ASN1C C++ Common Runtime Classes

Version 7.6

Objective Systems, Inc.

January 2022

ASN1C C++ Common Runtime Classes

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

Chapter 1. C++ Common 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. Module Documentation

C++ Run-Time Classes

Detailed Description

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.

Classes

- struct ASN1TTime

Modules

- OSRT Message Buffer Classes
- Control (ASN1C_) Base Classes
- ASN.1 Type (ASN1T_) Base Classes
- Date and Time Runtime Classes
- Generic Input Stream Classes
- Generic Output Stream Classes
- TCP/IP or UDP Socket Classes
- Event Handlers

OSRT Message Buffer Classes

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

- struct ASN1MessageBuffer
- struct OSRTMessageBuffer

- struct OSRTMessageBufferIF

Control (ASN1C_) Base Classes

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.

Classes

- struct ASN1CType
- struct ASN1CBitStrSizeHolder
- struct ASN1CBitStrSizeHolder8
- struct ASN1CBitStrSizeHolder16
- struct ASN1CBitStrSizeHolder32
- struct ASN1CBitStrSizeHolder64
- struct ASN1CBitStr
- struct ASN1CGeneralizedTime
- struct ASN1CUTCTime
- struct ASN1CSeqOfListIterator
- struct ASN1CSeqOfList

Variables

- class EXTRTCLASS ASN1CSeqOfList

ASN.1 Type (ASN1T_) Base Classes

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.

Classes

- struct ASN1TDynBitStr
- struct ASN1TDynBitStr64
- struct ASN1TBitStr32
- struct ASN1TBMPString

- struct ASN1TUniversalString
- struct ASN1TOpenType
- struct Asn1TObject
- struct ASN1TSeqExt
- struct ASN1TPDU
- struct ASN1TSeqOfList
- struct ASN1TPDUSeqOfList
- struct ASN1TObjId
- struct ASN1TDynOctStr
- struct ASN1TTime

Typedefs

- typedef Asn1TObject ASN1TObject

Functions

- int operator== (const ASN1OBJID & lhs, const ASN1OBJID & rhs)
- int operator== (const ASN1OBJID & 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 ASN1TObjId & lhs, const ASN1TObjId & rhs)
- int operator> (const ASN1TObjId & 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 ASN1TObjId & lhs, const ASN1TObjId & rhs)
- int operator>= (const ASN1TObjId & lhs, const char * dotted_oid_string)
- int operator>= (const ASN1OBJID & lhs, const ASN1OBJID & rhs)
- int operator>= (const ASN1OBJID & lhs, const char * dotted_oid_string)
- ASN1TObjId operator+ (const ASN1TObjId & lhs, const ASN1TObjId & rhs)
- 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)

Function Documentation

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

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

Table 2.1. Parameters

lhs	- C object identifier value.
rhs	- C object identifier value.

Returns: . - True if values are equal.

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.

Table 2.2. Parameters

lhs	- C object identifier value.
dotted_oid_string	- String containing OID value to compare.

Returns: . - True if values are equal.

int operator!= (const ASN1TObjId &lhs, const ASN1TObjId &rhs)

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

Table 2.3. Parameters

lhs	- C++ object identifier value.
rhs	- C++ object identifier value

Returns: . - True if values are equal.

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.

Table 2.4. Parameters

lhs	- C object identifier value.
rhs	- C object identifier value

Returns: . - True if values are equal.

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.

Table 2.5. Parameters

lhs	- C object identifier value.
dotted_oid_string	- String containing OID value to compare.

Returns: . - True if values are equal.

int operator!= (const ASN1TObjId &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.

Table 2.6. Parameters

lhs	- C++ object identifier value.
dotted_oid_string	- String containing OID value to compare.

Returns: . - True if values are equal.

int operator< (const ASN1TObjId &lhs, const ASN1TObjId &rhs)

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

Table 2.7. Parameters

lhs	- C++ object identifier value.
rhs	- C++ object identifier value.

Returns: . - True if values are equal.

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.

Table 2.8. Parameters

lhs	- C object identifier value.
rhs	- C object identifier value.

Returns: . - True if values are equal.

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.

Table 2.9. Parameters

lhs	- C object identifier value.
dotted_oid_string	- String containing OID value to compare.

Returns: . - True if values are equal.

int operator< (const ASN1TObjId &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.

Table 2.10. Parameters

lhs	- C++ object identifier value.
dotted_oid_string	- String containing OID value to compare.

Returns: . - True if values are equal.

int operator<= (const ASN1TObjId &lhs, const ASN1TObjId &rhs)

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

Table 2.11. Parameters

lhs	- C++ object identifier value.
rhs	- C++ object identifier value

Returns: . - True if values are equal.

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.

Table 2.12. Parameters

lhs	- C object identifier value.
rhs	- C object identifier value

Returns: . - True if values are equal.

int operator<= (const ASN1TObjId &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.

Table 2.13. Parameters

lhs	- C++ object identifier value.
dotted_oid_string	- String containing OID value to compare.

Returns: . - True if values are equal.

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.

Table 2.14. Parameters

lhs	- C object identifier value.
dotted_oid_string	- String containing OID value to compare.

Returns: . - True if values are equal.

int operator> (const ASN1TObjId &lhs, const ASN1TObjId &rhs)

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

Table 2.15. Parameters

lhs	- C++ object identifier value.
rhs	- C++ object identifier value.

Returns: . - True if values are equal.

int operator> (const ASN1TObjId &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.

Table 2.16. Parameters

lhs	- C++ object identifier value.
dotted_oid_string	- String containing OID value to compare.

Returns: . - True if values are equal.

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.

Table 2.17. Parameters

lhs	- C object identifier value.
rhs	- C object identifier value.

Returns: . - True if values are equal.

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.

Table 2.18. Parameters

lhs	- C object identifier value.
dotted_oid_string	- String containing OID value to compare.

Returns: . - True if values are equal.

int operator>= (const ASN1TObjId &lhs, const ASN1TObjId &rhs)

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

Table 2.19. Parameters

lhs	- C++ object identifier value.
rhs	- C++ object identifier value.

Returns: . - True if values are equal.

int operator>= (const ASN1TObjId &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.

Table 2.20. Parameters

lhs	- C++ object identifier value.

dotted_oid_string	- String containing OID value to compare.

Returns: . - True if values are equal.

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.

Table 2.21. Parameters

lhs	- C object identifier value.
rhs	- C object identifier value.

Returns: . - True if values are equal.

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.

Table 2.22. Parameters

lhs	- C object identifier value.
dotted_oid_string	- String containing OID value to compare.

Returns: . - True if values are equal.

ASN1TObjId operator+ (const ASN1TObjId &lhs, const ASN1TObjId &rhs)

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

Table 2.23. Parameters

lhs	- C++ object identifier value.
rhs	- C++ object identifier value.

int operator== (const ASN1TDynOctStr &lhs, const ASN1TDynOctStr &rhs)

The equality test operator: compares two dynamic octet strings to each other.

Table 2.24. Parameters

lhs	The left-hand ASN1TDynOctStr class.
rhs	The right-hand ASN1TDynOctStr class.

Returns: . 1 if the two octet strings are equal; 0 otherwise.

int operator== (const ASN1TDynOctStr &lhs, const char *string)

The equality test operator: compares a dynamic octet string against a character string.

Table 2.25. Parameters

lhs	The left-hand ASN1TDynOctStr class.
string	A character string to be compared against.

Returns: . 1 if the two strings are equal; 0 otherwise.

int operator== (const ASN1DynOctStr &lhs, const ASN1DynOctStr &rhs)

The equality test operator: compares two dynamic octet strings to each other.

Table 2.26. Parameters

lhs	The left-hand ASN1DynOctStr class.
rhs	The right-hand ASN1DynOctStr structure.

Returns: . 1 if the two octet strings are equal; 0 otherwise.

int operator== (const ASN1DynOctStr &lhs, const char *string)

The equality test operator: compares a dynamic octet string against a character string.

Table 2.27. Parameters

lhs	The left-hand ASN1DynOctStr class.
string	A character string to be compared against.

Returns: . 1 if the two strings are equal; 0 otherwise.

int operator!= (const ASN1TDynOctStr &lhs, const ASN1TDynOctStr &rhs)

The inequality test operator: compares two dynamic octet strings to each other.

Table 2.28. Parameters

lhs	The left-hand ASN1TDynOctStr class.
rhs	The right-hand ASN1TDynOctStr class.

Returns: . 1 if the two octet strings are equal; 0 otherwise.

int operator!= (const ASN1TDynOctStr &lhs, const char *string)

The inequality test operator: compares a dynamic octet string against a character string.

Table 2.29. Parameters

lhs	The left-hand ASN1TDynOctStr class.
string	A character string to be compared against.

Returns: . 1 if the two strings are equal; 0 otherwise.

int operator!= (const ASN1DynOctStr &lhs, const ASN1DynOctStr &rhs)

The inequality test operator: compares two dynamic octet strings to each other.

Table 2.30. Parameters

lhs	The left-hand ASN1DynOctStr class.
rhs	The right-hand ASN1DynOctStr structure.

Returns: . 1 if the two octet strings are equal; 0 otherwise.

int operator!= (const ASN1DynOctStr &lhs, const char *string)

The inequality test operator: compares a dynamic octet string against a character string.

Table 2.31. Parameters

lhs	The left-hand ASN1DynOctStr class.
string	A character string to be compared against.

Returns: . 1 if the two strings are equal; 0 otherwise.

int operator< (const ASN1TDynOctStr &lhs, const ASN1TDynOctStr &rhs)

The less-than test operator: compares two dynamic octet strings to each other.

Table 2.32. Parameters

lhs	The left-hand ASN1TDynOctStr class.
rhs	The right-hand ASN1TDynOctStr class.

Returns: . 1 if the two octet strings are equal; 0 otherwise.

int operator< (const ASN1TDynOctStr &lhs, const char *string)

The less-than test operator: compares a dynamic octet string against a character string.

Table 2.33. Parameters

lhs	The left-hand ASN1TDynOctStr class.
string	A character string to be compared against.

Returns: . 1 if the two strings are equal; 0 otherwise.

int operator< (const ASN1DynOctStr &lhs, const ASN1DynOctStr &rhs)

The inequality test operator: compares two dynamic octet strings to each other.

Table 2.34. Parameters

lhs	The left-hand ASN1DynOctStr class.
rhs	The right-hand ASN1DynOctStr structure.

Returns: . 1 if the two octet strings are equal; 0 otherwise.

int operator< (const ASN1DynOctStr &lhs, const char *string)

The inequality test operator: compares a dynamic octet string against a character string.

Table 2.35. Parameters

lhs	The left-hand ASN1DynOctStr class.
string	A character string to be compared against.

Returns: . 1 if the two strings are equal; 0 otherwise.

int operator<= (const ASN1TDynOctStr &lhs, const ASN1TDynOctStr &rhs)

The less-equal test operator: compares two dynamic octet strings to each other.

Table 2.36. Parameters

lhs	The left-hand ASN1TDynOctStr class.
rhs	The right-hand ASN1TDynOctStr class.

Returns: . 1 if the two octet strings are equal; 0 otherwise.

int operator<= (const ASN1TDynOctStr &lhs, const char *string)

The less-than test operator: compares a dynamic octet string against a character string.

Table 2.37. Parameters

lhs	The left-hand ASN1TDynOctStr class.
string	A character string to be compared against.

Returns: . 1 if the two strings are equal; 0 otherwise.

int operator<= (const ASN1DynOctStr &lhs, const ASN1DynOctStr &rhs)

The inequality test operator: compares two dynamic octet strings to each other.

Table 2.38. Parameters

lhs	The left-hand ASN1DynOctStr class.
rhs	The right-hand ASN1DynOctStr structure.

Returns: . 1 if the two octet strings are equal; 0 otherwise.

int operator<= (const ASN1DynOctStr &lhs, const char *string)

The inequality test operator: compares a dynamic octet string against a character string.

Table 2.39. Parameters

lhs	The left-hand ASN1DynOctStr class.
string	A character string to be compared against.

Returns: . 1 if the two strings are equal; 0 otherwise.

int operator> (const ASN1TDynOctStr &lhs, const ASN1TDynOctStr &rhs)

The greater-than test operator: compares two dynamic octet strings to each other.

Table 2.40. Parameters

lhs	The left-hand ASN1TDynOctStr class.
rhs	The right-hand ASN1TDynOctStr class.

Returns: . 1 if the two octet strings are equal; 0 otherwise.

int operator> (const ASN1TDynOctStr &lhs, const char *string)

The greater-than test operator: compares a dynamic octet string against a character string.

Table 2.41. Parameters

lhs	The left-hand ASN1TDynOctStr class.

string	A character string to be compared against.

Returns: . 1 if the two strings are equal; 0 otherwise.

int operator> (const ASN1DynOctStr &lhs, const ASN1DynOctStr &rhs)

The inequality test operator: compares two dynamic octet strings to each other.

Table 2.42. Parameters

lhs	The left-hand ASN1DynOctStr class.
rhs	The right-hand ASN1DynOctStr structure.

Returns: . 1 if the two octet strings are equal; 0 otherwise.

int operator> (const ASN1DynOctStr &lhs, const char *string)

The inequality test operator: compares a dynamic octet string against a character string.

Table 2.43. Parameters

lhs	The left-hand ASN1DynOctStr class.
string	A character string to be compared against.

Returns: . 1 if the two strings are equal; 0 otherwise.

int operator>= (const ASN1TDynOctStr &lhs, const ASN1TDynOctStr &rhs)

The greater-equal test operator: compares two dynamic octet strings to each other.

Table 2.44. Parameters

lhs	The left-hand ASN1TDynOctStr class.
rhs	The right-hand ASN1TDynOctStr class.

Returns: . 1 if the two octet strings are equal; 0 otherwise.

int operator>= (const ASN1TDynOctStr &lhs, const char *string)

The less-than test operator: compares a dynamic octet string against a character string.

Table 2.45. Parameters

lhs	The left-hand ASN1TDynOctStr class.
string	A character string to be compared against.

Returns: . 1 if the two strings are equal; 0 otherwise.

int operator>= (const ASN1DynOctStr &lhs, const ASN1DynOctStr &rhs)

The inequality test operator: compares two dynamic octet strings to each other.

Table 2.46. Parameters

lhs	The left-hand ASN1DynOctStr class.
rhs	The right-hand ASN1DynOctStr structure.

Returns: . 1 if the two octet strings are equal; 0 otherwise.

int operator>= (const ASN1DynOctStr &lhs, const char *string)

The inequality test operator: compares a dynamic octet string against a character string.

Table 2.47. Parameters

lhs	The left-hand ASN1DynOctStr class.
string	A character string to be compared against.

Returns: . 1 if the two strings are equal; 0 otherwise.

Date and Time Runtime Classes

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

- struct ASN1TTime
- struct ASN1CGeneralizedTime
- struct ASN1CUTCTime

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

Generic Input Stream Classes

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

- struct OSRTInputStream
- struct OSRTInputStreamIF
- struct OSRTInputStreamPtr

Generic Output Stream Classes

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

- struct OSRTOutputStream
- struct OSRTOutputStreamIF
- struct OSRTOutputStreamPtr

TCP/IP or UDP Socket Classes

Detailed Description

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

Classes

- struct OSRTSocket

Event Handlers

Detailed Description

Event Handler Classes include base classes from which user-defined error handler and event handler classes are derived. There are two kinds of event handlers:

- Named Event Handlers use element names and provide decoded values. Represented by class Asn1NamedEventHandler.
- Raw Event Handlers use identifiers and provide access to the underlying, raw data. Represented by Asn1RawEventHandler.

Classes

- struct Asn1NamedEventHandler
- struct Asn1NullEventHandler
- struct Asn1ErrorHandler

Variables

- class EXTRTCLASS ASN1MessageBuffer

Macros

- #define OS_UNUSED_ARG (void)arg

Variable Documentation

class EXTRTCLASS ASN1MessageBuffer

This definition is a forward reference to the ASN1MessageBuffer class.

Macro Definition Documentation

#define OS_UNUSED_ARG

This macro is used to prevent warnings of unused arguments passed to generated functions and methods.

Table 2.48. Parameters

arg	The argument to be passed.

Definition at line 63 of file asn1CppEvtHndlr.h

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

- asn1CppEvtHndlr.h

Context Management Classes

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

- struct ASN1Context

ASN.1 Stream Classes

Detailed Description

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

Classes

- struct OSRTStream

- struct OSRTStreamIF

Run-time error status codes.

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 in raised. This additional information can be output using the `rtxErrPrint` or `rtxErrLogUsingCB` run-time functions.

Macros

- #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)
- #define ASN_E_INVPERENC (ASN_E_BASE-10)
- #define ASN_E_NOTINSEQ (ASN_E_BASE-11)
- #define ASN_E_BAD_ALIGN (ASN_E_BASE-12)
- #define ASN_E_UNKNOWNPDU (ASN_E_BASE-13)
- #define ASN_E_NOTCANON (ASN_E_BASE-14)
- #define ASN_E_VALTYPMIS (ASN_E_BASE-15)
- #define ASN_E_LEN_FORM (ASN_E_BASE-16)
- #define ASN_E_LEN_NOT_MIN (ASN_E_BASE-17)
- #define ASN_E_PRIM_REQ (ASN_E_BASE-18)
- #define ASN_E_FRAGMENTS (ASN_E_BASE-19)

Macro Definition Documentation

#define ASN_OK_FRAG

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.

Definition at line 50 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_BASE

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

Definition at line 56 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_INVOBJID

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).

Definition at line 66 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_INVLEN

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.

Definition at line 74 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_BADTAG

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.

Definition at line 81 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_INVBINS

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.

Definition at line 88 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_INVINDEX

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.

Definition at line 95 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_INVTCVAL

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.

Definition at line 102 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_CONCMODF

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.

Definition at line 109 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_ILLSTATE

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.

Definition at line 118 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_NOTPDU

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.

Definition at line 125 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_UNDEFTYP

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.

Definition at line 132 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_INVPERENC

Invalid PER encoding. This occurs when a given element within an ASN.1 specification is configured to have an expected PER encoding and the decoded value does not match this encoding.

Definition at line 139 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_NOTINSEQ

Element not in sequence. This occurs when an element is parsed within a SEQUENCE but is found to not match any of the elements defined for that SEQUENCE.

Definition at line 146 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_BAD_ALIGN

Encoding has bad alignment. This occurs when an encoding is expected to align with a byte boundary and does not. It may happen with reference to the start of the message or with reference to the start of a length-constrained container. Typically this means some bit string element has too few bits, where the encoding of that element is supposed to end the encoding so that padding cannot be used to acheive the desired alignment.

Definition at line 157 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_UNKNOWNPDU

Unknown PDU type. This error occurs in interpreted BER decoders when the PDU type for the message can't be automatically determined. The user in this case must manually specify the PDU type by some external means. For example, if the application has a command-line interface, it may have a '-pdu' option for specifying the PDU type.

Definition at line 166 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_NOTCANON

Encoded data does not conform to canonical rules for the current encoding rules.

Definition at line 172 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_VALTYPMIS

Value/type mismatch. The given value is not valid for the declared type.

Definition at line 177 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_LEN_FORM

Incorrect length form for CER/DER.

Definition at line 182 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_LEN_NOT_MIN

Length does not use minimal # of octets.

Definition at line 187 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_PRIM_REQ

Primitive form encoding is required.

Definition at line 192 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

#define ASN_E_FRAGMENTS

A value is not correctly fragmented as required by the given encoding rules.

Definition at line 198 of file asn1ErrCodes.h

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

- asn1ErrCodes.h

Chapter 3. Class Documentation

ASN1BitStr32 class Reference

ASN1BMPString class Reference

ASN1CBitStr class Reference

`#include <ASN1CBitStr.h>`

Private Attributes

- OSSIZE _numbits
- OSOCTET * _units

Protected Attributes

- OSOCTET ** mpUnits
- OSOCTET ** mppExtData
- OSSIZE mMaxNumBits
- ASN1CBitStrSizeHolder * mpNumBits
- OSSIZE mUnitsUsed
- OSSIZE mUnitsAllocated
- OSBOOL mDynAlloc

- OSOCTET * allocateMemory (OSSIZE sz)
- OSOCTET * reallocateMemory (OSOCTET * old, OSSIZE oldBufSz, OSSIZE newBufSz)
- void freeMemory (OSOCTET * mem)
- void recalculateUnitsUsed ( )
- int checkCapacity (OSSIZE unitsRequired)
- OSOCTET getBits (OSSIZE j)
- void privateInit (OSSIZE nbits)
- void privateInit (OSOCTET * bitStr, OSSIZE maxNumbits_, OSOCTET ** ppExtData)
- void privateInit (ASN1TDynBitStr & bitStr)
- void privateInit (ASN1TDynBitStr64 & bitStr)

- EXTRTMETHOD ASN1CBitStr (OSRTMessageBufferIF & msgBuf)
- EXTRTMETHOD ASN1CBitStr ( )
- EXTRTMETHOD ASN1CBitStr (OSRTContext & ctxt)
- EXTRTMETHOD ASN1CBitStr (OSOCTET * pBits, OSUINT32 & numbits, OSSIZE maxNumbits, OSOCTET ** ppExtData)
- EXTRTMETHOD ASN1CBitStr (OSOCTET * pBits, OSUINT8 & numbits, OSSIZE maxNumbits, OSOCTET ** ppExtData)
- EXTRTMETHOD ASN1CBitStr (OSOCTET * pBits, OSUINT16 & numbits, OSSIZE maxNumbits, OSOCTET ** ppExtData)
- EXTRTMETHOD ASN1CBitStr (OSOCTET * pBits, OSSIZE maxNumbits)
- EXTRTMETHOD ASN1CBitStr (ASN1TDynBitStr & bitStr)
- EXTRTMETHOD ASN1CBitStr (ASN1TDynBitStr64 & bitStr)
- void initBase (OSOCTET * pBits, OSSIZE numbits, OSSIZE maxNumbits)
- EXTRTMETHOD void init (OSOCTET * pBits, OSUINT32 & numbits, OSSIZE maxNumbits)
- EXTRTMETHOD void init (OSOCTET * pBits, OSUINT8 & numbits, OSSIZE maxNumbits)
- EXTRTMETHOD void init (OSOCTET * pBits, OSUINT16 & numbits, OSSIZE maxNumbits)
- EXTRTMETHOD void init (ASN1TDynBitStr & bitStr)

- EXTRTMETHOD ASN1CBitStr (OSRTMessageBufferIF & msgbuf, OSSIZE nbits)
- EXTRTMETHOD ASN1CBitStr (OSRTMessageBufferIF & msgbuf, OSOCTET * bitStr, OSUINT32 & numbits, OSSIZE maxNumbits_, OSOCTET ** ppExtData)
- EXTRTMETHOD ASN1CBitStr (OSRTMessageBufferIF & msgbuf, OSOCTET * bitStr, OSUINT8 & numbits, OSSIZE maxNumbits_, OSOCTET ** ppExtData)
- EXTRTMETHOD ASN1CBitStr (OSRTMessageBufferIF & msgbuf, OSOCTET * bitStr, OSUINT16 & numbits, OSSIZE maxNumbits_, OSOCTET ** ppExtData)
- EXTRTMETHOD ASN1CBitStr (OSRTMessageBufferIF & msgbuf, OSOCTET * bitStr, OSSIZE maxNumbits_)
- EXTRTMETHOD ASN1CBitStr (OSRTMessageBufferIF & msgBuf, ASN1TDynBitStr & bitStr)
- EXTRTMETHOD ASN1CBitStr (OSRTMessageBufferIF & msgBuf, ASN1TDynBitStr64 & bitStr)
- EXTRTMETHOD ASN1CBitStr (OSRTContext & ctxt, OSSIZE nbits)
- EXTRTMETHOD ASN1CBitStr (OSRTContext & ctxt, OSOCTET * bitStr, OSUINT32 & octsNumbits, OSSIZE maxNumbits_, OSOCTET ** ppExtData)
- EXTRTMETHOD ASN1CBitStr (OSRTContext & ctxt, OSOCTET * bitStr, OSUINT8 & octsNumbits, OSSIZE maxNumbits_, OSOCTET ** ppExtData)
- EXTRTMETHOD ASN1CBitStr (OSRTContext & ctxt, OSOCTET * bitStr, OSUINT16 & octsNumbits, OSSIZE maxNumbits_, OSOCTET ** ppExtData)

- EXTRTMETHOD ASN1CBitStr (OSRTContext & ctxt, OSOCTET * bitStr, OSSIZE maxNumbits_)
- EXTRTMETHOD ASN1CBitStr (OSRTContext & ctxt, ASN1TDynBitStr & bitStr)
- EXTRTMETHOD ASN1CBitStr (OSRTContext & ctxt, ASN1TDynBitStr64 & bitStr)
- EXTRTMETHOD ASN1CBitStr (const ASN1CBitStr & bitStr)
- EXTRTMETHOD ASN1CBitStr (const ASN1CBitStr & bitStr, OSBOOL extendable)
- EXTRTMETHOD ~ASN1CBitStr ( )
- EXTRTMETHOD int set (OSSIZE bitIndex)
- EXTRTMETHOD int set (OSSIZE fromIndex, OSSIZE toIndex)
- int change (OSSIZE bitIndex, OSBOOL value)
- EXTRTMETHOD int clear (OSSIZE bitIndex)
- EXTRTMETHOD int clear (OSSIZE fromIndex, OSSIZE toIndex)
- EXTRTMETHOD void clear ( )
- EXTRTMETHOD int invert (OSSIZE bitIndex)
- EXTRTMETHOD int invert (OSSIZE fromIndex, OSSIZE toIndex)
- EXTRTMETHOD OSBOOL get (OSSIZE bitIndex)
- OSBOOL isSet (OSSIZE bitIndex)
- OSBOOL isEmpty ( )
- EXTRTMETHOD OSSIZE size ( )
- EXTRTMETHOD OSSIZE length ( )
- EXTRTMETHOD OSSIZE cardinality ( )
- EXTRTMETHOD int getBytes (OSOCTET * pBuf, OSSIZE bufSz)
- EXTRTMETHOD int get (OSSIZE fromIndex, OSSIZE toIndex, OSOCTET * pBuf, OSSIZE bufSz)
- EXTRTMETHOD int doAnd (const OSOCTET * pOctstr, OSSIZE octsNumbits)
- int doAnd (const ASN1TDynBitStr & bitStr)
- int doAnd (const ASN1CBitStr & bitStr)
- EXTRTMETHOD int doOr (const OSOCTET * pOctstr, OSSIZE octsNumbits)
- int doOr (const ASN1TDynBitStr & bitStr)
- int doOr (const ASN1CBitStr & bitStr)
- EXTRTMETHOD int doXor (const OSOCTET * pOctstr, OSSIZE octsNumbits)
- int doXor (const ASN1TDynBitStr & bitStr)
- int doXor (const ASN1CBitStr & bitStr)

- EXTRTMETHOD int doAndNot (const OSOCTET * pOctstr, OSSIZE octsNumbits)
- int doAndNot (const ASN1TDynBitStr & bitStr)
- int doAndNot (const ASN1CBitStr & bitStr)
- EXTRTMETHOD int shiftLeft (OSSIZE shift)
- EXTRTMETHOD int shiftRight (OSSIZE shift)
- EXTRTMETHOD OSUINT32 unusedBitsInLastUnit ( )
- EXTRTMETHOD operator ASN1TDynBitStr ( )
- EXTRTMETHOD operator ASN1TDynBitStr * ( )

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.

Definition at line 169 of file ASN1CBitStr.h

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

- ASN1CBitStr.h

EXTRTMETHOD ASN1CBitStr::ASN1CBitStr (OSRTMessageBufferIF &msgbuf, OSSIZE 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`.

Table 3.1. Parameters

msgbuf	- ASN.1 message buffer or stream object.
nbits	- Number of bits this bit string can contain (zero if unbounded.

EXTRTMETHOD ASN1CBitStr::ASN1CBitStr (OSRTMessageBufferIF &msgbuf, OSOCTET *bitStr, OSUINT32 &numbits, OSSIZE maxNumbits_, OSOCTET **ppExtData=0)

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.

Table 3.2. 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

EXTRTMETHOD int ASN1CBitStr::set (OSSIZE bitIndex)

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

Table 3.3. Parameters

bitIndex	Relative index of the bit to set in the string. The bit idex 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.

EXTRTMETHOD int ASN1CBitStr::set (OSSIZE fromIndex, OSSIZE toIndex)

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

Table 3.4. 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.

int ASN1CBitStr::change (OSSIZE bitIndex, OSBOOL value)

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

Table 3.5. 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.

EXTRTMETHOD int ASN1CBitStr::clear (OSSIZE bitIndex)

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

Table 3.6. 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.

EXTRTMETHOD int ASN1CBitStr::clear (OSSIZE fromIndex, OSSIZE toIndex)

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

Table 3.7. 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.

EXTRTMETHOD void ASN1CBitStr::clear ()

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

Table 3.8. Parameters

-	none

Returns: . Completion status of operation:

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

EXTRTMETHOD int ASN1CBitStr::invert (OSSIZE 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.

Table 3.9. 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.

EXTRTMETHOD int ASN1CBitStr::invert (OSSIZE fromIndex, OSSIZE 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.

Table 3.10. 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.

EXTRTMETHOD OSBOOL ASN1CBitStr::get (OSSIZE bitIndex)

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

Table 3.11. 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.

OSBOOL ASN1CBitStr::isSet (OSSIZE bitIndex)

This method is the same as ASN1CBitStr::get.

See also: . get (OSSIZE bitIndex)

OSBOOL ASN1CBitStr::isEmpty ()

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

Table 3.12. Parameters

-	none

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

EXTRTMETHOD OSSIZE ASN1CBitStr::size () const

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

Table 3.13. Parameters

-	none

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

EXTRTMETHOD OSSIZE ASN1CBitStr::length () const

This method calculates the "logical size" of the bit 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 least significant bit in the last octet set to 1.

Table 3.14. Parameters

-	none

Returns: . Returns the "logical size" of this bit string.

EXTRTMETHOD OSSIZE 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.

Table 3.15. Parameters

-	none

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

EXTRTMETHOD int ASN1CBitStr::getBytes (OSOCTET *pBuf, OSSIZE bufSz)

This method copies the bit string to the given buffer.

Table 3.16. 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.

EXTRTMETHOD int ASN1CBitStr::get (OSSIZE fromIndex, OSSIZE toIndex, OSOCTET *pBuf, OSSIZE 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.

Table 3.17. 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).

EXTRTMETHOD int ASN1CBitStr::doAnd (const OSOCTET *pOctstr, OSSIZE octsNumbits)

Performs a logical AND of this target bit set with the argument bit set.

Returns: 0 - if succeed, or ASN_E_INVLEN - if 'octsNumbits' is negative, or RTERR_INVPARAM - if pOctstr is the same bit string as this or null, or other error codes (see asn1type.h).

Table 3.18. 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.

int ASN1CBitStr::doAnd (const ASN1TDynBitStr &bitStr)

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

Table 3.19. 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.

int ASN1CBitStr::doAnd (const ASN1CBitStr &bitStr)

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

Table 3.20. 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.

EXTRTMETHOD int ASN1CBitStr::doOr (const OSOCTET *pOctstr, OSSIZE octsNumbits)

Performs a logical OR of this target bit set with the argument bit set.

Returns: 0 - if succeed, or ASN_E_INVLEN - if 'octsNumbits' is negative, or RTERR_INVPARAM - if pOctstr is the same bit string as this or null, or other error codes (see asn1type.h).

Table 3.21. 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.

int ASN1CBitStr::doOr (const ASN1TDynBitStr &bitStr)

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

Table 3.22. 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.

int ASN1CBitStr::doOr (const ASN1CBitStr &bitStr)

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

Table 3.23. 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.

EXTRTMETHOD int ASN1CBitStr::doXor (const OSOCTET *pOctstr, OSSIZE octsNumbits)

Performs a logical XOR of this target bit set with the argument bit set.

Returns: 0 - if succeed, or ASN_E_INVLEN - if 'octsNumbits' is negative, or RTERR_INVPARAM - if pOctstr is null, or other error codes (see asn1type.h).

Table 3.24. 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.

int ASN1CBitStr::doXor (const ASN1TDynBitStr &bitStr)

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

Table 3.25. 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.

int ASN1CBitStr::doXor (const ASN1CBitStr &bitStr)

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

Table 3.26. 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.

EXTRTMETHOD int ASN1CBitStr::doAndNot (const OSOCTET *pOctstr, OSSIZE octsNumbits)

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

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

Table 3.27. 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.

int ASN1CBitStr::doAndNot (const ASN1TDynBitStr &bitStr)

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.

Table 3.28. 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.

int ASN1CBitStr::doAndNot (const ASN1CBitStr &bitStr)

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.

Table 3.29. 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.

EXTRTMETHOD int ASN1CBitStr::shiftLeft (OSSIZE 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.

Table 3.30. Parameters

shift	Number of bits to be shifted.

Returns: . Completion status of operation:

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

EXTRTMETHOD int ASN1CBitStr::shiftRight (OSSIZE 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.

Table 3.31. Parameters

shift	Number of bits to be shifted.

Returns: . Completion status of operation:

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

EXTRTMETHOD OSUINT32 ASN1CBitStr::unusedBitsInLastUnit ()

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

Returns: . Number of bits in the last octet. It is equal to length() % 8.

EXTRTMETHOD 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, and could become invalid if this object is modified. Upon return, retval.data will point to this bit string and retval.numbits will equal length().

Do not call this function if length() is outside the range of OSUINT32.

Table 3.32. Parameters

-	none

Returns: . Filled ASN1TDynBitStr.

EXTRTMETHOD ASN1CBitStr::operator ASN1TDynBitStr * ()

This method returns a pointer to a newly allocated ASN1TDynBitStr object. Upon return, retval->data will point to newly allocated copy of the data in this bit string and retval->numbits will equal length().

Do not call this function if length() is outside the range of OSUINT32.

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

Table 3.33. Parameters

-	none

Returns: . Pointer to a filled ASN1TDynBitStr.

ASN1CBitStrSizeHolder class Reference

`#include <ASN1CBitStr.h>`

- virtual ASN1CBitStrSizeHolder * clone ( )
- virtual OSSIZE getValue ( )
- virtual int setValue (OSSIZE value)
- virtual ~ASN1CBitStrSizeHolder ( )

Detailed Description

The ASN1CBitStrSizeHolder is a class used to hold sizes for the ASN1CBitStr control class. This base class is abstract and is implemented by the 8-bit, 16-bit, and 32-bit varieties.

Definition at line 51 of file ASN1CBitStr.h

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

- ASN1CBitStr.h

virtual int ASN1CBitStrSizeHolder::setValue (OSSIZE value)=0

Set the size to the given value. Implementations of this method will return an error if the given value is outside of the range of the field being used to hold the size value.

ASN1CBitStrSizeHolder16 class Reference

`#include <ASN1CBitStr.h>`

Protected Attributes

- OSUINT16 & mSize

- ASN1CBitStrSizeHolder16 (OSUINT16 & value)
- virtual ASN1CBitStrSizeHolder * clone ( )
- virtual OSSIZE getValue ( )
- virtual int setValue (OSSIZE value)
- virtual ~ASN1CBitStrSizeHolder16 ( )

- ASN1CBitStrSizeHolder16 & operator= (const ASN1CBitStrSizeHolder16 &)

Detailed Description

This is the 16-bit implementation of the ASN1CBitStrSizeHolder class.

Definition at line 93 of file ASN1CBitStr.h

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

- ASN1CBitStr.h

virtual int ASN1CBitStrSizeHolder16::setValue (OSSIZE value)

Set the size to the given value. Implementations of this method will return an error if the given value is outside of the range of the field being used to hold the size value.

ASN1CBitStrSizeHolder32 class Reference

`#include <ASN1CBitStr.h>`

Protected Attributes

- OSUINT32 & mSize

- ASN1CBitStrSizeHolder32 (OSUINT32 & value)
- virtual ASN1CBitStrSizeHolder * clone ( )
- virtual OSSIZE getValue ( )
- virtual int setValue (OSSIZE value)
- virtual ~ASN1CBitStrSizeHolder32 ( )

- ASN1CBitStrSizeHolder32 & operator= (const ASN1CBitStrSizeHolder32 &)

Detailed Description

This is the 32-bit implementation of the ASN1CBitStrSizeHolder class.

Definition at line 116 of file ASN1CBitStr.h

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

- ASN1CBitStr.h

virtual int ASN1CBitStrSizeHolder32::setValue (OSSIZE value)

Set the size to the given value. Implementations of this method will return an error if the given value is outside of the range of the field being used to hold the size value.

ASN1CBitStrSizeHolder64 class Reference

`#include <ASN1CBitStr.h>`

Protected Attributes

- OSSIZE & mSize

- ASN1CBitStrSizeHolder64 (OSSIZE & value)
- virtual ASN1CBitStrSizeHolder * clone ( )
- virtual OSSIZE getValue ( )
- virtual int setValue (OSSIZE value)
- virtual ~ASN1CBitStrSizeHolder64 ( )

- ASN1CBitStrSizeHolder64 & operator= (const ASN1CBitStrSizeHolder64 &)

Detailed Description

This is the OSSIZE implementation of the ASN1CBitStrSizeHolder class.

Definition at line 139 of file ASN1CBitStr.h

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

- ASN1CBitStr.h

virtual int ASN1CBitStrSizeHolder64::setValue (OSSIZE value)

Set the size to the given value. Implementations of this method will return an error if the given value is outside of the range of the field being used to hold the size value.

ASN1CBitStrSizeHolder8 class Reference

`#include <ASN1CBitStr.h>`

Protected Attributes

- OSUINT8 & mSize

- ASN1CBitStrSizeHolder8 (OSUINT8 & value)
- virtual ASN1CBitStrSizeHolder * clone ( )
- virtual OSSIZE getValue ( )

- virtual int setValue (OSSIZE value)
- virtual ~ASN1CBitStrSizeHolder8 ( )
- ASN1CBitStrSizeHolder8 & operator= (const ASN1CBitStrSizeHolder8 &)

Detailed Description

This is the 8-bit implementation of the ASN1CBitStrSizeHolder class.

Definition at line 70 of file ASN1CBitStr.h

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

- ASN1CBitStr.h

virtual int ASN1CBitStrSizeHolder8::setValue (OSSIZE value)

Set the size to the given value. Implementations of this method will return an error if the given value is outside of the range of the field being used to hold the size value.

ASN1CGeneralizedTime class Reference

`#include <ASN1CGeneralizedTime.h>`

Protected Attributes

- ASN1TGeneralizedTime timeObj
- virtual ASN1TTime & getTimeObj ( )
- virtual const ASN1TTime & getTimeObj ( )
- EXTRTMETHOD ASN1CGeneralizedTime (char *& buf, int bufSize, OSBOOL useDerRules)
- EXTRTMETHOD ASN1CGeneralizedTime (ASN1GeneralizedTime & buf, OSBOOL useDerRules)
- EXTRTMETHOD int compileString ( )
- EXTRTMETHOD ASN1CGeneralizedTime (OSRTMessageBufferIF & msgBuf, char *& buf, int bufSize, OSBOOL useDerRules)
- EXTRTMETHOD ASN1CGeneralizedTime (OSRTMessageBufferIF & msgBuf, ASN1GeneralizedTime & buf, OSBOOL useDerRules)
- EXTRTMETHOD ASN1CGeneralizedTime (OSRTContext & ctxt, char *& buf, int bufSize, OSBOOL useDerRules)
- EXTRTMETHOD ASN1CGeneralizedTime (OSRTContext & ctxt, ASN1GeneralizedTime & buf, OSBOOL useDerRules)

- ASN1CGeneralizedTime (const ASN1CGeneralizedTime & original)
- EXTRTMETHOD int getCentury ( )
- EXTRTMETHOD int setCentury (short century)
- EXTRTMETHOD int setTime (time_t time, OSBOOL diffTime)
- const ASN1CGeneralizedTime & operator= (const ASN1CGeneralizedTime & tm)

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).

Definition at line 64 of file ASN1CGeneralizedTime.h

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

- ASN1CGeneralizedTime.h

EXTRTMETHOD int ASN1CGeneralizedTime::compileString ()

Compiles new time string accoring X.680 (clause 41) and ISO 8601. Returns 0, if succeed, or error code, if error.

Returns: . 0 on success, or an error code otherwise.

EXTRTMETHOD 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.

Table 3.34. 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.

EXTRTMETHOD 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 complier-generated time string variable.

Table 3.35. 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.

EXTRTMETHOD ASN1CGeneralizedTime::ASN1CGeneralizedTime (OSRTContext &ctxt, char *&buf, int bufSize, OSBOOL useDerRules=FALSE)

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.

Table 3.36. Parameters

ctxt	Reference to an OSRTContext data structure.
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.

EXTRTMETHOD ASN1CGeneralizedTime::ASN1CGeneralizedTime (OSRTContext &ctxt, ASN1GeneralizedTime &buf, OSBOOL useDerRules=FALSE)

This constructor creates a time string from an ::ASN1GeneralizedTime object.

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.

Table 3.37. Parameters

ctxt	Reference to an OSRTContext data structure.
buf	Reference to a pointer to a time string buffer.
useDerRules	Use the Distinguished Encoding Rules (DER) to operate on this time value.

ASN1CGeneralizedTime::ASN1CGeneralizedTime (const ASN1CGeneralizedTime &original)

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

Table 3.38. Parameters

original	The original time string object value.

EXTRTMETHOD int ASN1CGeneralizedTime::getCentury ()

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

Table 3.39. Parameters

-	none

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

EXTRTMETHOD int ASN1CGeneralizedTime::setCentury (short century)

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

Table 3.40. Parameters

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

Returns: . Completion status of operation:

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

EXTRTMETHOD int ASN1CGeneralizedTime::setTime (time_t time, OSBOOL diffTime)

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.

Table 3.41. 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.

ASN1Context class Reference

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

- EXTRTMETHOD ASN1Context ( )
- virtual EXTRTMETHOD int setRunTimeKey (const OSOCTET * key, size_t keylen)
- OSCTXT * GetPtr ( )
- void PrintErrorInfo ( )

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.

Definition at line 47 of file ASN1Context.h

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

- ASN1Context.h

EXTRTMETHOD ASN1Context::ASN1Context ()

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

virtual EXTRTMETHOD int ASN1Context::setRunTimeKey (const OSOCTET *key, size_t keylen)

This method sets run-time key in the context. When using an unlimited runtime, this method will still set the key, but the runtime does not actually check it.

Table 3.42. 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.

ASN1CSeqOfList class Reference

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

•

Protected Attributes

- OSRTDList * pList
- volatile int modCount
- OSBOOL wasAssigned

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

- EXTRTMETHOD ASN1CSeqOfList (OSRTMessageBufferIF & msgBuf, OSRTDList & lst, OSBOOL initBeforeUse)
- EXTRTMETHOD ASN1CSeqOfList (OSRTMessageBufferIF & msgBuf)

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

- EXTRTMETHOD ASN1CSeqOfListIterator * iteratorFrom (void * data)
- EXTRTMETHOD void * toArray (OSSIZE elemSize)
- EXTRTMETHOD void * toArray (void * pArray, OSSIZE elemSize, OSSIZE allocatedElems)
- void * operator[] (int index)
- operator OSRTDList * ( )

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.

Definition at line 204 of file ASN1CSeqOfList.h

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

- ASN1CSeqOfList.h

EXTRTMETHOD 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.

Table 3.43. 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).

EXTRTMETHOD ASN1CSeqOfList::ASN1CSeqOfList (OSRTMessageBufferIF &msgBuf)

This constructor creates an empty linked list.

Table 3.44. Parameters

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

EXTRTMETHOD ASN1CSeqOfList::ASN1CSeqOfList (ASN1CType &ccobj)

This constructor creates an empty linked list.

Table 3.45. Parameters

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

EXTRTMETHOD 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.

Table 3.46. Parameters

msgBuf	Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).
lst	Reference to a linked list holder.

EXTRTMETHOD 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.

Table 3.47. Parameters

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

EXTRTMETHOD 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.

Table 3.48. Parameters

msgBuf	Reference to an ASN1Message buffer derived object (for example, an ASN1BEREncodeBuffer).
lst	Reference to a linked list holder.

EXTRTMETHOD 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.

Table 3.49. Parameters

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

Returns: . - none

EXTRTMETHOD void ASN1CSeqOfList::appendArray (const void *data, OSSIZE numElems, OSSIZE 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.

Table 3.50. 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

EXTRTMETHOD void ASN1CSeqOfList::appendArrayCopy (const void *data, OSSIZE numElems, OSSIZE 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.

Table 3.51. 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

void ASN1CSeqOfList::init ()

This method initializes the linked list structure.

EXTRTMETHOD 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.

Table 3.52. 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

EXTRTMETHOD 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.

Table 3.53. Parameters

index	Index of the item to be removed.

Returns: . - none

EXTRTMETHOD 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.

Table 3.54. Parameters

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

void ASN1CSeqOfList::removeFirst ()

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

Table 3.55. Parameters

-	none

Returns: . - none

void ASN1CSeqOfList::removeLast ()

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

Table 3.56. Parameters

-	none

Returns: . - none

EXTRTMETHOD 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.

Table 3.57. 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.

OSBOOL ASN1CSeqOfList::contains (void *data) const

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.

Table 3.58. Parameters

data	- Pointer to data item.

Returns: . TRUE if this pointer value found in the list.

EXTRTMETHOD 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.

EXTRTMETHOD 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.

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

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

Table 3.59. Parameters

index	Index of the item to be returned.

Returns: . The item at the specified index in the list.

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

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

Table 3.60. 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.

EXTRTMETHOD void ASN1CSeqOfList::clear ()

This method removes all items from the list.

virtual void ASN1CSeqOfList::freeMemory ()

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

EXTRTMETHOD OSBOOL ASN1CSeqOfList::isEmpty () const

This method returns TRUE if the list is empty.

Returns: . TRUE if this list is empty.

EXTRTMETHOD OSSIZE ASN1CSeqOfList::size () const

This method returns the number of nodes in the list.

Returns: . The number of items in this list.

EXTRTMETHOD ASN1CSeqOfListIterator* ASN1CSeqOfList::iterator ()

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

Returns: . The iterator over this linked list.

EXTRTMETHOD ASN1CSeqOfListIterator* ASN1CSeqOfList::iteratorFromLast ()

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

Table 3.61. Parameters

-	none

Returns: . The reverse iterator over this linked list.

EXTRTMETHOD 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.

Table 3.62. Parameters

data	The item of the list to be iterated first.

Returns: . The iterator over this linked list.

EXTRTMETHOD void* ASN1CSeqOfList::toArray (OSSIZE elemSize)

This method converts the linked list into a new array.

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

Table 3.63. Parameters

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

Returns: . The point to converted array.

EXTRTMETHOD void* ASN1CSeqOfList::toArray (void *pArray, OSSIZE elemSize, OSSIZE 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.

Table 3.64. 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.

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

This method is the overloaded operator[].

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

See also: . get (int index)

ASN1CSeqOfListIterator class Reference

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

•

Protected Attributes

- ASN1CSeqOfList * pSeqList
- OSRTDListNode * nextNode
- OSRTDListNode * lastNode
- volatile int expectedModCount
- int stat

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

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

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.

Definition at line 75 of file ASN1CSeqOfList.h

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

- ASN1CSeqOfList.h

OSBOOL ASN1CSeqOfListIterator::hasNext ()

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.

OSBOOL ASN1CSeqOfListIterator::hasPrev ()

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.

EXTRTMETHOD 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.

EXTRTMETHOD 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.

Table 3.65. Parameters

-	none

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

EXTRTMETHOD 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.

Table 3.66. Parameters

-	none

Returns: . Completion status of operation:

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

EXTRTMETHOD 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.

Table 3.67. 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.

EXTRTMETHOD 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.

Table 3.68. Parameters

data	The element to be inserted

Returns: . Completion status of operation:

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

ASN1CTime class Reference

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

- enum @1 {

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
}

Protected Attributes

- OSBOOL parsed
- OSBOOL derRules
- char *& timeStr
- OSSIZE strSize

Private Attributes

- char timeStrBuf[MAX_TIMESTR_SIZE]
- char * pTimeStr

- EXTRTMETHOD void checkCapacity ( )
- EXTRTMETHOD char *& getTimeStringPtr ( )
- virtual ASN1TTime & getTimeObj ( )
- virtual const ASN1TTime & getTimeObj ( )
- EXTRTMETHOD ASN1CTime (char *& buf, OSSIZE bufSize, OSBOOL useDerRules)
- EXTRTMETHOD ASN1CTime (ASN1VisibleString & buf, OSBOOL useDerRules)
- virtual int compileString ( )

- EXTRTMETHOD void privateInit ( )
- EXTRTMETHOD ASN1CTime (OSRTMessageBufferIF & msgBuf, char *& buf, OSSIZE bufSize, OSBOOL useDerRules)
- EXTRTMETHOD ASN1CTime (OSRTMessageBufferIF & msgBuf, ASN1VisibleString & buf, OSBOOL useDerRules)
- EXTRTMETHOD ASN1CTime (OSRTContext & ctxt, char *& buf, OSSIZE bufSize, OSBOOL useDerRules)
- EXTRTMETHOD ASN1CTime (OSRTContext & ctxt, ASN1VisibleString & buf, OSBOOL useDerRules)
- EXTRTMETHOD ASN1CTime (const ASN1CTime & original)
- EXTRTMETHOD ~ASN1CTime ( )
- virtual EXTRTMETHOD int getYear ( )
- virtual EXTRTMETHOD int getMonth ( )
- virtual EXTRTMETHOD int getDay ( )
- virtual EXTRTMETHOD int getHour ( )
- virtual EXTRTMETHOD int getMinute ( )
- virtual EXTRTMETHOD int getSecond ( )
- virtual EXTRTMETHOD int getFraction ( )
- virtual EXTRTMETHOD double getFractionAsDouble ( )
- virtual EXTRTMETHOD int getFractionStr (char *const pBuf, size_t bufSize)
- virtual EXTRTMETHOD int getFractionLen ( )
- virtual EXTRTMETHOD int getDiffHour ( )
- virtual EXTRTMETHOD int getDiffMinute ( )
- virtual EXTRTMETHOD int getDiff ( )
- virtual EXTRTMETHOD OSBOOL getUTC ( )
- virtual EXTRTMETHOD time_t getTime ( )
- void setDER (OSBOOL bvalue)
- virtual EXTRTMETHOD int setUTC (OSBOOL utc)
- virtual EXTRTMETHOD int setYear (short year_)
- virtual EXTRTMETHOD int setMonth (short month_)
- virtual EXTRTMETHOD int setDay (short day_)
- virtual EXTRTMETHOD int setHour (short hour_)

- virtual EXTRTMETHOD int setMinute (short minute_)
- virtual EXTRTMETHOD int setSecond (short second_)
- virtual EXTRTMETHOD int setFraction (int fraction, int fracLen)
- virtual EXTRTMETHOD int setFraction (double frac, int fracLen)
- virtual EXTRTMETHOD int setFraction (char const * frac)
- virtual int setTime (time_t time, OSBOOL diffTime)
- virtual EXTRTMETHOD int setDiffHour (short dhour)
- virtual EXTRTMETHOD int setDiff (short dhour, short dminute)
- virtual EXTRTMETHOD int setDiff (short inMinutes)
- virtual EXTRTMETHOD int parseString (const char * string)
- virtual EXTRTMETHOD void clear ( )
- virtual EXTRTMETHOD int equals (ASN1CTime &)
- EXTRTMETHOD OSSIZE getTimeStringLen ( )
- EXTRTMETHOD const char * getTimeString (char * pbuf, OSSIZE bufsize)
- EXTRTMETHOD const ASN1CTime & operator= (const ASN1CTime &)
- virtual EXTRTMETHOD OSBOOL operator== (ASN1CTime &)
- virtual EXTRTMETHOD OSBOOL operator!= (ASN1CTime &)
- virtual EXTRTMETHOD OSBOOL operator> (ASN1CTime &)
- virtual EXTRTMETHOD OSBOOL operator< (ASN1CTime &)
- virtual EXTRTMETHOD OSBOOL operator>= (ASN1CTime &)
- virtual EXTRTMETHOD OSBOOL operator<= (ASN1CTime &)

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).

Definition at line 85 of file ASN1CTime.h

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

- ASN1CTime.h

virtual int ASN1CTime::compileString ()=0

Compiles new time string accoring X.680 and ISO 8601.

Returns: . 0 on success, or an error code if there was an error.

EXTRTMETHOD ASN1CTime::ASN1CTime (OSRTMessageBufferIF &msgBuf, char *&buf, OSSIZE 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.

Table 3.69. 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.

EXTRTMETHOD 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.

Table 3.70. 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.

EXTRTMETHOD ASN1CTime::ASN1CTime (OSRTContext &ctxt, char *&buf, OSSIZE 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.

Table 3.71. Parameters

ctxt	Reference to an OSRTContext data structure.
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.

EXTRTMETHOD ASN1CTime::ASN1CTime (OSRTContext &ctxt, 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 array to an internal reference variable. The object will then directly operate on the given data variable.

Table 3.72. Parameters

ctxt	Reference to an OSRTContext data structure.
buf	Reference to a pointer to a time string buffer.
useDerRules	Use the Distinguished Encoding Rules (DER) to operate on this time value.

EXTRTMETHOD ASN1CTime::ASN1CTime (const ASN1CTime &original)

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

Table 3.73. Parameters

original	The original time string object value.

EXTRTMETHOD ASN1CTime::~ASN1CTime ()

The destructor; this cleans up the ASN1CTime object.

virtual EXTRTMETHOD int ASN1CTime::getYear ()

This method returns the year component of the time value.

Note that the return value may differ for different inherited ASN1CTime classes.

Table 3.74. Parameters

-	none

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

virtual EXTRTMETHOD int ASN1CTime::getMonth ()

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 valued 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.

Table 3.75. Parameters

-	none

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

virtual EXTRTMETHOD int ASN1CTime::getDay ()

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 be differ for different inherited ASN1CTime classes.

Table 3.76. 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.

virtual EXTRTMETHOD int ASN1CTime::getHour ()

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.

Table 3.77. Parameters

-	none

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

virtual EXTRTMETHOD int ASN1CTime::getMinute ()

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.

Table 3.78. Parameters

-	none

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

virtual EXTRTMETHOD int ASN1CTime::getSecond ()

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.

Table 3.79. Parameters

-	none

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

virtual EXTRTMETHOD int ASN1CTime::getFraction ()

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.

Table 3.80. 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.

virtual EXTRTMETHOD double ASN1CTime::getFractionAsDouble ()

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.

virtual EXTRTMETHOD int ASN1CTime::getFractionStr (char *const pBuf, size_t bufSize)

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.

virtual EXTRTMETHOD int ASN1CTime::getFractionLen ()

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.

virtual EXTRTMETHOD int ASN1CTime::getDiffHour ()

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 of negative time difference. Note that the return value may differ for different inherited ASN1CTime classes.

Table 3.81. 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.

virtual EXTRTMETHOD int ASN1CTime::getDiffMinute ()

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 of negative time difference. Note that the return value may differ for different inherited ASN1CTime classes.

Table 3.82. 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.

virtual EXTRTMETHOD int ASN1CTime::getDiff ()

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 of negative time difference. Note that the return value may differ for different inherited ASN1CTime classes.

Table 3.83. 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.

virtual EXTRTMETHOD OSBOOL ASN1CTime::getUTC ()

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.

Table 3.84. Parameters

-	none

Returns: . UTC flag state is returned.

virtual EXTRTMETHOD time_t ASN1CTime::getTime ()

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 represned 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.

Table 3.85. 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.

void ASN1CTime::setDER (OSBOOL bvalue)

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

virtual EXTRTMETHOD int ASN1CTime::setUTC (OSBOOL utc)

This method sets teh 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.

Table 3.86. Parameters

utc	UTC flag state.

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD int ASN1CTime::setYear (short year_)

This method sets the year component of the time value.

Note that the action of this method may differ for different inherited ASN1CTime classes.

Table 3.87. Parameters

year_	Year component (full 4 digits).

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD int ASN1CTime::setMonth (short month_)

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.

Table 3.88. Parameters

month_	Month component (1 - 12).

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD int ASN1CTime::setDay (short day_)

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.

Table 3.89. Parameters

day_	Day of month component (1 - 31).

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD int ASN1CTime::setHour (short hour_)

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.

Table 3.90. Parameters

hour_	Hour component (0 - 23).

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD int ASN1CTime::setMinute (short minute_)

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.

Table 3.91. Parameters

minute_	Minute component (0 - 59).

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD int ASN1CTime::setSecond (short second_)

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 form different inherited ASN1CTime classes.

Table 3.92. Parameters

second_	Second component (0 - 59).

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD int ASN1CTime::setFraction (int fraction, int fracLen=-1)

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.

Table 3.93. 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.

virtual EXTRTMETHOD int ASN1CTime::setFraction (double frac, int fracLen)

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.

Table 3.94. 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.

virtual EXTRTMETHOD int ASN1CTime::setFraction (char const *frac)

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.

Table 3.95. Parameters

frac	Second's decimal fraction component.

Returns: . Completion status of operation:

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

virtual int ASN1CTime::setTime (time_t time, OSBOOL diffTime)=0

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.

Table 3.96. 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.

virtual EXTRTMETHOD int ASN1CTime::setDiffHour (short dhour)

This method sets teh 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.

Table 3.97. 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.

virtual EXTRTMETHOD int ASN1CTime::setDiff (short dhour, short dminute)

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.

Table 3.98. 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.

virtual EXTRTMETHOD int ASN1CTime::setDiff (short in-Minutes)

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.

Table 3.99. 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.

virtual EXTRTMETHOD int ASN1CTime::parseString (const char *string)

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.

Table 3.100. Parameters

string	The time string value to be parsed.

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD void ASN1CTime::clear ()

This method clears the time string.

Note the action of this method may differ for different inherited ASN1CTime classes.

Table 3.101. Parameters

-	none

Returns: . - none

virtual EXTRTMETHOD int ASN1CTime::equals (ASN1CTime &)

This method compares times.

EXTRTMETHOD OSSIZE ASN1CTime::getTimeStringLen ()

This method returns the length of the compiled time string.

Returns: . The length of the compiled time string.

EXTRTMETHOD const char* ASN1CTime::getTimeString (char *pbuf, OSSIZE bufsize)

This method copies the compiled time string into the given buffer.

Table 3.102. Parameters

pbuf	The buffer to receive the time string.
bufsize	The size of the buffer.

Returns: . A character string containing the compiled time string.

EXTRTMETHOD const ASN1CTime& ASN1CTime::operator= (const ASN1CTime &)

This operator assigns this ASN1CTime object to the given ASN1CTime reference.

Returns: . A constant reference to the given ASN1CTime object.

virtual EXTRTMETHOD OSBOOL ASN1CTime::operator== (ASN1CTime &)

The ASN1CTime equality test.

Returns: . TRUE if the two times are equal.

virtual EXTRTMETHOD OSBOOL ASN1CTime::operator!= (ASN1CTime &)

The ASN1CTime inequality test.

Returns: . TRUE if the two times are not equal.

virtual EXTRTMETHOD OSBOOL ASN1CTime::operator> (ASN1CTime &)

The ASN1CTime greater-than test.

Returns: . TRUE if this time is greater than the given time.

virtual EXTRTMETHOD OSBOOL ASN1CTime::operator< (ASN1CTime &)

The ASN1CTime less-than test.

Returns: . TRUE if this time is less than the given time.

virtual EXTRTMETHOD OSBOOL ASN1CTime::operator>= (ASN1CTime &)

The ASN1CTime greater-equal test.

Returns: . TRUE if this time is greater-than or equal to the given time.

virtual EXTRTMETHOD OSBOOL ASN1CTime::operator<= (ASN1CTime &)

The ASN1CTime less-equal test.

Returns: . TRUE if this time is less-than or equal to the given time.

ASN1CType class Reference

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

Protected Attributes

- OSRTCtxtPtr mpContext
- OSRTMessageBufferIF * mpMsgBuf

- EXTRTMETHOD ASN1CType ( )
- EXTRTMETHOD ASN1CType (OSRTContext & ctxt)

- EXTRTMETHOD int setMsgBuf (OSRTMessageBufferIF & msgBuf, OSBOOL initBuf)
- EXTRTMETHOD int setRunTimeKey (const OSOCTET * key, OSSIZE keylen)
- EXTRTMETHOD ASN1CType (OSRTMessageBufferIF & msgBuf)
- EXTRTMETHOD ASN1CType (const ASN1CType & orig)
- virtual ~ASN1CType ( )
- void append (OSRTDList & llist, void * pdata)
- OSRTCtxtPtr getContext ( )
- OSCTXT * getCtxtPtr ( )
- char * getErrorText (char * textbuf, OSSIZE bufsize)
- int getStatus ( )
- void * memAlloc (OSSIZE numocts)
- void * memAllocZ (OSSIZE numocts)
- void memFreeAll ( )
- void * memRealloc (void * ptr, OSSIZE numocts)
- void memReset ( )
- void memFreePtr (void * ptr)
- void printErrorInfo ( )
- void resetError ( )
- OSBOOL setDiag (OSBOOL value)
- virtual EXTRTMETHOD int Encode ( )
- virtual EXTRTMETHOD int Decode (OSBOOL free)
- virtual int EncodeTo (OSRTMessageBufferIF &)
- virtual int DecodeFrom (OSRTMessageBufferIF & , OSBOOL free)
- virtual void MemFree ( )

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.

Definition at line 321 of file asn1CppTypes.h

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

- asn1CppTypes.h

Member Data Documentation

OSRTCtxtPtr ASN1CType::mpContext

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.

Definition at line 329 of file asn1CppTypes.h

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

- asn1CppTypes.h

OSRTMessageBufferIF* ASN1CType::mpMsgBuf

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.

Definition at line 335 of file asn1CppTypes.h

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

- asn1CppTypes.h

EXTRTMETHOD ASN1CType::ASN1CType ()

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

EXTRTMETHOD ASN1CType::ASN1CType (OSRTContext &ctxt)

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

Table 3.103. Parameters

ctxt	- Reference to a context object.

EXTRTMETHOD int ASN1CType::setRunTimeKey (const OSOCTET *key, OSSIZE keylen)

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

Table 3.104. 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.

EXTRTMETHOD ASN1CType::ASN1CType (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.

Table 3.105. Parameters

msgBuf	- Reference to a message buffer or stream object.

EXTRTMETHOD ASN1CType::ASN1CType (const ASN1CType &orig)

The copy constructor sets the internal message buffer pointer and context to point at the message buffer and context from the original ASN1CType object.

Table 3.106. Parameters

orig	- Reference to a message buffer or stream object.

virtual ASN1CType::~ASN1CType ()

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

void ASN1CType::append (OSRTDList &llist, void *pdata)

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

Table 3.107. Parameters

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

OSRTCtxtPtr ASN1CType::getContext ()

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

Returns: . Context smart pointer object.

OSCTXT* ASN1CType::getCtxtPtr ()

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

char* ASN1CType::getErrorText (char *textbuf=(char *) 0, OSSIZE bufsize=0)

This method returns the error text associated with the last run-time error. If buffer pointer and buffer size are specified in parameters (not NULL) then error text will be copied in the passed buffer. Otherwise, this function allocates memory using the 'new' operator. It is the user's responsibility to free this memory using 'delete'.

Table 3.108. Parameters

textbuf	A pointer to a destination buffer to hold the error text. If NULL, a dynamic buffer will be allocated.
bufsize	The size of the output buffer size.

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.

int ASN1CType::getStatus () const

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

Returns: . Runtime status code:

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

void* ASN1CType::memAlloc (OSSIZE numocts)

The memAlloc method allocates memory using the C runtime memory management functions. The memory is tracked in the underlying context structure. When both this ASN1CType derived control class object and the message buffer object are destroyed, this memory will be freed.

Table 3.109. 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.

void* ASN1CType::memAllocZ (OSSIZE numocts)

The memAllocZ method allocates memory using the C runtime memory management functions. The memory is tracked in the underlying context structure. When both this ASN1CType derived control class object and the message buffer object are destroyed, this memory will be freed. This memory will be zeroed out upon allocation.

Table 3.110. 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.

void ASN1CType::memFreeAll ()

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.

void* ASN1CType::memRealloc (void *ptr, OSSIZE numocts)

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

Table 3.111. Parameters

ptr	Original pointer containing dynamic memory to be resized.
numocts	Number of bytes of memory to allocate

Returns: . Reallocated memory pointer

void ASN1CType::memReset ()

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

void ASN1CType::memFreePtr (void *ptr)

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

Table 3.112. Parameters

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

void ASN1CType::printErrorInfo ()

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

void ASN1CType::resetError ()

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

OSBOOL ASN1CType::setDiag (OSBOOL value)

This method turns diagnostic tracing on or off.

Table 3.113. Parameters

value	Boolean value; TRUE = turn tracing on.

Returns: . Previous state.

virtual EXTRTMETHOD int ASN1CType::Encode ()

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.

virtual EXTRTMETHOD int ASN1CType::Decode (OSBOOL free=FALSE)

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

Table 3.114. Parameters

free	Indicates whether memory for existing objects should be freed prior to decoding as part of object (re)initialization.

Returns: . Completion status of operation:

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

virtual int ASN1CType::EncodeTo (OSRTMessageBufferIF &)

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

Table 3.115. 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.

virtual int ASN1CType::DecodeFrom (OSRTMessage-BufferIF &, OSBOOL free=TRUE)

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

Table 3.116. Parameters

msgBuf	Message buffer or stream containing message to decode.
free	Indicates whether memory for existing objects should be freed prior to decoding as part of object (re)initialization.

Returns: . Completion status of operation:

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

virtual void ASN1CType::MemFree ()

The `MemFree` method invokes the asn1Free* function on the message using this object's OSCTXT. The implementation here does nothing. Subclasses must override this method to invoke the correct asn1Free function when there is one (the function is only generated when required).

ASN1CUTCTime class Reference

`#include <ASN1CUTCTime.h>`

Protected Attributes

- ASN1TUTCTime timeObj

- virtual ASN1TTime & getTimeObj ( )
- virtual const ASN1TTime & getTimeObj ( )
- EXTRTMETHOD ASN1CUTCTime (char *& buf, int bufSize, OSBOOL useDerRules)
- EXTRTMETHOD ASN1CUTCTime (ASN1UTCTime & buf, OSBOOL useDerRules)
- EXTRTMETHOD int compileString ( )

- EXTRTMETHOD int getFraction ( )
- EXTRTMETHOD int setFraction (int fraction)
- EXTRTMETHOD ASN1CUTCTime (OSRTMessageBufferIF & msgBuf, char *& buf, int bufSize, OSBOOL useDerRules)
- EXTRTMETHOD ASN1CUTCTime (OSRTMessageBufferIF & msgBuf, ASN1UTCTime & buf, OSBOOL useDerRules)
- EXTRTMETHOD ASN1CUTCTime (OSRTContext & ctxt, char *& buf, int bufSize, OSBOOL useDerRules)
- EXTRTMETHOD ASN1CUTCTime (OSRTContext & ctxt, ASN1UTCTime & buf, OSBOOL useDerRules)
- ASN1CUTCTime (const ASN1CUTCTime & original)
- EXTRTMETHOD int setTime (time_t time, OSBOOL diffTime)
- const ASN1CUTCTime & operator= (const ASN1CUTCTime & tm)

Detailed Description

ASN.1 UTCTime control class. The ASN1CUTTime 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).

Definition at line 65 of file ASN1CUTCTime.h

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

- ASN1CUTCTime.h

EXTRTMETHOD int ASN1CUTCTime::compileString ()

Compiles new time string accoring X.680 and ISO 8601.

Returns: . 0 on success, or an error code if there was an error.

EXTRTMETHOD int ASN1CUTCTime::getFraction ()

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.

Table 3.117. 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.

EXTRTMETHOD 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.

Table 3.118. 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,

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

This constructor creates a time string suing the ASN1UTCTime argument. c The constructor does not deep-copy the variable, it assigns a referene 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.

Table 3.119. 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,

EXTRTMETHOD int ASN1CUTCTime::setTime (time_t time, OSBOOL diffTime)

Converts time_t to time string.

Table 3.120. 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.

ASN1DynBitStr class Reference

ASN1DynBitStr64 class Reference

ASN1DynOctStr class Reference

Asn1ErrorHandler class Reference

`#include <asn1CppEvtHndlr.h>`

- Asn1ErrorHandler ( )
- virtual ~Asn1ErrorHandler ( )
- virtual int error (OSCTXT * pCtxt, ASN1CCB * pCCB, int stat)
- static EXTRTMETHOD int invoke (OSCTXT * pCtxt, ASN1CCB * pCCB, int stat)
- static EXTRTMETHOD int invoke (OSCTXT * pCtxt, OSOCTET * ptr, int len, int stat)
- static EXTRTMETHOD void setErrorHandler (OSCTXT * pCtxt, Asn1ErrorHandler * pHandler)

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.

Definition at line 555 of file asn1CppEvtHndlr.h

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

- asn1CppEvtHndlr.h

virtual int Asn1ErrorHandler::error (OSCTXT *pCtxt, ASN1CCB *pCCB, int stat)=0

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

Table 3.121. 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.

static EXTRTMETHOD void Asn1ErrorHandler::setErrorHandler (OSCTXT *pCtxt, Asn1ErrorHandler *pHandler)

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..

Table 3.122. Parameters

pCtxt	- Pointer to a context block structure.
pHandler	- Pointer to error handler object to register.

ASN1MessageBuffer class Reference

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

- EXTRTMETHOD ASN1MessageBuffer (Type bufferType)
- EXTRTMETHOD ASN1MessageBuffer (Type bufferType, OSRTContext * pContext)
- virtual int setStatus (int stat)
- virtual ~ASN1MessageBuffer ( )
- virtual void addEventHandler (Asn1NamedEventHandler * pEventHandler)
- void addRawEventHandler (Asn1RawEventHandler * pHandler)
- ASN1BMPString * CStringToBMPString (const char * cstring, ASN1BMPString * pBMPString, Asn116BitCharSet * pCharSet)
- virtual void * getAppInfo ( )
- virtual EXTRTMETHOD int initBuffer (OSRTMEMBUF & membuf)
- virtual EXTRTMETHOD int initBuffer (OSUNICHAR * unistr)
- virtual int initBuffer (const OSUTF8CHAR *)
- virtual OSBOOL isA (Type)
- virtual void removeEventHandler (Asn1NamedEventHandler * pEventHandler)
- void removeRawEventHandler ( )

- virtual void resetErrorInfo ( )
- virtual void setAppInfo (void *)
- virtual void setErrorHandler (Asn1ErrorHandler * pErrorHandler)
- EXTRTMETHOD int setRunTimeKey (const OSOCTET * key, OSSIZE keylen)
- size_t getBitOffset ( )
- OSOCTET * GetMsgCopy ( )
- const OSOCTET * GetMsgPtr ( )
- void PrintErrorInfo ( )

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.

Definition at line 104 of file asn1CppTypes.h

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

- asn1CppTypes.h

EXTRTMETHOD ASN1MessageBuffer::ASN1MessageBuffer (Type bufferType)

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

Table 3.123. Parameters

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

EXTRTMETHOD ASN1MessageBuffer::ASN1MessageBuffer (Type bufferType, OSRTContext *pContext)

This constructor sets the buffer class type and also a pointer to a context created by the user.

Table 3.124. Parameters

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

pContext	A pointer to an OSRTContext structure previously created by the user.

virtual int ASN1MessageBuffer::setStatus (int stat)

This method sets error status to the context.

Returns: . Error status value being set.

virtual ASN1MessageBuffer::~ASN1MessageBuffer ()

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

virtual void ASN1MessageBuffer::addEventHandler (Asn1NamedEventHandler *pEventHandler)

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

Table 3.125. Parameters

pEventHandler	- Pointer to named event handler object to register.

void ASN1MessageBuffer::addRawEventHandler (Asn1RawEventHandler *pHandler)

Add the given event handler to this object's context, replacing any previous event handler. This does not take ownership of the handler.

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

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.

Table 3.126. 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.

virtual void* ASN1MessageBuffer::getAppInfo ()

Returns a pointer to application-specific information block

virtual EXTRTMETHOD int ASN1MessageBuffer::initBuffer (OSRTMEMBUF &membuf)

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

Table 3.127. Parameters

membuf	OSRTMEMBUF memory buffer class object reference.

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD int ASN1MessageBuffer::initBuffer (OSUNICHAR *unistr)

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.

Table 3.128. Parameters

unistr	Pointer to a Unicode character string.

Returns: . Completion status of operation:

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

virtual OSBOOL ASN1MessageBuffer::isA (Type)

This method checks the type of the message buffer.

Table 3.129. Parameters

bufferType	Enumerated identifier specifying a derived class. Possible values are: BEREncode, BERDecode, PEREncode, PERDecode, XMLEncode, XMLDecode, Stream.

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

virtual void ASN1MessageBuffer::removeEventHandler (Asn1NamedEventHandler *pEventHandler)

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

Table 3.130. Parameters

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

void ASN1MessageBuffer::removeRawEventHandler ()

Remove the event handler from this object's context.

virtual void ASN1MessageBuffer::resetErrorInfo ()

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

virtual void ASN1MessageBuffer::setAppInfo (void *)

Sets the application-specific information block.

virtual void ASN1MessageBuffer::setErrorHandler (Asn1ErrorHandler *pErrorHandler)

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.

Table 3.131. Parameters

pErrorHandler	- Pointer to error handler object to register.

EXTRTMETHOD int ASN1MessageBuffer::setRunTimeKey (const OSOCTET *key, OSSIZE keylen)

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

Table 3.132. 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.

size_t ASN1MessageBuffer::getBitOffset ()

This method returns the current bit offset in the message buffer.

Table 3.133. Parameters

-	none

Asn1NamedEventHandler class Reference

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

- Asn1NamedEventHandler ( )
- virtual ~Asn1NamedEventHandler ( )
- virtual void startElement (const char * name, int index)
- virtual void endElement (const char * name, int index)
- virtual void boolValue (OSBOOL value)
- virtual void intValue (OSINT32 value)
- virtual void uIntValue (OSUINT32 value)
- virtual void int64Value (OSINT64 value)
- virtual void uInt64Value (OSUINT64 value)
- virtual void bitStrValue (OSUINT32 numbits, const OSOCTET * data)
- virtual void octStrValue (OSUINT32 numocts, const OSOCTET * data)
- virtual void charStrValue (const char * value)
- virtual void charStrValue (OSUINT32 nchars, const OSUTF8CHAR * value)
- virtual void charStrValue (OSUINT32 nchars, OSUNICHAR * data)
- virtual void charStrValue (OSUINT32 nchars, OS32BITCHAR * data)
- virtual void nullValue ( )
- virtual void oidValue (OSUINT32 numSubIds, OSUINT32 * pSubIds)
- virtual void realValue (double value)

- virtual void enumValue (OSUINT32 value, const OSUTF8CHAR * text)
- virtual void openTypeValue (OSUINT32 numocts, const OSOCTET * data)

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

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.

Definition at line 75 of file asn1CppEvtHndlr.h

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

- asn1CppEvtHndlr.h

virtual void Asn1NamedEventHandler::startElement (const char *name, int index)=0

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.

Table 3.134. Parameters

name	For SEQUENCE, SET, or CHOICE, this is the name of the element as defined in the ASN.1 defination. 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

virtual void Asn1NamedEventHandler::endElement (const char *name, int index)=0

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.

Table 3.135. Parameters

name	For SEQUENCE, SET, or CHOICE, this is the name of the element as defined in the ASN.1 defination. 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

virtual void Asn1NamedEventHandler::boolValue (OSBOOL value)

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

Table 3.136. Parameters

value	Parsed value.

Returns: . - none

virtual void Asn1NamedEventHandler::intValue (OSINT32 value)

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

Table 3.137. Parameters

value	Parsed value.

Returns: . - none

virtual void Asn1NamedEventHandler::uIntValue (OSUINT32 value)

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.

Table 3.138. Parameters

value	Parsed value.

Returns: . - none

virtual void Asn1NamedEventHandler::int64Value (OSINT64 value)

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

Table 3.139. Parameters

value	Parsed value.

Returns: . - none

virtual void Asn1NamedEventHandler::uInt64Value (OSUINT64 value)

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

Table 3.140. Parameters

value	Parsed value.

Returns: . - none

virtual void Asn1NamedEventHandler::bitStrValue (OSUINT32 numbits, const OSOCTET *data)

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

Table 3.141. Parameters

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

Returns: . - none

virtual void Asn1NamedEventHandler::octStrValue (OSUINT32 numocts, const OSOCTET *data)

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

Table 3.142. Parameters

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

Returns: . - none

virtual void Asn1NamedEventHandler::charStrValue (const char *value)

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.

Table 3.143. Parameters

value	Null terminated character string value.

Returns: . - none

virtual void Asn1NamedEventHandler::charStrValue (OSUINT32 nchars, const OSUTF8CHAR *value)

This method is invoked from within a decode function when a value of a UTF-8 character string type is parsed.

Table 3.144. Parameters

nchars	Number of characters in the parsed value.
value	A UTF-8 character string.

Returns: . - none

virtual void Asn1NamedEventHandler::charStrValue (OSUINT32 nchars, OSUNICHAR *data)

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.

Table 3.145. 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

virtual void Asn1NamedEventHandler::charStrValue (OSUINT32 nchars, OS32BITCHAR *data)

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

This is used for the ASN.1 UniversalString type.

Table 3.146. 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

virtual void Asn1NamedEventHandler::nullValue ()

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

Table 3.147. Parameters

-	none

Returns: . - none

virtual void Asn1NamedEventHandler::oidValue (OSUINT32 numSubIds, OSUINT32 *pSubIds)

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

Table 3.148. Parameters

numSubIds	Number of subidentifiers in the object identifier.
pSubIds	Pointer to array containing the subidentifier values.

Returns: . -none

virtual void Asn1NamedEventHandler::realValue (double value)

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

Table 3.149. Parameters

value	Parsed value.

Returns: . - none

virtual void Asn1NamedEventHandler::enumValue (OSUINT32 value, const OSUTF8CHAR *text)

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

Table 3.150. Parameters

value	- Parsed enumerated value
text	- Textual value of enumerated identifier

Returns: . - none

virtual void Asn1NamedEventHandler::openTypeValue (OSUINT32 numocts, const OSOCTET *data)

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

Table 3.151. Parameters

numocts	Number of octets in the parsed value.
data	Pointer to byet array contain in tencoded ASN.1 value.

Returns: . - none

static EXTRTMETHOD void Asn1NamedEventHandler::addEventHandler (OSCTXT *pCtxt, Asn1NamedEventHandler *pHandler)

This method is called to add a new event handler to the context event handler list. It is a static method.

Table 3.152. Parameters

pCtxt	A pointer-to an ::OSCTXT data structure.
pHandler	A pointer to an Asn1NamedEventHandler.

static EXTRTMETHOD void Asn1NamedEventHandler::removeEventHandler (OSCTXT *pCtxt, Asn1NamedEventHandler *pHandler)

This method removes the given event handler from the context event handler list. This method does not delete the handler, so memory allocated for it will need to be released elsewhere.

Table 3.153. Parameters

pCtxt	A pointer-to an ::OSCTXT data structure.
pHandler	A pointer to an Asn1NamedEventHandler.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeStartElement (OSCTXT *pCtxt, const char *name, int index)

This method is called by generated code to invoke the event handlers' start element methods. It is static.

Table 3.154. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
name	The name of the element.
index	The index of the element, if it is in a SEQUENCE or SET OF type.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeEndElement (OSCTXT *pCtxt, const char *name, int index)

This method is called by generated code to invoke the event handlers' end element methods. It is static.

Table 3.155. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
name	The name of the element.
index	The index of the element, if it is in a SEQUENCE or SET OF type.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeBoolValue (OSCTXT *pCtxt, OSBOOL value)

This method is called by generated code to invoke the event handlers' boolean method. It is static.

Table 3.156. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
value	A decoded boolean value.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeIntValue (OSCTXT *pCtxt, OSINT32 value)

This method is called by generated code to invoke the event handlers' integer method. It is static.

Table 3.157. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
value	A decoded 32-bit integer value.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeUIntValue (OSCTXT *pCtxt, OSUINT32 value)

This method is called by generated code to invoke the event handlers' unsigned integer method. It is static.

Table 3.158. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
value	A decoded unsigned 32-bit integer value.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeInt64Value (OSCTXT *pCtxt, OSINT64 value)

This method is called by generated code to invoke the event handlers' 64-bit integer method. It is static.

Table 3.159. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
value	A decoded 64-bit integer value.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeUInt64Value (OSCTXT *pCtxt, OSUINT64 value)

This method is called by generated code to invoke the event handlers' unsigned 64-bit integer method. It is static.

Table 3.160. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
value	A decoded 64-bit integer value.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeBitStrValue (OSCTXT *pCtxt, OSUINT32 numbits, const OSOCTET *data)

This method is called by generated code to invoke the event handlers' bit string method. It is static.

Table 3.161. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
numbits	The number of bits in the decoded bit string.
data	The decoded bit string data.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeOctStrValue (OSCTXT *pCtxt, OSUINT32 numocts, const OSOCTET *data)

This method is called by generated code to invoke the event handlers' octet string method. It is static.

Table 3.162. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
numocts	The number of octets in the decoded octet string.
data	The decoded octet string data.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeCharStrValue (OSCTXT *pCtxt, const char *value)

This method is called by generated code to invoke the event handlers' character string method. It is static.

Table 3.163. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
value	The decoded character string.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeCharStrValue (OSCTXT *pCtxt, OSUINT32 nchars, OSUNICHAR *data)

This method is called by generated code to invoke the event handlers' 16-bit character string method. It is static.

Table 3.164. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
nchars	The number of characters in the string.
value	The decoded 16-bit character string.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeCharStrValue (OSCTXT *pCtxt, OSUINT32 nchars, OS32BITCHAR *data)

This method is called by generated code to invoke the event handlers' 32-bit character string method. It is static.

Table 3.165. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
nchars	The number of characters in the string.
value	The decoded character string.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeCharStrValue (OSCTXT *pCtxt, OSUINT32 nchars, const OSUTF8CHAR *data)

This method is called by generated code to invoke the event handlers' UTF-8 character string method. It is static.

Table 3.166. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
nchars	The number of characters in the string.
value	The decoded character string.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeNullValue (OSCTXT *pCtxt)

This method is called by generated code to invoke the event handlers' null method. It is static.

Table 3.167. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeOidValue (OSCTXT *pCtxt, OSUINT32 numSubIds, OSUINT32 *pSubIds)

This method is called by generated code to invoke the event handlers' object identifier method. It is static.

Table 3.168. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
numSubIds	The number of OID subids.
value	The decoded OID ids.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeRealValue (OSCTXT *pCtxt, double value)

This method is called by generated code to invoke the event handlers' real method. It is static.

Table 3.169. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
value	A decoded real value.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeEnumValue (OSCTXT *pCtxt, OSUINT32 value, const OSUTF8CHAR *text)

This method is called by generated code to invoke the event handlers' enumerated method. It is static.

Table 3.170. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
value	A decoded 64-bit integer value.
text	The character string representation of the value.

static EXTRTMETHOD void Asn1NamedEventHandler::invokeOpenTypeValue (OSCTXT *pCtxt, OSUINT32 numocts, const OSOCTET *data)

This method is called by generated code to invoke the event handlers' open type method. It is static.

Table 3.171. Parameters

pCtxt	A pointer to an ::OSCTXT data structure.
numocts	The number of octets in the open type.
data	The data octets that comprise the open type value.

Asn1NullEventHandler class Reference

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

- virtual void startElement (const char * , int)

- virtual void endElement (const char * , int)

Detailed Description

The Asn1NullEventHandler contains a completely empty implementation of all user methods.

Definition at line 533 of file asn1CppEvtHndlr.h

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

- asn1CppEvtHndlr.h

virtual void Asn1NullEventHandler::startElement (const char *, int)

This startElement method does nothing.

virtual void Asn1NullEventHandler::endElement (const char *, int)

This endElement method does nothing.

Asn1Object class Reference

ASN1OBJID class Reference

ASN1OpenType class Reference

ASN1TBitStr32 struct Reference

`#include <asn1CppTypes.h>`

- ASN1TBitStr32 ( )
- ASN1TBitStr32 (OSUINT32 _numbits, const OSOCTET * _data)
- ASN1TBitStr32 (ASN1BitStr32 & _bs)

Detailed Description

Fixed-size bit string. This is the base class for generated C++ data type classes for sized BIT STRING's with size <= 32 bits.

Definition at line 708 of file asn1CppTypes.h

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

- asn1CppTypes.h

ASN1TBitStr32::ASN1TBitStr32 ()

The default constructor creates an empty bit string.

ASN1TBitStr32::ASN1TBitStr32 (OSUINT32 _numbits, const OSOCTET *_data)

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

Table 3.172. Parameters

_numbits	Number of bits in the bit string.
_data	The binary bit data values.

ASN1TBitStr32::ASN1TBitStr32 (ASN1BitStr32 &_bs)

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

Table 3.173. Parameters

_bs	- C bit string structure.

ASN1TBMPString struct Reference

`#include <asn1CppTypes.h>`

- ASN1TBMPString ( )

Detailed Description

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

Definition at line 748 of file asn1CppTypes.h

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

- asn1CppTypes.h

ASN1TBMPString::ASN1TBMPString ()

The default constructor creates an empty BMPString value.

ASN1TDynBitStr struct Reference

`#include <asn1CppTypes.h>`

- ASN1TDynBitStr ( )
- ASN1TDynBitStr (OSUINT32 _numbits, const OSOCTET * _data)
- ASN1TDynBitStr (ASN1DynBitStr & _bs)

Detailed Description

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

Definition at line 641 of file asn1CppTypes.h

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

- asn1CppTypes.h

ASN1TDynBitStr::ASN1TDynBitStr ()

The default constructor creates an empty bit string.

ASN1TDynBitStr::ASN1TDynBitStr (OSUINT32 _numbits, const OSOCTET *_data)

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

Table 3.174. Parameters

_numbits	- Number of bits in the bit string.
_data	- The binary bit data values.

ASN1TDynBitStr::ASN1TDynBitStr (ASN1DynBitStr &_bs)

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

Table 3.175. Parameters

_bs	- C bit string structure.

ASN1TDynBitStr64 struct Reference

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

- ASN1TDynBitStr64 ( )
- ASN1TDynBitStr64 (OSSIZE _numbits, const OSOCTET * _data)

- ASN1TDynBitStr64 (ASN1DynBitStr64 & _bs)

Detailed Description

64-bit dynamic bit string.

Definition at line 674 of file asn1CppTypes.h

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

- asn1CppTypes.h

ASN1TDynBitStr64::ASN1TDynBitStr64 ()

The default constructor creates an empty bit string.

ASN1TDynBitStr64::ASN1TDynBitStr64 (OSSIZE _numbits, const OSOCTET *_data)

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

Table 3.176. Parameters

_numbits	- Number of bits in the bit string.
_data	- The binary bit data values.

ASN1TDynBitStr64::ASN1TDynBitStr64 (ASN1DynBitStr64 &_bs)

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

Table 3.177. Parameters

_bs	- C bit string structure.

ASN1TDynOctStr struct Reference

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

- ASN1TDynOctStr ( )
- ASN1TDynOctStr (OSUINT32 _numocts, const OSOCTET * _data)
- ASN1TDynOctStr (const ASN1DynOctStr & _os)
- ASN1TDynOctStr (const ASN1TDynOctStr & _os)
- ASN1TDynOctStr (const char * cstring)

- ASN1TDynOctStr & operator= (const char * cstring)
- EXTRTMETHOD ASN1TDynOctStr & operator= (const ASN1TDynOctStr & octet)
- EXTRTMETHOD const char * toString (OSCTXT * pctxt)
- EXTRTMETHOD const char * toHexString (OSCTXT * pctxt)
- EXTRTMETHOD int nCompare (OSUINT32 n, const ASN1TDynOctStr & o)

Detailed Description

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

Definition at line 48 of file ASN1TOctStr.h

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

- ASN1TOctStr.h

ASN1TDynOctStr::ASN1TDynOctStr ()

The default constructor creates an empty octet string.

ASN1TDynOctStr::ASN1TDynOctStr (OSUINT32 _numocts, const OSOCTET *_data)

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

Table 3.178. Parameters

_numocts	- Number of octet in the octet string.
_data	- The binary octet data values.

ASN1TDynOctStr::ASN1TDynOctStr (const ASN1DynOctStr &_os)

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

Table 3.179. Parameters

_os	- C octet string structure.

ASN1TDynOctStr::ASN1TDynOctStr (const ASN1TDynOctStr &_os)

Copy constructor.

Table 3.180. Parameters

_os	- C++ octet string structure.

ASN1TDynOctStr::ASN1TDynOctStr (const char *cstring)

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

Table 3.181. Parameters

cstring	- C null-terminated string.

ASN1TDynOctStr& ASN1TDynOctStr::operator= (const char *cstring)

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

Table 3.182. Parameters

cstring	- C null-terminated string.

EXTRTMETHOD ASN1TDynOctStr& ASN1TDynOctStr::operator= (const ASN1TDynOctStr &octet)

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

Table 3.183. Parameters

octet	- Octet string object reference

EXTRTMETHOD 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.

Table 3.184. Parameters

pctxt	- Pointer to a context structure.

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

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

Table 3.185. Parameters

pctxt	- Pointer to a context structure.

EXTRTMETHOD int ASN1TDynOctStr::nCompare (OSUINT32 n, const ASN1TDynOctStr &o) const

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

Table 3.186. 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.

ASN1TGeneralizedTime class Reference

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

- ASN1TGeneralizedTime ( )
- EXTRTMETHOD ASN1TGeneralizedTime (const char * buf, OSBOOL useDerRules)
- ASN1TGeneralizedTime (OSBOOL useDerRules)
- ASN1TGeneralizedTime (const ASN1TGeneralizedTime & original)
- EXTRTMETHOD int getCentury ( )
- EXTRTMETHOD int setCentury (short century)
- EXTRTMETHOD int setTime (time_t time, OSBOOL diffTime)
- EXTRTMETHOD int parseString (const char * string)
- const ASN1TGeneralizedTime & operator= (const ASN1TGeneralizedTime & tm)

- EXTRTMETHOD int compileString (char * pbuf, OSSIZE bufsize)

Detailed Description

ASN.1 GeneralizedTime utility class. The ASN1TGeneralizedTime class is derived from the ASN1TTime base class.

Definition at line 690 of file ASN1TTime.h

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

- ASN1TTime.h

ASN1TGeneralizedTime::ASN1TGeneralizedTime ()

A default constructor.

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

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

Table 3.187. Parameters

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

ASN1TGeneralizedTime::ASN1TGeneralizedTime (OSBOOL useDerRules)

This constructor creates an empty time object.

Table 3.188. Parameters

useDerRules	An OSBOOL value.

ASN1TGeneralizedTime::ASN1TGeneralizedTime (const ASN1TGeneralizedTime &original)

A copy constructor.

EXTRTMETHOD int ASN1TGeneralizedTime::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 sucessful. If the operation fails, one of the negative status codes is returned.

EXTRTMETHOD int ASN1TGeneralizedTime::setCentury (short century)

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

Table 3.189. Parameters

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

Returns: . Completion status of operation:

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

EXTRTMETHOD int ASN1TGeneralizedTime::setTime (time_t time, OSBOOL diffTime)

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.

Table 3.190. 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.

EXTRTMETHOD int ASN1TGeneralizedTime::parseString (const char *string)

Parses sting.

Returns: . Completion status of operation:

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

EXTRTMETHOD int ASN1TGeneralizedTime::compileString (char *pbuf, OSSIZE bufsize) const

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

Table 3.191. 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.

Asn1TObject struct Reference

`#include <asn1CppTypes.h>`

- Asn1TObject ( )

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.

Definition at line 791 of file asn1CppTypes.h

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

- asn1CppTypes.h

Asn1TObject::Asn1TObject ()

The default constructor creates an empty object value.

ASN1TObjId struct Reference

`#include <ASN1TObjId.h>`

- ASN1TObjId ( )
- virtual EXTRTMETHOD ~ASN1TObjId ( )

- EXTRTMETHOD ASN1TObjId (OSOCTET _numids, const OSUINT32 * _subids)
- EXTRTMETHOD ASN1TObjId (const ASN1OBJID & oid)
- EXTRTMETHOD ASN1TObjId (const ASN1TObjId & oid)
- EXTRTMETHOD ASN1TObjId (const char * dotted_oid_string)
- EXTRTMETHOD ASN1TObjId & operator= (const char * dotted_oid_string)
- EXTRTMETHOD void operator= (const ASN1OBJID & rhs)
- EXTRTMETHOD void operator= (const ASN1TObjId & rhs)
- EXTRTMETHOD ASN1TObjId & operator+= (const char * dotted_oid_string)
- EXTRTMETHOD ASN1TObjId & operator+= (const OSUINT32 i)
- EXTRTMETHOD ASN1TObjId & operator+= (const ASN1TObjId & o)
- EXTRTMETHOD const char * toString (OSCTXT * pctxt)
- EXTRTMETHOD void set_data (const OSUINT32 * raw_oid, OSUINT32 oid_len)
- EXTRTMETHOD int nCompare (const OSUINT32 n, const ASN1TObjId & o)
- EXTRTMETHOD int RnCompare (const OSUINT32 n, const ASN1TObjId & o)
- EXTRTMETHOD void trim (const OSUINT32 n)

Detailed Description

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

Definition at line 49 of file ASN1TObjId.h

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

- ASN1TObjId.h

ASN1TObjId::ASN1TObjId ()

The default constructor creates an empty object identifier value.

virtual EXTRTMETHOD ASN1TObjId::~ASN1TObjId ()

The Virtual Destructor

EXTRTMETHOD ASN1TObjId::ASN1TObjId (OSOCTET _numids, const OSUINT32 *_subids)

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

Table 3.192. Parameters

_numids	- Number of subidentifiers in the OID.
_subids	- Array of subidentifier values.

EXTRTMETHOD ASN1TObjId::ASN1TObjId (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.

Table 3.193. Parameters

oid	- C object identifier value.

EXTRTMETHOD ASN1TObjId::ASN1TObjId (const ASN1TObjId &oid)

The copy constructor.

Table 3.194. Parameters

oid	- C++ object identifier value.

EXTRTMETHOD ASN1TObjId::ASN1TObjId (const char *dotted_oid_string)

Construct an OID from a dotted string.

Table 3.195. Parameters

dotted_oid_string	- for example "1.3.1.6.1.10"

EXTRTMETHOD ASN1TObjId& ASN1TObjId::operator= (const char *dotted_oid_string)

Assignment from a string.

Table 3.196. Parameters

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

EXTRTMETHOD void ASN1TObjId::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.

Table 3.197. Parameters

rhs	- C object identifier value.

EXTRTMETHOD void ASN1TObjId::operator= (const ASN1TObjId &rhs)

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

Table 3.198. Parameters

rhs	- C++ object identifier value.

EXTRTMETHOD ASN1TObjId& ASN1TObjId::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.

Table 3.199. Parameters

dotted_oid_string	- C++ object identifier value.

Returns: . - True if values are equal.

EXTRTMETHOD ASN1TObjId& ASN1TObjId::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.

Table 3.200. Parameters

i	- Subidentifier to append.

Returns: . - True if values are equal.

EXTRTMETHOD ASN1TObjId& ASN1TObjId::operator+= (const ASN1TObjId &o)

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

Table 3.201. Parameters

o	- C++ object identifier value.

Returns: . - True if values are equal.

EXTRTMETHOD const char* ASN1TObjId::toString (OSCTXT *pctxt) const

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

Table 3.202. Parameters

pctxt	- Pointer to a context structure.

Returns: . - Dotted OID string (for example "3.6.1.6")

EXTRTMETHOD void ASN1TObjId::set_data (const OSUINT32 *raw_oid, OSUINT32 oid_len)

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

Table 3.203. Parameters

raw_oid	- Pointer to an array of subidentifier values.
oid_len	- Number of subids in the array,

EXTRTMETHOD int ASN1TObjId::nCompare (const OSUINT32 n, const ASN1TObjId &o) const

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

Table 3.204. 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.

EXTRTMETHOD int ASN1TObjId::RnCompare (const OSUINT32 n, const ASN1TObjId &o) const

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

Table 3.205. 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.

EXTRTMETHOD void ASN1TObjId::trim (const OSUINT32 n)

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

Table 3.206. Parameters

n	- number of subids to trim from OID

ASN1TOpenType struct Reference

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

- ASN1TOpenType ( )

Detailed Description

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

Definition at line 776 of file asn1CppTypes.h

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

- asn1CppTypes.h

ASN1TOpenType::ASN1TOpenType ()

The default constructor creates an empty open type value.

ASN1TPDU struct Reference

`#include <asn1CppTypes.h>`

Protected Attributes

- OSRTCtxtPtr mpContext

- void setContext (OSRTContext * ctxt)
- virtual ~ASN1TPDU ( )

Detailed Description

Base class for PDU types. This class is used as the base class for all compiler-generated PDU types. Control classes do not inherit from this class.

Definition at line 826 of file asn1CppTypes.h

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

- asn1CppTypes.h

Member Data Documentation

OSRTCtxtPtr ASN1TPDU::mpContext

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.

Definition at line 834 of file asn1CppTypes.h

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

- asn1CppTypes.h

void ASN1TPDU::setContext (OSRTContext *ctxt)

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 destuctor 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 dynamicly 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.

Table 3.207. Parameters

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

virtual ASN1TPDU::~ASN1TPDU ()

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

ASN1TPDUSeqOfList struct Reference

`#include <asn1CppTypes.h>`

- ASN1TPDUSeqOfList ( )

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.

Definition at line 884 of file asn1CppTypes.h

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

- asn1CppTypes.h

ASN1TPDUSeqOfList::ASN1TPDUSeqOfList ()

The default constructor creates an empty list.

ASN1TSeqExt struct Reference

`#include <asn1CppTypes.h>`

- ASN1TSeqExt ( )

Detailed Description

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

Definition at line 812 of file asn1CppTypes.h

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

- asn1CppTypes.h

ASN1TSeqExt::ASN1TSeqExt ()

The default constructor creates an empty open extension element.

ASN1TSeqOfList struct Reference

`#include <asn1CppTypes.h>`

- ASN1TSeqOfList ( )

Detailed Description

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

Definition at line 870 of file asn1CppTypes.h

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

- asn1CppTypes.h

ASN1TSeqOfList::ASN1TSeqOfList ()

The default constructor creates an empty list.

ASN1TTime class Reference

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

- enum @0 {
  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 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

- EXTRTMETHOD ASN1TTime ( )
- EXTRTMETHOD ASN1TTime (OSBOOL useDerRules)
- EXTRTMETHOD ASN1TTime (const ASN1TTime & original)
- virtual EXTRTMETHOD ~ASN1TTime ( )
- virtual EXTRTMETHOD int getYear ( )
- virtual EXTRTMETHOD int getMonth ( )
- virtual EXTRTMETHOD int getDay ( )
- virtual EXTRTMETHOD int getHour ( )
- virtual EXTRTMETHOD int getMinute ( )
- virtual EXTRTMETHOD int getSecond ( )
- virtual EXTRTMETHOD int getFraction ( )
- virtual EXTRTMETHOD double getFractionAsDouble ( )
- virtual EXTRTMETHOD int getFractionStr (char *const pBuf, OSSIZE bufSize)
- virtual EXTRTMETHOD int getFractionLen ( )
- virtual EXTRTMETHOD int getDiffHour ( )
- virtual EXTRTMETHOD int getDiffMinute ( )
- virtual EXTRTMETHOD int getDiff ( )
- virtual EXTRTMETHOD OSBOOL getUTC ( )
- virtual EXTRTMETHOD time_t getTime ( )

- void setDER (OSBOOL bvalue)
- virtual EXTRTMETHOD int setUTC (OSBOOL utc)
- virtual EXTRTMETHOD int setYear (short year_)
- virtual EXTRTMETHOD int setMonth (short month_)
- virtual EXTRTMETHOD int setDay (short day_)
- virtual EXTRTMETHOD int setHour (short hour_)
- virtual EXTRTMETHOD int setMinute (short minute_)
- virtual EXTRTMETHOD int setSecond (short second_)
- virtual EXTRTMETHOD int setFraction (int fraction, int fracLen)
- virtual EXTRTMETHOD int setFraction (double frac, int fracLen)
- virtual EXTRTMETHOD int setFraction (char const * frac)
- virtual int setTime (time_t time, OSBOOL diffTime)
- virtual EXTRTMETHOD int setDiffHour (short dhour)
- virtual EXTRTMETHOD int setDiff (short dhour, short dminute)
- virtual EXTRTMETHOD int setDiff (short inMinutes)
- virtual int parseString (const char * string)
- virtual EXTRTMETHOD void clear ( )
- virtual int compileString (char * pbuf, OSSIZE bufsize)
- virtual EXTRTMETHOD int equals (const ASN1TTime &)
- EXTRTMETHOD const char * toString (char * pbuf, OSSIZE bufsize)
- EXTRTMETHOD char * toString (OSCTXT * pctxt)
- EXTRTMETHOD char * toString ( )
- EXTRTMETHOD const ASN1TTime & operator= (const ASN1TTime &)
- virtual EXTRTMETHOD OSBOOL operator== (const ASN1TTime &)
- virtual EXTRTMETHOD OSBOOL operator!= (const ASN1TTime &)
- virtual EXTRTMETHOD OSBOOL operator> (const ASN1TTime &)
- virtual EXTRTMETHOD OSBOOL operator< (const ASN1TTime &)
- virtual EXTRTMETHOD OSBOOL operator>= (const ASN1TTime &)
- virtual EXTRTMETHOD OSBOOL operator<= (const ASN1TTime &)

- static EXTRTMETHOD int checkDate (int day, int month, int year)

- static EXTRTMETHOD void addMilliseconds (int deltaMs, short & year, short & month, short & day, short & hour, short & minute, short & second, int & secFraction, int secFracLen)
- static EXTRTMETHOD void addDays (int deltaDays, short & year, short & month, short & day)
- static EXTRTMETHOD short daysInMonth (int i)
- static EXTRTMETHOD int daysAfterMonth (int i)
- EXTRTMETHOD void privateInit ( )
- EXTRTMETHOD int getDaysNum ( )
- EXTRTMETHOD long getMillisNum ( )
- int ncharsToInt (const char * str, OSSIZE nchars, int & value)

Detailed Description

ASN.1 Time utility base class.

Definition at line 66 of file ASN1TTime.h

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

- ASN1TTime.h

Member Data Documentation

short ASN1TTime::mYear

This member variable holds this time's year.

Definition at line 71 of file ASN1TTime.h

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

- ASN1TTime.h

short ASN1TTime::mMonth

This member variable holds this time's month.

Definition at line 76 of file ASN1TTime.h

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

- ASN1TTime.h

short ASN1TTime::mDay

This member variable holds this time's day.

Definition at line 81 of file ASN1TTime.h

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

- ASN1TTime.h

short ASN1TTime::mHour

This member variable holds this time's hour.

Definition at line 86 of file ASN1TTime.h

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

- ASN1TTime.h

short ASN1TTime::mMinute

This member variable holds this time's minute.

Definition at line 91 of file ASN1TTime.h

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

- ASN1TTime.h

short ASN1TTime::mSecond

This member variable holds this time's second.

Definition at line 96 of file ASN1TTime.h

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

- ASN1TTime.h

short ASN1TTime::mDiffHour

This member variable holds this time's hour differential.

Definition at line 101 of file ASN1TTime.h

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

- ASN1TTime.h

short ASN1TTime::mDiffMin

This member variable holds this time's minute differential.

Definition at line 106 of file ASN1TTime.h

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

- ASN1TTime.h

int ASN1TTime::mSecFraction

This member variable holds this time's fractional seconds.

Definition at line 111 of file ASN1TTime.h

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

- ASN1TTime.h

int ASN1TTime::mSecFracLen

This member variable holds this time's fractional seconds length.

Definition at line 116 of file ASN1TTime.h

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

- ASN1TTime.h

int ASN1TTime::mStatus

This member variable holds this time's status.

Definition at line 121 of file ASN1TTime.h

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

- ASN1TTime.h

OSBOOL ASN1TTime::mbUtcFlag

This member variable tells whether this time is UTC time or not.

Definition at line 126 of file ASN1TTime.h

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

- ASN1TTime.h

OSBOOL ASN1TTime::mbDerRules

This member variable tells whether we will encode this time according to the Distinguished Encoding Rules (DER) or not.

Definition at line 132 of file ASN1TTime.h

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

- ASN1TTime.h

EXTRTMETHOD ASN1TTime::ASN1TTime ()

This constructpor creates an empty time class.

EXTRTMETHOD ASN1TTime::ASN1TTime (OSBOOL useDerRules)

This constructor creates an empty time class.

Table 3.208. Parameters

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

EXTRTMETHOD ASN1TTime::ASN1TTime (const ASN1TTime &original)

The copy constructor.

Table 3.209. Parameters

original	The original time string object value.

virtual EXTRTMETHOD ASN1TTime::~ASN1TTime ()

The destructor.

virtual EXTRTMETHOD int ASN1TTime::getYear () const

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.

virtual EXTRTMETHOD int ASN1TTime::getMonth () const

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 valued 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.

virtual EXTRTMETHOD int ASN1TTime::getDay () const

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 be 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.

virtual EXTRTMETHOD int ASN1TTime::getHour () const

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.

virtual EXTRTMETHOD int ASN1TTime::getMinute () const

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.

virtual EXTRTMETHOD int ASN1TTime::getSecond () const

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.

virtual EXTRTMETHOD int ASN1TTime::getFraction () const

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.

virtual EXTRTMETHOD double ASN1TTime::getFractionAsDouble () const

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.

virtual EXTRTMETHOD int ASN1TTime::getFractionStr (char *const pBuf, OSSIZE bufSize) const

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.

virtual EXTRTMETHOD int ASN1TTime::getFractionLen () const

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.

virtual EXTRTMETHOD int ASN1TTime::getDiffHour () const

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 of 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.

virtual EXTRTMETHOD int ASN1TTime::getDiffMinute () const

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 of 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.

virtual EXTRTMETHOD int ASN1TTime::getDiff () const

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 of 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.

virtual EXTRTMETHOD OSBOOL ASN1TTime::getUTC () const

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.

virtual EXTRTMETHOD time_t ASN1TTime::getTime () const

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 represned 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 ASN1TTime classes.

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

void ASN1TTime::setDER (OSBOOL bvalue)

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

virtual EXTRTMETHOD int ASN1TTime::setUTC (OSBOOL utc)

This method sets teh 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.

Table 3.210. Parameters

utc	UTC flag state.

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD int ASN1TTime::setYear (short year_)

This method sets the year component of the time value.

Note that the action of this method may differ for different inherited ASN1TTime classes.

Table 3.211. Parameters

year_	Year component (full 4 digits).

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD int ASN1TTime::setMonth (short month_)

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.

Table 3.212. Parameters

month_	Month component (1 - 12).

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD int ASN1TTime::setDay (short day_)

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.

Table 3.213. Parameters

day_	Day of month component (1 - 31).

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD int ASN1TTime::setHour (short hour_)

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.

Table 3.214. Parameters

hour_	Hour component (0 - 23).

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD int ASN1TTime::setMinute (short minute_)

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 ASN1TTime classes.

Table 3.215. Parameters

minute_	Minute component (0 - 59).

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD int ASN1TTime::setSecond (short second_)

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 form different inherited ASN1TTime classes.

Table 3.216. Parameters

second_	Second component (0 - 59).

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD int ASN1TTime::setFraction (int fraction, int fracLen=-1)

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.

Table 3.217. 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.

virtual EXTRTMETHOD int ASN1TTime::setFraction (double frac, int fracLen)

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.

Table 3.218. 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.

virtual EXTRTMETHOD int ASN1TTime::setFraction (char const *frac)

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.

Table 3.219. Parameters

frac	Second's decimal fraction component.

Returns: . Completion status of operation:

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

virtual int ASN1TTime::setTime (time_t time, OSBOOL diffTime)=0

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.

Table 3.220. 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.

virtual EXTRTMETHOD int ASN1TTime::setDiffHour (short dhour)

This method sets teh 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.

Table 3.221. 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.

virtual EXTRTMETHOD int ASN1TTime::setDiff (short dhour, short dminute)

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.

Table 3.222. 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.

virtual EXTRTMETHOD int ASN1TTime::setDiff (short inMinutes)

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.

Table 3.223. 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.

virtual int ASN1TTime::parseString (const char *string)=0

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.

Table 3.224. Parameters

string	The time string value to be parsed.

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD void ASN1TTime::clear ()

This method clears the time object.

Note the action of this method may differ for different inherited ASN1TTime classes.

virtual int ASN1TTime::compileString (char *pbuf, OSSIZE bufsize) const =0

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

Table 3.225. 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.

virtual EXTRTMETHOD int ASN1TTime::equals (const ASN1TTime &) const

This method compares times.

EXTRTMETHOD const char* ASN1TTime::toString (char *pbuf, OSSIZE bufsize) const

Get a printable ASCII string of the time value into the specified buffer.

Table 3.226. Parameters

pbuf	Pointer to a destination buffer.
bufsize	Size of destination buffer.

Returns: . Compiled time string. NULL, if error occurs.

EXTRTMETHOD char* ASN1TTime::toString (OSCTXT *pctxt) const

Get a printable ASCII string of the time value. Memory for the returned string is allocated using the rtxMemAlloc function and is freed either when context memory is freed or explicitly via a call to rtxMemFreePtr.

Table 3.227. Parameters

pctxt	Pointer to a context structure.

Returns: . Compiled time string. NULL, if error occurs.

EXTRTMETHOD 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.

ASN1TUniversalString struct Reference

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

- ASN1TUniversalString ( )

Detailed Description

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

Definition at line 762 of file asn1CppTypes.h

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

- asn1CppTypes.h

ASN1TUniversalString::ASN1TUniversalString ()

The default constructor creates an empty UniversalString value.

ASN1TUTCTime class Reference

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

- EXTRTMETHOD ASN1TUTCTime ( )
- EXTRTMETHOD ASN1TUTCTime (const char * timeStr, OSBOOL useDerRules)
- EXTRTMETHOD ASN1TUTCTime (OSBOOL useDerRules)
- ASN1TUTCTime (const ASN1TUTCTime & original)
- EXTRTMETHOD int setYear (short year_)
- EXTRTMETHOD int setTime (time_t time, OSBOOL diffTime)
- EXTRTMETHOD int setUTC (OSBOOL utc)
- EXTRTMETHOD void clear ( )
- EXTRTMETHOD int compileString (char * pbuf, OSSIZE bufsize)
- EXTRTMETHOD int parseString (const char * string)
- const ASN1TUTCTime & operator= (const ASN1TUTCTime & tm)

- EXTRTMETHOD int getFraction ( )
- EXTRTMETHOD int setFraction (int fraction, int fracLen)

Detailed Description

ASN.1 UTCTime utility class. The ASN1TUTTime class is derived from the ASN1TTime base class.

Definition at line 790 of file ASN1TTime.h

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

- ASN1TTime.h

EXTRTMETHOD ASN1TUTCTime::ASN1TUTCTime ()

A default constructor.

EXTRTMETHOD ASN1TUTCTime::ASN1TUTCTime (const char *timeStr, OSBOOL useDerRules=FALSE)

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

Table 3.228. Parameters

timeStr	A pointer to the time string to be parsed.
useDerRules	Create object using DER rules.

EXTRTMETHOD ASN1TUTCTime::ASN1TUTCTime (OSBOOL useDerRules)

This constructor creates an empty time object.

Table 3.229. Parameters

useDerRules	An OSBOOL value.

ASN1TUTCTime::ASN1TUTCTime (const ASN1TUTCTime &original)

A copy constructor.

EXTRTMETHOD int ASN1TUTCTime::setYear (short year_)

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.

Table 3.230. 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.

EXTRTMETHOD int ASN1TUTCTime::setTime (time_t time, OSBOOL diffTime)

Converts time_t to time string.

Table 3.231. 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.

EXTRTMETHOD int ASN1TUTCTime::setUTC (OSBOOL utc)

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.

Table 3.232. Parameters

utc	UTC flag state.

Returns: . Completion status of operation:

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

EXTRTMETHOD void ASN1TUTCTime::clear ()

Clears out the time object.

EXTRTMETHOD int ASN1TUTCTime::compileString (char *pbuf, OSSIZE bufsize) const

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

Table 3.233. 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.

EXTRTMETHOD int ASN1TUTCTime::parseString (const char *string)

Parses the given time string. The string is assumed to be in standard UTC time format.

Table 3.234. Parameters

string	UTC time string to be parsed.

Returns: . Completion status of operation:

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

EXTRTMETHOD int ASN1TUTCTime::getFraction () const

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.

EXTRTMETHOD int ASN1TUTCTime::setFraction (int fraction, int fracLen=-1)

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.

Table 3.235. 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.

ASN1UniversalString class Reference

OSRTBaseType class Reference

```
#include <OSRTBaseType.h>
```

- OSRTBaseType ( )
- virtual ~OSRTBaseType ( )

- virtual OSRTBaseType * clone ( )

Detailed Description

C++ structured type base class. This is the base class for all generated structured types.

Definition at line 37 of file OSRTBaseType.h

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

- OSRTBaseType.h

OSRTContext class Reference

`#include <OSRTContext.h>`

Protected Attributes

- OSCTXT mCtxt
- OSUINT32 mCount
- OSBOOL mbInitialized
- int mStatus

- EXTRTMETHOD OSRTContext (OSCTXT *)

- EXTRTMETHOD OSRTContext ( )
- virtual EXTRTMETHOD ~OSRTContext ( )
- OSCTXT * getPtr ( )
- const OSCTXT * getPtr ( )
- EXTRTMETHOD OSUINT32 getRefCount ( )
- int getStatus ( )
- OSBOOL isInitialized ( )
- EXTRTMETHOD void _ref ( )
- EXTRTMETHOD void _unref ( )
- EXTRTMETHOD char * getErrorInfo ( )
- EXTRTMETHOD char * getErrorInfo (size_t * pBufSize)
- EXTRTMETHOD char * getErrorInfo (char * pBuf, size_t & bufSize)
- void * memAlloc (size_t numocts)
- void * memAllocZ (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)
- virtual EXTRTMETHOD int setRunTimeKey (const OSOCTET * key, size_t keylen)
- int setStatus (int stat)

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.

Definition at line 64 of file OSRTContext.h

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

- OSRTContext.h

Member Data Documentation

OSCTXT OSRTContext::mCtxt

The mCtxt member variable is a standard C runtime context variable used in most C runtime function calls.

Definition at line 73 of file OSRTContext.h

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

- OSRTContext.h

OSUINT32 OSRTContext::mCount

The mCount member variable holds the reference count of this context.

Definition at line 78 of file OSRTContext.h

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

- OSRTContext.h

OSBOOL OSRTContext::mbInitialized

TRUE, if initialized correctly, FALSE otherwise

Definition at line 83 of file OSRTContext.h

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

- OSRTContext.h

int OSRTContext::mStatus

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.

Definition at line 90 of file OSRTContext.h

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

- OSRTContext.h

EXTRTMETHOD OSRTContext::OSRTContext ()

The default constructor initializes the mCtxt member variable and sets the reference count variable (mCount) to zero.

virtual EXTRTMETHOD OSRTContext::~OSRTContext ()

The destructor frees all memory held by the context.

OSCTXT* OSRTContext::getPtr ()

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.

EXTRTMETHOD OSUINT32 OSRTContext::getRefCount ()

The getRefCount method returns the current reference count.

int OSRTContext::getStatus () const

The getStatus method returns the runtime status code value.

Returns: . Runtime status code:

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

OSBOOL OSRTContext::isInitialized ()

Returns TRUE, if initialized correctly, FALSE otherwise.

Returns: . TRUE, if initialized correctly, FALSE otherwise.

EXTRTMETHOD void OSRTContext::_ref ()

The _ref method increases the reference count by one.

EXTRTMETHOD void OSRTContext::_unref ()

The _unref method decreases the reference count by one.

EXTRTMETHOD 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, or NULL if an error occurred.

EXTRTMETHOD 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 []'.

Table 3.236. Parameters

pBufSize	A pointer to buffer size. It will receive the size of allocated dynamic buffer, or (size_t)-1 if an error occurred.

Returns: . A pointer to a newly allocated buffer with error text, or NULL if an error occurred.

EXTRTMETHOD 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 []'.

Table 3.237. 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.

void* OSRTContext::memAlloc (size_t numocts)

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.

Table 3.238. Parameters

numocts	- Number of bytes of memory to allocate

void* OSRTContext::memAllocZ (size_t numocts)

The memAllocZ method allocates and zeroes 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.

Table 3.239. Parameters

numocts	- Number of bytes of memory to allocate

void OSRTContext::memFreeAll ()

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.

void OSRTContext::memFreePtr (void *ptr)

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

Table 3.240. Parameters

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

void* OSRTContext::memRealloc (void *ptr, size_t numocts)

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

Table 3.241. Parameters

ptr	- Original pointer containing dynamic memory to be resized.
numocts	- Number of bytes of memory to allocate

Returns: . Reallocated memory pointer

void OSRTContext::memReset ()

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

void OSRTContext::printErrorInfo ()

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

void OSRTContext::resetErrorInfo ()

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

OSBOOL OSRTContext::setDiag (OSBOOL value=TRUE)

The setDiag method will turn diagnostic tracing on or off.

Table 3.242. Parameters

value	- Boolean value (default = TRUE = on)

Returns: . - Previous state of the diagnostics enabled boolean

virtual EXTRTMETHOD int OSRTContext::setRunTimeKey (const OSOCTET *key, size_t keylen)

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

Table 3.243. 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.

int OSRTContext::setStatus (int stat)

This method sets error status in the context.

Table 3.244. Parameters

stat	Status value.

Returns: . Error status value being set.

OSRTCtxtPtr class Reference

`#include <OSRTContext.h>`

Protected Attributes

- OSRTContext * mPointer

- OSRTCtxtPtr (OSRTContext * rf)
- OSRTCtxtPtr (const OSRTCtxtPtr & o)
- virtual ~OSRTCtxtPtr ( )
- OSRTCtxtPtr & operator= (const OSRTCtxtPtr & rf)
- OSRTCtxtPtr & operator= (OSRTContext * rf)
- operator OSRTContext * ( )
- operator const OSRTContext * ( )
- OSRTContext * operator-> ( )
- const OSRTContext * operator-> ( )
- OSBOOL operator== (const OSRTContext * o)
- OSBOOL isNull ( )
- OSCTXT * getCtxtPtr ( )

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 OSRTContext object.

Definition at line 310 of file OSRTContext.h

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

- OSRTContext.h

Member Data Documentation

OSRTContext* OSRTCtxtPtr::mPointer

The mPointer member variable is a pointer to a reference-counted ASN.1 context wrapper class object.

Definition at line 316 of file OSRTContext.h

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

- OSRTContext.h

OSRTCtxtPtr::OSRTCtxtPtr (OSRTContext *rf=0)

This constructor set the internal context pointer to the given value and, if it is non-zero, increases the reference count by one.

Table 3.245. Parameters

rf	- Pointer to OSRTContext object

OSRTCtxtPtr::OSRTCtxtPtr (const OSRTCtxtPtr &o)

The copy constructor copies the pointer from the source pointer object and, if it is non-zero, increases the reference count by one.

Table 3.246. Parameters

o	- Reference to OSRTCtxtPtr object to be copied

virtual OSRTCtxtPtr::~OSRTCtxtPtr ()

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.

OSRTCtxtPtr& OSRTCtxtPtr::operator= (const OSRTCtxtPtr &rf)

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.

Table 3.247. Parameters

rf	- Pointer to OSRTCtxtPtr smart-pointer object

OSRTCtxtPtr& OSRTCtxtPtr::operator= (OSRTContext *rf)

This assignment operator assigns does a direct assignment of an OSRTContext object to this OSRTCtxtPtr object.

OSRTCtxtPtr::operator OSRTContext * ()

The 'OSRTContext*' operator returns the context object pointer.

OSRTContext* OSRTCtxtPtr::operator-> ()

The '->' operator returns the context object pointer.

OSBOOL OSRTCtxtPtr::operator== (const OSRTContext *o) const

The '==' operator compares two OSRTContext pointer values.

OSBOOL OSRTCtxtPtr::isNull () const

The isNull method returns TRUE if the underlying context pointer is NULL.

OSCTXT* OSRTCtxtPtr::getCtxtPtr ()

This method returns the standard context pointer used in C function calls.

OSRTDList class Reference

OSRTElemNameGuard class Reference

`#include <OSRTContext.h>`

Protected Attributes

- OSCTXT * mpCtxt

- OSRTElemNameGuard (OSCTXT * pctxt, const char * elemName)
- ~OSRTElemNameGuard ( )

Detailed Description

Element name guard class. This class pushes and pops element names from the element name stack within the context for diagnostic and error logging.

Definition at line 421 of file OSRTContext.h

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

- OSRTContext.h

OSRTFastString class Reference

`#include <OSRTFastString.h>`

Protected Attributes

- const OSUTF8CHAR * mValue

- OSRTFastString ( )
- OSRTFastString (const char * strval)
- OSRTFastString (const OSUTF8CHAR * strval)
- OSRTFastString (const OSRTFastString & str)
- virtual ~OSRTFastString ( )
- virtual OSRTStringIF * clone ( )
- virtual const char * getValue ( )
- virtual const OSUTF8CHAR * getUTF8Value ( )
- virtual void print (const char * name)
- virtual void setValue (const char * str)
- virtual void setValue (const OSUTF8CHAR * str)
- OSRTFastString & operator= (const OSRTFastString & original)

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.

Definition at line 43 of file OSRTFastString.h

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

- OSRTFastString.h

OSRTFastString::OSRTFastString ()

The default constructor sets the internal string member variable pointer to null.

OSRTFastString::OSRTFastString (const char *strval)

This constructor initializes the string to contain the given standard ASCII string value.

Table 3.248. Parameters

strval	- Null-terminated C string value

OSRTFastString::OSRTFastString (const OSUTF8CHAR *strval)

This constructor initializes the string to contain the given UTF-8 string value.

Table 3.249. Parameters

strval	- Null-terminated C string value

OSRTFastString::OSRTFastString (const OSRT-FastString &str)

Copy constructor. String data is not copied; the pointer is simply assigned to the target class member variable.

Table 3.250. Parameters

str	- C++ string object to be copied.

virtual OSRTFastString::~OSRTFastString ()

The destructor does nothing.

virtual OSRTStringIF* OSRTFastString::clone ()

This method creates a copy of the given string object.

virtual const char* OSRTFastString::getValue () const

This method returns the pointer to UTF-8 null terminated string as a standard ASCII string.

virtual const OSUTF8CHAR* OSRTFastString::getUTF8Value () const

This method returns the pointer to UTF-8 null terminated string as a UTF-8 string.

virtual void OSRTFastString::print (const char *name)

This method prints the string value to standard output.

Table 3.251. Parameters

name	- Name of generated string variable.

virtual void OSRTFastString::setValue (const char *str)

This method sets the string value to the given string.

Table 3.252. Parameters

str	- C null-terminated string.

virtual void OSRTFastString::setValue (const OSUTF8CHAR *str)

This method sets the string value to the given UTF-8 string value.

Table 3.253. Parameters

str	- C null-terminated UTF-8 string.

OSRTFastString& OSRTFastString::operator= (const OSRTFastString &original)

Assignment operator.

OSRTFileInputStream class Reference

`#include <OSRTFileInputStream.h>`

- EXTRTMETHOD OSRTFileInputStream (const char * pFilename)
- EXTRTMETHOD OSRTFileInputStream (OSRTContext * pContext, const char * pFilename)
- EXTRTMETHOD OSRTFileInputStream (FILE * file)
- EXTRTMETHOD OSRTFileInputStream (OSRTContext * pContext, FILE * file)
- virtual EXTRTMETHOD ~OSRTFileInputStream ( )
- virtual OSBOOL isA (StreamID id)

Detailed Description

Generic file input stream. This class opens an existing file for input in binary mode and reads data from it.

Definition at line 37 of file OSRTFileInputStream.h

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

- OSRTFileInputStream.h

EXTRTMETHOD OSRTFileInputStream::OSRTFileInputStream (const char *pFilename)

Creates and initializes a file input stream using the name of file.

Table 3.254. Parameters

pFilename	Name of file.

See also: . ::rtxStreamFileOpen

EXTRTMETHOD OSRTFileInputStream::OSRTFileInputStream (OSRTContext *pContext, const char *pFilename)

Creates and initializes a file input stream using the name of file.

Table 3.255. Parameters

pContext	Pointer to a context to use.
pFilename	Name of file.

See also: . ::rtxStreamFileOpen

EXTRTMETHOD OSRTFileInputStream::OSRTFileInputStream (FILE *file)

Initializes the file input stream using the opened FILE structure descriptor.

Table 3.256. Parameters

file	Pointer to FILE structure.

See also: . ::rtxStreamFileAttach

EXTRTMETHOD OSRTFileInputStream::OSRTFileInputStream (OSRTContext *pContext, FILE *file)

Initializes the file input stream using the opened FILE structure descriptor.

Table 3.257. Parameters

pContext	Pointer to a context to use.

file	Pointer to FILE structure.

See also: . ::rtxStreamFileAttach

virtual OSBOOL OSRTFileInputStream::isA (StreamID id) const

This method is used to query a stream object in order to determine its actual type.

Table 3.258. Parameters

id	Enumerated stream identifier

Returns: . True if the stream matches the identifier

OSRTFileOutputStream class Reference

`#include <OSRTFileOutputStream.h>`

- EXTRTMETHOD OSRTFileOutputStream (const char * pFilename)
- EXTRTMETHOD OSRTFileOutputStream (OSRTContext * pContext, const char * pFilename)
- EXTRTMETHOD OSRTFileOutputStream (FILE * file)
- EXTRTMETHOD OSRTFileOutputStream (OSRTContext * pContext, FILE * file)
- virtual EXTRTMETHOD ~OSRTFileOutputStream ( )
- virtual OSBOOL isA (StreamID id)

Detailed Description

Generic file output stream. This class opens an existing file for output in binary mode and reads data from it.

Definition at line 37 of file OSRTFileOutputStream.h

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

- OSRTFileOutputStream.h

EXTRTMETHOD OSRTFileOutputStream::OSRTFileOutputStream (const char *pFilename)

Creates and initializes a file output stream using the name of file.

Table 3.259. Parameters

pFilename	Name of file.

Table 3.260. Exceptions

OSStreamException	Stream create or initialize failed.

See also: . ::rtxStreamFileOpen

EXTRTMETHOD OSRTFileOutputStream::OSRTFileOutputStream (OSRT-Context *pContext, const char *pFilename)

Creates and initializes a file output stream using the name of file.

Table 3.261. Parameters

pContext	Pointer to a context to use.
pFilename	Name of file.

Table 3.262. Exceptions

OSStreamException	Stream create or initialize failed.

See also: . ::rtxStreamFileOpen

EXTRTMETHOD OSRTFileOutputStream::OSRTFileOutputStream (FILE *file)

Initializes the file output stream using the opened FILE structure descriptor.

Table 3.263. Parameters

file	Pointer to FILE structure.

Table 3.264. Exceptions

OSStreamException	Stream create or initialize failed.

See also: . ::rtxStreamFileAttach

EXTRTMETHOD OSRTFileOutputStream::OSRTFileOutputStream (OSRT-Context *pContext, FILE *file)

Initializes the file output stream using the opened FILE structure descriptor.

Table 3.265. Parameters

pContext	Pointer to a context to use.
file	Pointer to FILE structure.

Table 3.266. Exceptions

OSStreamException	Stream create or initialize failed.

See also: . ::rtxStreamFileAttach

virtual OSBOOL OSRTFileOutputStream::isA (StreamID id) const

This method is used to query a stream object in order to determine its actual type.

Table 3.267. Parameters

id	Enumerated stream identifier

Returns: . True if the stream matches the identifier

OSRTHexTextInputStream class Reference

`#include <OSRTHexTextInputStream.h>`

Protected Attributes

- OSRTInputStream * mpUnderStream
- OSBOOL mbOwnUnderStream

- EXTRTMETHOD OSRTHexTextInputStream (OSRTInputStream * pstream)
- EXTRTMETHOD ~OSRTHexTextInputStream ( )

- virtual OSBOOL isA (StreamID id)
- void setOwnUnderStream (OSBOOL value)

Detailed Description

Hexadecimal text input stream filter class. This class is created on top of an existing stream class to provide conversion of hexadecimal text input into binary form.

Definition at line 38 of file OSRTHexTextInputStream.h

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

- OSRTHexTextInputStream.h

EXTRTMETHOD OSRTHexTextInputStream::OSRTHexTextInputStream (OSRTInputStream *pstream)

Initializes the input stream using the existing standard input stream. Only file and memory underlying stream types are supported.

Table 3.268. Parameters

pstream	The underlying input stream object. Note that this class will take control of the underlying stream object and delete it upon destruction.

See also: . ::rtxStreamHexTextAttach

EXTRTMETHOD OSRTHexTextInputStream::~OSRTHexTextInputStream ()

The destructor deletes the underlying stream object. That object should be used as nothing more to a surrogate to this object.

virtual OSBOOL OSRTHexTextInputStream::isA (StreamID id) const

This method is used to query a stream object in order to determine its actual type.

Table 3.269. Parameters

id	Enumerated stream identifier

Returns: . True if the stream matches the identifier

void OSRTHexTextInputStream::setOwnUnderStream (OSBOOL value=TRUE)

This method transfers ownership of the underlying stream to the class.

OSRTInputStream class Reference

`#include <OSRTInputStream.h>`

- EXTRTMETHOD OSRTInputStream ( )
- EXTRTMETHOD OSRTInputStream (OSRTContext * mpContext, OSBOOL attachStream)
- virtual EXTRTMETHOD ~OSRTInputStream ( )
- virtual EXTRTMETHOD int close ( )
- virtual EXTRTMETHOD size_t currentPos ( )
- virtual EXTRTMETHOD int flush ( )
- virtual OSBOOL isA (StreamID id)
- virtual OSRTCtxtPtr getContext ( )
- virtual OSCTXT * getCtxtPtr ( )
- virtual char * getErrorInfo ( )
- virtual char * getErrorInfo (char * pBuf, size_t & bufSize)
- virtual int getPosition (size_t * ppos)
- virtual int getStatus ( )
- virtual EXTRTMETHOD OSBOOL isOpened ( )
- virtual EXTRTMETHOD OSBOOL markSupported ( )
- virtual EXTRTMETHOD int mark (size_t readAheadLimit)
- void printErrorInfo ( )
- void resetErrorInfo ( )
- virtual EXTRTMETHOD long read (OSOCTET * pDestBuf, size_t maxToRead)
- virtual EXTRTMETHOD long readBlocking (OSOCTET * pDestBuf, size_t toReadBytes)
- virtual EXTRTMETHOD int reset ( )
- virtual int setPosition (size_t pos)
- virtual EXTRTMETHOD int skip (size_t n)

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.

Definition at line 41 of file OSRTInputStream.h

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

- OSRTInputStream.h

EXTRTMETHOD 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.

Table 3.270. Exceptions

OSRTStreamException	Stream create or initialize failed.

virtual EXTRTMETHOD OSRTInputStream::~OSRTInputStream ()

Virtual destructor. Closes the stream if it was opened.

virtual EXTRTMETHOD int OSRTInputStream::close ()

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

Returns: . Completion status of operation:

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

See also: . ::rtxStreamClose

virtual EXTRTMETHOD size_t OSRTInputStream::currentPos ()

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

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

virtual EXTRTMETHOD int OSRTInputStream::flush ()

Flushes the buffered data to the stream.

Returns: . Completion status of operation:

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

See also: . ::rtxStreamFlush

virtual OSBOOL OSRTInputStream::isA (StreamID id) const

This method is used to query a stream object in order to determine its actual type.

Table 3.271. Parameters

id	Enumerated stream identifier

Returns: . True if the stream matches the identifier

virtual OSRTCtxtPtr OSRTInputStream::getContext ()

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.

virtual OSCTXT* OSRTInputStream::getCtxtPtr ()

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.

virtual char* OSRTInputStream::getErrorInfo ()

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

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

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

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

Table 3.272. 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.

virtual int OSRTInputStream::getPosition (size_t *ppos)

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

Table 3.273. Parameters

ppos	Pointer to a variable to receive position.

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

virtual int OSRTInputStream::getStatus () const

This method returns the completion status of previous operation. It can be used to check completion status of constructors or methods, which do not return completion status.

Returns: . Runtime status code:

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

virtual EXTRTMETHOD OSBOOL OSRTInputStream::isOpened ()

Checks, is the stream opened or not.

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

See also: . ::rtxStreamIsOpened

virtual EXTRTMETHOD OSBOOL OSRTInputStream::markSupported ()

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

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

See also: . ::rtxStreamMarkSupported

virtual EXTRTMETHOD int OSRTInputStream::mark (size_t readAheadLimit)

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.

Table 3.274. 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

void OSRTInputStream::printErrorInfo ()

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

void OSRTInputStream::resetErrorInfo ()

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

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

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

Table 3.275. Parameters

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

See also: . ::rtxStreamRead

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

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

Table 3.276. Parameters

pDestBuf	Pointer to a buffer to receive a data.
toReadBytes	Number of bytes to be read.

See also: . ::rtxStreamRead

virtual EXTRTMETHOD int OSRTInputStream::reset ()

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

Returns: . Completion status of operation:

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

See also: . ::rtxStreamMark, ::rtxStreamReset

virtual int OSRTInputStream::setPosition (size_t pos)

Sets the current stream position to the given offset.

Table 3.277. Parameters

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

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

virtual EXTRTMETHOD int OSRTInputStream::skip (size_t n)

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

Table 3.278. Parameters

n	The number of octets to be skipped.

Returns: . Completion status of operation:

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

See also: . ::rtxStreamSkip

OSRTInputStreamIF class Reference

- enum StreamID {
  Unknown,
  File,
  Memory,
  Socket
  }

- virtual EXTRTMETHOD ~OSRTInputStreamIF ( )
- virtual OSBOOL isA (StreamID id)
- virtual size_t currentPos ( )
- virtual int getPosition (size_t * ppos)
- virtual OSBOOL markSupported ( )
- virtual int mark (size_t readAheadLimit)
- virtual long read (OSOCTET * pDestBuf, size_t maxToRead)
- virtual long readBlocking (OSOCTET * pDestBuf, size_t toReadBytes)
- virtual int reset ( )
- virtual int setPosition (size_t pos)
- virtual int skip (size_t n)

virtual EXTRTMETHOD OSRTInputStreamIF::~OSRTInputStreamIF ()

Virtual destructor. Closes the stream if it was opened.

virtual OSBOOL OSRTInputStreamIF::isA (StreamID id) const =0

This method is used to query a stream object in order to determine its actual type.

Table 3.279. Parameters

id	Enumerated stream identifier

Returns: . True if the stream matches the identifier

virtual size_t OSRTInputStreamIF::currentPos ()=0

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

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

virtual int OSRTInputStreamIF::getPosition (size_t *ppos)=0

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

Table 3.280. Parameters

ppos	Pointer to a variable to receive position.

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

virtual OSBOOL OSRTInputStreamIF::markSupported ()=0

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

virtual int OSRTInputStreamIF::mark (size_t readAhead-Limit)=0

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.

Table 3.281. 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

virtual long OSRTInputStreamIF::read (OSOCTET *pDestBuf, size_t maxToRead)=0

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

Table 3.282. Parameters

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

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

See also: . ::rtxStreamRead

virtual long OSRTInputStreamIF::readBlocking (OSOCTET *pDestBuf, size_t toReadBytes)=0

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

Table 3.283. Parameters

pDestBuf	Pointer to a buffer to receive a data.
toReadBytes	Number of bytes to be read.

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

See also: . ::rtxStreamRead

virtual int OSRTInputStreamIF::reset ()=0

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

virtual int OSRTInputStreamIF::setPosition (size_t pos)=0

Sets the current stream position to the given offset.

Table 3.284. Parameters

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

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

virtual int OSRTInputStreamIF::skip (size_t n)=0

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

Table 3.285. Parameters

n	The number of octets to be skipped.

Returns: . Completion status of operation:

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

See also: . ::rtxStreamSkip

OSRTInputStreamPtr class Reference

Private Attributes

- OSRTInputStreamIF * mPtr

- OSRTInputStreamPtr (OSRTInputStreamIF * ptr)
- ~OSRTInputStreamPtr ( )
- operator OSRTInputStreamIF * ( )
- OSRTInputStreamIF * operator-> ( )

OSRTMemoryInputStream class Reference

`#include <OSRTMemoryInputStream.h>`

- EXTRTMETHOD OSRTMemoryInputStream (const OSOCTET * pMemBuf, size_t bufSize)
- EXTRTMETHOD OSRTMemoryInputStream (OSRTContext * pContext, const OSOCTET * pMemBuf, size_t bufSize)
- virtual EXTRTMETHOD ~OSRTMemoryInputStream ( )
- virtual OSBOOL isA (StreamID id)

Detailed Description

Generic memory input stream. This class provides methods for streaming data from an input memory buffer.

Definition at line 37 of file OSRTMemoryInputStream.h

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

- OSRTMemoryInputStream.h

EXTRTMETHOD OSRTMemoryInputStream::OSRTMemoryInputStream (const OSOCTET *pMemBuf, size_t bufSize)

Initializes the memory input stream using the specified memory buffer.

Table 3.286. Parameters

pMemBuf	The pointer to the buffer.
bufSize	The size of the buffer.

See also: . ::rtxStreamMemoryAttach

EXTRTMETHOD OSRTMemoryInputStream::OSRTMemoryInputStream (OSRTContext *pContext, const OSOCTET *pMemBuf, size_t bufSize)

Initializes the memory input stream using the specified memory buffer.

Table 3.287. Parameters

pContext	Pointer to a context to use.
pMemBuf	The pointer to the buffer.
bufSize	The size of the buffer.

See also: . ::rtxStreamMemoryAttach

virtual OSBOOL OSRTMemoryInputStream::isA (StreamID id) const

This method is used to query a stream object in order to determine its actual type.

Table 3.288. Parameters

id	Enumerated stream identifier

Returns: . True if the stream matches the identifier

OSRTMemoryOutputStream class Reference

```
#include <OSRTMemoryOutputStream.h>
```

- EXTRTMETHOD OSRTMemoryOutputStream ( )
- EXTRTMETHOD OSRTMemoryOutputStream (OSOCTET * pMemBuf, size_t bufSize)
- EXTRTMETHOD OSRTMemoryOutputStream (OSRTContext * pContext, OSOCTET * pMemBuf, size_t bufSize)
- virtual EXTRTMETHOD ~OSRTMemoryOutputStream ( )
- EXTRTMETHOD OSOCTET * getBuffer (size_t * pSize)
- virtual OSBOOL isA (StreamID id)
- int reset ( )

Detailed Description

Generic memory output stream. This class provides methods for streaming data to an output memory buffer.

Definition at line 37 of file OSRTMemoryOutputStream.h

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

- OSRTMemoryOutputStream.h

EXTRTMETHOD OSRTMemoryOutputStream::OSRTMemoryOutputStream ()

The default constructor initializes the memory output stream to use a dynamic memory output buffer. The status of the construction can be obtained by calling the `getStatus` method.

See also: . ::rtxStreamMemoryCreate

EXTRTMETHOD OSRTMemoryOutputStream::OSRTMemoryOutputStream (OSOCTET *pMemBuf, size_t bufSize)

Initializes the memory output stream using the specified memory buffer. The status of the construction can be obtained by calling the `getStatus` method.

Table 3.289. Parameters

pMemBuf	The pointer to the buffer.
bufSize	The size of the buffer.

See also: . ::rtxStreamMemoryAttach

EXTRTMETHOD OSRTMemoryOutputStream::OSRTMemoryOutputStream (OSRTContext *pContext, OSOCTET *pMemBuf, size_t bufSize)

Initializes the memory output stream using the specified memory buffer. The status of the construction can be obtained by calling the `getStatus` method.

Table 3.290. Parameters

pContext	Pointer to a context to use.
pMemBuf	The pointer to the buffer.
bufSize	The size of the buffer.

See also: . ::rtxStreamMemoryAttach

EXTRTMETHOD OSOCTET* OSRTMemoryOutputStream::getBuffer (size_t *pSize=0)

This method returns the address of the memory buffer to which data was written. If the buffer memory is dynamic, it may be freed using the rtxMemFreePtr function or it will be freed when the stream object is destroyed.

Table 3.291. Parameters

pSize	Pointer to a size variable to receive the number of bytes written to the stream. This is an optional parameter, if a null pointer is passed, size is not returned.

Returns: . Pointer to memory buffer.

virtual OSBOOL OSRTMemoryOutputStream::isA (StreamID id) const

This method is used to query a stream object in order to determine its actual type.

Table 3.292. Parameters

id	Enumerated stream identifier

Returns: . True if the stream matches the identifier

int OSRTMemoryOutputStream::reset ()

This method resets the output memory stream internal buffer to allow it to be overwritten with new data. Memory for the buffer is not freed.

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

OSRTMessageBuffer class Reference

`#include <OSRTMsgBuf.h>`

Protected Attributes

- OSRTCtxtHolder mCtxtHolder
- Type mBufferType
- EXTRTMETHOD OSRTMessageBuffer (Type bufferType, OSRTContext * pContext)
- 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 ( )
- virtual size_t getMsgLen ( )
- int getStatus ( )
- virtual int init ( )
- virtual EXTRTMETHOD int initBuffer (OSOCTET * pMsgBuf, size_t msgBufLen)
- virtual void printErrorInfo ( )
- virtual void resetErrorInfo ( )
- virtual void setAppInfo (void *)
- virtual EXTRTMETHOD void setDiag (OSBOOL value)

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.

Definition at line 47 of file OSRTMsgBuf.h

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

- OSRTMsgBuf.h

Member Data Documentation

Type OSRTMessageBuffer::mBufferType

The mBufferType member variable holds information on the derived message buffer class type (for example, XMLEncode).

Definition at line 55 of file OSRTMsgBuf.h

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

- OSRTMsgBuf.h

EXTRTMETHOD OSRTMessageBuffer::OSRTMessageBuffer (Type bufferType, OSRTContext *pContext=0)

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

Table 3.293. 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.

virtual OSRTMessageBuffer::~OSRTMessageBuffer ()

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

virtual void* OSRTMessageBuffer::getAppInfo ()

Returns a pointer to application-specific information block

virtual size_t OSRTMessageBuffer::getByteIndex ()

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.

virtual OSRTCtxtPtr OSRTMessageBuffer::getContext ()

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

virtual OSCTXT* OSRTMessageBuffer::getCtxtPtr ()

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

virtual char* OSRTMessageBuffer::getErrorInfo ()

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.

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

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

Table 3.294. 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.

virtual OSOCTET* OSRTMessageBuffer::getMsgCopy ()

The getMsgCopy method will return a copy of the encoded message managed by the object.

virtual const OSOCTET* OSRTMessageBuffer::getMsgPtr ()

The getMsgPtr method will return a const pointer to the encoded message managed by the object.

virtual size_t OSRTMessageBuffer::getMsgLen ()

This method returns the length in bytes of an encoded message.

int OSRTMessageBuffer::getStatus () const

This method returns the completion status of previous operation. It can be used to check completion status of constructors or methods, which do not return completion status.

Returns: . Runtime status code:

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

virtual int OSRTMessageBuffer::init ()

Initializes message buffer.

Returns: . Completion status of operation:

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

virtual EXTRTMETHOD int OSRTMessageBuffer::initBuffer (OSOCTET *pMsgBuf, size_t msgBufLen)

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

Table 3.295. Parameters

pMsgBuf	Pointer to message buffer.
msgBufLen	Length of message buffer in bytes.

Returns: . Completion status of operation:

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

virtual void OSRTMessageBuffer::printErrorInfo ()

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

virtual void OSRTMessageBuffer::resetErrorInfo ()

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

virtual void OSRTMessageBuffer::setAppInfo (void *)

Sets the application-specific information block.

virtual EXTRTMETHOD void OSRTMessageBuffer::setDiag (OSBOOL value=TRUE)

The setDiag method will turn diagnostic tracing on or off.

Table 3.296. Parameters

value	- Boolean value (default = TRUE = on)

OSRTMessageBufferIF class Reference

`#include <OSRTMsgBufIF.h>`

- enum Type {
  BEREncode,
  BERDecode,
  PEREncode,
  PERDecode,
  XMLEncode,
  XMLDecode,
  JSONEncode,
  JSONDecode,
  Stream,
  OEREncode,
  OERDecode,
  CBOREncode,
  CBORDecode,
  AVNEncode,
  AVNDecode
  }

- virtual ~OSRTMessageBufferIF ( )

- virtual void * getAppInfo ( )
- virtual size_t getByteIndex ( )
- virtual OSRTCtxtPtr getContext ( )
- virtual OSCTXT * getCtxtPtr ( )
- virtual OSOCTET * getMsgCopy ( )
- virtual const OSOCTET * getMsgPtr ( )
- virtual int init ( )
- virtual int initBuffer (OSOCTET * pMsgBuf, size_t msgBufLen)
- virtual OSBOOL isA (Type bufferType)
- virtual void setAppInfo (void * pAppInfo)
- virtual void setNamespace (const OSUTF8CHAR * , const OSUTF8CHAR * , OSRTDList *)
- virtual void setDiag (OSBOOL value)

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.

Definition at line 47 of file OSRTMsgBufIF.h

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

- OSRTMsgBufIF.h

virtual OSRTMessageBufferIF::~OSRTMessageBufferIF ()

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

virtual void* OSRTMessageBufferIF::getAppInfo ()=0

Returns a pointer to application-specific information block

virtual size_t OSRTMessageBufferIF::getByteIndex ()=0

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.

virtual OSOCTET* OSRTMessageBufferIF::getMsgCopy ()=0

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.

virtual const OSOCTET* OSRTMessageBufferIF::getMsgPtr ()=0

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.

virtual int OSRTMessageBufferIF::init ()=0

Initializes message buffer.

Returns: . Completion status of operation:

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

virtual int OSRTMessageBufferIF::initBuffer (OSOCTET *pMsgBuf, size_t msgBufLen)=0

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

Table 3.297. Parameters

pMsgBuf	Pointer to message buffer.
msgBufLen	Length of message buffer in bytes. string.

Returns: . Completion status of operation:

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

virtual OSBOOL OSRTMessageBufferIF::isA (Type bufferType)=0

This method checks the type of the message buffer.

Table 3.298. Parameters

bufferType	Enumerated identifier specifying a derived class. Possible values are: BEREncode, BERDecode, PEREncode, PERDecode, XMLEncode, XMLDecode, Stream, OEREncode, OERDecode, CBOREncode, CBORDecode.

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

virtual void OSRTMessageBufferIF::setAppInfo (void *pAppInfo)=0

Sets the application-specific information block.

virtual void OSRTMessageBufferIF::setNamespace (const OSUTF8CHAR *, const OSUTF8CHAR *, OSRTDList *=0)

Sets the namespace information.

virtual void OSRTMessageBufferIF::setDiag (OSBOOL value=TRUE)=0

The setDiag method will turn diagnostic tracing on or off.

Table 3.299. Parameters

value	- Boolean value (default = TRUE = on)

OSRTOutputStream class Reference

```
#include <OSRTOutputStream.h>
```

- EXTRTMETHOD OSRTOutputStream ( )
- EXTRTMETHOD OSRTOutputStream (OSRTContext * mpContext, OSBOOL attachStream)
- virtual EXTRTMETHOD ~OSRTOutputStream ( )
- virtual EXTRTMETHOD int close ( )
- virtual EXTRTMETHOD int flush ( )
- virtual OSRTCtxtPtr getContext ( )
- virtual OSCTXT * getCtxtPtr ( )
- virtual char * getErrorInfo ( )
- virtual char * getErrorInfo (char * pBuf, size_t & bufSize)
- virtual int getStatus ( )
- virtual OSBOOL isA (StreamID id)
- virtual EXTRTMETHOD OSBOOL isOpened ( )
- void printErrorInfo ( )
- void resetErrorInfo ( )
- virtual EXTRTMETHOD long write (const OSOCTET * pdata, size_t size)
- virtual EXTRTMETHOD long write (const char * pdata)

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.

Definition at line 45 of file OSRTOutputStream.h

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

- OSRTOutputStream.h

EXTRTMETHOD 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.

virtual EXTRTMETHOD OSRTOutputStream::~OSRTOutputStream ()

Virtual destructor. Closes the stream if it was opened.

virtual EXTRTMETHOD int OSRTOutputStream::close ()

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

virtual EXTRTMETHOD int OSRTOutputStream::flush ()

Flushes the buffered data to the stream.

Returns: . Completion status of operation:

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

See also: . ::rtxStreamFlush

virtual OSRTCtxtPtr OSRTOutputStream::getContext ()

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.

virtual OSCTXT* OSRTOutputStream::getCtxtPtr ()

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.

virtual char* OSRTOutputStream::getErrorInfo ()

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

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

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

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

Table 3.300. 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.

virtual int OSRTOutputStream::getStatus () const

This method returns the completion status of previous operation. It can be used to check completion status of constructors or methods, which do not return completion status.

Returns: . Runtime status code:

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

virtual OSBOOL OSRTOutputStream::isA (StreamID id) const

This method is used to query a stream object in order to determine its actual type.

Table 3.301. Parameters

id	Enumerated stream identifier

Returns: . True if the stream matches the identifier

virtual EXTRTMETHOD OSBOOL OSRTOutputStream::isOpened ()

Checks if the stream open or not.

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

See also: . ::rtxStreamIsOpened

void OSRTOutputStream::printErrorInfo ()

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

void OSRTOutputStream::resetErrorInfo ()

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

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

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

Table 3.302. 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

virtual EXTRTMETHOD long OSRTOutputStream::write (const char *pdata)

Write data to the stream. This method writes data from a null-terminated character string to the output stream.

Table 3.303. Parameters

pdata	The pointer to the data to be written.

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

See also: . ::rtxStreamWrite

OSRTOutputStreamIF class Reference

- enum StreamID {
  Unknown,
  File,
  Memory,
  Socket
  }

- virtual EXTRTMETHOD ~OSRTOutputStreamIF ( )
- virtual OSBOOL isA (StreamID id)
- virtual long write (const OSOCTET * pdata, size_t size)

virtual EXTRTMETHOD OSRTOutputStreamIF::~OSRTOutputStreamIF ()

Virtual destructor. Closes the stream if it was opened.

virtual OSBOOL OSRTOutputStreamIF::isA (StreamID id) const =0

This method is used to query a stream object in order to determine its actual type.

Table 3.304. Parameters

id	Enumerated stream identifier

Returns: . True if the stream matches the identifier

virtual long OSRTOutputStreamIF::write (const OSOCTET *pdata, size_t size)=0

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

Table 3.305. Parameters

pdata	Pointer to the data to be written.
size	The number of octets to write.

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

See also: . ::rtxStreamWrite

OSRTOutputStreamPtr class Reference

Private Attributes

- OSRTOutputStreamIF * mPtr
- OSRTOutputStreamPtr (OSRTOutputStreamIF * ptr)

- ~OSRTOutputStreamPtr ( )
- operator OSRTOutputStreamIF * ( )
- OSRTOutputStreamIF * operator-> ( )

OSRTSocket class Reference

`#include <OSRTSocket.h>`

Protected Attributes

- OSRTSOCKET mSocket

  handle of the socket

- int mInitStatus
- int mStatus
- int mRetryCount

  number of time to retry socker connect

- OSBOOL mOwner

  indicates this class owns the socket

- OSBOOL isInitialized ( )

- EXTRTMETHOD OSRTSocket ( )
- EXTRTMETHOD OSRTSocket (OSRTSOCKET socket, OSBOOL ownership, int retryCount)
- EXTRTMETHOD OSRTSocket (const OSRTSocket & socket)
- EXTRTMETHOD ~OSRTSocket ( )
- EXTRTMETHOD OSRTSocket * accept (OSIPADDR * destIP, int * port)
- EXTRTMETHOD int bind (OSIPADDR addr, int port)
- EXTRTMETHOD int bindUrl (const char * url)
- EXTRTMETHOD int bind (const char * pAddrStr, int port)
- int bind (int port)
- EXTRTMETHOD int blockingRead (OSOCTET * pbuf, size_t readBytes)
- EXTRTMETHOD int close ( )
- EXTRTMETHOD int connect (const char * host, int port)
- EXTRTMETHOD int connectTimed (const char * host, int port, int nsecs)

- EXTRTMETHOD int connectUrl (const char * url)
- OSBOOL getOwnership ( )
- OSRTSOCKET getSocket ( )
- int getStatus ( )
- EXTRTMETHOD int listen (int maxConnections)
- EXTRTMETHOD int recv (OSOCTET * pbuf, size_t bufsize)
- EXTRTMETHOD int send (const OSOCTET * pdata, size_t size)
- void setOwnership (OSBOOL ownership)
- void setRetryCount (int value)

- static EXTRTMETHOD const char * addrToString (OSIPADDR ipAddr, char * pAddrStr, size_t bufsize)
- static EXTRTMETHOD OSIPADDR stringToAddr (const char * pAddrStr)

Detailed Description

Wrapper class for TCP/IP or UDP sockets.

Definition at line 50 of file OSRTSocket.h

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

- OSRTSocket.h

EXTRTMETHOD OSRTSocket::OSRTSocket ()

This is the default constructor. It initializes all internal members with default values and creates a new socket structure. Use getStatus() method to determine has error occurred during the initialization or not.

EXTRTMETHOD OSRTSocket::OSRTSocket (OSRTSOCKET socket, OSBOOL ownership=FALSE, int retryCount=1)

This constructor initializes an instance by using an existing socket.

Table 3.306. 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.

retryCount	Number of times to retry a socket connect operation.

EXTRTMETHOD OSRTSocket::OSRTSocket (const OSRTSocket &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.

EXTRTMETHOD OSRTSocket::~OSRTSocket ()

The destructor. This closes socket if the instance is the owner of the socket.

EXTRTMETHOD OSRTSocket* OSRTSocket::accept (OSIPADDR *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.

Table 3.307. 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 OSRTSocket::getStatus method to obtain error code.

See also: . ::rtxSocketAccept

EXTRTMETHOD 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 OSRTSocket::connect or OSRTSocket::listen methods.

Table 3.308. 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: . ::rtxSocketBind

EXTRTMETHOD int OSRTSocket::bindUrl (const char *url)

This method associates a local address with a socket. It is used on an unconnected socket before subsequent calls to the OSRTSocket::connect or OSRTSocket::listen methods. This version of the method allows a URL to be used instead of address and port number.

Table 3.309. Parameters

url	Univeral resource locator (URL) string.

Returns: . Completion status of operation:

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

See also: . ::rtxSocketBind

EXTRTMETHOD 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 OSRTSocket::connect or OSRTSocket::listen methods.

Table 3.310. 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: . ::rtxSocketBind

int OSRTSocket::bind (int port)

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.

Table 3.311. 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: . ::rtxSocketBind
bind ()

EXTRTMETHOD int OSRTSocket::blockingRead (OSOCTET *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.

Table 3.312. 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.

EXTRTMETHOD int OSRTSocket::close ()

This method closes this socket.

Returns: . Completion status of operation:

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

See also: . ::rtxSocketClose

EXTRTMETHOD 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.

Table 3.313. 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: . ::rtxSocketConnect

EXTRTMETHOD int OSRTSocket::connectTimed (const char *host, int port, int nsecs)

This method establishes a connection to this socket. It is similar to the socketConnect method except that it will only wait the given number of seconds to establish a connection before giving up.

Table 3.314. Parameters

host	Null-terminated character string representing a number expressed in the Internet standard "." (dotted) notation.
port	The destination port to connect.
nsecs	Number of seconds to wait before failing.

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

EXTRTMETHOD int OSRTSocket::connectUrl (const char *url)

This method establishes a connection to this socket. It is used to create a connection to the specified destination. In this version, destination is specified using a URL.

Table 3.315. Parameters

url	Univeral resource locator (URL) string.

Returns: . Completion status of operation:

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

See also: . ::rtxSocketConnect

OSBOOL OSRTSocket::getOwnership ()

Returns the ownership of underlying O/S socket.

Returns: . TRUE, if the socket object has the ownership of underlying O/S socket.

OSRTSOCKET OSRTSocket::getSocket () const

This method returns the handle of the socket.

Returns: . The handle of the socket.

int OSRTSocket::getStatus ()

Returns a completion status of last operation.

Returns: . Completion status of last operation:

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

EXTRTMETHOD int OSRTSocket::listen (int maxConnections)

This method places a socket into a state where it is listening for an incoming connection.

Table 3.316. Parameters

maxConnections	Maximum length of the queue of pending connections.

Returns: . Completion status of operation:

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

See also: . ::rtxSocketListen

EXTRTMETHOD int OSRTSocket::recv (OSOCTET *pbuf, size_t 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.

Table 3.317. 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: . ::rtxSocketRecv

EXTRTMETHOD int OSRTSocket::send (const OSOCTET *pdata, size_t size)

This method sends data on a connected socket. It is used to write outgoing data on a connected socket.

Table 3.318. 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: . ::rtxSocketSend

void OSRTSocket::setOwnership (OSBOOL ownership)

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.

Table 3.319. Parameters

ownership	TRUE, if socket object should have ownership of the underlying O/S socket; FALSE, otherwise.

void OSRTSocket::setRetryCount (int value)

This method sets the socket connect retry count. The connect operation will be retried this many times if the operation fails. By default, the connect operation is not retried.

Table 3.320. Parameters

value	Retry count.

static EXTRTMETHOD const char* OSRTSocket::addrToString (OSIPADDR ipAddr, char *pAddrStr, size_t bufsize)

This method converts an IP address to its string representation.

Table 3.321. 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.

static EXTRTMETHOD OSIPADDR OSRTSocket::stringToAddr (const char *pAddrStr)

This method converts a string containing an Internet Protocol dotted address into a proper OSIPADDR address.

Table 3.322. 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.

OSRTSocketInputStream class Reference

`#include <OSRTSocketInputStream.h>`

Protected Attributes

- OSRTSocket mSocket

  a socket

- EXTRTMETHOD OSRTSocketInputStream (OSRTSocket & socket)
- EXTRTMETHOD OSRTSocketInputStream (OSRTContext * pContext, OSRTSocket & socket)
- EXTRTMETHOD OSRTSocketInputStream (OSRTSOCKET socket, OSBOOL ownership)
- OSRTSocketInputStream (OSRTContext * pContext, OSRTSOCKET socket, OSBOOL ownership)
- virtual EXTRTMETHOD ~OSRTSocketInputStream ( )
- virtual OSBOOL isA (StreamID id)

Detailed Description

Generic socket input stream. This class opens an existing socket for input in binary mode and reads data from it.

Definition at line 40 of file OSRTSocketInputStream.h

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

- OSRTSocketInputStream.h

EXTRTMETHOD OSRTSocketInputStream::OSRTSocketInputStream (OSRTSocket &socket)

Creates and initializes a socket input stream using the OSRTSocket instance of socket.

Table 3.323. Parameters

socket	Reference to OSRTSocket instance.

See also: . ::rtxStreamSocketAttach

EXTRTMETHOD OSRTSocketInputStream::OSRTSocketInputStream (OSRTContext *pContext, OSRTSocket &socket)

Creates and initializes a socket input stream using the OSRTSocket instance of socket.

Table 3.324. Parameters

pContext	Pointer to a context to use.
socket	Reference to OSRTSocket instance.

See also: . ::rtxStreamSocketAttach

EXTRTMETHOD OSRTSocketInputStream::OSRTSocketInputStream (OSRTSOCKET socket, OSBOOL ownership=FALSE)

Creates and initializes the socket input stream using the socket handle.

Table 3.325. 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

OSRTSocketInputStream::OSRTSocketInputStream (OSRTContext *pContext, OSRTSOCKET socket, OSBOOL ownership=FALSE)

Creates and initializes the socket input stream using the socket handle.

Table 3.326. 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

virtual OSBOOL OSRTSocketInputStream::isA (StreamID id) const

This method is used to query a stream object in order to determine its actual type.

Table 3.327. Parameters

id	Enumerated stream identifier

Returns: . True if the stream matches the identifier

OSRTSocketOutputStream class Reference

`#include <OSRTSocketOutputStream.h>`

Protected Attributes

- OSRTSocket mSocket

  a socket

- EXTRTMETHOD OSRTSocketOutputStream (OSRTSocket & socket)
- EXTRTMETHOD OSRTSocketOutputStream (OSRTContext * pContext, OSRTSocket & socket)
- EXTRTMETHOD OSRTSocketOutputStream (OSRTSOCKET socket, OSBOOL ownership)
- OSRTSocketOutputStream (OSRTContext * pContext, OSRTSOCKET socket, OSBOOL ownership)
- virtual EXTRTMETHOD ~OSRTSocketOutputStream ( )
- virtual OSBOOL isA (StreamID id)

Detailed Description

Generic socket output stream. This class opens an existing socket for output in binary mode and reads data from it.

Definition at line 40 of file OSRTSocketOutputStream.h

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

- OSRTSocketOutputStream.h

EXTRTMETHOD OSRTSocketOutputStream::OSRTSocketOutputStream (OSRTSocket &socket)

Creates and initializes a socket output stream using the OSRTSocket instance of socket.

Table 3.328. Parameters

socket	Reference to OSRTSocket instance.

See also: . ::rtxStreamSocketAttach

EXTRTMETHOD OSRTSocketOutputStream::OSRTSocketOutputStream (OSRTContext *pContext, OSRTSocket &socket)

Creates and initializes a socket output stream using the OSRTSocket instance of socket.

Table 3.329. Parameters

pContext	Pointer to a context to use.
socket	Reference to OSRTSocket instance.

See also: . ::rtxStreamSocketAttach

EXTRTMETHOD OSRTSocketOutputStream::OSRTSocketOutputStream (OSRTSOCKET socket, OSBOOL ownership=FALSE)

Initializes the socket output stream using the socket handle.

Table 3.330. 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

OSRTSocketOutputStream::OSRTSocketOutputStream (OSRTContext *pContext, OSRTSOCKET socket, OSBOOL ownership=FALSE)

Initializes the socket output stream using the socket handle.

Table 3.331. 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

virtual OSBOOL OSRTSocketOutputStream::isA (StreamID id) const

This method is used to query a stream object in order to determine its actual type.

Table 3.332. Parameters

id	Enumerated stream identifier

Returns: . True if the stream matches the identifier

OSRTStream class Reference

`#include <OSRTStream.h>`

Protected Attributes

- OSRTCtxtHolder mCtxtHolder
- OSBOOL mbAttached

  Flag, TRUE for "attached" streams.
- int mStatus

  Last stream operation status.

- int mInitStatus

  Initialization status. 0 if initialized successfully.

- EXTRTMETHOD OSRTStream (OSRTContext * pContext, OSBOOL attachStream)
- EXTRTMETHOD OSRTStream (OSRTStream & original)
- EXTRTMETHOD OSRTStream ( )
- EXTRTMETHOD char * getErrorInfo (size_t * pBufSize)

- virtual EXTRTMETHOD ~OSRTStream ( )
- virtual EXTRTMETHOD int close ( )
- virtual EXTRTMETHOD int flush ( )
- virtual OSRTCtxtPtr getContext ( )
- virtual OSCTXT * getCtxtPtr ( )
- virtual char * getErrorInfo ( )
- virtual char * getErrorInfo (char * pBuf, size_t & bufSize)
- int getStatus ( )
- OSBOOL isInitialized ( )
- virtual EXTRTMETHOD OSBOOL isOpened ( )
- void printErrorInfo ( )
- void resetErrorInfo ( )

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.

Definition at line 44 of file OSRTStream.h

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

- OSRTStream.h

EXTRTMETHOD OSRTStream::OSRTStream ()

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.

virtual EXTRTMETHOD OSRTStream::~OSRTStream ()

Virtual destructor. Closes the stream if it was opened.

virtual EXTRTMETHOD int OSRTStream::close ()

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

Returns: . Completion status of operation:

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

See also: . ::rtxStreamClose

virtual EXTRTMETHOD int OSRTStream::flush ()

Flushes the buffered data to the stream.

Returns: . Completion status of operation:

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

See also: . ::rtxStreamFlush

virtual OSRTCtxtPtr OSRTStream::getContext ()

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.

virtual OSCTXT* OSRTStream::getCtxtPtr ()

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.

virtual char* OSRTStream::getErrorInfo ()

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

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

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

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

Table 3.333. 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.

int OSRTStream::getStatus () const

This method returns the completion status of previous operation. It can be used to check completion status of constructors or methods, which do not return completion status.

Returns: . Runtime status code:

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

virtual EXTRTMETHOD OSBOOL OSRTStream::isOpened ()

Checks, is the stream opened or not.

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

See also: . ::rtxStreamIsOpened

void OSRTStream::printErrorInfo ()

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

void OSRTStream::resetErrorInfo ()

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

OSRTStreamIF class Reference

- virtual int close ( )
- virtual int flush ( )
- virtual OSRTCtxtPtr getContext ( )
- virtual OSCTXT * getCtxtPtr ( )
- virtual OSBOOL isOpened ( )
- virtual ~OSRTStreamIF ( )

virtual int OSRTStreamIF::close ()=0

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

Returns: . Completion status of operation:

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

See also: . ::rtxStreamClose

virtual int OSRTStreamIF::flush ()=0

Flushes buffered data to the stream.

Returns: . Completion status of operation:

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

See also: . ::rtxStreamFlush

virtual OSBOOL OSRTStreamIF::isOpened ()=0

Checks if the stream is opened or not.

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

See also: . ::rtxStreamIsOpened

OSRTString class Reference

`#include <OSRTString.h>`

Protected Attributes

- char * mpValue

- OSRTString ( )
- OSRTString (const char * strval)
- OSRTString (const OSUTF8CHAR * strval)
- OSRTString (const OSRTString & str)
- virtual ~OSRTString ( )
- virtual OSRTStringIF * clone ( )
- const char * data ( )

- virtual const char * getValue ( )
- virtual const OSUTF8CHAR * getUTF8Value ( )
- int indexOf (char ch)
- size_t length ( )
- virtual void print (const char * name)
- virtual EXTRTMETHOD void setValue (const char * strval)
- virtual EXTRTMETHOD void setValue (const OSUTF8CHAR * strval)
- virtual EXTRTMETHOD void setValuePtr (char * strval)
- bool toInt (OSINT32 & value)
- bool toSize (OSSIZE & value)
- bool toUInt (OSUINT32 & value)
- bool toUInt64 (OSUINT64 & value)
- EXTRTMETHOD OSRTString & operator= (const OSRTString & original)
- operator const char * (void)

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.

Definition at line 49 of file OSRTString.h

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

- OSRTString.h

OSRTString::OSRTString ()

The default constructor creates an empty string.

OSRTString::OSRTString (const char *strval)

This constructor initializes the string to contain the given standard ASCII string value.

Table 3.334. Parameters

strval	- Null-terminated C string value

OSRTString::OSRTString (const OSUTF8CHAR *strval)

This constructor initializes the string to contain the given UTF-8 string value.

Table 3.335. Parameters

strval	- Null-terminated C string value

OSRTString::OSRTString (const OSRTString &str)

Copy constructor.

Table 3.336. Parameters

str	- C++ string object to be copied.

virtual OSRTString::~OSRTString ()

The destructor frees string memory using the standard 'delete' operator.

virtual OSRTStringIF* OSRTString::clone ()

This method creates a copy of the given string object.

const char* OSRTString::data () const

This method is a synonym for getValue().

virtual const char* OSRTString::getValue () const

This method returns the pointer to UTF-8 null terminated string as a standard ASCII string.

virtual const OSUTF8CHAR* OSRTString::getUTF8Value () const

This method returns the pointer to UTF-8 null terminated string as a UTF-8 string.

int OSRTString::indexOf (char ch) const

This method returns the index of the first occurence of the given characetr within the string or -1 if the character is not found.

size_t OSRTString::length () const

This method returns the length of the string.

virtual void OSRTString::print (const char *name)

This method prints the string value to standard output.

Table 3.337. Parameters

name	- Name of generated string variable.

virtual EXTRTMETHOD void OSRTString::setValue (const char *strval)

This method sets the string value to the given string.

Table 3.338. Parameters

str	- C null-terminated string.

virtual EXTRTMETHOD void OSRTString::setValue (const OSUTF8CHAR *strval)

This method sets the string value to the given UTF-8 string value.

Table 3.339. Parameters

str	- C null-terminated UTF-8 string.

virtual EXTRTMETHOD void OSRTString::setValuePtr (char *strval)

This method sets the string value to the given string value pointer. This is assumed to be a mutable string allocated with the new operator. This class will assume ownership of the string memory.

Table 3.340. Parameters

str	- Mutable null-terminated string allocated with new.

bool OSRTString::toInt (OSINT32 &value) const

This method converts the string to a signed 32-bit integer value.

Table 3.341. Parameters

value	Reference to variable to receive converted integer value.

Returns: . Boolean result, true if successful or false if failed.

bool OSRTString::toSize (OSSIZE &value) const

This method converts the string to a size typed (site_t) value.

Table 3.342. Parameters

value	Reference to variable to receive converted integer value.

Returns: . Boolean result, true if successful or false if failed.

bool OSRTString::toUInt (OSUINT32 &value) const

This method converts the string to an unsigned 32-bit integer value.

Table 3.343. Parameters

value	Reference to variable to receive converted integer value.

Returns: . Boolean result, true if successful or false if failed.

bool OSRTString::toUInt64 (OSUINT64 &value) const

This method converts the string to an unsigned 64-bit integer value.

Table 3.344. Parameters

value	Reference to variable to receive converted integer value.

Returns: . Boolean result, true if successful or false if failed.

EXTRTMETHOD OSRTString& OSRTString::operator= (const OSRTString &original)

Assignment operator.

OSRTStringIF class Reference

```
#include <OSRTStringIF.h>
```

- OSRTStringIF ( )
- OSRTStringIF (const char *)
- OSRTStringIF (const OSUTF8CHAR *)
- virtual ~OSRTStringIF ( )

- virtual OSRTStringIF * clone ( )
- virtual const char * getValue ( )
- virtual const OSUTF8CHAR * getUTF8Value ( )
- virtual void print (const char * name)
- virtual void setValue (const char * str)
- virtual void setValue (const OSUTF8CHAR * utf8str)

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).

Definition at line 49 of file OSRTStringIF.h

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

- OSRTStringIF.h

OSRTStringIF::OSRTStringIF ()

The default constructor creates an empty string.

OSRTStringIF::OSRTStringIF (const char *)

This constructor initializes the string to contain the given standard ASCII string value.

Table 3.345. Parameters

-	Null-terminated C string value

OSRTStringIF::OSRTStringIF (const OSUTF8CHAR *)

This constructor initializes the string to contain the given UTF-8 string value.

Table 3.346. Parameters

-	Null-terminated C string value

virtual OSRTStringIF::~OSRTStringIF ()

The destructor frees string memory using the standard 'delete' operator.

virtual OSRTStringIF* OSRTStringIF::clone ()=0

This method creates a copy of the given string object.

virtual const char* OSRTStringIF::getValue () const =0

This method returns the pointer to UTF-8 null terminated string as a standard ASCII string.

virtual const OSUTF8CHAR* OSRTStringIF::getUTF8Value () const =0

This method returns the pointer to UTF-8 null terminated string as a UTF-8 string.

virtual void OSRTStringIF::print (const char *name)=0

This method prints the string value to standard output.

Table 3.347. Parameters

name	- Name of generated string variable.

virtual void OSRTStringIF::setValue (const char *str)=0

This method sets the string value to the given string.

Table 3.348. Parameters

str	- C null-terminated string.

virtual void OSRTStringIF::setValue (const OSUTF8CHAR *utf8str)=0

This method sets the string value to the given UTF-8 string value.

Table 3.349. Parameters

utf8str	- C null-terminated UTF-8 string.

OSRTUTF8String class Reference

`#include <OSRTUTF8String.h>`

Private Attributes

- const OSUTF8CHAR * mValue

- OSRTUTF8String ( )

- OSRTUTF8String (const char * strval)
- OSRTUTF8String (const OSUTF8CHAR * strval)
- OSRTUTF8String (const OSRTUTF8String & str)
- virtual ~OSRTUTF8String ( )
- OSRTBaseType * clone ( )
- void copyValue (const char * str)
- const char * c_str ( )
- const char * getValue ( )
- void print (const char * name)
- void setValue (const char * str)
- OSRTUTF8String & operator= (const OSRTUTF8String & original)

Detailed Description

UTF-8 string. This is the base class for generated C++ data type classes for XSD string types (string, token, NMTOKEN, etc.).

Definition at line 39 of file OSRTUTF8String.h

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

- OSRTUTF8String.h

OSRTUTF8String::OSRTUTF8String ()

The default constructor creates an empty string.

OSRTUTF8String::OSRTUTF8String (const char *strval)

This constructor initializes the string to contain the given character string value.

Table 3.350. Parameters

strval	- String value

OSRTUTF8String::OSRTUTF8String (const OSUTF8CHAR *strval)

This constructor initializes the string to contain the given UTF-8 character string value.

Table 3.351. Parameters

strval	- String value

OSRTUTF8String::OSRTUTF8String (const OSRTUTF8String &str)

Copy constructor.

Table 3.352. Parameters

str	- C++ XML string class.

virtual OSRTUTF8String::~OSRTUTF8String ()

The destructor frees string memory if the memory ownership flag is set.

OSRTBaseType* OSRTUTF8String::clone () const

Clone method. Creates a copied instance and returns pointer to OSRTBaseType.

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.

Table 3.353. Parameters

str	- C null-terminated string.

const char* OSRTUTF8String::c_str () const

This method returns the pointer to C null terminated string.

const char* OSRTUTF8String::getValue () const

This method returns the pointer to UTF-8 null terminated string.

void OSRTUTF8String::print (const char *name)

This method prints the string value to standard output.

Table 3.354. Parameters

name	- Name of generated string variable.

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.

Table 3.355. Parameters

str	- C null-terminated string.

OSRTUTF8String& OSRTUTF8String::operator= (const OSRTUTF8String &original)

Assignment operator.

Chapter 4. File Documentation

ASN1CBitStr.h File Reference

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

Classes

- struct ASN1CBitStrSizeHolder
- struct ASN1CBitStrSizeHolder8
- struct ASN1CBitStrSizeHolder16
- struct ASN1CBitStrSizeHolder32
- struct ASN1CBitStrSizeHolder64
- struct ASN1CBitStr

Detailed Description

Bit string control class definitions.

Definition in file ASN1CBitStr.h

ASN1CGeneralizedTime.h File Reference

```
#include "rtsrc/ASN1CTime.h"
```

Classes

- struct ASN1CGeneralizedTime

Detailed Description

GeneralizedTime control class definition.

Definition in file ASN1CGeneralizedTime.h

ASN1Context.h File Reference

```
#include "rtxsrc/rtxDiag.h"
#include "rtxsrc/rtxError.h"
#include "rtxsrc/OSRTContext.h"
```

Classes

- struct ASN1Context

Detailed Description

Common C++ type and class definitions.

Definition in file ASN1Context.h

asn1CppEvtHndlr.h File Reference

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

Classes

- struct Asn1NamedEventHandler
- struct Asn1NullEventHandler
- struct Asn1ErrorHandler

Macros

- #define OS_UNUSED_ARG (void)arg

Variables

- class EXTRTCLASS ASN1MessageBuffer

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.

Definition in file asn1CppEvtHndlr.h

asn1CppTypes.h File Reference

```
#include <new>

#include "rtxsrc/rtxMemory.h"

#include "rtxsrc/rtxDiag.h"

#include "rtxsrc/rtxError.h"

#include "rtxsrc/rtxMemBuf.h"

#include "rtsrc/asn1CppEvtHndlr64.h"

#include "rtsrc/asn1CppRawEvtHndlr.h"

#include "rtsrc/ASN1Context.h"
```

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

#include "rtsrc/ASN1TOctStr.h"

#include "rtsrc/ASN1TOctStr64.h"

#include "rtsrc/ASN1TObjId.h"
```

Classes

- struct ASN1MessageBuffer
- struct ASN1CType
- struct ASN1TDynBitStr
- struct ASN1TDynBitStr64
- struct ASN1TBitStr32
- struct ASN1TBMPString
- struct ASN1TUniversalString
- struct ASN1TOpenType
- struct Asn1TObject
- struct ASN1TSeqExt
- struct ASN1TPDU
- struct ASN1TSeqOfList
- struct ASN1TPDUSeqOfList

Macros

- #define ASN1TRY try
- #define ASN1RTLTHROW exit (-1)
- #define ASN1THROW exit (-1)
- #define ASN1CATCH if (0) { body; }

Typedefs

- typedef Asn1TObject ASN1TObject

Detailed Description

Common C++ type and class definitions.

Definition in file asn1CppTypes.h

ASN1CSeqOfList.h File Reference

```
#include <stdlib.h>

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

Classes

- struct ASN1CSeqOfListIterator
- struct ASN1CSeqOfList

Variables

- class EXTRTCLASS ASN1CSeqOfList

Detailed Description

ASN1CSeqOfList linked list control class definition.

Definition in file ASN1CSeqOfList.h

ASN1CTime.h File Reference

```
#include <time.h>

#include "rtsrc/asn1CppTypes.h"

#include "rtsrc/ASN1TTime.h"
```

Classes

- struct ASN1CTime

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

Detailed Description

ASN1CTime abstract class definition. This is used as the base class for other ASN.1 time class definitions.

Definition in file ASN1CTime.h

ASN1CUTCTime.h File Reference

```
#include "rtsrc/ASN1CTime.h"
```

Classes

- struct ASN1CUTCTime

Detailed Description

ASN1CUTCTime control class definition.

Definition in file ASN1CUTCTime.h

asn1ErrCodes.h File Reference

Macros

- #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)
- #define ASN_E_INVPERENC (ASN_E_BASE-10)
- #define ASN_E_NOTINSEQ (ASN_E_BASE-11)
- #define ASN_E_BAD_ALIGN (ASN_E_BASE-12)
- #define ASN_E_UNKNOWNPDU (ASN_E_BASE-13)
- #define ASN_E_NOTCANON (ASN_E_BASE-14)
- #define ASN_E_VALTYPMIS (ASN_E_BASE-15)
- #define ASN_E_LEN_FORM (ASN_E_BASE-16)
- #define ASN_E_LEN_NOT_MIN (ASN_E_BASE-17)
- #define ASN_E_PRIM_REQ (ASN_E_BASE-18)
- #define ASN_E_FRAGMENTS (ASN_E_BASE-19)

Detailed Description

List of numeric status codes that can be returned by ASN1C run-time functions and generated code.

Definition in file asn1ErrCodes.h

ASN1TObjId.h File Reference

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

Classes

- struct ASN1TObjId

Functions

- int operator== (const ASN1OBJID & lhs, const ASN1OBJID & rhs)
- int operator== (const ASN1OBJID & 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 ASN1TObjId & lhs, const ASN1TObjId & rhs)
- int operator> (const ASN1TObjId & 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 ASN1TObjId & lhs, const ASN1TObjId & rhs)
- int operator>= (const ASN1TObjId & lhs, const char * dotted_oid_string)
- int operator>= (const ASN1OBJID & lhs, const ASN1OBJID & rhs)
- int operator>= (const ASN1OBJID & lhs, const char * dotted_oid_string)

- ASN1TObjId operator+ (const ASN1TObjId & lhs, const ASN1TObjId & rhs)

Detailed Description

ASN.1 object identifier class definition.

Definition in file ASN1TObjId.h

ASN1TOctStr.h File Reference

```
#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)

Detailed Description

ASN.1 OCTET string class definition.

Definition in file ASN1TOctStr.h

ASN1TTime.h File Reference

```
#include <time.h>
```

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

Classes

- struct ASN1TTime
- struct ASN1TGeneralizedTime
- struct ASN1TUTCTime

Macros

- #define MAX_TIMESTR_SIZE 64
- #define LOG_TTMERR (mStatus = stat, stat)

Detailed Description

Definition in file ASN1TTime.h

OSRTBaseType.h File Reference

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

Classes

- struct OSRTBaseType

Detailed Description

C++ run-time base class for structured type definitions.

Definition in file OSRTBaseType.h

OSRTContext.h File Reference

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

#include "rtxsrc/rtxDiag.h"

#include "rtxsrc/rtxError.h"

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

Classes

- struct OSRTContext
- struct OSRTCtxtPtr
- struct OSRTElemNameGuard

Functions

- void * operator new (size_t nbytes, OSCTXT * pctxt)
- void operator delete (void * pmem, OSCTXT * pctxt)

Detailed Description

C++ run-time context class definition.

Definition in file OSRTContext.h

OSRTFastString.h File Reference

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

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

Classes

- struct OSRTFastString

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.

Definition in file OSRTFastString.h

OSRTFileInputStream.h File Reference

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

Classes

- struct OSRTFileInputStream

Detailed Description

C++ base class definitions for operations with input file streams.

Definition in file OSRTFileInputStream.h

OSRTFileOutputStream.h File Reference

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

Classes

- struct OSRTFileOutputStream

Detailed Description

C++ base class definitions for operations with output file streams.

Definition in file OSRTFileOutputStream.h

OSRTHexTextInputStream.h File Reference

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

Classes

- struct OSRTHexTextInputStream

Detailed Description

C++ hexadecimal text input stream filter class.

Definition in file OSRTHexTextInputStream.h

OSRTInputStream.h File Reference

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

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

Classes

- struct OSRTInputStream

Detailed Description

C++ base class definitions for operations with input streams.

Definition in file OSRTInputStream.h

OSRTInputStreamIF.h File Reference

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

Classes

- struct OSRTInputStreamIF
- struct OSRTInputStreamPtr

Detailed Description

C++ interface class definitions for operations with input streams.

Definition in file OSRTInputStreamIF.h

OSRTMemoryInputStream.h File Reference

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

Classes

- struct OSRTMemoryInputStream

Detailed Description

C++ base class definitions for operations with input memory streams.

Definition in file OSRTMemoryInputStream.h

OSRTMemoryOutputStream.h File Reference

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

Classes

- struct OSRTMemoryOutputStream

Detailed Description

C++ base class definitions for operations with output memory streams.

Definition in file OSRTMemoryOutputStream.h

OSRTMsgBuf.h File Reference

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

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

Classes

- struct OSRTMessageBuffer

Detailed Description

C++ run-time message buffer class definition.

Definition in file OSRTMsgBuf.h

OSRTMsgBuflF.h File Reference

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

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

Classes

- struct OSRTMessageBufferIF

Detailed Description

C++ run-time message buffer interface class definition.

Definition in file OSRTMsgBufIF.h

OSRTOutputStream.h File Reference

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

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

Classes

- struct OSRTOutputStream

Detailed Description

C++ base class definitions for operations with output streams.

Definition in file OSRTOutputStream.h

OSRTOutputStreamIF.h File Reference

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

Classes

- struct OSRTOutputStreamIF

- struct OSRTOutputStreamPtr

Detailed Description

C++ interface class definitions for operations with output streams.

Definition in file OSRTOutputStreamIF.h

OSRTSocket.h File Reference

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

Classes

- struct OSRTSocket

Detailed Description

TCP/IP or UDP socket class definitions.

Definition in file OSRTSocket.h

OSRTSocketInputStream.h File Reference

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

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

Classes

- struct OSRTSocketInputStream

Detailed Description

C++ base class definitions for operations with input socket streams.

Definition in file OSRTSocketInputStream.h

OSRTSocketOutputStream.h File Reference

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

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

Classes

- struct OSRTSocketOutputStream

Detailed Description

C++ base class definitions for operations with output socket streams.

Definition in file OSRTSocketOutputStream.h

OSRTStream.h File Reference

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

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

Classes

- struct OSRTStream

Detailed Description

C++ base class definitions for operations with I/O streams.

Definition in file OSRTStream.h

OSRTStreamIF.h File Reference

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

Classes

- struct OSRTStreamIF

Detailed Description

C++ interface class definitions for operations with I/O streams.

Definition in file OSRTStreamIF.h

OSRTString.h File Reference

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

#include "rtxsrc/rtxPrint.h"

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

Classes

- struct OSRTString

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.

Definition in file OSRTString.h

OSRTStringIF.h File Reference

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

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

Classes

- struct OSRTStringIF

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.

Definition in file OSRTStringIF.h

OSRTUTF8String.h File Reference

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

#include "rtxsrc/rtxPrint.h"

#include "rtxsrc/rtxUTF8.h"
```

Classes

- struct OSRTUTF8String

Detailed Description

C++ UTF-8 string class definition.

Definition in file OSRTUTF8String.h