxt *pContext=0)`

Protected Attributes

- Type `mBufferType`

3.36.2 Constructor & Destructor Documentation

3.36.2.1 OSRTMessageBuffer::OSRTMessageBuffer (Type *bufferType*, OSRTContext * *pContext* = 0) [protected]

The protected constructor creates a new context and sets the buffer class type.

Parameters:

bufferType Type of message buffer that is being created (for example, XMLEncode).

pContext Pointer to a context to use. If NULL, new context will be allocated.

3.36.2.2 virtual OSRTMessageBuffer::~~OSRTMessageBuffer () [inline, virtual]

The virtual destructor does nothing. It is overridden by derived versions of this class.

3.36.3 Member Function Documentation

3.36.3.1 virtual void* OSRTMessageBuffer::getAppInfo () [inline, virtual]

Returns a pointer to application-specific information block

Implements [OSRTMessageBufferIF](#).

Reimplemented in [ASNIMessageBuffer](#).

3.36.3.2 virtual size_t OSRTMessageBuffer::getByteIndex () [inline, virtual]

The getByteIndex method is used to fetch the current byte offset within the current working buffer. For encoding, this is the next location that will be written to. For decoding, this is the next byte the parser will read.

Implements [OSRTMessageBufferIF](#).

3.36.3.3 virtual OSRTCtxPtr OSRTMessageBuffer::getContext () [inline, virtual]

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

3.36.3.4 virtual OSCTXT* OSRTMessageBuffer::getCtxtPtr () [inline, virtual]

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

3.36.3.5 virtual char* OSRTMessageBuffer::getErrorInfo () [inline, virtual]

Returns error text in a dynamic memory buffer. The buffer is allocated using '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.

3.36.3.6 `virtual char* OSRTMessageBuffer::getErrorInfo (char * pBuf, size_t & bufSize)` [inline, virtual]

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

Parameters:

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

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

Returns:

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

3.36.3.7 `virtual OSOCTET* OSRTMessageBuffer::getMsgCopy ()` [inline, virtual]

The getMsgCopy method will return a copy of the encoded message managed by the object.

Implements [OSOCTET* OSRTMessageBufferIF](#).

3.36.3.8 `virtual const OSOCTET* OSRTMessageBuffer::getMsgPtr ()` [inline, virtual]

The getMsgPtr method will return a const pointer to the encoded message managed by the object.

Implements [OSOCTET* OSRTMessageBufferIF](#).

3.36.3.9 `int OSRTMessageBuffer::getStatus () const` [inline]

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.

Returns:

Runtime status code:

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

3.36.3.10 `virtual int OSRTMessageBuffer::init ()` [inline, virtual]

Initializes message buffer.

Returns:

Completion status of operation:

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

Implements [OSOCTET* OSRTMessageBufferIF](#).

3.36.3.11 virtual int OSRTMessageBuffer::initBuffer (OSOCKET * *pMsgBuf*, size_t *msgBufLen*)
[virtual]

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

Implements [OSRTMessageBufferIF](#).

3.36.3.12 virtual void OSRTMessageBuffer::printErrorInfo () [inline, virtual]

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

3.36.3.13 virtual void OSRTMessageBuffer::resetErrorInfo () [inline, virtual]

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

Reimplemented in [ASN1MessageBuffer](#).

3.36.3.14 virtual void OSRTMessageBuffer::setAppInfo (void * *pAppInfo*) [inline, virtual]

Sets the application-specific information block.

Implements [OSRTMessageBufferIF](#).

Reimplemented in [ASN1MessageBuffer](#).

3.36.3.15 virtual void OSRTMessageBuffer::setDiag (OSBOOL *value* = TRUE) [virtual]

The setDiag method will turn diagnostic tracing on or off.

Parameters:

value - Boolean value (default = TRUE = on)

Implements [OSRTMessageBufferIF](#).

3.36.4 Member Data Documentation

3.36.4.1 Type OSRTMessageBuffer::mBufferType [protected]

The mBufferType member variable holds information on the derived message buffer class type (for example, XMLEncode).

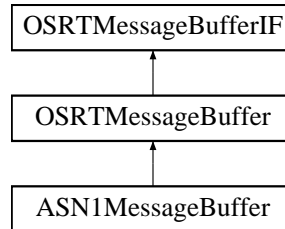
The documentation for this class was generated from the following file:

- [OSRTMsgBuf.h](#)

3.37 OSRTMessageBufferIF Class Reference

```
#include <OSRTMsgBufIF.h>
```

Inheritance diagram for OSRTMessageBufferIF::



3.37.1 Detailed Description

Abstract message buffer or stream interface class. This is the base class for both the in-memory message buffer classes and the run-time stream classes.

Public Types

- enum **Type** {
 BEREncode, **BERDecode**, **PEREncode**, **PERDecode**,
 XEREncode, **XERDecode**, **XMLEncode**, **XMLDecode**,
 Stream }

Public Member Functions

- virtual void * [getAppInfo](#) ()=0
- virtual size_t [getByteIndex](#) ()=0
- virtual OSOCTET * [getMsgCopy](#) ()=0
- virtual const OSOCTET * [getMsgPtr](#) ()=0
- virtual int [init](#) ()=0
- virtual int [initBuffer](#) (OSOCTET *pMsgBuf, size_t msgBufLen)=0
- virtual OSBOOL [isA](#) (int bufferType)=0
- virtual void [setAppInfo](#) (void *pAppInfo)=0
- virtual void [setNamespace](#) (const OSUTF8CHAR *, const OSUTF8CHAR *)
- virtual void [setDiag](#) (OSBOOL value=TRUE)=0

Protected Member Functions

- virtual [~OSRTMessageBufferIF](#) ()

3.37.2 Constructor & Destructor Documentation

3.37.2.1 virtual OSRTMessageBufferIF::~OSRTMessageBufferIF () [inline, protected, virtual]

The virtual destructor does nothing. It is overridden by derived versions of this class.

3.37.3 Member Function Documentation

3.37.3.1 virtual void* OSRTMessageBufferIF::getAppInfo () [pure virtual]

Returns a pointer to application-specific information block

Implemented in [ASN1MessageBuffer](#), and [OSRTMessageBuffer](#).

3.37.3.2 virtual size_t OSRTMessageBufferIF::getByteIndex () [pure virtual]

The getByteIndex method is used to fetch the current byte offset within the current working buffer. For encoding, this is the next location that will be written to. For decoding, this is the next byte the parser will read.

Implemented in [OSRTMessageBuffer](#).

3.37.3.3 virtual OSOCTET* OSRTMessageBufferIF::getMsgCopy () [pure virtual]

The getMsgCopy method will return a copy of the encoded ASN.1 message managed by the object. The memory for the copy is allocated by new [] operator, user is responsible to free it by delete [] operator.

Returns:

The pointer to copied encoded ASN.1 message. NULL, if error occurred.

Implemented in [OSRTMessageBuffer](#).

3.37.3.4 virtual const OSOCTET* OSRTMessageBufferIF::getMsgPtr () [pure virtual]

The getMsgPtr method will return a const pointer to the encoded ASN.1 message managed by the object.

Returns:

The pointer to the encoded ASN.1 message.

Implemented in [OSRTMessageBuffer](#).

3.37.3.5 virtual int OSRTMessageBufferIF::init () [pure virtual]

Initializes message buffer.

Returns:

Completion status of operation:

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

Implemented in [OSRTMessageBuffer](#).

3.37.3.6 `virtual int OSRTMessageBufferIF::initBuffer (OSOCKET * pMsgBuf, size_t msgBufLen)` [pure virtual]

This version of the overloaded `initBuffer` method initializes the message buffer to point at the given null-terminated 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.

Implemented in [OSRTMessageBuffer](#).

3.37.3.7 `virtual OSBOOL OSRTMessageBufferIF::isA (int bufferType)` [pure virtual]

This method checks the type of the message buffer.

Parameters:

bufferType Enumerated identifier specifying a derived class. 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.

3.37.3.8 `virtual void OSRTMessageBufferIF::setAppInfo (void * pAppInfo)` [pure virtual]

Sets the application-specific information block.

Implemented in [ASN1MessageBuffer](#), and [OSRTMessageBuffer](#).

3.37.3.9 `virtual void OSRTMessageBufferIF::setNamespace (const OSUTF8CHAR *, const OSUTF8CHAR *)` [inline, virtual]

Sets the namespace information.

3.37.3.10 `virtual void OSRTMessageBufferIF::setDiag (OSBOOL value = TRUE)` [pure virtual]

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

Parameters:

value - Boolean value (default = TRUE = on)

Implemented in [OSRTMessageBuffer](#).

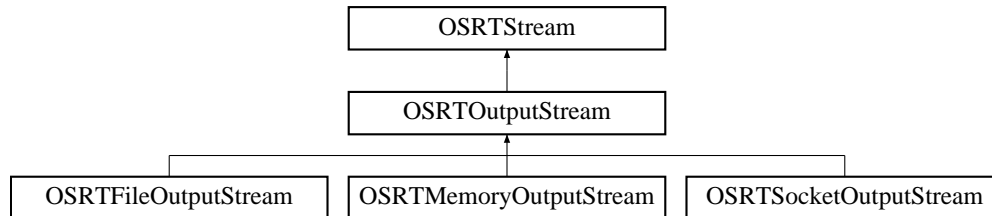
The documentation for this class was generated from the following file:

- [OSRTMsgBufIF.h](#)

3.38 OSRTOutputStream Class Reference

```
#include <OSRTOutputStream.h>
```

Inheritance diagram for OSRTOutputStream::



3.38.1 Detailed Description

The base class definition for operations with output streams. As with the input stream, this implementation is backed by memory buffers to improve I/O performance.

Public Member Functions

- [OSRTOutputStream \(\)](#)
- [OSRTOutputStream \(OSRTContext *mpContext, OSBOOL attachStream=FALSE\)](#)
- [virtual ~OSRTOutputStream \(\)](#)
- [virtual int close \(\)](#)
- [virtual int flush \(\)](#)
- [virtual OSRTCtxPtr getContext \(\)](#)
- [virtual OSCTXT * getCtxPtr \(\)](#)
- [virtual char * getErrorInfo \(\)](#)
- [virtual char * getErrorInfo \(char *pBuf, size_t &bufSize\)](#)
- [virtual int getStatus \(\) const](#)
- [virtual OSBOOL isOpened \(\)](#)
- [void printErrorInfo \(\)](#)
- [void resetErrorInfo \(\)](#)
- [virtual long write \(const OSOCTET *pdata, size_t size\)](#)

3.38.2 Constructor & Destructor Documentation

3.38.2.1 OSRTOutputStream::OSRTOutputStream ()

The default constructor. It initializes a buffered stream. A buffered stream maintains data in memory before reading or writing to the device. This generally provides better performance than an unbuffered stream.

3.38.2.2 virtual OSRTOutputStream::~~OSRTOutputStream () [virtual]

Virtual destructor. Closes the stream if it was opened.

3.38.3 Member Function Documentation

3.38.3.1 `virtual int OSRTOutputStream::close ()` [virtual]

Closes the output 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`, `rtxStreamBufClose`

Reimplemented from [OSRTStream](#).

3.38.3.2 `virtual int OSRTOutputStream::flush ()` [virtual]

Flushes the buffered data to the stream.

Returns:

Completion status of operation:

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

See also:

`rtxStreamFlush`, `rtxStreamBufFlush`

Reimplemented from [OSRTStream](#).

3.38.3.3 `virtual OSRTCtxtPtr OSRTOutputStream::getContext ()` [inline, virtual]

This method returns a pointer to the underlying [OSRTContext](#) object.

Returns:

A reference-counted pointer to an [OSRTContext](#) object. The [OSRTContext](#) object will not be released until all referenced-counted pointer variables go out of scope. This allows safe sharing of the context between different run-time classes.

Reimplemented from [OSRTStream](#).

3.38.3.4 `virtual OSCTXT* OSRTOutputStream::getCtxtPtr ()` [inline, virtual]

This method returns a pointer to the underlying OSCTXT object. This is the structure used in calls to low-level C encode/decode functions.

Returns:

Pointer to a context (OSCTXT) structure.

Reimplemented from [OSRTStream](#).

3.38.3.5 `virtual char* OSRTOutputStream::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.

Reimplemented from [OSRTStream](#).

3.38.3.6 `virtual char* OSRTOutputStream::getErrorInfo (char * pBuf, size_t & bufSize)` [inline, virtual]

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

Parameters:

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

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

Returns:

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

Reimplemented from [OSRTStream](#).

3.38.3.7 `virtual int OSRTOutputStream::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.

Returns:

Runtime status code:

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

Reimplemented from [OSRTStream](#).

3.38.3.8 `virtual OSBOOL OSRTOutputStream::isOpened ()` [virtual]

Checks, is the stream opened or not.

Returns:

s TRUE, if the stream is opened, FALSE otherwise.

See also:

rtxStreamIsOpened

Reimplemented from [OSRTStream](#).

3.38.3.9 void OSRTOutputStream::printErrorInfo () [inline]

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

Reimplemented from [OSRTStream](#).

3.38.3.10 void OSRTOutputStream::resetErrorInfo () [inline]

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

Reimplemented from [OSRTStream](#).

3.38.3.11 virtual long OSRTOutputStream::write (const OSOCTET * *pdata*, size_t *size*) [virtual]

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

Parameters:

pdata The 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:

- [OSRTOutputStream.h](#)

3.39 OSRTSocket Class Reference

```
#include <OSRTSocket.h>
```

3.39.1 Detailed Description

Wrapper class for TCP/IP or UDP sockets.

Public Member Functions

- [OSRTSocket](#) (bool udp=false)
- [OSRTSocket](#) (OSRTSOCKET socket, OSBOOL ownership=FALSE)
- [OSRTSocket](#) (const [OSRTSocket](#) &socket)
- [~OSRTSocket](#) ()
- [OSRTSocket * accept](#) (OSIPADDR *destIP=0, int *port=0)
- int [bind](#) (OSIPADDR addr, int port)
- int [bind](#) (const char *pAddrStr, int port)
- int [bind](#) (int port)
- int [blockingRead](#) (OSOCKET *pbuf, size_t readBytes)
- int [close](#) ()
- int [connect](#) (const char *host, int port)
- OSBOOL [getOwnership](#) ()
- OSRTSOCKET [getSocket](#) () const
- int [getStatus](#) ()
- int [listen](#) (int maxConnections)
- int [recv](#) (OSOCKET *pbuf, OSUINT32 bufsize)
- void [setOwnership](#) (OSBOOL ownership)
- int [send](#) (const OSOCKET *pdata, OSUINT32 size)

Static Public Member Functions

- static const char * [addrToString](#) (OSIPADDR ipAddr, char *pAddrStr, int bufsize)
- static OSIPADDR [stringToAddr](#) (const char *pAddrStr)

Protected Member Functions

- OSBOOL [isInitialized](#) ()

Protected Attributes

- OSRTSOCKET [mSocket](#)
- int [mInitStatus](#)
- int [mStatus](#)
- OSBOOL [mOwner](#)
- OSBOOL [mUDP](#)

3.39.2 Constructor & Destructor Documentation

3.39.2.1 OSRSocket::OSRSocket (bool *udp* = false)

This is the default constructor. It initializes all internal members with default values and creates a new socket structure. Use the `getStatus()` method to determine if an error has occurred during the initialization.

Parameters:

udp A flag that specifies whether the socket should be a UDP socket instead of a TCP socket; the default value is FALSE.

3.39.2.2 OSRSocket::OSRSocket (OSR_SOCKET *socket*, OS_BOOL *ownership* = FALSE)

This constructor initializes an instance by using an existing socket.

Parameters:

socket An existing socket handle.

ownership Boolean flag that specifies who is the owner of the socket. If it is TRUE then the socket will be destroyed in the destructor. Otherwise, the user is responsible to close and destroy the socket.

3.39.2.3 OSRSocket::OSRSocket (const OSR_SOCKET & *socket*)

The copy constructor. The copied instance will have the same socket handle as the original one, but will not be the owner of the handle.

3.39.2.4 OSRSocket::~OSRSocket ()

The destructor. This closes socket if the instance is the owner of the socket.

3.39.3 Member Function Documentation

3.39.3.1 OSR_SOCKET* OSRSocket::accept (OS_IPADDR * *destIP* = 0, int * *port* = 0)

This method permits an incoming connection attempt on a socket. It extracts the first connection on the queue of pending connections on the socket. It then creates a new socket and returns an instance of the new socket. The newly created socket will handle the actual connection and has the same properties as the original socket.

Parameters:

destIP Optional pointer to a buffer that receives the IP address of the connecting entity. It may be NULL.

port Optional pointer to a buffer that receives the port of the connecting entity. It may be NULL.

Returns:

An instance of the new socket class. NULL, if error occur. Use `getStatus` method to obtain error code.

See also:

`rtSocketAccept`

3.39.3.2 static const char* OSRTSocket::addrToString (OSIPADDR *ipAddr*, char * *pAddrStr*, int *bufsize*)
[static]

This method converts an IP address to its string representation.

Parameters:

ipAddr The IP address to be converted.

pAddrStr Pointer to the buffer to receive a string with the IP address.

bufsize Size of the buffer.

Returns:

Pointer to a string with IP-address. NULL, if error occur.

3.39.3.3 int OSRTSocket::bind (OSIPADDR *addr*, int *port*)

This method associates a local address with a socket. It is used on an unconnected socket before subsequent calls to the connect or listen methods.

Parameters:

addr The local IP address to assign to the socket.

port The local port number to assign to the socket.

Returns:

Completion status of operation:

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

See also:

rtSocketBind

3.39.3.4 int OSRTSocket::bind (const char * *pAddrStr*, int *port*)

This method associates a local address with a socket. It is used on an unconnected socket before subsequent calls to the connect or listen methods.

Parameters:

pAddrStr Null-terminated character string representing a number expressed in the Internet standard "." (dotted) notation.

port The local port number to assign to the socket.

Returns:

Completion status of operation:

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

See also:

rtSocketBind

3.39.3.5 int OSRTSocket::bind (int *port*) [inline]

This method associates only a local port with a socket. It is used on an unconnected socket before subsequent calls to the [OSRTSocket::connect](#) or [OSRTSocket::listen](#) methods.

Parameters:

port The local port number to assign to the socket.

Returns:

Completion status of operation:

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

See also:

rtSocketBind
[bind \(\)](#)

3.39.3.6 int OSRTSocket::blockingRead (OSOCKET * *pbuf*, size_t *readBytes*)

This method receives data from the connected socket. In this case, the connection is blocked until either the requested number of bytes is received or the socket is closed or an error occurs.

Parameters:

pbuf Pointer to the buffer for the incoming data.

readBytes Number of bytes to receive.

Returns:

If no error occurs, returns the number of bytes received. Otherwise, the negative value is error code.

3.39.3.7 int OSRTSocket::close ()

This method closes this socket.

Returns:

Completion status of operation:

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

See also:

rtSocketClose

3.39.3.8 `int OSRTSocket::connect (const char * host, int port)`

This method establishes a connection to this socket. It is used to create a connection to the specified destination. When the socket call completes successfully, the socket is ready to send and receive data.

Parameters:

host Null-terminated character string representing a number expressed in the Internet standard "." (dotted) notation.

port The destination port to connect.

Returns:

Completion status of operation:

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

See also:

`rtSocketConnect`

3.39.3.9 `OSBOOL OSRTSocket::getOwnership () [inline]`

Returns the ownership of underlying O/S socket.

Returns:

TRUE, if the socket object has the ownership of underlying O/S socket.

3.39.3.10 `OSRTSOCKET OSRTSocket::getSocket () const [inline]`

This method returns the handle of the socket.

Returns:

The handle of the socket.

3.39.3.11 `int OSRTSocket::getStatus () [inline]`

Returns a completion status of last operation.

Returns:

Completion status of last operation:

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

3.39.3.12 `int OSRTSocket::listen (int maxConnections)`

This method places a socket into a state where it is listening for an incoming connection.

Parameters:

maxConnections Maximum length of the queue of pending connections.

Returns:

Completion status of operation:

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

See also:

`rtSocketListen`

3.39.3.13 `int OSRTSocket::recv (OSOCKET * pbuf, OSUINT32 bufsize)`

This method receives data from a connected socket. It is used to read incoming data on sockets. The socket must be connected before calling this function.

Parameters:

pbuf Pointer to the buffer for the incoming data.

bufsize Length of the buffer.

Returns:

If no error occurs, returns the number of bytes received. Negative error code if error occurred.

See also:

`rtSocketRecv`

3.39.3.14 `void OSRTSocket::setOwnership (OSBOOL ownership)` `[inline]`

Transfers an ownership of the underlying O/S socket to or from the socket object. If the socket object has the ownership of the underlying O/S socket it will close the O/S socket when the socket object is being closed or destroyed.

Parameters:

ownership TRUE, if socket object should have ownership of the underlying O/S socket; FALSE, otherwise.

3.39.3.15 `int OSRTSocket::send (const OSOCKET * pdata, OSUINT32 size)`

This method sends data on a connected socket. It is used to write outgoing data on a connected socket.

Parameters:

pdata Buffer containing the data to be transmitted.

size Length of the data in pdata.

Returns:

Completion status of operation:

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

See also:

rtSocketSend

3.39.3.16 static OSIPADDR OSRTSocket::stringToAddr (const char * *pAddrStr*) [static]

This method converts a string containing an Internet Protocol dotted address into a proper OSIPADDR address.

Parameters:

pAddrStr Null-terminated character string representing a number expressed in the Internet standard "." (dotted) notation.

Returns:

If no error occurs, returns OSIPADDR. OSIPADDR_INVALID, if error occurred.

3.39.4 Member Data Documentation

3.39.4.1 OSRTSOCKET OSRTSocket::mSocket [protected]

handle of the socket

3.39.4.2 OSBOOL OSRTSocket::mOwner [protected]

indicates this class owns the socket

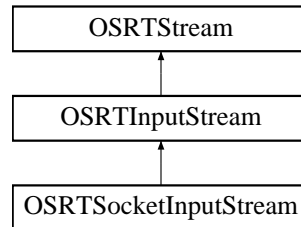
The documentation for this class was generated from the following file:

- [OSRTSocket.h](#)

3.40 OSRTSocketInputStream Class Reference

```
#include <OSRTSocketInputStream.h>
```

Inheritance diagram for OSRTSocketInputStream::



3.40.1 Detailed Description

Generic socket input stream. This class opens an existing socket for input in binary mode and reads data from it.

Public Member Functions

- [OSRTSocketInputStream](#) ([OSRTSocket](#) &socket)
- [OSRTSocketInputStream](#) ([OSRTContext](#) *pContext, [OSRTSocket](#) &socket)
- [OSRTSocketInputStream](#) (OSRTSOCKET socket, OSBOOL ownership=FALSE)
- [OSRTSocketInputStream](#) ([OSRTContext](#) *pContext, OSRTSOCKET socket, OSBOOL ownership=FALSE)

Protected Attributes

- [OSRTSocket](#) mSocket

3.40.2 Constructor & Destructor Documentation

3.40.2.1 OSRTSocketInputStream::OSRTSocketInputStream ([OSRTSocket](#) & socket)

Creates and initializes a socket input stream using the [OSRTSocket](#) instance of socket.

Parameters:

socket Reference to [OSRTSocket](#) instance.

See also:

[rtxStreamSocketOpen](#)

3.40.2.2 OSRTSocketInputStream::OSRTSocketInputStream ([OSRTContext](#) * pContext, [OSRTSocket](#) & socket)

Creates and initializes a socket input stream using the [OSRTSocket](#) instance of socket.

Parameters:

pContext Pointer to a context to use.
socket Reference to [OSRTSocket](#) instance.

See also:

[rtxStreamSocketOpen](#)

3.40.2.3 OSRTSocketInputStream::OSRTSocketInputStream (OSRTSOCKET *socket*, OSBOOL *ownership* = FALSE)

Creates and initializes the socket input stream using the socket handle.

Parameters:

socket Handle of the socket.
ownership Indicates ownership of the socket. Set to TRUE to pass ownership to this object instance. The socket will be closed when this object instance is deleted or goes out of scope.

See also:

[rtxStreamSocketAttach](#)

3.40.2.4 OSRTSocketInputStream::OSRTSocketInputStream (OSRTContext * *pContext*, OSRTSOCKET *socket*, OSBOOL *ownership* = FALSE)

Creates and initializes the socket input stream using the socket handle.

Parameters:

pContext Pointer to a context to use.
socket Handle of the socket.
ownership Indicates ownership of the socket. Set to TRUE to pass ownership to this object instance. The socket will be closed when this object instance is deleted or goes out of scope.

See also:

[rtxStreamSocketAttach](#)

3.40.3 Member Data Documentation

3.40.3.1 OSRTSocket OSRTSocketInputStream::mSocket [protected]

a socket

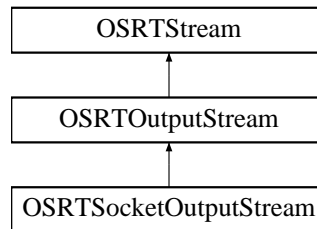
The documentation for this class was generated from the following file:

- [OSRTSocketInputStream.h](#)

3.41 OSRTSocketOutputStream Class Reference

```
#include <OSRTSocketOutputStream.h>
```

Inheritance diagram for OSRTSocketOutputStream::



3.41.1 Detailed Description

Generic socket output stream. This class opens an existing socket for output in binary mode and reads data from it.

Public Member Functions

- [OSRTSocketOutputStream](#) ([OSRTSocket](#) &socket)
- [OSRTSocketOutputStream](#) ([OSRTContext](#) *pContext, [OSRTSocket](#) &socket)
- [OSRTSocketOutputStream](#) ([OSRTSOCKET](#) socket, [OSBOOL](#) ownership=FALSE)
- [OSRTSocketOutputStream](#) ([OSRTContext](#) *pContext, [OSRTSOCKET](#) socket, [OSBOOL](#) ownership=FALSE)

Protected Attributes

- [OSRTSocket](#) mSocket

3.41.2 Constructor & Destructor Documentation

3.41.2.1 OSRTSocketOutputStream::OSRTSocketOutputStream ([OSRTSocket](#) & socket)

Creates and initializes a socket output stream using the [OSRTSocket](#) instance of socket.

Parameters:

socket Reference to [OSRTSocket](#) instance.

See also:

[rtxStreamSocketOpen](#)

3.41.2.2 OSRTSocketOutputStream::OSRTSocketOutputStream ([OSRTContext](#) * pContext, [OSRTSocket](#) & socket)

Creates and initializes a socket output stream using the [OSRTSocket](#) instance of socket.

Parameters:

pContext Pointer to a context to use.
socket Reference to [OSRSocket](#) instance.

See also:

[rtxStreamSocketOpen](#)

3.41.2.3 OSRSocketOutputStream::OSRSocketOutputStream (OSR_SOCKET *socket*, OS_BOOL *ownership* = FALSE)

Initializes the socket output stream using the socket handle.

Parameters:

socket Handle of the socket.
ownership Indicates ownership of the socket. Set to TRUE to pass ownership to this object instance. The socket will be closed when this object instance is deleted or goes out of scope.

See also:

[rtxStreamSocketAttach](#)

3.41.2.4 OSRSocketOutputStream::OSRSocketOutputStream (OSR_CONTEXT **pContext*, OSR_SOCKET *socket*, OS_BOOL *ownership* = FALSE)

Initializes the socket output stream using the socket handle.

Parameters:

pContext Pointer to a context to use.
socket Handle of the socket.
ownership Indicates ownership of the socket. Set to TRUE to pass ownership to this object instance. The socket will be closed when this object instance is deleted or goes out of scope.

See also:

[rtxStreamSocketAttach](#)

3.41.3 Member Data Documentation

3.41.3.1 OSRSocket OSRSocketOutputStream::mSocket [protected]

a socket

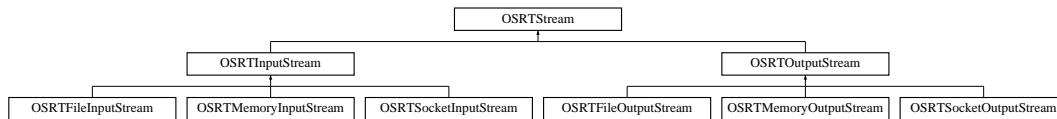
The documentation for this class was generated from the following file:

- [OSRSocketOutputStream.h](#)

3.42 OSRTStream Class Reference

```
#include <OSRTStream.h>
```

Inheritance diagram for OSRTStream::



3.42.1 Detailed Description

The default base class for using I/O streams. This class may be subclassed, as in the case of [OSRTInputStream](#) and [OSRTOutputStream](#) or other custom implementations.

Public Member Functions

- virtual [~OSRTStream](#) ()
- virtual int [close](#) ()
- virtual int [flush](#) ()
- virtual [OSRTCtxtPtr](#) [getContext](#) ()
- virtual [OSCTXT](#) * [getCtxtPtr](#) ()
- virtual char * [getErrorInfo](#) ()
- virtual char * [getErrorInfo](#) (char *pBuf, size_t &bufSize)
- int [getStatus](#) () const
- OSBOOL [isInitialized](#) ()
- virtual OSBOOL [isOpen](#) ()
- void [printErrorInfo](#) ()
- void [resetErrorInfo](#) ()

Protected Member Functions

- [OSRTStream](#) ([OSRTContext](#) *pContext, OSBOOL attachStream=FALSE)
- [OSRTStream](#) ([OSRTStream](#) &original)
- [OSRTStream](#) ()
- char * [getErrorInfo](#) (size_t *pBufSize)

Protected Attributes

- OSBOOL [mbAttached](#)
- int [mStatus](#)
- int [mInitStatus](#)

3.42.2 Constructor & Destructor Documentation

3.42.2.1 OSRTStream::OSRTStream () [protected]

The default constructor. It initializes a buffered stream. A buffered stream maintains data in memory before reading or writing to the device. This generally provides better performance than an unbuffered stream.

3.42.2.2 virtual OSRTStream::~~OSRTStream () [virtual]

Virtual destructor. Closes the stream if it was opened.

3.42.3 Member Function Documentation

3.42.3.1 virtual int OSRTStream::close () [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

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

3.42.3.2 virtual int OSRTStream::flush () [virtual]

Flushes the buffered data to the stream.

Returns:

Completion status of operation:

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

See also:

rtxStreamFlush

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

3.42.3.3 virtual OSRTCtxPtr OSRTStream::getContext () [inline, virtual]

This method returns a pointer to the underlying [OSRTContext](#) object.

Returns:

A reference-counted pointer to an [OSRTContext](#) object. The [OSRTContext](#) object will not be released until all referenced-counted pointer variables go out of scope. This allows safe sharing of the context between different run-time classes.

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

3.42.3.4 **virtual OSCTXT* OSRTStream::getCtxtPtr ()** [inline, virtual]

This method returns a pointer to the underlying OSCTXT object. This is the structure used in calls to low-level C encode/decode functions.

Returns:

Pointer to a context (OSCTXT) structure.

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

3.42.3.5 **virtual char* OSRTStream::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.

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

3.42.3.6 **virtual char* OSRTStream::getErrorInfo (char * pBuf, size_t & bufSize)** [inline, virtual]

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

Parameters:

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

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

Returns:

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

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

3.42.3.7 **int OSRTStream::getStatus () const** [inline]

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.

Returns:

Runtime status code:

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

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

3.42.3.8 **virtual OSBOOL OSRTStream::isOpen ()** [virtual]

Checks, is the stream opened or not.

Returns:

TRUE, if the stream is opened, FALSE otherwise.

See also:

[rtxStreamIsOpened](#)

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

3.42.3.9 **void OSRTStream::printErrorInfo ()** [inline]

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

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

3.42.3.10 **void OSRTStream::resetErrorInfo ()** [inline]

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

Reimplemented in [OSRTInputStream](#), and [OSRTOutputStream](#).

3.42.4 Member Data Documentation

3.42.4.1 **OSBOOL OSRTStream::mbAttached** [protected]

Flag, TRUE for "attached" streams.

3.42.4.2 **int OSRTStream::mStatus** [protected]

Last stream operation status.

3.42.4.3 **int OSRTStream::mInitStatus** [protected]

Initialization status. 0 if initialized successfully.

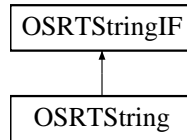
The documentation for this class was generated from the following file:

- [OSRTStream.h](#)

3.43 OSRTString Class Reference

```
#include <OSRTString.h>
```

Inheritance diagram for OSRTString::



3.43.1 Detailed Description

C++ string class definition. This can be used to hold standard ASCII or UTF-8 strings. The standard C++ 'new' and 'delete' operators are used to allocate/free memory for the strings. All strings are deep-copied.

Public Member Functions

- [OSRTString](#) ()
- [OSRTString](#) (const char *strval)
- [OSRTString](#) (const OSUTF8CHAR *strval)
- [OSRTString](#) (const [OSRTString](#) &str)
- virtual [~OSRTString](#) ()
- virtual [OSRTStringIF](#) * [clone](#) ()
- virtual const char * [getValue](#) () const
- virtual const OSUTF8CHAR * [getUTF8Value](#) () const
- virtual void [print](#) (const char *name)
- virtual void [setValue](#) (const char *str)
- virtual void [setValue](#) (const OSUTF8CHAR *str)
- [OSRTString](#) & [operator=](#) (const [OSRTString](#) &original)

Protected Attributes

- OSUTF8CHAR * [mValue](#)

3.43.2 Constructor & Destructor Documentation

3.43.2.1 OSRTString::OSRTString ()

The default constructor creates an empty string.

3.43.2.2 OSRTString::OSRTString (const char * strval)

This constructor initializes the string to contain the given standard ASCII string value.

Parameters:

strval - Null-terminated C string value

3.43.2.3 OSRTString::OSRTString (const OSUTF8CHAR * *strval*)

This constructor initializes the string to contain the given UTF-8 string value.

Parameters:

strval - Null-terminated C string value

3.43.2.4 OSRTString::OSRTString (const OSRTString & *str*)

Copy constructor.

Parameters:

str - C++ string object to be copied.

3.43.2.5 virtual OSRTString::~~OSRTString () [virtual]

The destructor frees string memory using the standard 'delete' operator.

3.43.3 Member Function Documentation

3.43.3.1 virtual OSRTStringIF* OSRTString::clone () [inline, virtual]

This method creates a copy of the given string object.

Implements [OSRTStringIF](#).

3.43.3.2 virtual const char* OSRTString::getValue () const [inline, virtual]

This method returns the pointer to UTF-8 null terminated string as a standard ASCII string.

Implements [OSRTStringIF](#).

3.43.3.3 virtual const OSUTF8CHAR* OSRTString::getUTF8Value () const [inline, virtual]

This method returns the pointer to UTF-8 null terminated string as a UTF-8 string.

Implements [OSRTStringIF](#).

3.43.3.4 virtual void OSRTString::print (const char * *name*) [inline, virtual]

This method prints the string value to standard output.

Parameters:

name - Name of generated string variable.

Implements [OSRTStringIF](#).

3.43.3.5 virtual void OSRTString::setValue (const char * *str*) [virtual]

This method sets the string value to the given string.

Parameters:

str - C null-terminated string.

Implements [OSRTStringIF](#).

3.43.3.6 virtual void OSRTString::setValue (const OSUTF8CHAR * *str*) [virtual]

This method sets the string value to the given UTF-8 string value.

Parameters:

utf8str - C null-terminated UTF-8 string.

Implements [OSRTStringIF](#).

3.43.3.7 OSRTString& OSRTString::operator= (const OSRTString & *original*)

Assignment operator.

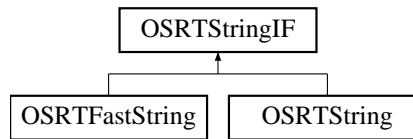
The documentation for this class was generated from the following file:

- [OSRTString.h](#)

3.44 OSRTStringIF Class Reference

```
#include <OSRTStringIF.h>
```

Inheritance diagram for OSRTStringIF::



3.44.1 Detailed Description

C++ string class interface. This defines an interface to allow different types of string derived classes to be implemented. Currently, implementations include a standard string class ([OSRTString](#)) which deep-copies all values using new/delete, and a fast string class ([OSRTFastString](#)) that just copies pointers (i.e does no memory management).

Public Member Functions

- virtual `~OSRTStringIF ()`
- virtual `OSRTStringIF * clone ()=0`
- virtual `const char * getValue () const=0`
- virtual `const OSUTF8CHAR * getUTF8Value () const=0`
- virtual `void print (const char *name)=0`
- virtual `void setValue (const char *str)=0`
- virtual `void setValue (const OSUTF8CHAR *utf8str)=0`

Protected Member Functions

- `OSRTStringIF ()`
- `OSRTStringIF (const char *strval)`
- `OSRTStringIF (const OSUTF8CHAR *strval)`

3.44.2 Constructor & Destructor Documentation

3.44.2.1 OSRTStringIF::OSRTStringIF () [inline, protected]

The default constructor creates an empty string.

3.44.2.2 OSRTStringIF::OSRTStringIF (const char * strval) [inline, protected]

This constructor initializes the string to contain the given standard ASCII string value.

Parameters:

strval - Null-terminated C string value

3.44.2.3 OSRTStringIF::OSRTStringIF (const OSUTF8CHAR * *strval*) [inline, protected]

This constructor initializes the string to contain the given UTF-8 string value.

Parameters:

strval - Null-terminated C string value

3.44.2.4 virtual OSRTStringIF::~~OSRTStringIF () [inline, virtual]

The destructor frees string memory using the standard 'delete' operator.

3.44.3 Member Function Documentation

3.44.3.1 virtual OSRTStringIF* OSRTStringIF::clone () [pure virtual]

This method creates a copy of the given string object.

Implemented in [OSRTString](#), and [OSRTFastString](#).

3.44.3.2 virtual const char* OSRTStringIF::getValue () const [pure virtual]

This method returns the pointer to UTF-8 null terminated string as a standard ASCII string.

Implemented in [OSRTString](#), and [OSRTFastString](#).

3.44.3.3 virtual const OSUTF8CHAR* OSRTStringIF::getUTF8Value () const [pure virtual]

This method returns the pointer to UTF-8 null terminated string as a UTF-8 string.

Implemented in [OSRTString](#), and [OSRTFastString](#).

3.44.3.4 virtual void OSRTStringIF::print (const char * *name*) [pure virtual]

This method prints the string value to standard output.

Parameters:

name - Name of generated string variable.

Implemented in [OSRTString](#), and [OSRTFastString](#).

3.44.3.5 virtual void OSRTStringIF::setValue (const char * *str*) [pure virtual]

This method sets the string value to the given string.

Parameters:

str - C null-terminated string.

Implemented in [OSRTString](#), and [OSRTFastString](#).

3.44.3.6 virtual void OSRTStringIF::setValue (const OSUTF8CHAR * *utf8str*) [pure virtual]

This method sets the string value to the given UTF-8 string value.

Parameters:

utf8str - C null-terminated UTF-8 string.

Implemented in [OSRTString](#), and [OSRTFastString](#).

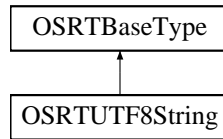
The documentation for this class was generated from the following file:

- [OSRTStringIF.h](#)

3.45 OSRTUTF8String Class Reference

```
#include <OSRTUTF8String.h>
```

Inheritance diagram for OSRTUTF8String::



3.45.1 Detailed Description

UTF-8 string. This is the base class for generated C++ data type classes for XSD string types (string, token, NMTOKEN, etc.).

Public Member Functions

- [OSRTUTF8String \(\)](#)
- [OSRTUTF8String \(const char *strval\)](#)
- [OSRTUTF8String \(const OSUTF8CHAR *strval\)](#)
- [OSRTUTF8String \(const OSRTUTF8String &str\)](#)
- virtual [~OSRTUTF8String \(\)](#)
- [OSRTBaseType * clone \(\) const](#)
- void [copyValue \(const char *str\)](#)
- const char * [c_str \(\) const](#)
- const char * [getValue \(\) const](#)
- void [print \(const char *name\)](#)
- void [setValue \(const char *str\)](#)
- [OSRTUTF8String & operator= \(const OSRTUTF8String &original\)](#)

3.45.2 Constructor & Destructor Documentation

3.45.2.1 OSRTUTF8String::OSRTUTF8String ()

The default constructor creates an empty string.

3.45.2.2 OSRTUTF8String::OSRTUTF8String (const char * *strval*)

This constructor initializes the string to contain the given character string value.

Parameters:

strval - String value

3.45.2.3 OSRTUTF8String::OSRTUTF8String (const OSUTF8CHAR * *strval*)

This constructor initializes the string to contain the given UTF-8 character string value.

Parameters:

strval - String value

3.45.2.4 OSRTUTF8String::OSRTUTF8String (const OSRTUTF8String & *str*)

Copy constructor.

Parameters:

str - C++ XML string class.

3.45.2.5 virtual OSRTUTF8String::~~OSRTUTF8String () [virtual]

The destructor frees string memory if the memory ownership flag is set.

3.45.3 Member Function Documentation

3.45.3.1 OSRTBaseType* OSRTUTF8String::clone () const [inline, virtual]

Clone method. Creates a copied instance and returns pointer to OSRTBaseType.

Reimplemented from OSRTBaseType.

3.45.3.2 void OSRTUTF8String::copyValue (const char * *str*)

This method copies the given string value to the internal string storage variable. A deep-copy of the given value is done; the class will delete this memory when the object is deleted.

Parameters:

utf8str - C null-terminated string.

3.45.3.3 const char* OSRTUTF8String::c_str () const [inline]

This method returns the pointer to C null terminated string.

3.45.3.4 const char* OSRTUTF8String::getValue () const [inline]

This method returns the pointer to UTF-8 null terminated string.

3.45.3.5 void OSRTUTF8String::print (const char * *name*) [inline]

This method prints the string value to standard output.

Parameters:

name - Name of generated string variable.

3.45.3.6 void OSRTUTF8String::setValue (const char * *str*)

This method sets the string value to the given string. A deep-copy of the given value is not done; the pointer is stored directly in the class member variable.

Parameters:

utf8str - C null-terminated string.

3.45.3.7 OSRTUTF8String& OSRTUTF8String::operator= (const OSRTUTF8String & *original*)

Assignment operator.

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

- [OSRTUTF8String.h](#)

Chapter 4

ASN1C C++ Common Runtime Classes File Documentation

4.1 ASN1CBitStr.h File Reference

4.1.1 Detailed Description

Bit string control class definitions.

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

Classes

- class [ASN1CBitStr](#)

4.2 ASN1CGeneralizedTime.h File Reference

4.2.1 Detailed Description

GeneralizedTime control class definition.

```
#include "rtsrc/ASN1CTime.h"
```

Classes

- class [ASN1CGeneralizedTime](#)

4.3 ASN1Context.h File Reference

4.3.1 Detailed Description

Common C++ type and class definitions.

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

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

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

Classes

- class [ASN1Context](#)

4.4 asn1CppEvtHndlr.h File Reference

4.4.1 Detailed Description

Named event handler base class. The Asn1Named Event Handler class is an abstract base class from which user-defined event handlers are derived. This class contains pure virtual function definitions for all of the methods that must be implemented to create a customized event handler class.

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

Classes

- class [Asn1NamedEventHandler](#)
- class [Asn1ErrorHandler](#)

4.5 `asn1CppType.h` File Reference

4.5.1 Detailed Description

Common C++ type and class definitions.

```
#include "rtxsrc/rtxMemory.h"
#include "rtxsrc/rtxDiag.h"
#include "rtxsrc/rtxError.h"
#include "rtxsrc/rtxMemBuf.h"
#include "rtsrc/asn1CppTypeEvtHndlr.h"
#include "rtsrc/ASN1Context.h"
#include "rtxsrc/OSRTMsgBuf.h"
#include "rtsrc/ASN1TOctStr.h"
#include "rtsrc/ASN1TObjId.h"
```

Classes

- class [ASN1MessageBuffer](#)
- class [ASN1CType](#)
- struct [ASN1TDynBitStr](#)
- struct [ASN1TBMPString](#)
- struct [ASN1TUniversalString](#)
- struct [ASN1TOpenType](#)
- struct [Asn1TObject](#)
- struct [ASN1TObjId64](#)
- struct [ASN1TSeqExt](#)
- struct [ASN1TPDU](#)
- struct [ASN1TSeqOfList](#)
- struct [ASN1TPDUSeqOfList](#)

Defines

- `#define ASN1TRY try`
- `#define ASN1RTLTHROW(stat) exit (-1)`
- `#define ASN1THROW(ex) exit (-1)`
- `#define ASN1CATCH(exType, ex, body) if (0) { body; }`

Typedefs

- typedef [Asn1TObject](#) `ASN1TObject`

4.6 ASN1CSeqOfList.h File Reference

4.6.1 Detailed Description

[ASN1CSeqOfList](#) linked list control class definition.

```
#include <new>
```

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

Classes

- class [ASN1CSeqOfListIterator](#)
- class [ASN1CSeqOfList](#)

4.7 ASN1CTime.h File Reference

4.7.1 Detailed Description

[ASN1CTime](#) abstract class definition. This is used as the base class for other ASN.1 time class definitions.

```
#include <time.h>
#include "rtsrc/asn1CppTypes.h"
#include "rtsrc/ASN1TTime.h"
```

Classes

- class [ASN1CTime](#)

Defines

- #define **LOG_TMERR**(pctxt, stat) ((pctxt != 0) ? LOG_RTERR (pctxt, stat) : stat)

4.8 ASN1CUTCTime.h File Reference

4.8.1 Detailed Description

[ASN1CUTCTime](#) control class definition.

```
#include "rtsrc/ASN1CTime.h"
```

Classes

- class [ASN1CUTCTime](#)

4.9 asn1ErrCodes.h File Reference

4.9.1 Detailed Description

List of numeric status codes that can be returned by ASN1C run-time functions and generated code.

Defines

- #define `ASN_OK_FRAG` 2
- #define `ASN_E_BASE` -100
- #define `ASN_E_INVOBJID` (ASN_E_BASE)
- #define `ASN_E_INVLEN` (ASN_E_BASE-1)
- #define `ASN_E_BADTAG` (ASN_E_BASE-2)
- #define `ASN_E_INVBINS` (ASN_E_BASE-3)
- #define `ASN_E_INVINDEX` (ASN_E_BASE-4)
- #define `ASN_E_INVTCVAL` (ASN_E_BASE-5)
- #define `ASN_E_CONCMODF` (ASN_E_BASE-6)
- #define `ASN_E_ILLSTATE` (ASN_E_BASE-7)
- #define `ASN_E_NOTPDU` (ASN_E_BASE-8)
- #define `ASN_E_UNDEFTYP` (ASN_E_BASE-9)

4.10 ASN1ObjId.h File Reference

4.10.1 Detailed Description

ASN.1 object identifier class definition.

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

Classes

- struct [ASN1ObjId](#)

Functions

- int [operator==](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator==](#) (const [ASN1ObjId](#) &lhs, const [ASN1ObjId](#) &rhs)
- int [operator==](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator==](#) (const [ASN1ObjId](#) &lhs, const char *dotted_oid_string)
- int [operator!=](#) (const [ASN1ObjId](#) &lhs, const [ASN1ObjId](#) &rhs)
- int [operator!=](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator!=](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator!=](#) (const [ASN1ObjId](#) &lhs, const char *dotted_oid_string)
- int [operator<](#) (const [ASN1ObjId](#) &lhs, const [ASN1ObjId](#) &rhs)
- int [operator<](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator<](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator<](#) (const [ASN1ObjId](#) &lhs, const char *dotted_oid_string)
- int [operator<=](#) (const [ASN1ObjId](#) &lhs, const [ASN1ObjId](#) &rhs)
- int [operator<=](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator<=](#) (const [ASN1ObjId](#) &lhs, const char *dotted_oid_string)
- int [operator<=](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator>](#) (const [ASN1ObjId](#) &lhs, const [ASN1ObjId](#) &rhs)
- int [operator>](#) (const [ASN1ObjId](#) &lhs, const char *dotted_oid_string)
- int [operator>](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator>](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- int [operator>=](#) (const [ASN1ObjId](#) &lhs, const [ASN1ObjId](#) &rhs)
- int [operator>=](#) (const [ASN1ObjId](#) &lhs, const char *dotted_oid_string)
- int [operator>=](#) (const ASN1OBJID &lhs, const ASN1OBJID &rhs)
- int [operator>=](#) (const ASN1OBJID &lhs, const char *dotted_oid_string)
- [ASN1ObjId operator+](#) (const [ASN1ObjId](#) &lhs, const [ASN1ObjId](#) &rhs)

4.11 ASN1OctStr.h File Reference

4.11.1 Detailed Description

ASN.1 OCTET string class definition.

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

Classes

- struct [ASN1TDynOctStr](#)

Functions

- int **operator**== (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int **operator**== (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int **operator**== (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int **operator**== (const [ASN1DynOctStr](#) &lhs, const char *string)
- int **operator**!= (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int **operator**!= (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int **operator**!= (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int **operator**!= (const [ASN1DynOctStr](#) &lhs, const char *string)
- int **operator**< (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int **operator**< (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int **operator**< (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int **operator**< (const [ASN1DynOctStr](#) &lhs, const char *string)
- int **operator**<= (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int **operator**<= (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int **operator**<= (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int **operator**<= (const [ASN1DynOctStr](#) &lhs, const char *string)
- int **operator**> (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int **operator**> (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int **operator**> (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int **operator**> (const [ASN1DynOctStr](#) &lhs, const char *string)
- int **operator**>= (const [ASN1TDynOctStr](#) &lhs, const [ASN1TDynOctStr](#) &rhs)
- int **operator**>= (const [ASN1TDynOctStr](#) &lhs, const char *string)
- int **operator**>= (const [ASN1DynOctStr](#) &lhs, const [ASN1DynOctStr](#) &rhs)
- int **operator**>= (const [ASN1DynOctStr](#) &lhs, const char *string)

4.12 ASN1TTime.h File Reference

4.12.1 Detailed Description

```
#include <time.h>
#include "rtsrc/asn1CppTypes.h"
```

Classes

- class [ASN1TTime](#)
- class [ASN1TGeneralizedTime](#)
- class [ASN1TUTCTime](#)

Defines

- #define **MAX_TIMESTR_SIZE** 64
- #define **LOG_TTMERR**(stat) (mStatus = stat, stat)

4.13 OSRTBaseType.h File Reference

4.13.1 Detailed Description

C++ run-time base class for structured type definitions.

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

Classes

- class [OSRTBaseType](#)

4.14 OSRTContext.h File Reference

4.14.1 Detailed Description

C++ run-time context class definition.

```
#include "rtxsrc/rtxContext.h"
#include "rtxsrc/rtxDiag.h"
#include "rtxsrc/rtxError.h"
#include "rtxsrc/rtxMemory.h"
```

Classes

- class [OSRTContext](#)
- class [OSRTCtxtPtr](#)

Functions

- EXTERNRTX void * [operator new](#) (size_t nbytes, OSCTXT *pctx)
- EXTERNRTX void [operator delete](#) (void *pmem, OSCTXT *pctx)

4.14.2 Function Documentation

4.14.2.1 EXTERNRTX void operator delete (void * *pmem*, OSCTXT * *pctx*)

Custom placement delete function to free memory using context memory-management functions.

4.14.2.2 EXTERNRTX void* operator new (size_t *nbytes*, OSCTXT * *pctx*)

Custom placement new function to allocate memory using context memory-management functions.

4.15 OSRTFastString.h File Reference

4.15.1 Detailed Description

C++ fast string class definition. This can be used to hold standard ASCII or UTF-8 strings. This string class implementations directly assigns any assigned pointers to internal member variables. It does no memory management.

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

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

Classes

- class [OSRTFastString](#)

4.16 OSRTFileInputStream.h File Reference

4.16.1 Detailed Description

C++ base class definitions for operations with input file streams.

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

Classes

- class [OSRTFileInputStream](#)

4.17 OSRTFileOutputStream.h File Reference

4.17.1 Detailed Description

C++ base class definitions for operations with output file streams.

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

Classes

- class [OSRTFileOutputStream](#)

4.18 OSRTInputStream.h File Reference

4.18.1 Detailed Description

C++ base class definitions for operations with input streams.

```
#include "rtxsrc/OSRTInputStreamIF.h"  
#include "rtxsrc/OSRTStream.h"
```

Classes

- class [OSRTInputStream](#)

4.19 OSRTInputStreamIF.h File Reference

4.19.1 Detailed Description

C++ interface class definitions for operations with input streams.

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

Classes

- class **OSRTInputStreamIF**
- class **OSRTInputStreamPtr**

4.20 OSRTMemoryInputStream.h File Reference

4.20.1 Detailed Description

C++ base class definitions for operations with input memory streams.

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

Classes

- class [OSRTMemoryInputStream](#)

4.21 OSRTMemoryOutputStream.h File Reference

4.21.1 Detailed Description

C++ base class definitions for operations with output memory streams.

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

Classes

- class [OSRTMemoryOutputStream](#)

4.22 OSRTMsgBuf.h File Reference

4.22.1 Detailed Description

C++ run-time message buffer class definition.

```
#include "rtxsrc/OSRTCtxtHolder.h"  
#include "rtxsrc/OSRTMsgBufIF.h"
```

Classes

- class [OSRTMessageBuffer](#)

4.23 OSRTMsgBufIF.h File Reference

4.23.1 Detailed Description

C++ run-time message buffer interface class definition.

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

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

Classes

- class [OSRTMessageBufferIF](#)

4.24 OSRTOutputStream.h File Reference

4.24.1 Detailed Description

C++ base class definitions for operations with output streams.

```
#include "rtxsrc/OSRTOutputStreamIF.h"  
#include "rtxsrc/OSRTStream.h"
```

Classes

- class [OSRTOutputStream](#)

4.25 OSRTOutputStreamIF.h File Reference

4.25.1 Detailed Description

C++ interface class definitions for operations with output streams.

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

Classes

- class **OSRTOutputStreamIF**
- class **OSRTOutputStreamPtr**

4.26 OSRTSocket.h File Reference

4.26.1 Detailed Description

TCP/IP or UDP socket class definitions.

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

Classes

- class [OSRTSocket](#)

4.27 OSRTSocketInputStream.h File Reference

4.27.1 Detailed Description

C++ base class definitions for operations with input socket streams.

```
#include "rtxsrc/OSRTSocket.h"  
#include "rtxsrc/OSRTInputStream.h"
```

Classes

- class [OSRTSocketInputStream](#)

4.28 OSRTSocketOutputStream.h File Reference

4.28.1 Detailed Description

C++ base class definitions for operations with output socket streams.

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

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

Classes

- class [OSRTSocketOutputStream](#)

4.29 OSRTStream.h File Reference

4.29.1 Detailed Description

C++ base class definitions for operations with I/O streams.

```
#include "rtxsrc/OSRCTxtHolder.h"  
#include "rtxsrc/OSRTStreamIF.h"
```

Classes

- class [OSRTStream](#)

4.30 OSRTStreamIF.h File Reference

4.30.1 Detailed Description

C++ interface class definitions for operations with I/O streams.

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

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

Classes

- class **OSRTStreamIF**

4.31 OSRTString.h File Reference

4.31.1 Detailed Description

C++ string class definition. This can be used to hold standard ASCII or UTF-8 strings. The standard C++ 'new' and 'delete' operators are used to allocate/free memory for the strings. All strings are deep-copied.

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

Classes

- class [OSRTString](#)

4.32 OSRTStringIF.h File Reference

4.32.1 Detailed Description

C++ string class interface. This defines an interface to allow different types of string derived classes to be implemented. Currently, implementations include a standard string class ([OSRTString](#)) which deep-copies all values using new/delete, and a fast string class ([OSRTFastString](#)) that just copies pointers (i.e does no memory management).

These classes can be used to hold standard ASCII or UTF-8 strings.

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

Classes

- class [OSRTStringIF](#)

4.33 OSRTUTF8String.h File Reference

4.33.1 Detailed Description

C++ UTF-8 string class definition.

```
#include "rtxsrc/OSRTBaseType.h"  
#include "rtxsrc/rtxPrint.h"  
#include "rtxsrc/rtxUTF8.h"
```

Classes

- class [OSRTUTF8String](#)

Index

- ~ASN1Type
 - ASN1Type, 65
- ~ASN1MessageBuffer
 - ASN1MessageBuffer, 76
- ~ASN1ObjId
 - ASN1ObjId, 95
- ~ASN1Time
 - ASN1Time, 107
- ~OSRTContext
 - OSRTContext, 125
- ~OSRTCtxtPtr
 - OSRTCtxtPtr, 129
- ~OSRTFastString
 - OSRTFastString, 132
- ~OSRTInputStream
 - OSRTInputStream, 139
- ~OSRTMessageBuffer
 - OSRTMessageBuffer, 149
- ~OSRTMessageBufferIF
 - OSRTMessageBufferIF, 153
- ~OSRTOutputStream
 - OSRTOutputStream, 156
- ~OSRTSocket
 - OSRTSocket, 161
- ~OSRTStream
 - OSRTStream, 171
- ~OSRTString
 - OSRTString, 176
- ~OSRTStringIF
 - OSRTStringIF, 179
- ~OSRTUTF8String
 - OSRTUTF8String, 182
- _ref
 - OSRTContext, 125
- _unref
 - OSRTContext, 125
- accept
 - OSRTSocket, 161
- addEventHandler
 - ASN1MessageBuffer, 76
- addrToString
 - OSRTSocket, 161
- append
 - ASN1CSeqOfList, 41
 - ASN1Type, 66
- appendArray
 - ASN1CSeqOfList, 42
- appendArrayCopy
 - ASN1CSeqOfList, 42
- ASN.1 Stream Classes, 15
- ASN.1 Type (ASN1T_) Base Classes, 4
- ASN1CBitStr, 22
 - ASN1CBitStr, 24
- ASN1CBitStr
 - ASN1CBitStr, 24
 - cardinality, 28
 - change, 25
 - clear, 25, 26
 - doAnd, 29
 - doAndNot, 31, 32
 - doOr, 30
 - doXor, 31
 - get, 27, 28
 - getBytes, 28
 - invert, 26
 - isEmpty, 27
 - isSet, 27
 - length, 27
 - operator ASN1TDynBitStr, 33
 - operator ASN1TDynBitStr *, 34
 - set, 24
 - shiftLeft, 32
 - shiftRight, 33
 - size, 27
 - unusedBitsInLastUnit, 33
- ASN1CBitStr.h, 184
- ASN1CGeneralizedTime, 35
 - ASN1CGeneralizedTime, 36
- ASN1CGeneralizedTime
 - ASN1CGeneralizedTime, 36
 - getCentury, 36
 - setCentury, 36
 - setTime, 37
- ASN1CGeneralizedTime.h, 185
- ASN1Context, 38
 - ASN1Context, 38
 - setRunTimeKey, 38
- ASN1Context.h, 186
- asn1CppEvtHndlr.h, 187

- asn1CppTypes.h, 188
- ASN1CSeqOfList, 39
 - ASN1CSeqOfList, 40, 41
- ASN1CSeqOfList
 - append, 41
 - appendArray, 42
 - appendArrayCopy, 42
 - ASN1CSeqOfList, 40, 41
 - clear, 45
 - contains, 44
 - get, 44
 - getFirst, 44
 - getLast, 44
 - indexOf, 44
 - init, 42
 - insert, 42
 - isEmpty, 45
 - iterator, 45
 - iteratorFrom, 46
 - iteratorFromLast, 46
 - operator[], 47
 - OSCRTLFREE, 45
 - remove, 43
 - removeFirst, 43
 - removeLast, 43
 - set, 45
 - size, 45
 - toArray, 46
- ASN1CSeqOfList.h, 189
- ASN1CSeqOfListIterator, 48
- ASN1CSeqOfListIterator
 - hasNext, 48
 - hasPrev, 48
 - insert, 50
 - next, 49
 - prev, 49
 - remove, 49
 - set, 49
- ASN1CTime, 51
 - ASN1CTime, 53
 - clear, 62
 - equals, 63
 - getDay, 54
 - getDiff, 57
 - getDiffHour, 56
 - getDiffMinute, 56
 - getFraction, 55
 - getFractionAsDouble, 55
 - getFractionLen, 56
 - getFractionStr, 56
 - getHour, 54
 - getMinute, 54
 - getMonth, 54
 - getSecond, 55
 - getTime, 57
 - getUTC, 57
 - getYear, 53
 - parseString, 62
 - setDay, 58
 - setDER, 57
 - setDiff, 61, 62
 - setDiffHour, 61
 - setFraction, 60
 - setHour, 59
 - setMinute, 59
 - setMonth, 58
 - setSecond, 59
 - setTime, 61
 - setUTC, 58
 - setYear, 58
- ASN1CTime.h, 190
- ASN1CType, 64
 - ~ASN1CType, 65
 - append, 66
 - ASN1CType, 65
 - Decode, 68
 - DecodeFrom, 68
 - Encode, 68
 - EncodeTo, 68
 - getContext, 66
 - getCtxtPtr, 66
 - getStatus, 66
 - memAlloc, 66
 - memFreeAll, 67
 - memFreePtr, 67
 - memRealloc, 67
 - memReset, 67
 - mpContext, 69
 - mpMsgBuf, 69
 - printErrorInfo, 67
 - resetError, 67
 - setDiag, 68
 - setRunTimeKey, 65
- ASN1CUTCTime, 70
 - ASN1CUTCTime, 71
 - getFraction, 71
 - setTime, 71
- ASN1CUTCTime.h, 191
- asn1data
 - operator!=, 5, 6
 - operator+, 6
 - operator<, 6, 7
 - operator<=, 7, 8
 - operator==, 8, 9
 - operator>, 9, 10
 - operator>=, 10, 11
- asn1ErrCodes
 - ASN_E_BADTAG, 20

- ASN_E_BASE, 20
- ASN_E_CONCMODF, 20
- ASN_E_ILLSTATE, 20
- ASN_E_INVBINS, 20
- ASN_E_INVINDEX, 21
- ASN_E_INVLEN, 21
- ASN_E_INVOBJID, 21
- ASN_E_INVTCVAL, 21
- ASN_E_NOTPDU, 21
- ASN_E_UNDEFTYP, 21
- ASN_OK_FRAG, 21
- asn1ErrCodes.h, 192
- Asn1ErrorHandler, 73
- Asn1ErrorHandler
 - error, 73
 - setErrorHandler, 73
- ASN1MessageBuffer, 75
 - ASN1MessageBuffer, 76
- ASN1MessageBuffer
 - ~ASN1MessageBuffer, 76
 - addEventHandler, 76
 - ASN1MessageBuffer, 76
 - CStringToBMPString, 76
 - getAppInfo, 76
 - initBuffer, 76, 77
 - isA, 77
 - removeEventHandler, 78
 - resetErrorInfo, 78
 - setAppInfo, 78
 - setErrorHandler, 78
 - setRunTimeKey, 78
 - setStatus, 76
- Asn1NamedEventHandler, 19, 80
- Asn1NamedEventHandler
 - bitStringValue, 82
 - boolValue, 81
 - charStringValue, 83, 84
 - endElement, 81
 - enumValue, 85
 - int64Value, 82
 - intValue, 81
 - nullValue, 84
 - octStringValue, 83
 - oidValue, 84
 - openTypeValue, 85
 - realValue, 84
 - startElement, 81
 - uint64Value, 82
 - uintValue, 82
- ASN1TBMPString, 86
 - ASN1TBMPString, 86
- ASN1TDynBitStr, 87
 - ASN1TDynBitStr, 87
- ASN1TDynBitStr
- ASN1TDynBitStr, 87
- ASN1TDynOctStr, 88
 - ASN1TDynOctStr, 88
- ASN1TDynOctStr
 - ASN1TDynOctStr, 88
 - nCompare, 89
 - operator=, 89
 - toHexString, 89
 - toString, 89
- ASN1TGeneralizedTime, 91
 - ASN1TGeneralizedTime, 91, 92
- ASN1TGeneralizedTime
 - ASN1TGeneralizedTime, 91, 92
 - compileString, 93
 - getCentury, 92
 - parseString, 93
 - setCentury, 92
 - setTime, 92
- Asn1TObject, 94
 - Asn1TObject, 94
- ASN1TObjId, 95
 - ASN1TObjId, 95, 96
- ASN1TObjId
 - ~ASN1TObjId, 95
 - ASN1TObjId, 95, 96
 - nCompare, 98
 - operator+=", 96, 97
 - operator=, 96
 - RnCompare, 98
 - set_data, 97
 - toString, 97
 - trim, 98
- ASN1TObjId.h, 193
- ASN1TObjId64, 99
- ASN1TOctStr.h, 194
- ASN1TOpenType, 100
 - ASN1TOpenType, 100
- ASN1TOpenType
 - ASN1TOpenType, 100
- ASN1TPDU, 101
 - mpContext, 101
 - setContext, 101
- ASN1TPDUSeqOfList, 102
 - ASN1TPDUSeqOfList, 102
- ASN1TPDUSeqOfList
 - ASN1TPDUSeqOfList, 102
- ASN1TSeqExt, 103
 - ASN1TSeqExt, 103
- ASN1TSeqExt
 - ASN1TSeqExt, 103
- ASN1TSeqOfList, 104
 - ASN1TSeqOfList, 104
- ASN1TSeqOfList
 - ASN1TSeqOfList, 104

- ASN1TTime, 105
 - ~ASN1TTime, 107
 - ASN1TTime, 107
 - clear, 116
 - compileString, 116
 - equals, 116
 - getDay, 108
 - getDiff, 110
 - getDiffHour, 109
 - getDiffMinute, 110
 - getFraction, 109
 - getFractionAsDouble, 109
 - getFractionLen, 109
 - getFractionStr, 109
 - getHour, 108
 - getMinute, 108
 - getMonth, 107
 - getSecond, 108
 - getTime, 110
 - getUTC, 110
 - getYear, 107
 - parseString, 116
 - setDay, 112
 - setDER, 111
 - setDiff, 115
 - setDiffHour, 114
 - setFraction, 113, 114
 - setHour, 112
 - setMinute, 112
 - setMonth, 111
 - setSecond, 113
 - setTime, 114
 - setUTC, 111
 - setYear, 111
 - toString, 116, 117
- ASN1TTime.h, 195
- ASN1TUniversalString, 118
 - ASN1TUniversalString, 118
- ASN1TUniversalString
 - ASN1TUniversalString, 118
- ASN1TUTCTime, 119
 - ASN1TUTCTime, 119, 120
 - clear, 121
 - compileString, 121
 - getFraction, 121
 - parseString, 121
 - setFraction, 122
 - setTime, 120
 - setUTC, 120
 - setYear, 120
- ASN_E_BADTAG
 - asn1ErrCodes, 20
- ASN_E_BASE
 - asn1ErrCodes, 20
- ASN_E_CONCMODF
 - asn1ErrCodes, 20
- ASN_E_ILLSTATE
 - asn1ErrCodes, 20
- ASN_E_INVBINS
 - asn1ErrCodes, 20
- ASN_E_INVINDEX
 - asn1ErrCodes, 21
- ASN_E_INVLEN
 - asn1ErrCodes, 21
- ASN_E_INVOBJID
 - asn1ErrCodes, 21
- ASN_E_INVTCVAL
 - asn1ErrCodes, 21
- ASN_E_NOTPDU
 - asn1ErrCodes, 21
- ASN_E_UNDEFTYP
 - asn1ErrCodes, 21
- ASN_OK_FRAG
 - asn1ErrCodes, 21
- bind
 - OSRTSocket, 162
- bitStrValue
 - Asn1NamedEventHandler, 82
- blockingRead
 - OSRTSocket, 163
- boolValue
 - Asn1NamedEventHandler, 81
- C++ Run-Time Classes, 2
- c_str
 - OSRTUTF8String, 182
- cardinality
 - ASN1CBitStr, 28
- change
 - ASN1CBitStr, 25
- charStrValue
 - Asn1NamedEventHandler, 83, 84
- clear
 - ASN1CBitStr, 25, 26
 - ASN1CSeqOfList, 45
 - ASN1CTime, 62
 - ASN1TTime, 116
 - ASN1TUTCTime, 121
- clone
 - OSRTFastString, 132
 - OSRTString, 176
 - OSRTStringIF, 179
 - OSRTUTF8String, 182
- close
 - OSRTInputStream, 139
 - OSRTOutputStream, 157
 - OSRTSocket, 163

- OSRTStream, 172
- compileString
 - ASN1TGeneralizedTime, 93
 - ASN1TTime, 116
 - ASN1TUTCTime, 121
- connect
 - OSRTSocket, 163
- contains
 - ASN1CSeqOfList, 44
- Context Management Classes, 12
- Control (ASN1C_) Base Classes, 3
- copyValue
 - OSRTUTF8String, 182
- CStringToBMPString
 - ASN1MessageBuffer, 76
- currentPos
 - OSRTInputStream, 139
- Date and Time Runtime Classes, 13
- Decode
 - ASN1CType, 68
- DecodeFrom
 - ASN1CType, 68
- doAnd
 - ASN1CBitStr, 29
- doAndNot
 - ASN1CBitStr, 31, 32
- doOr
 - ASN1CBitStr, 30
- doXor
 - ASN1CBitStr, 31
- Encode
 - ASN1CType, 68
- EncodeTo
 - ASN1CType, 68
- endElement
 - Asn1NamedEventHandler, 81
- enumValue
 - Asn1NamedEventHandler, 85
- equals
 - ASN1CTime, 63
 - ASN1TTime, 116
- error
 - Asn1ErrorHandler, 73
- flush
 - OSRTInputStream, 139
 - OSRTOutputStream, 157
 - OSRTStream, 172
- Generic Input Stream Classes, 16
- Generic Output Stream Classes, 17
- get
 - ASN1CBitStr, 27, 28
 - ASN1CSeqOfList, 44
- getAppInfo
 - ASN1MessageBuffer, 76
 - OSRTMessageBuffer, 149
 - OSRTMessageBufferIF, 154
- getByteIndex
 - OSRTMessageBuffer, 149
 - OSRTMessageBufferIF, 154
- getBytes
 - ASN1CBitStr, 28
- getCentury
 - ASN1CGeneralizedTime, 36
 - ASN1TGeneralizedTime, 92
- getContext
 - ASN1CType, 66
 - OSRTInputStream, 139
 - OSRTMessageBuffer, 149
 - OSRTOutputStream, 157
 - OSRTStream, 172
- getCtxtPtr
 - ASN1CType, 66
 - OSRTCtxtPtr, 130
 - OSRTInputStream, 140
 - OSRTMessageBuffer, 149
 - OSRTOutputStream, 157
 - OSRTStream, 172
- getDay
 - ASN1CTime, 54
 - ASN1TTime, 108
- getDiff
 - ASN1CTime, 57
 - ASN1TTime, 110
- getDiffHour
 - ASN1CTime, 56
 - ASN1TTime, 109
- getDiffMinute
 - ASN1CTime, 56
 - ASN1TTime, 110
- getErrorInfo
 - OSRTContext, 125, 126
 - OSRTInputStream, 140
 - OSRTMessageBuffer, 149
 - OSRTOutputStream, 157, 158
 - OSRTStream, 173
- getFirst
 - ASN1CSeqOfList, 44
- getFraction
 - ASN1CTime, 55
 - ASN1CUTCTime, 71
 - ASN1TTime, 109
 - ASN1TUTCTime, 121
- getFractionAsDouble
 - ASN1CTime, 55
 - ASN1TTime, 109

- getFractionLen
 - ASN1CTime, 56
 - ASN1TTime, 109
- getFractionStr
 - ASN1CTime, 56
 - ASN1TTime, 109
- getHour
 - ASN1CTime, 54
 - ASN1TTime, 108
- getLast
 - ASN1CSeqOfList, 44
- getMinute
 - ASN1CTime, 54
 - ASN1TTime, 108
- getMonth
 - ASN1CTime, 54
 - ASN1TTime, 107
- getMsgCopy
 - OSRTMessageBuffer, 150
 - OSRTMessageBufferIF, 154
- getMsgPtr
 - OSRTMessageBuffer, 150
 - OSRTMessageBufferIF, 154
- getOwnership
 - OSRTSocket, 164
- getPtr
 - OSRTContext, 125
- getRefCount
 - OSRTContext, 125
- getSecond
 - ASN1CTime, 55
 - ASN1TTime, 108
- getSocket
 - OSRTSocket, 164
- getStatus
 - ASN1CType, 66
 - OSRTContext, 125
 - OSRTInputStream, 140
 - OSRTMessageBuffer, 150
 - OSRTOutputStream, 158
 - OSRTSocket, 164
 - OSRTStream, 173
- getTime
 - ASN1CTime, 57
 - ASN1TTime, 110
- getUTC
 - ASN1CTime, 57
 - ASN1TTime, 110
- getUTF8Value
 - OSRTFastString, 132
 - OSRTString, 176
 - OSRTStringIF, 179
- getValue
 - OSRTFastString, 132
 - OSRTString, 176
 - OSRTStringIF, 179
 - OSRTUTF8String, 182
- getYear
 - ASN1CTime, 53
 - ASN1TTime, 107
- hasNext
 - ASN1CSeqOfListIterator, 48
- hasPrev
 - ASN1CSeqOfListIterator, 48
- indexOf
 - ASN1CSeqOfList, 44
- init
 - ASN1CSeqOfList, 42
 - OSRTMessageBuffer, 150
 - OSRTMessageBufferIF, 154
- initBuffer
 - ASN1MessageBuffer, 76, 77
 - OSRTMessageBuffer, 150
 - OSRTMessageBufferIF, 154
- insert
 - ASN1CSeqOfList, 42
 - ASN1CSeqOfListIterator, 50
- int64Value
 - Asn1NamedEventHandler, 82
- intValue
 - Asn1NamedEventHandler, 81
- invert
 - ASN1CBitStr, 26
- isA
 - ASN1MessageBuffer, 77
 - OSRTMessageBufferIF, 155
- isEmpty
 - ASN1CBitStr, 27
 - ASN1CSeqOfList, 45
- isInitialized
 - OSRTContext, 125
- isNull
 - OSRTCtxtPtr, 130
- isOpened
 - OSRTInputStream, 141
 - OSRTOutputStream, 158
 - OSRTStream, 173
- isSet
 - ASN1CBitStr, 27
- iterator
 - ASN1CSeqOfList, 45
- iteratorFrom
 - ASN1CSeqOfList, 46
- iteratorFromLast
 - ASN1CSeqOfList, 46
- length

- ASN1CBitStr, [27](#)
- listen
 - OSRSocket, [164](#)
- mark
 - OSRTInputStream, [141](#)
- markSupported
 - OSRTInputStream, [141](#)
- mbAttached
 - OSRTStream, [174](#)
- mbInitialized
 - OSRTContext, [128](#)
- mBufferType
 - OSRTMessageBuffer, [151](#)
- mCount
 - OSRTContext, [128](#)
- mCtxt
 - OSRTContext, [128](#)
- memAlloc
 - ASN1CType, [66](#)
 - OSRTContext, [126](#)
- memFreeAll
 - ASN1CType, [67](#)
 - OSRTContext, [126](#)
- memFreePtr
 - ASN1CType, [67](#)
 - OSRTContext, [126](#)
- memRealloc
 - ASN1CType, [67](#)
 - OSRTContext, [127](#)
- memReset
 - ASN1CType, [67](#)
 - OSRTContext, [127](#)
- Message Buffer Classes, [14](#)
- mInitStatus
 - OSRTStream, [174](#)
- mOwner
 - OSRSocket, [166](#)
- mpContext
 - ASN1CType, [69](#)
 - ASN1TPDU, [101](#)
- mpMsgBuf
 - ASN1CType, [69](#)
- mPointer
 - OSRTCtxtPtr, [130](#)
- mSocket
 - OSRSocket, [166](#)
 - OSRSocketInputStream, [168](#)
 - OSRSocketOutputStream, [170](#)
- mStatus
 - OSRTContext, [128](#)
 - OSRTStream, [174](#)
- nCompare
 - ASN1TDynOctStr, [89](#)
 - ASN1ObjId, [98](#)
- next
 - ASN1CSeqOfListIterator, [49](#)
- nullValue
 - Asn1NamedEventHandler, [84](#)
- octStrValue
 - Asn1NamedEventHandler, [83](#)
- oidValue
 - Asn1NamedEventHandler, [84](#)
- openTypeValue
 - Asn1NamedEventHandler, [85](#)
- operator ASN1TDynBitStr
 - ASN1CBitStr, [33](#)
- operator ASN1TDynBitStr *
 - ASN1CBitStr, [34](#)
- operator delete
 - OSRTContext.h, [197](#)
- operator new
 - OSRTContext.h, [197](#)
- operator OSRTContext *
 - OSRTCtxtPtr, [130](#)
- operator!=
 - asn1data, [5, 6](#)
- operator+
 - asn1data, [6](#)
- operator+=
 - ASN1ObjId, [96, 97](#)
- operator->
 - OSRTCtxtPtr, [130](#)
- operator<
 - asn1data, [6, 7](#)
- operator<=
 - asn1data, [7, 8](#)
- operator=
 - ASN1TDynOctStr, [89](#)
 - ASN1ObjId, [96](#)
 - OSRTCtxtPtr, [130](#)
 - OSRTFastString, [133](#)
 - OSRTString, [177](#)
 - OSRTUTF8String, [183](#)
- operator==
 - asn1data, [8, 9](#)
 - OSRTCtxtPtr, [130](#)
- operator>
 - asn1data, [9, 10](#)
- operator>=
 - asn1data, [10, 11](#)
- operator[]
 - ASN1CSeqOfList, [47](#)
- OSCTRLFREE
 - ASN1CSeqOfList, [45](#)
- OSRTBaseType, [123](#)

- OSRTBaseType
 - setOwnMemory, 123
- OSRTBaseType.h, 196
- OSRTContext, 124
 - ~OSRTContext, 125
 - _ref, 125
 - _unref, 125
 - getErrorInfo, 125, 126
 - getPtr, 125
 - getRefCount, 125
 - getStatus, 125
 - isInitialized, 125
 - mbInitialized, 128
 - mCount, 128
 - mCtxt, 128
 - memAlloc, 126
 - memFreeAll, 126
 - memFreePtr, 126
 - memRealloc, 127
 - memReset, 127
 - mStatus, 128
 - OSRTContext, 125
 - printErrorInfo, 127
 - resetErrorInfo, 127
 - setDiag, 127
 - setRunTimeKey, 127
 - setStatus, 128
- OSRTContext.h, 197
 - operator delete, 197
 - operator new, 197
- OSRTCtxtPtr, 129
 - OSRTCtxtPtr, 129
- OSRTCtxtPtr
 - ~OSRTCtxtPtr, 129
 - getCtxtPtr, 130
 - isNull, 130
 - mPointer, 130
 - operator OSRTContext *, 130
 - operator->, 130
 - operator=, 130
 - operator==, 130
 - OSRTCtxtPtr, 129
- OSRTFastString, 131
 - OSRTFastString, 131, 132
- OSRTFastString
 - ~OSRTFastString, 132
 - clone, 132
 - getUTF8Value, 132
 - getValue, 132
 - operator=, 133
 - OSRTFastString, 131, 132
 - print, 132
 - setValue, 132, 133
- OSRTFastString.h, 198
- OSRTFileInputStream, 134
 - OSRTFileInputStream, 134, 135
- OSRTFileInputStream
 - OSRTFileInputStream, 134, 135
- OSRTFileInputStream.h, 199
- OSRTFileOutputStream, 136
 - OSRTFileOutputStream, 136, 137
- OSRTFileOutputStream
 - OSRTFileOutputStream, 136, 137
- OSRTFileOutputStream.h, 200
- OSRTInputStream, 138
 - OSRTInputStream, 138
- OSRTInputStream
 - ~OSRTInputStream, 139
 - close, 139
 - currentPos, 139
 - flush, 139
 - getContext, 139
 - getCtxtPtr, 140
 - getErrorInfo, 140
 - getStatus, 140
 - isOpened, 141
 - mark, 141
 - markSupported, 141
 - OSRTInputStream, 138
 - printErrorInfo, 142
 - read, 142
 - readBlocking, 142
 - reset, 142
 - resetErrorInfo, 142
 - skip, 143
- OSRTInputStream.h, 201
- OSRTInputStreamIF.h, 202
- OSRTMemoryInputStream, 144
 - OSRTMemoryInputStream, 144
- OSRTMemoryInputStream
 - OSRTMemoryInputStream, 144
- OSRTMemoryInputStream.h, 203
- OSRTMemoryOutputStream, 146
 - OSRTMemoryOutputStream, 146
- OSRTMemoryOutputStream
 - OSRTMemoryOutputStream, 146
- OSRTMemoryOutputStream.h, 204
- OSRTMessageBuffer, 148
 - OSRTMessageBuffer, 149
- OSRTMessageBuffer
 - ~OSRTMessageBuffer, 149
 - getAppInfo, 149
 - getByteIndex, 149
 - getContext, 149
 - getCtxtPtr, 149
 - getErrorInfo, 149
 - getMsgCopy, 150
 - getMsgPtr, 150

- getStatus, 150
- init, 150
- initBuffer, 150
- mBufferType, 151
- OSRTMessageBuffer, 149
- printErrorInfo, 151
- resetErrorInfo, 151
- setAppInfo, 151
- setDiag, 151
- OSRTMessageBufferIF, 153
- OSRTMessageBufferIF
 - ~OSRTMessageBufferIF, 153
 - getAppInfo, 154
 - getByteIndex, 154
 - getMsgCopy, 154
 - getMsgPtr, 154
 - init, 154
 - initBuffer, 154
 - isA, 155
 - setAppInfo, 155
 - setDiag, 155
 - setNamespace, 155
- OSRTMsgBuf.h, 205
- OSRTMsgBufIF.h, 206
- OSRTOutputStream, 156
 - OSRTOutputStream, 156
- OSRTOutputStream
 - ~OSRTOutputStream, 156
 - close, 157
 - flush, 157
 - getContext, 157
 - getCtxtPtr, 157
 - getErrorInfo, 157, 158
 - getStatus, 158
 - isOpened, 158
 - OSRTOutputStream, 156
 - printErrorInfo, 158
 - resetErrorInfo, 159
 - write, 159
- OSRTOutputStream.h, 207
- OSRTOutputStreamIF.h, 208
- OSRTSocket, 160
 - ~OSRTSocket, 161
 - accept, 161
 - addrToString, 161
 - bind, 162
 - blockingRead, 163
 - close, 163
 - connect, 163
 - getOwnership, 164
 - getSocket, 164
 - getStatus, 164
 - listen, 164
 - mOwner, 166
 - mSocket, 166
 - OSRTSocket, 161
 - recv, 165
 - send, 165
 - setOwnership, 165
 - stringToAddr, 166
- OSRTSocket.h, 209
- OSRTSocketInputStream, 167
 - OSRTSocketInputStream, 167, 168
- OSRTSocketInputStream
 - mSocket, 168
 - OSRTSocketInputStream, 167, 168
- OSRTSocketInputStream.h, 210
- OSRTSocketOutputStream, 169
 - OSRTSocketOutputStream, 169, 170
- OSRTSocketOutputStream
 - mSocket, 170
 - OSRTSocketOutputStream, 169, 170
- OSRTSocketOutputStream.h, 211
- OSRTStream, 171
 - ~OSRTStream, 171
 - close, 172
 - flush, 172
 - getContext, 172
 - getCtxtPtr, 172
 - getErrorInfo, 173
 - getStatus, 173
 - isOpened, 173
 - mbAttached, 174
 - mInitStatus, 174
 - mStatus, 174
 - OSRTStream, 171
 - printErrorInfo, 174
 - resetErrorInfo, 174
- OSRTStream.h, 212
- OSRTStreamIF.h, 213
- OSRTString, 175
 - ~OSRTString, 176
 - clone, 176
 - getUTF8Value, 176
 - getValue, 176
 - operator=, 177
 - OSRTString, 175, 176
 - print, 176
 - setValue, 176, 177
- OSRTString.h, 214
- OSRTStringIF, 178
 - OSRTStringIF, 178
- OSRTStringIF
 - ~OSRTStringIF, 179
 - clone, 179
 - getUTF8Value, 179
 - getValue, 179
 - OSRTStringIF, 178

- print, 179
- setValue, 179
- OSRTStringIF.h, 215
- OSRTUTF8String, 181
 - ~OSRTUTF8String, 182
 - c_str, 182
 - clone, 182
 - copyValue, 182
 - getValue, 182
 - operator=, 183
 - OSRTUTF8String, 181, 182
 - print, 182
 - setValue, 183
- OSRTUTF8String.h, 216
- parseString
 - ASN1CTime, 62
 - ASN1TGeneralizedTime, 93
 - ASN1TTime, 116
 - ASN1TUTCTime, 121
- prev
 - ASN1CSeqOfListIterator, 49
- print
 - OSRTFastString, 132
 - OSRTString, 176
 - OSRTStringIF, 179
 - OSRTUTF8String, 182
- printErrorInfo
 - ASN1CType, 67
 - OSRTContext, 127
 - OSRTInputStream, 142
 - OSRTMessageBuffer, 151
 - OSRTOutputStream, 158
 - OSRTStream, 174
- read
 - OSRTInputStream, 142
- readBlocking
 - OSRTInputStream, 142
- realValue
 - Asn1NamedEventHandler, 84
- recv
 - OSRTSocket, 165
- remove
 - ASN1CSeqOfList, 43
 - ASN1CSeqOfListIterator, 49
- removeEventHandler
 - ASN1MessageBuffer, 78
- removeFirst
 - ASN1CSeqOfList, 43
- removeLast
 - ASN1CSeqOfList, 43
- reset
 - OSRTInputStream, 142
- resetError
 - ASN1CType, 67
- resetErrorInfo
 - ASN1MessageBuffer, 78
 - OSRTContext, 127
 - OSRTInputStream, 142
 - OSRTMessageBuffer, 151
 - OSRTOutputStream, 159
 - OSRTStream, 174
- RnCompare
 - ASN1TObjId, 98
- Run-time error status codes., 20
- send
 - OSRTSocket, 165
- set
 - ASN1CBitStr, 24
 - ASN1CSeqOfList, 45
 - ASN1CSeqOfListIterator, 49
- set_data
 - ASN1TObjId, 97
- setAppInfo
 - ASN1MessageBuffer, 78
 - OSRTMessageBuffer, 151
 - OSRTMessageBufferIF, 155
- setCentury
 - ASN1CGeneralizedTime, 36
 - ASN1TGeneralizedTime, 92
- setContext
 - ASN1TPDU, 101
- setDay
 - ASN1CTime, 58
 - ASN1TTime, 112
- setDER
 - ASN1CTime, 57
 - ASN1TTime, 111
- setDiag
 - ASN1CType, 68
 - OSRTContext, 127
 - OSRTMessageBuffer, 151
 - OSRTMessageBufferIF, 155
- setDiff
 - ASN1CTime, 61, 62
 - ASN1TTime, 115
- setDiffHour
 - ASN1CTime, 61
 - ASN1TTime, 114
- setErrorHandler
 - Asn1ErrorHandler, 73
 - ASN1MessageBuffer, 78
- setFraction
 - ASN1CTime, 60
 - ASN1TTime, 113, 114
 - ASN1TUTCTime, 122

- setHour
 - ASN1CTime, 59
 - ASN1TTime, 112
- setMinute
 - ASN1CTime, 59
 - ASN1TTime, 112
- setMonth
 - ASN1CTime, 58
 - ASN1TTime, 111
- setNamespace
 - OSRTMessageBufferIF, 155
- setOwnership
 - OSRTSocket, 165
- setOwnMemory
 - OSRTBaseType, 123
- setRunTimeKey
 - ASN1Context, 38
 - ASN1CTYPE, 65
 - ASN1MessageBuffer, 78
 - OSRTContext, 127
- setSecond
 - ASN1CTime, 59
 - ASN1TTime, 113
- setStatus
 - ASN1MessageBuffer, 76
 - OSRTContext, 128
- setTime
 - ASN1CGeneralizedTime, 37
 - ASN1CTime, 61
 - ASN1CUTCTime, 71
 - ASN1TGeneralizedTime, 92
 - ASN1TTime, 114
 - ASN1TUTCTime, 120
- setUTC
 - ASN1CTime, 58
 - ASN1TTime, 111
 - ASN1TUTCTime, 120
- setValue
 - OSRTFastString, 132, 133
 - OSRTString, 176, 177
 - OSRTStringIF, 179
 - OSRTUTF8String, 183
- setYear
 - ASN1CTime, 58
 - ASN1TTime, 111
 - ASN1TUTCTime, 120
- shiftLeft
 - ASN1CBitStr, 32
- shiftRight
 - ASN1CBitStr, 33
- size
 - ASN1CBitStr, 27
 - ASN1CSeqOfList, 45
- skip
 - OSRTInputStream, 143
- startElement
 - Asn1NamedEventHandler, 81
- stringToAddr
 - OSRTSocket, 166
- TCP/IP or UDP Socket Classes, 18
- toArray
 - ASN1CSeqOfList, 46
- toHexString
 - ASN1TDynOctStr, 89
- toString
 - ASN1TDynOctStr, 89
 - ASN1TObjId, 97
 - ASN1TTime, 116, 117
- trim
 - ASN1TObjId, 98
- uInt64Value
 - Asn1NamedEventHandler, 82
- uIntValue
 - Asn1NamedEventHandler, 82
- unusedBitsInLastUnit
 - ASN1CBitStr, 33
- write
 - OSRTOutputStream, 159