XBinder C++ Common Runtime Classes

Version 3.0

Objective Systems, Inc.

December 2024

XBinder C++ Common Runtime Classes

Author's Contact Information. Comments, suggestions, and inquiries regarding XBinder 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

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 OSRTMessageBuffer
- struct OSRTMessageBufferIF

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

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

Chapter 3. Class Documentation

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 3.1. 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 3.2. 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 3.3. 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 3.4. 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 3.5. 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 3.6. 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 3.7. 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 3.8. 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 3.9. 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 3.10. 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 3.11. 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 OSRTC-txtPtr &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 3.12. 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.

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 3.13. 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 3.14. 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 3.15. 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 3.16. 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 3.17. 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 3.18. 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 3.19. 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 3.20. 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 3.21. 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 3.22. 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 3.23. 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 3.24. Parameters

pFilename	Name of file.

Table 3.25. 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 3.26. Parameters

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

Table 3.27. 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 3.28. Parameters

file	Pointer to FILE structure.

Table 3.29. 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 3.30. Parameters

pContext	Pointer to a context to use.
file	Pointer to FILE structure.

Table 3.31. 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 3.32. 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 3.33. 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 3.34. 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 3.35. 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 3.36. 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 3.37. 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 3.38. 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 3.39. 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 3.40. 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 3.41. 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 3.42. 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 3.43. 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 3.44. 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 3.45. 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 3.46. 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 3.47. 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 3.48. 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 3.49. 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 3.50. 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 3.51. 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 3.52. 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 3.53. 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 3.54. 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 3.55. 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 3.56. 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 3.57. 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 3.58. 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 3.59. 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 3.60. 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 3.61. 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 3.62. 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 3.63. 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 3.64. 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 3.65. 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 3.66. 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 3.67. 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 3.68. 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 3.69. 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 3.70. 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 3.71. 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 3.72. 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 3.73. 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 3.74. 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 3.75. 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 3.76. 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 3.77. 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 3.78. 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 3.79. 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 3.80. 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 3.81. 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 3.82. 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 3.83. 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 3.84. 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 3.85. 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 3.86. 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 3.87. 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 3.88. 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 3.89. 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 3.90. 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 3.91. 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 3.92. 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 3.93. 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 3.94. 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 3.95. 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 3.96. 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 3.97. 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 3.98. 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 3.99. 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 3.100. Parameters

strval	- Null-terminated C string value

OSRTString::OSRTString (const OSRTString &str)

Copy constructor.

Table 3.101. 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 3.102. 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 3.103. 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 3.104. 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 3.105. 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 3.106. 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 3.107. 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 3.108. 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 3.109. 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 3.110. Parameters

-	Null-terminated C string value

OSRTStringIF::OSRTStringIF (const OSUTF8CHAR *)

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

Table 3.111. 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 3.112. 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 3.113. 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 3.114. 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 3.115. Parameters

strval	- String value

OSRTUTF8String::OSRTUTF8String (const OSUTF8CHAR *strval)

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

Table 3.116. Parameters

strval	- String value

OSRTUTF8String::OSRTUTF8String (const OSRTUTF8String &str)

Copy constructor.

Table 3.117. 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 3.118. 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 3.119. 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 3.120. Parameters

str	- C null-terminated string.

OSRTUTF8String& OSRTUTF8String::operator= (const OSRTUTF8String &original)

Assignment operator.

Chapter 4. File Documentation

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

- EXTERNRT void * operator new (size_t nbytes, OSCTXT * pctxt)
- EXTERNRT 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

OSRTMsgBufIF.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