XBinder XML Runtime

Version 3.0

Objective Systems, Inc.

December 2024

XBinder XML Runtime

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 XML Runtime Library Functions

The **C run-time XML library** contains functions used to encode/decode XML data. These functions are identified by their *rtXml* prefixes.

The categories of functions provided are as follows:

- XML pull-parser code.
- Functions functions to encode C types to XML.
- Functions to decode XML to C data types.
- Functions to encode XML element tags.
- Functions to encode XML attributes in sorted order for C14N.
- SAX parser interfaces.
- Context management functions.

Chapter 2. Module Documentation

XML decode functions.

Detailed Description

Functions

- EXTERNXML int rtXmlDecBase64Binary (OSRTMEMBUF * pMemBuf, const OSUTF8CHAR * inpdata, OSSIZE length)

  This function decodes the contents of a Base64-encoded binary data type into a memory buffer.

- EXTERNXML int rtXmlDecBase64Str (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocts, OSINT32 bufsize)

  This function decodes a contents of a Base64-encode binary string into a static memory structure.

- EXTERNXML int rtXmlDecBase64Str64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnocts, OSSIZE bufsize)

  This function is identical to rtXmlDecBase64Str except that is supports a 64-bit integer length on 64-bit systems.

- EXTERNXML int rtXmlDecBase64StrValue (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocts, OSSIZE bufSize, OSSIZE srcDataLen)

  This function decodes a contents of a Base64-encode binary string into the specified octet array.

- EXTERNXML int rtXmlDecBase64StrValue64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnocts, OSSIZE bufSize, OSSIZE srcDataLen)

  This function decodes is identical to rtXmlDecBase64StrValue except that it supports a 64-bit integer length on 64-bit systems.

- EXTERNXML int rtXmlDecBigInt (OSCTXT * pctxt, const OSUTF8CHAR ** ppvalue)

  This function will decode a variable of the XSD integer type.

- EXTERNXML int rtXmlDecBool (OSCTXT * pctxt, OSBOOL * pvalue)

  This function decodes a variable of the boolean type.

- EXTERNXML int rtXmlDecDate (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'date' type.

- EXTERNXML int rtXmlDecTime (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'time' type.

- EXTERNXML int rtXmlDecDateTime (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'dateTime' type.

- EXTERNXML int rtXmlDecDecimal (OSCTXT * pctxt, OSREAL * pvalue)

  This function decodes the contents of a decimal data type.

- EXTERNXML int rtXmlDecDouble (OSCTXT * pctxt, OSREAL * pvalue)

  This function decodes the contents of a float or double data type.

- EXTERNXML int rtXmlDecDynBase64Str (OSCTXT * pctxt, OSDynOctStr * pvalue)

  This function decodes a contents of a Base64-encode binary string.

- EXTERNXML int rtXmlDecDynBase64Str64 (OSCTXT * pctxt, OSDynOctStr64 * pvalue)

  This function is identical to rtXmlDecDynBase64Str except that it supports a 64-bit integer length on 64-bit systems.

- EXTERNXML int rtXmlDecDynHexStr (OSCTXT * pctxt, OSDynOctStr * pvalue)

  This function decodes a contents of a hexBinary string.

- EXTERNXML int rtXmlDecDynHexStr64 (OSCTXT * pctxt, OSDynOctStr64 * pvalue)

  This function is identical to rtXmlDecDynHexStr except that it supports a 64-bit integer length on 64-bit systems.

- EXTERNXML int rtXmlDecEmptyElement (OSCTXT * pctxt)

  This function is used to enforce a requirement that an element be empty.

- EXTERNXML int rtXmlDecUTF8Str (OSCTXT * pctxt, OSUTF8CHAR * outdata, OSSIZE max_len)

  This function decodes the contents of a UTF-8 string data type.

- EXTERNXML int rtXmlDecDynUTF8Str (OSCTXT * pctxt, const OSUTF8CHAR ** outdata)

  This function decodes the contents of a UTF-8 string data type.

- EXTERNXML int rtXmlDecHexBinary (OSRTMEMBUF * pMemBuf, const OSUTF8CHAR * inpdata, OSSIZE length)

  This function decodes the contents of a hex-encoded binary data type into a memory buffer.

- EXTERNXML int rtXmlDecHexStr (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocts, OSINT32 bufsize)

  This function decodes the contents of a hexBinary string into a static memory structure.

- EXTERNXML int rtXmlDecHexStr64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnocts, OSSIZE bufsize)

  This function is identical to rtXmlDecHexStr except that it supports a 64-bit integer length on 64-bit systems.

- EXTERNXML int rtXmlDecHexStrValue (OSCTXT * pctxt, const OSUTF8CHAR *const inpdata, OSSIZE nbytes, OSOCTET * pvalue, OSUINT32 * pnbits, OSINT32 bufsize)

- EXTERNXML int rtXmlDecHexStrValue64 (OSCTXT * pctxt, const OSUTF8CHAR *const inpdata, OSSIZE nbytes, OSOCTET * pvalue, OSSIZE * pnbits, OSSIZE bufsize)

- EXTERNXML int rtXmlDecGYear (OSCTXT * pctxt, OSXSDDateTime * pvalue)

This function decodes a variable of the XSD 'gYear' type.

- EXTERNXML int rtXmlDecGYearMonth (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gYearMonth' type.

- EXTERNXML int rtXmlDecGMonth (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gMonth' type.

- EXTERNXML int rtXmlDecGMonthDay (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gMonthDay' type.

- EXTERNXML int rtXmlDecGDay (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gDay' type.

- EXTERNXML int rtXmlDecInt (OSCTXT * pctxt, OSINT32 * pvalue)

  This function decodes the contents of a 32-bit integer data type.

- EXTERNXML int rtXmlDecInt8 (OSCTXT * pctxt, OSINT8 * pvalue)

  This function decodes the contents of an 8-bit integer data type (i.e.

- EXTERNXML int rtXmlDecInt16 (OSCTXT * pctxt, OSINT16 * pvalue)

  This function decodes the contents of a 16-bit integer data type.

- EXTERNXML int rtXmlDecInt64 (OSCTXT * pctxt, OSINT64 * pvalue)

  This function decodes the contents of a 64-bit integer data type.

- EXTERNXML int rtXmlDecUInt (OSCTXT * pctxt, OSUINT32 * pvalue)

  This function decodes the contents of an unsigned 32-bit integer data type.

- EXTERNXML int rtXmlDecUInt8 (OSCTXT * pctxt, OSUINT8 * pvalue)

  This function decodes the contents of an unsigned 8-bit integer data type (i.e.

- EXTERNXML int rtXmlDecUInt16 (OSCTXT * pctxt, OSUINT16 * pvalue)

  This function decodes the contents of an unsigned 16-bit integer data type.

- EXTERNXML int rtXmlDecUInt64 (OSCTXT * pctxt, OSUINT64 * pvalue)

  This function decodes the contents of an unsigned 64-bit integer data type.

- EXTERNXML int rtXmlDecNSAttr (OSCTXT * pctxt, const OSUTF8CHAR * attrName, const OSUTF8CHAR * attrValue, OSRTDList * pNSAttrs, const OSUTF8CHAR * nsTable, OSUINT32 nsTableRowCount)

  This function decodes an XML namespac attribute (xmlns).

- EXTERNXML const OSUTF8CHAR * rtXmlDecQName (OSCTXT * pctxt, const OSUTF8CHAR * qname, const OSUTF8CHAR ** prefix)

This function decodes an XML qualified name string (QName) type.

- EXTERNXML int rtXmlDecXSIAttr (OSCTXT * pctxt, const OSUTF8CHAR * attrName, const OSUTF8CHAR * attrValue)

  This function decodes XML schema instance (XSI) attribute.

- EXTERNXML int rtXmlDecXSIAttrs (OSCTXT * pctxt, const OSUTF8CHAR *const * attrs, const char * typeName)

  This function decodes XML schema instance (XSI) attributes.

- EXTERNXML int rtXmlDecXmlStr (OSCTXT * pctxt, OSXMLSTRING * outdata)

  This function decodes the contents of an XML string data type.

- EXTERNXML int rtXmlParseElementName (OSCTXT * pctxt, OSUTF8CHAR ** ppName)

  This function parses the initial tag from an XML message.

- EXTERNXML int rtXmlParseElemQName (OSCTXT * pctxt, OSXMLQName * pQName)

  This function parses the initial tag from an XML message.

Function Documentation

EXTERNXML int rtXmlDecBase64Binary (OSRTMEMBUF *pMemBuf, const OSUTF8CHAR *inpdata, OSSIZE length)

This function decodes the contents of a Base64-encoded binary data type into a memory buffer.

Input is expected to be a string of UTF-8 characters returned by an XML parser. The decoded data will be put into the memory buffer starting from the current position and bit offset. After all data is decoded the octet string may be fetched out.

This function is normally used in the 'characters' SAX handler.

Table 2.1. Parameters

pMemBuf	Memory buffer to which decoded binary data is to be appended.
inpdata	Pointer to a source string to be decoded.
length	Length of the source string (in characters).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecBase64Str (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSINT32 bufsize)

This function decodes a contents of a Base64-encode binary string into a static memory structure.

The octet string must be Base64 encoded. This function call is used to decode a sized base64Binary string production.

Table 2.2. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.
pnocts	A pointer to an integer value to receive the decoded number of octets.
bufsize	The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecBase64Str64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnocts, OSSIZE bufsize)

This function is identical to rtXmlDecBase64Str except that is supports a 64-bit integer length on 64-bit systems.

Table 2.3. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.
pnocts	A pointer to an integer value to receive the decoded number of octets.
bufsize	The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecBase64StrValue (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSSIZE bufSize, OSSIZE srcDataLen)

This function decodes a contents of a Base64-encode binary string into the specified octet array.

The octet string must be Base64 encoded. This function call is used internally to decode both sized and non-sized base64binary string production.

Table 2.4. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.
pnocts	A pointer to an integer value to receive the decoded number of octets.
bufSize	A maximum size (in octets) of `pvalue` buffer. An error will occur if the number of octets in the decoded string is larger than this value.
srcDataLen	An actual source data length (in octets) without whitespaces.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecBase64StrValue64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnocts, OSSIZE bufSize, OSSIZE srcDataLen)

This function decodes is identical to rtXmlDecBase64StrValue except that it supports a 64-bit integer length on 64-bit systems.

Table 2.5. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.
pnocts	A pointer to an integer value to receive the decoded number of octets.
bufSize	A maximum size (in octets) of `pvalue` buffer. An error will occur if the number of octets in the decoded string is larger than this value.
srcDataLen	An actual source data length (in octets) without whitespaces.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecBigInt (OSCTXT *pctxt, const OSUTF8CHAR **ppvalue)

This function will decode a variable of the XSD integer type.

In this case, the integer is assumed to be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits). For example, parameters used to calculate security values are typically larger than these sizes.

These variables are stored in character string constant variables. The radix should be 10. If it is necessary to convert to another radix, then use rtxBigIntSetStr or rtxBigIntToString functions.

Table 2.6. Parameters

pctxt	Pointer to context block structure.
ppvalue	Pointer to a pointer to receive decoded UTF-8 string. Dynamic memory is allocated for the variable using the rtMemAlloc function. The decoded variable is represented as a string starting with appropriate prefix.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecBool (OSCTXT *pctxt, OSBOOL *pvalue)

This function decodes a variable of the boolean type.

Table 2.7. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to a variable to receive the decoded boolean value.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecDate (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'date' type.

Input is expected to be a string of characters returned by an XML parser. The string should have CCYY-MM-DD format.

Table 2.8. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecTime (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'time' type.

Input is expected to be a string of characters returned by an XML parser. The string should have one of following formats:

(1) hh-mm-ss.ss used if tz_flag = false
(2) hh-mm-ss.ssZ used if tz_flag = false and tzo = 0
(3) hh-mm-ss.ss+HH:MM if tz_flag = false and tzo > 0
(4) hh-mm-ss.ss-HH:MM-HH:MM
if tz_flag = false and tzo < 0

Table 2.9. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecDateTime (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'dateTime' type.

Input is expected to be a string of characters returned by an XML parser.

Table 2.10. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecDecimal (OSCTXT *pctxt, OSREAL *pvalue)

This function decodes the contents of a decimal data type.

Input is expected to be a string of characters returned by an XML parser.

Table 2.11. Parameters

pctxt	Pointer to context block structure.

pvalue	Pointer to 64-bit double value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecDouble (OSCTXT *pctxt, OSREAL *pvalue)

This function decodes the contents of a float or double data type.

Input is expected to be a string of characters returned by an XML parser.

Table 2.12. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to 64-bit double value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecDynBase64Str (OSCTXT *pctxt, OSDynOctStr *pvalue)

This function decodes a contents of a Base64-encode binary string.

The octet string must be Base64 encoded. This function will allocate dynamic memory to store the decoded result.

Table 2.13. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the ::rtxMemAlloc function.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecDynBase64Str64 (OSCTXT *pctxt, OSDynOctStr64 *pvalue)

This function is identical to rtXmlDecDynBase64Str except that it supports a 64-bit integer length on 64-bit systems.

Table 2.14. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the ::rtxMemAlloc function.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecDynHexStr (OSCTXT *pctxt, OSDynOctStr *pvalue)

This function decodes a contents of a hexBinary string.

This function will allocate dynamic memory to store the decoded result.

Table 2.15. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the ::rtxMemAlloc function.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecDynHexStr64 (OSCTXT *pctxt, OSDynOctStr64 *pvalue)

This function is identical to rtXmlDecDynHexStr except that it supports a 64-bit integer length on 64-bit systems.

Table 2.16. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the ::rtxMemAlloc function.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecEmptyElement (OSCTXT *pctxt)

This function is used to enforce a requirement that an element be empty.

An error is returned in the current element has any element or character children. The last event must be the start tag.

Table 2.17. Parameters

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

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecUTF8Str (OSCTXT *pctxt, OSUTF8CHAR *outdata, OSSIZE max_len)

This function decodes the contents of a UTF-8 string data type.

Input is expected to be a string of UTF-8 or Unicode characters returned by an XML parser.

Table 2.18. Parameters

pctxt	Pointer to context block structure.
outdata	Pointer to a block of memory to receive decoded UTF8 string.
max_len	Size of memory block.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecDynUTF8Str (OSCTXT *pctxt, const OSUTF8CHAR **outdata)

This function decodes the contents of a UTF-8 string data type.

Input is expected to be a string of UTF-8 or Unicode characters returned by an XML parser.

Table 2.19. Parameters

pctxt	Pointer to context block structure.
outdata	Pointer to a pointer to receive decoded UTF-8 string. Memory is allocated for this string using the run-time memory manager.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecHexBinary (OSRTMEMBUF *pMemBuf, const OSUTF8CHAR *inpdata, OSSIZE length)

This function decodes the contents of a hex-encoded binary data type into a memory buffer.

Input is expected to be a string of UTF-8 characters returned by an XML parser. The decoded data will be put into the given memory buffer starting from the current position and bit offset. After all data is decoded the octet string may be fetched out.

This function is normally used in the 'characters' SAX handler.

Table 2.20. Parameters

pMemBuf	Pointer to memory buffer onto which the decoded binary data will be appended.
inpdata	Pointer to a source string to be decoded.
length	Length of the source string (in characters).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecHexStr (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSINT32 bufsize)

This function decodes the contents of a hexBinary string into a static memory structure.

Table 2.21. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.
pnocts	A pointer to an integer value to receive the decoded number of octets.
bufsize	The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecHexStr64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnocts, OSSIZE bufsize)

This function is identical to rtXmlDecHexStr except that it supports a 64-bit integer length on 64-bit systems.

Table 2.22. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.
pnocts	A pointer to an integer value to receive the decoded number of octets.
bufsize	The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecHexStrValue (OSCTXT *pctxt, const OSUTF8CHAR *const inpdata, OSSIZE nbytes, OSOCTET *pvalue, OSUINT32 *pnbits, OSINT32 bufsize)

EXTERNXML int rtXmlDecHexStrValue64 (OSCTXT *pctxt, const OSUTF8CHAR *const inpdata, OSSIZE nbytes, OSOCTET *pvalue, OSSIZE *pnbits, OSSIZE bufsize)

EXTERNXML int rtXmlDecGYear (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'gYear' type.

Input is expected to be a string of characters returned by an XML parser. The string should have CCYY[-+hh:mm|Z] format.

Table 2.23. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecGYearMonth (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'gYearMonth' type.

Input is expected to be a string of characters returned by an XML parser. The string should have CCYY-MM[-+hh:mm|Z] format.

Table 2.24. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecGMonth (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'gMonth' type.

Input is expected to be a string of characters returned by an XML parser. The string should have –MM[-+hh:mm|Z] format.

Table 2.25. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecGMonthDay (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'gMonthDay' type.

Input is expected to be a string of characters returned by an XML parser. The string should have –MM-DD[-+hh:mm|Z] format.

Table 2.26. Parameters

pctxt	Pointer to context block structure.

pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecGDay (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'gDay' type.

Input is expected to be a string of characters returned by an XML parser. The string should have —DD[-+hh:mm|Z] format.

Table 2.27. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecInt (OSCTXT *pctxt, OSINT32 *pvalue)

This function decodes the contents of a 32-bit integer data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML parser.

Table 2.28. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to 32-bit integer value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecInt8 (OSCTXT *pctxt, OSINT8 *pvalue)

This function decodes the contents of an 8-bit integer data type (i.e.

a signed byte type in the range of -128 to 127). Input is expected to be a string of OSUTF8CHAR characters returned by an XML parser.

Table 2.29. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to 8-bit integer value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecInt16 (OSCTXT *pctxt, OSINT16 *pvalue)

This function decodes the contents of a 16-bit integer data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML parser.

Table 2.30. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to 16-bit integer value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecInt64 (OSCTXT *pctxt, OSINT64 *pvalue)

This function decodes the contents of a 64-bit integer data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML parser.

Table 2.31. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to 64-bit integer value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecUInt (OSCTXT *pctxt, OSUINT32 *pvalue)

This function decodes the contents of an unsigned 32-bit integer data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML parser.

Table 2.32. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to unsigned 32-bit integer value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecUInt8 (OSCTXT *pctxt, OSUINT8 *pvalue)

This function decodes the contents of an unsigned 8-bit integer data type (i.e.

a signed byte type in the range of 0 to 255). Input is expected to be a string of OSUTF8CHAR characters returned by an XML parser.

Table 2.33. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to unsigned 8-bit integer value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecUInt16 (OSCTXT *pctxt, OSUINT16 *pvalue)

This function decodes the contents of an unsigned 16-bit integer data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML parser.

Table 2.34. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to unsigned 16-bit integer value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecUInt64 (OSCTXT *pctxt, OSUINT64 *pvalue)

This function decodes the contents of an unsigned 64-bit integer data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML parser.

Table 2.35. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to unsigned 64-bit integer value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecNSAttr (OSCTXT *pctxt, const OSUTF8CHAR *attrName, const OSUTF8CHAR *attrValue, OSRTDList *pNSAttrs, const OSUTF8CHAR *nsTable[], OSUINT32 nsTableRowCount)

This function decodes an XML namespac attribute (xmlns).

It is assumed that the given attribute name is either 'xmlns' or 'xmlns:prefix'. The XML namespace prefix and URI are added to the given attribute list structure. The parsed namespace is also added to the namespace stack in the given context.

Table 2.36. Parameters

pctxt	Pointer to context structure.
attrName	Name of the XML namespace attribute. This is assumed to contain either 'xmlns' (a default namespace declaration) or 'xmlns:prefix' where prefix is a namespace prefix.
attrValue	XML namespace attribute value (a URI).
pNSAttrs	List to receive parsed namespace values.
nsTable	Namespace URI's parsed from schema.
nsTableRowCount	Number of rows (URI's) in namespace table.

Returns: . Zero if success or negative error code.

EXTERNXML const OSUTF8CHAR* rtXmlDecQName (OSCTXT *pctxt, const OSUTF8CHAR *qname, const OSUTF8CHAR **prefix)

This function decodes an XML qualified name string (QName) type.

This is a type that contains an optional namespace prefix followed by a colon and the local part of the name:

[NS 5] QName ::= (Prefix ':')? LocalPart

[NS 6] Prefix ::= NCName

[NS 7] LocalPart ::= NCName

Table 2.37. Parameters

pctxt	Pointer to context block structure.
qname	String containing XML QName to be decoded.
prefix	Pointer to string pointer to receive decoded prefix. This is optional. If NULL, the prefix will not be returned. If not-NULL, the prefix will be returned in memory allocated using `rtxMemAlloc` which must be freed using one of the `rtxMemFree` functions.

Returns: . Pointer to local part of string. This is pointer to a location within the given string (i.e. a pointer to the character after the ':' or to the beginning of the string if it contains no colon). This pointer does not need to be freed.

EXTERNXML int rtXmlDecXSIAttr (OSCTXT *pctxt, const OSUTF8CHAR *attrName, const OSUTF8CHAR *attrValue)

This function decodes XML schema instance (XSI) attribute.

These attributes include the XSI namespace declaration, the XSD schema location attribute, and the XSD no namespace schema location attribute.

Table 2.38. Parameters

pctxt	Pointer to context block structure.
attrName	Attribute's name to be decoded
attrValue	Attribute's value to be decoded

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecXSIAttrs (OSCTXT *pctxt, const OSUTF8CHAR *const *attrs, const char *typeName)

This function decodes XML schema instance (XSI) attributes.

These attributes include the XSI namespace declaration, the XSD schema location attribute, and the XSD no namespace schema location attribute.

Table 2.39. Parameters

pctxt	Pointer to context block structure.
attrs	Attributes-values array [attr, value]. Should be null-terminated.
typeName	Name of parent type to add in error log, if would be necessary.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlDecXmlStr (OSCTXT *pctxt, OSXMLSTRING *outdata)

This function decodes the contents of an XML string data type.

This type contains a pointer to a UTF-8 characer string plus flags that can be set to alter the encoding of the string (for example, the cdata flag allows the string to be encoded in a CDATA section). Input is expected to be a string of UTF-8 characters returned by an XML parser.

Table 2.40. Parameters

pctxt	Pointer to context block structure.
outdata	Pointer to an XML string structure to receive the decoded string. Memory is allocated for the string using the run-time memory manager.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlParseElementName (OSCTXT *pctxt, OSUTF8CHAR **ppName)

This function parses the initial tag from an XML message.

If the tag is a QName, only the local part of the name is returned.

Table 2.41. Parameters

pctxt	Pointer to OSCTXT structure
ppName	Pointer to a pointer to receive decoded UTF-8 string. Dynamic memory is allocated for the variable using the rtxMemAlloc function.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlParseElemQName (OSCTXT *pctxt, OSXMLQName *pQName)

This function parses the initial tag from an XML message.

Table 2.42. Parameters

pctxt	Pointer to OSCTXT structure
pQName	Pointer to a QName structure to receive parsed name prefix and local name. Dynamic memory is allocated for both name parts using the rtxMemAlloc function.

Returns: . Completion status of operation:

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

XML encode functions.

Detailed Description

Functions

- EXTERNXML int rtXmlEncAny (OSCTXT * pctxt, OSXMLSTRING * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD any type.

- EXTERNXML int rtXmlEncAnyStr (OSCTXT * pctxt, const OSUTF8CHAR * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

- EXTERNXML int rtXmlEncAnyTypeValue (OSCTXT * pctxt, const OSUTF8CHAR * pvalue)

  This function encodes a variable of the XSD anyType type.

- EXTERNXML int rtXmlEncAnyAttr (OSCTXT * pctxt, OSRTDList * pAnyAttrList)

  This function encodes a list of OSAnyAttr attributes in which the name and value are given as a UTF-8 string.

- EXTERNXML int rtXmlEncBase64Binary (OSCTXT * pctxt, OSSIZE nocts, const OSOCTET * value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD base64Binary type.

- EXTERNXML int rtXmlEncBase64BinaryAttr (OSCTXT * pctxt, OSUINT32 nocts, const OSOCTET * value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen)

  This function encodes a variable of the XSD base64Binary type as an attribute.

- EXTERNXML int rtXmlEncBase64StrValue (OSCTXT * pctxt, OSSIZE nocts, const OSOCTET * value)

  This function encodes a variable of the XSD base64Binary type.

- EXTERNXML int rtXmlEncBigInt (OSCTXT * pctxt, const OSUTF8CHAR * value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD integer type.

- EXTERNXML int rtXmlEncBigIntAttr (OSCTXT * pctxt, const OSUTF8CHAR * value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen)

This function encodes an XSD integer attribute value.

- EXTERNXML int rtXmlEncBigIntValue (OSCTXT * pctxt, const OSUTF8CHAR * value)

  This function encodes an XSD integer attribute value.

- EXTERNXML int rtXmlEncBitString (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

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

- EXTERNXML int rtXmlEncBitStringExt (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * value, OSSIZE dataSize, const OSOCTET * extValue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

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

- EXTERNXML int rtXmlEncBinStrValue (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data)

  This function encodes a binary string value as a sequence of '1's and '0's.

- EXTERNXML int rtXmlEncBool (OSCTXT * pctxt, OSBOOL value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD boolean type.

- EXTERNXML int rtXmlEncBoolValue (OSCTXT * pctxt, OSBOOL value)

  This function encodes a variable of the XSD boolean type.

- EXTERNXML int rtXmlEncBoolAttr (OSCTXT * pctxt, OSBOOL value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen)

  This function encodes an XSD boolean attribute value.

- EXTERNXML int rtXmlEncCanonicalSort (OSCTXT * pctxt, OSCTXT * pBufCtxt, OSRTSList * pList)

  Sort (as for CXER, Canonical-XER) and encode previously encoded SET OF components.

- EXTERNXML int rtXmlEncComment (OSCTXT * pctxt, const OSUTF8CHAR * comment)

  This function encodes an XML comment.

- EXTERNXML int rtXmlEncDate (OSCTXT * pctxt, const OSXSDDateTime * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD 'date' type as a string.

- EXTERNXML int rtXmlEncDateValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)

  This function encodes a variable of the XSD 'date' type as a string.

- EXTERNXML int rtXmlEncTime (OSCTXT * pctxt, const OSXSDDateTime * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD 'time' type as an string.

- EXTERNXML int rtXmlEncTimeValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)

This function encodes a variable of the XSD 'time' type as an string.

- EXTERNXML int rtXmlEncDateTime (OSCTXT * pctxt, const OSXSDDateTime * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a numeric date/time value into an XML string representation.

- EXTERNXML int rtXmlEncDateTimeValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)

  This function encodes a numeric date/time value into an XML string representation.

- EXTERNXML int rtXmlEncDecimal (OSCTXT * pctxt, OSREAL value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, const OSDecimalFmt * pFmtSpec)

  This function encodes a variable of the XSD decimal type.

- EXTERNXML int rtXmlEncDecimalAttr (OSCTXT * pctxt, OSREAL value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen, const OSDecimalFmt * pFmtSpec)

  This function encodes a variable of the XSD decimal type as an attribute.

- EXTERNXML int rtXmlEncDecimalValue (OSCTXT * pctxt, OSREAL value, const OSDecimalFmt * pFmtSpec, char * pDestBuf, OSSIZE destBufSize)

  This function encodes a value of the XSD decimal type.

- EXTERNXML int rtXmlEncDouble (OSCTXT * pctxt, OSREAL value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, const OSDoubleFmt * pFmtSpec)

  This function encodes a variable of the XSD double type.

- EXTERNXML int rtXmlEncDoubleAttr (OSCTXT * pctxt, OSREAL value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen, const OSDoubleFmt * pFmtSpec)

  This function encodes a variable of the XSD double type as an attribute.

- EXTERNXML int rtXmlEncDoubleNormalValue (OSCTXT * pctxt, OSREAL value, const OSDoubleFmt * pFmtSpec, int defaultPrecision)

  This function encodes a normal (not +/-INF or NaN) value of the XSD double or float type.

- EXTERNXML int rtXmlEncDoubleValue (OSCTXT * pctxt, OSREAL value, const OSDoubleFmt * pFmtSpec, int defaultPrecision)

  This function encodes a value of the XSD double or float type.

- EXTERNXML int rtXmlEncEmptyElement (OSCTXT * pctxt, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, OSRTDList * pNSAttrs, OSBOOL terminate)

  This function encodes an enpty element tag value (\<elemName/\>).

- EXTERNXML int rtXmlEncEndDocument (OSCTXT * pctxt)

  This function adds trailor information and a null terminator at the end of the XML document being encoded.

- EXTERNXML int rtXmlEncEndElement (OSCTXT * pctxt, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

This function encodes an end element tag value (\</elemName\>).

- EXTERNXML int rtXmlEncEndSoapEnv (OSCTXT * pctxt)

  This function encodes a SOAP envelope end element tag (\<SOAP-ENV:Envelope/\>).

- EXTERNXML int rtXmlEncEndSoapElems (OSCTXT * pctxt, OSXMLSOAPMsgType msgtype)

  This function encodes SOAP end element tags.

- EXTERNXML int rtXmlEncFloat (OSCTXT * pctxt, OSREAL value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, const OSDoubleFmt * pFmtSpec)

  This function encodes a variable of the XSD float type.

- EXTERNXML int rtXmlEncFloatAttr (OSCTXT * pctxt, OSREAL value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen, const OSDoubleFmt * pFmtSpec)

  This function encodes a variable of the XSD float type as an attribute.

- EXTERNXML int rtXmlEncGYear (OSCTXT * pctxt, const OSXSDDateTime * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a numeric gYear element into an XML string representation.

- EXTERNXML int rtXmlEncGYearMonth (OSCTXT * pctxt, const OSXSDDateTime * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a numeric gYearMonth element into an XML string representation.

- EXTERNXML int rtXmlEncGMonth (OSCTXT * pctxt, const OSXSDDateTime * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a numeric gMonth element into an XML string representation.

- EXTERNXML int rtXmlEncGMonthDay (OSCTXT * pctxt, const OSXSDDateTime * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a numeric gMonthDay element into an XML string representation.

- EXTERNXML int rtXmlEncGDay (OSCTXT * pctxt, const OSXSDDateTime * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a numeric gDay element into an XML string representation.

- EXTERNXML int rtXmlEncGYearValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)

  This function encodes a numeric gYear value into an XML string representation.

- EXTERNXML int rtXmlEncGYearMonthValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)

  This function encodes a numeric gYearMonth value into an XML string representation.

- EXTERNXML int rtXmlEncGMonthValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)

  This function encodes a numeric gMonth value into an XML string representation.

- EXTERNXML int rtXmlEncGMonthDayValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)

This function encodes a numeric gMonthDay value into an XML string representation.

- EXTERNXML int rtXmlEncGDayValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)

  This function encodes a numeric gDay value into an XML string representation.

- EXTERNXML int rtXmlEncHexBinary (OSCTXT * pctxt, OSSIZE nocts, const OSOCTET * value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD hexBinary type.

- EXTERNXML int rtXmlEncHexBinaryAttr (OSCTXT * pctxt, OSUINT32 nocts, const OSOCTET * value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen)

  This function encodes a variable of the XSD hexBinary type as an attribute.

- EXTERNXML int rtXmlEncHexStrValue (OSCTXT * pctxt, OSSIZE nocts, const OSOCTET * data)

  This function encodes a variable of the XSD hexBinary type.

- EXTERNXML int rtXmlEncIndent (OSCTXT * pctxt)

  This function adds indentation whitespace to the output stream.

- EXTERNXML int rtXmlEncInt (OSCTXT * pctxt, OSINT32 value, const OSUTF8CHAR * elemName, OSXML-Namespace * pNS)

  This function encodes a variable of the XSD integer type.

- EXTERNXML int rtXmlEncIntValue (OSCTXT * pctxt, OSINT32 value)

  This function encodes a variable of the XSD integer type.

- EXTERNXML int rtXmlEncIntAttr (OSCTXT * pctxt, OSINT32 value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen)

  This function encodes a variable of the XSD integer type as an attribute (name="value").

- EXTERNXML int rtXmlEncIntPattern (OSCTXT * pctxt, OSINT32 value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, const OSUTF8CHAR * pattern)

  This function encodes a variable of the XSD integer type using a pattern to specify the format of the integer value.

- EXTERNXML int rtXmlEncIntPatternValue (OSCTXT * pctxt, OSINT32 value, const OSUTF8CHAR * pattern)

- EXTERNXML int rtXmlEncUIntPattern (OSCTXT * pctxt, OSUINT32 value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, const OSUTF8CHAR * pattern)

- EXTERNXML int rtXmlEncUIntPatternValue (OSCTXT * pctxt, OSUINT32 value, const OSUTF8CHAR * pattern)

- EXTERNXML int rtXmlEncInt64 (OSCTXT * pctxt, OSINT64 value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD integer type.

- EXTERNXML int rtXmlEncInt64Pattern (OSCTXT * pctxt, OSINT64 value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, const OSUTF8CHAR * pattern)

- EXTERNXML int rtXmlEncInt64Value (OSCTXT * pctxt, OSINT64 value)

  This function encodes a variable of the XSD integer type.

- EXTERNXML int rtXmlEncInt64PatternValue (OSCTXT * pctxt, OSINT64 value, const OSUTF8CHAR * pattern)

- EXTERNXML int rtXmlEncInt64Attr (OSCTXT * pctxt, OSINT64 value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen)

  This function encodes a variable of the XSD integer type as an attribute (name="value").

- EXTERNXML int rtXmlEncNamedBits (OSCTXT * pctxt, const OSBitMapItem * pBitMap, OSSIZE nbits, const OSOCTET * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

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

- EXTERNXML int rtXmlEncNamedBitsValue (OSCTXT * pctxt, const OSBitMapItem * pBitMap, OSSIZE nbits, const OSOCTET * pvalue)

- EXTERNXML int rtXmlEncNSAttrs (OSCTXT * pctxt, OSRTDList * pNSAttrs)

  This function encodes namespace declaration attributes at the beginning of an XML document.

- EXTERNXML int rtXmlPrintNSAttrs (const char * name, const OSRTDList * data)

  This function prints a list of namespace attributes.

- EXTERNXML int rtXmlEncReal10 (OSCTXT * pctxt, const OSUTF8CHAR * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the ASN.1 REAL base 10 type.

- EXTERNXML int rtXmlEncSoapArrayTypeAttr (OSCTXT * pctxt, const OSUTF8CHAR * name, const OSUTF8CHAR * value, OSSIZE itemCount)

  This function encodes the special SOAP encoding attrType attribute which specifies the number and type of elements in a SOAP array.

- EXTERNXML int rtXmlEncSoapArrayTypeAttr2 (OSCTXT * pctxt, const OSUTF8CHAR * name, OSSIZE nameLen, const OSUTF8CHAR * value, OSSIZE valueLen, OSSIZE itemCount)

- EXTERNXML int rtXmlEncStartDocument (OSCTXT * pctxt)

  This function encodes the XML header text at the beginning of an XML document.

- EXTERNXML int rtXmlEncBOM (OSCTXT * pctxt)

  This function encodes the Unicode byte order mark header at the start of the document.

- EXTERNXML int rtXmlEncStartElement (OSCTXT * pctxt, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, OSRTDList * pNSAttrs, OSBOOL terminate)

  This function encodes a start element tag value (\<elemName\>).

- EXTERNXML int rtXmlEncStartSoapEnv (OSCTXT * pctxt, OSRTDList * pNSAttrs)

  This function encodes a SOAP envelope start element tag.

- EXTERNXML int rtXmlEncStartSoapElems (OSCTXT * pctxt, OSXMLSOAPMsgType msgtype)

  This function encodes a SOAP envelope start element tag and an optional SOAP body or fault tag.

- EXTERNXML int rtXmlEncString (OSCTXT * pctxt, OSXMLSTRING * pxmlstr, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD string type.

- EXTERNXML int rtXmlEncStringValue (OSCTXT * pctxt, const OSUTF8CHAR * value)

  This function encodes a variable of the XSD string type.

- EXTERNXML int rtXmlEncStringValue2 (OSCTXT * pctxt, const OSUTF8CHAR * value, OSSIZE valueLen)

  This function encodes a variable of the XSD string type.

- EXTERNXML int rtXmlEncTermStartElement (OSCTXT * pctxt)

  This function terminates a currently open XML start element by adding either a '>' or '/>' (if empty) terminator.

- EXTERNXML int rtXmlEncUnicodeStr (OSCTXT * pctxt, const OSUNICHAR * value, OSSIZE nchars, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a Unicode string value.

- EXTERNXML int rtXmlEncUTF8Attr (OSCTXT * pctxt, const OSUTF8CHAR * name, const OSUTF8CHAR * value)

  This function encodes an attribute in which the name and value are given as a null-terminated UTF-8 strings.

- EXTERNXML int rtXmlEncUTF8Attr2 (OSCTXT * pctxt, const OSUTF8CHAR * name, OSSIZE nameLen, const OSUTF8CHAR * value, OSSIZE valueLen)

  This function encodes an attribute in which the name and value are given as a UTF-8 strings with lengths.

- EXTERNXML int rtXmlEncUTF8Str (OSCTXT * pctxt, const OSUTF8CHAR * value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a UTF-8 string value.

- EXTERNXML int rtXmlEncUInt (OSCTXT * pctxt, OSUINT32 value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD unsigned integer type.

- EXTERNXML int rtXmlEncUIntValue (OSCTXT * pctxt, OSUINT32 value)

  This function encodes a variable of the XSD unsigned integer type.

- EXTERNXML int rtXmlEncUIntAttr (OSCTXT * pctxt, OSUINT32 value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen)

  This function encodes a variable of the XSD unsigned integer type as an attribute (name="value").

- EXTERNXML int rtXmlEncUInt64 (OSCTXT * pctxt, OSUINT64 value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

This function encodes a variable of the XSD integer type.

- EXTERNXML int rtXmlEncUInt64Pattern (OSCTXT * pctxt, OSUINT64 value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, const OSUTF8CHAR * pattern)
- EXTERNXML int rtXmlEncUInt64Value (OSCTXT * pctxt, OSUINT64 value)

  This function encodes a variable of the XSD integer type.
- EXTERNXML int rtXmlEncUInt64PatternValue (OSCTXT * pctxt, OSUINT64 value, const OSUTF8CHAR * pattern)
- EXTERNXML int rtXmlEncUInt64Attr (OSCTXT * pctxt, OSUINT64 value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen)

  This function encodes a variable of the XSD integer type as an attribute (name="value").
- EXTERNXML int rtXmlEncXSIAttrs (OSCTXT * pctxt, OSBOOL needXSI)

  This function encodes XML schema instance (XSI) attributes at the beginning of an XML document.
- EXTERNXML int rtXmlEncXSITypeAttr (OSCTXT * pctxt, const OSUTF8CHAR * value)

  This function encodes an XML schema instance (XSI) type attribute value (xsi:type="value").
- EXTERNXML int rtXmlEncXSITypeAttr2 (OSCTXT * pctxt, const OSUTF8CHAR * typeNsUri, const OSUTF8CHAR * typeName)

  This function encodes an XML schema instance (XSI) type attribute value (xsi:type="pfx:value").
- EXTERNXML int rtXmlEncXSINilAttr (OSCTXT * pctxt)

  This function encodes an XML nil attribute (xsi:nil="true").
- EXTERNXML int rtXmlFreeInputSource (OSCTXT * pctxt)

  This function closes an input source that was previously created with one of the create input source functions such as 'rtXmlCreateFileInputSource'.
- EXTERNXML OSBOOL rtXmlStrCmpAsc (const OSUTF8CHAR * text1, const char * text2)
- EXTERNXML OSBOOL rtXmlStrnCmpAsc (const OSUTF8CHAR * text1, const char * text2, OSSIZE len)
- EXTERNXML int rtXmlSetEncBufPtr (OSCTXT * pctxt, OSOCTET * bufaddr, OSSIZE bufsiz)

  This function is used to set the internal buffer within the run-time library encoding context.
- EXTERNXML int rtXmlGetIndent (OSCTXT * pctxt)

  This function returns current XML output indent value.
- EXTERNXML OSBOOL rtXmlGetWriteBOM (OSCTXT * pctxt)

  This function returns whether the Unicode byte order mark will be encoded.
- EXTERNXML int rtXmlGetIndentChar (OSCTXT * pctxt)

This function returns current XML output indent character value (default is space).

Macros

- #define rtxPrintNSAttrs rtXmlPrintNSAttrs(name,&data)
- #define rtXmlFinalizeMemBuf do { \ (pMemBuf)->pctxt->buffer.data = (pMemBuf)->buffer + (pMemBuf)->startidx; \ (pMemBuf)->pctxt->buffer.size = \ ((pMemBuf)->usedcnt - (pMemBuf)->startidx); \ (pMemBuf)->pctxt->buffer.dynamic = FALSE; \ (pMemBuf)->pctxt->buffer.byteIndex = 0; \ rtxMemBufReset (pMemBuf); \ } while(0)
- #define rtXmlGetEncBufPtr (pctxt)->buffer.data

  This macro returns the start address of the encoded XML message.
- #define rtXmlGetEncBufLen (pctxt)->buffer.byteIndex

  This macro returns the length of the encoded XML message.

Function Documentation

EXTERNXML int rtXmlEncAny (OSCTXT *pctxt, OSXMLSTRING *pvalue, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a variable of the XSD any type.

This is considered to be a fully-wrapped element of any type (for example: <myType>myData</myType>)

Table 2.43. Parameters

pctxt	Pointer to context block structure.
pvalue	Value to be encoded. This is a string containing the fully-encoded XML text to be copied to the output stream.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	Pointer to namespace structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncAnyStr (OSCTXT *pctxt, const OSUTF8CHAR *pvalue, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

EXTERNXML int rtXmlEncAnyTypeValue (OSCTXT *pctxt, const OSUTF8CHAR *pvalue)

This function encodes a variable of the XSD anyType type.

This is considered to be a fully-wrapped element of any type, possibly containing attributes. (for example: * <myType>myData</myType>)

Table 2.44. Parameters

pctxt	Pointer to context block structure.
pvalue	Value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncAnyAttr (OSCTXT *pctxt, OSRTDList *pAnyAttrList)

This function encodes a list of OSAnyAttr attributes in which the name and value are given as a UTF-8 string.

Table 2.45. Parameters

pctxt	Pointer to context block structure.
pAnyAttrList	List of attributes.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncBase64Binary (OSCTXT *pctxt, OSSIZE nocts, const OSOCTET *value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a variable of the XSD base64Binary type.

Table 2.46. Parameters

pctxt	Pointer to context block structure.
nocts	Number of octets in the value string.
value	Value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	Pointer to namespace structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncBase64BinaryAttr (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

This function encodes a variable of the XSD base64Binary type as an attribute.

Table 2.47. Parameters

pctxt	Pointer to context block structure.
nocts	Number of octets in the value string.
value	Value to be encoded.
attrName	XML attribute name. A name must be provided.
attrNameLen	Length in bytes of the attribute name.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncBase64StrValue (OSCTXT *pctxt, OSSIZE nocts, const OSOCTET *value)

This function encodes a variable of the XSD base64Binary type.

It just puts the encoded value in the destination buffer or stream without any tags.

Table 2.48. Parameters

pctxt	Pointer to context block structure.
nocts	Number of octets in the value string.
value	Value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncBigInt (OSCTXT *pctxt, const OSUTF8CHAR *value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a variable of the XSD integer type.

In this case, the integer is assumed to be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits). For example, parameters used to calculate security values are typically larger than these sizes.

Items of this type are stored in character string constant variables. They can be represented as decimal strings (with no prefixes), as hexadecimal strings starting with a "0x" prefix, as octal strings starting with a "0o" prefix or as binary strings starting with a "0b" prefix. Other radixes are currently not supported.

Table 2.49. Parameters

pctxt	Pointer to context block structure.
value	A pointer to a character string containing the value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	Pointer to namespace structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncBigIntAttr (OSCTXT *pctxt, const OSUTF8CHAR *value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

This function encodes an XSD integer attribute value.

In this case, the integer is assumed to be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits).

Table 2.50. Parameters

pctxt	Pointer to context block structure.
value	A pointer to a character string containing the value to be encoded.
attrName	XML attribute name. A name must be provided.
attrNameLen	Length in bytes of the attribute name.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncBigIntValue (OSCTXT *pctxt, const OSUTF8CHAR *value)

This function encodes an XSD integer attribute value.

In this case, the integer is assumed to be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits). This function just puts the encoded value in the destination buffer or stream without any tags.

Table 2.51. Parameters

pctxt	Pointer to context block structure.
value	A pointer to a character string containing the value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncBitString (OSCTXT *pctxt, OSSIZE nbits, const OSOCTET *value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

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

The encoded data is a sequence of '1's and '0's. This is only used if named bits are not specified in the string (

See also: . rtXmlEncNamedBits).

Table 2.52. Parameters

pctxt	Pointer to context block structure.
nbits	Number of bits in the bit string.
value	Value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	Pointer to namespace structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncBitStringExt (OSCTXT *pctxt, OSSIZE nbits, const OSOCTET *value, OSSIZE dataSize, const OSOCTET *extValue, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

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

The encoded data is a sequence of '1's and '0's. This is used for BIT STRINGs with extdata member present.

Table 2.53. Parameters

pctxt	Pointer to context block structure.

nbits	Number of bits in the bit string.
value	Value to be encoded.
dataSize	Size of data member.
extValue	Value of extdata to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	Pointer to namespace structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncBinStrValue (OSCTXT *pctxt, OSSIZE nbits, const OSOCTET *data)

This function encodes a binary string value as a sequence of '1's and '0's.

Table 2.54. Parameters

pctxt	Pointer to context block structure.
nbits	Number of bits in the value string.
data	Value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncBool (OSCTXT *pctxt, OSBOOL value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a variable of the XSD boolean type.

Table 2.55. Parameters

pctxt	Pointer to context block structure.
value	Boolean value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	Pointer to namespace structure.

Returns: . Completion status of operation:

- 0 = success,

- negative return value is error.

EXTERNXML int rtXmlEncBoolValue (OSCTXT *pctxt, OSBOOL value)

This function encodes a variable of the XSD boolean type.

It just puts the encoded value in the destination buffer or stream without any tags.

Table 2.56. Parameters

pctxt	Pointer to context block structure.
value	Boolean value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncBoolAttr (OSCTXT *pctxt, OSBOOL value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

This function encodes an XSD boolean attribute value.

Table 2.57. Parameters

pctxt	Pointer to context block structure.
value	Boolean value to be encoded.
attrName	XML attribute name. A name must be provided.
attrNameLen	Length in bytes of the attribute name.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncCanonicalSort (OSCTXT *pctxt, OSCTXT *pBufCtxt, OSRTSList *pList)

Sort (as for CXER, Canonical-XER) and encode previously encoded SET OF components.

Table 2.58. Parameters

pctxt	Context to use to encode the sorted encoding
pBufCtxt	Context previously used to encode the components which are to be sorted.

pList	List of OSRTBufLocDescr identifying the location of each of the components' encodings within pBufCtxt.

EXTERNXML int rtXmlEncComment (OSCTXT *pctxt, const OSUTF8CHAR *comment)

This function encodes an XML comment.

The given text will be inserted in between XML comment start and end elements ().

Table 2.59. Parameters

pctxt	Pointer to context block structure.
comment	The comment text.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncDate (OSCTXT *pctxt, const OSXSDDate-Time *pvalue, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a variable of the XSD 'date' type as a string.

This version of the function is used to encode an OSXSDDateTime value into CCYY-MM-DD format.

Table 2.60. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	Pointer to namespace structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncDateValue (OSCTXT *pctxt, const OSXSD-DateTime *pvalue)

This function encodes a variable of the XSD 'date' type as a string.

This version of the function is used to encode an OSXSDDateTime value into CCYY-MM-DD format. This function just puts the encoded value in the destination buffer or stream without any tags.

Table 2.61. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncTime (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a variable of the XSD 'time' type as an string.

This version of the function is used to encode OSXSDDateTime value into any of following format in different condition as stated below. (1) hh-mm-ss.ss used if tz_flag = false (2) hh-mm-ss.ssZ used if tz_flag = false and tzo = 0 (3) hh-mm-ss.ss+HH:MM if tz_flag = false and tzo > 0 (4) hh-mm-ss.ss-HH:MM-HH:MM if tz_flag = false and tzo < 0

Table 2.62. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	Pointer to namespace structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncTimeValue (OSCTXT *pctxt, const OSXSDDateTime *pvalue)

This function encodes a variable of the XSD 'time' type as an string.

This version of the function just puts the encoded value in the destination buffer or stream without any tags.

Table 2.63. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncDateTime (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a numeric date/time value into an XML string representation.

Table 2.64. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	Pointer to namespace structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncDateTimeValue (OSCTXT *pctxt, const OSXSDDateTime *pvalue)

This function encodes a numeric date/time value into an XML string representation.

It just puts the encoded value in the destination buffer or stream without any tags.

Table 2.65. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncDecimal (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS, const OSDecimalFmt *pFmtSpec)

This function encodes a variable of the XSD decimal type.

Table 2.66. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	Pointer to namespace structure.
pFmtSpec	Pointer to format specification structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncDecimalAttr (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen, const OSDecimalFmt *pFmtSpec)

This function encodes a variable of the XSD decimal type as an attribute.

Table 2.67. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
attrName	XML attribute name. A name must be provided.
attrNameLen	Length of XML attribute name.
pFmtSpec	Pointer to format specification structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncDecimalValue (OSCTXT *pctxt, OSREAL value, const OSDecimalFmt *pFmtSpec, char *pDestBuf, OSSIZE destBufSize)

This function encodes a value of the XSD decimal type.

It just puts the encoded value in the destination buffer or stream without any tags.

Table 2.68. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
pFmtSpec	Pointer to format specification structure.

pDestBuf	Pointer to a destination buffer. If NULL (destBufSize should be 0) the encoded value will be put in pctxt->buffer or in stream associated with the pctxt.
destBufSize	The size of the destination buffer. Must be 0, if pDestBuf is NULL.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncDouble (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS, const OSDoubleFmt *pFmtSpec)

This function encodes a variable of the XSD double type.

Table 2.69. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	Pointer to namespace structure.
pFmtSpec	Pointer to format specification structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncDoubleAttr (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen, const OSDoubleFmt *pFmtSpec)

This function encodes a variable of the XSD double type as an attribute.

Table 2.70. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
attrName	XML attribute name. A name must be provided.
attrNameLen	Length of XML attribute name.
pFmtSpec	Pointer to format specification structure.

Returns: . Completion status of operation:

- 0 = success,

- negative return value is error.

EXTERNXML int rtXmlEncDoubleNormalValue (OSCTXT *pctxt, OSREAL value, const OSDoubleFmt *pFmtSpec, int defaultPrecision)

This function encodes a normal (not +/-INF or NaN) value of the XSD double or float type.

It just puts the encoded value in the destination buffer or stream without any tags.

Table 2.71. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
pFmtSpec	Pointer to format specification structure.
defaultPrecision	Default precision of the value. For float, it is 6, for double it is 15.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncDoubleValue (OSCTXT *pctxt, OSREAL value, const OSDoubleFmt *pFmtSpec, int defaultPrecision)

This function encodes a value of the XSD double or float type.

It just puts the encoded value in the destination buffer or stream without any tags. Special real values +/-INF and NaN are encoded as "INF", "-INF", and "NaN", respectively.

Table 2.72. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
pFmtSpec	Pointer to format specification structure.
defaultPrecision	Default precision of the value. For float, it is 6, for double it is 15.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncEmptyElement (OSCTXT *pctxt, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS, OSRTDList *pNSAttrs, OSBOOL terminate)

This function encodes an enpty element tag value (\<elemName/\>).

Table 2.73. Parameters

pctxt	Pointer to context block structure.
elemName	XML element name.
pNS	XML namespace information (prefix and URI).
pNSAttrs	List of namespace attributes to be added to element.
terminate	Add closing '>' character.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncEndDocument (OSCTXT *pctxt)

This function adds trailor information and a null terminator at the end of the XML document being encoded.

Table 2.74. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncEndElement (OSCTXT *pctxt, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes an end element tag value (\</elemName\>).

Table 2.75. Parameters

pctxt	Pointer to context block structure.
elemName	XML element name.
pNS	XML namespace information (prefix and URI).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncEndSoapEnv (OSCTXT *pctxt)

This function encodes a SOAP envelope end element tag (\<SOAP-ENV:Envelope/\>).

Table 2.76. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncEndSoapElems (OSCTXT *pctxt, OSXMLSOAPMsgType msgtype)

This function encodes SOAP end element tags.

If will add a SOAP body or fault end tag based on the SOAP message type argument. It will then add an envelope end element tag.

Table 2.77. Parameters

pctxt	Pointer to context block structure.
msgtype	SOAP message type (body, fault, or none)

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncFloat (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS, const OSDoubleFmt *pFmtSpec)

This function encodes a variable of the XSD float type.

Table 2.78. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	XML namespace information (prefix and URI).
pFmtSpec	Pointer to format specification structure.

Returns: . Completion status of operation:

- 0 = success,

- negative return value is error.

EXTERNXML int rtXmlEncFloatAttr (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen, const OSDoubleFmt *pFmtSpec)

This function encodes a variable of the XSD float type as an attribute.

Table 2.79. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
attrName	XML attribute name. A name must be provided.
attrNameLen	Length of XML attribute name.
pFmtSpec	Pointer to format specification structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncGYear (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a numeric gYear element into an XML string representation.

Table 2.80. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	XML namespace information (prefix and URI).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncGYearMonth (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a numeric gYearMonth element into an XML string representation.

Table 2.81. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	XML namespace information (prefix and URI).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncGMonth (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a numeric gMonth element into an XML string representation.

Table 2.82. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	XML namespace information (prefix and URI).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncGMonthDay (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a numeric gMonthDay element into an XML string representation.

Table 2.83. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.

pNS	XML namespace information (prefix and URI).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncGDay (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a numeric gDay element into an XML string representation.

Table 2.84. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	XML namespace information (prefix and URI).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncGYearValue (OSCTXT *pctxt, const OSXSDDateTime *pvalue)

This function encodes a numeric gYear value into an XML string representation.

It just puts the encoded value into the buffer or stream without any tags.

Table 2.85. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncGYearMonthValue (OSCTXT *pctxt, const OSXSDDateTime *pvalue)

This function encodes a numeric gYearMonth value into an XML string representation.

It just puts the encoded value into the buffer or stream without any tags.

Table 2.86. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncGMonthValue (OSCTXT *pctxt, const OSXSDDateTime *pvalue)

This function encodes a numeric gMonth value into an XML string representation.

It just puts the encoded value into the buffer or stream without any tags.

Table 2.87. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncGMonthDayValue (OSCTXT *pctxt, const OSXSDDateTime *pvalue)

This function encodes a numeric gMonthDay value into an XML string representation.

It just puts the encoded value into the buffer or stream without any tags.

Table 2.88. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncGDayValue (OSCTXT *pctxt, const OSXSDDateTime *pvalue)

This function encodes a numeric gDay value into an XML string representation.

It just puts the encoded value into the buffer or stream without any tags.

Table 2.89. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncHexBinary (OSCTXT *pctxt, OSSIZE nocts, const OSOCTET *value, const OSUTF8CHAR *elemName, OSXML-Namespace *pNS)

This function encodes a variable of the XSD hexBinary type.

Table 2.90. Parameters

pctxt	Pointer to context block structure.
nocts	Number of octets in the value string.
value	Value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	XML namespace information (prefix and URI).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncHexBinaryAttr (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

This function encodes a variable of the XSD hexBinary type as an attribute.

Table 2.91. Parameters

pctxt	Pointer to context block structure.

nocts	Number of octets in the value string.
value	Value to be encoded.
attrName	XML attribute name. A name must be provided.
attrNameLen	Length of XML attribute name.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncHexStrValue (OSCTXT *pctxt, OSSIZE nocts, const OSOCTET *data)

This function encodes a variable of the XSD hexBinary type.

It just puts the encoded value in the destination buffer or stream without any tags.

Table 2.92. Parameters

pctxt	Pointer to context block structure.
nocts	Number of octets in the value string.
data	Value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncIndent (OSCTXT *pctxt)

This function adds indentation whitespace to the output stream.

The amount of indentation to add is determined by the level member variable in the context structure and the OSXMLINDENT constant value.

Table 2.93. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncInt (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a variable of the XSD integer type.

Table 2.94. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	XML namespace information (prefix and URI).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncIntValue (OSCTXT *pctxt, OSINT32 value)

This function encodes a variable of the XSD integer type.

It just puts the encoded value in the destination buffer or stream without any tags.

Table 2.95. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncIntAttr (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

This function encodes a variable of the XSD integer type as an attribute (name="value").

Table 2.96. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
attrName	XML attribute name.
attrNameLen	Length (in bytes) of the attribute name.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncIntPattern (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS, const OSUTF8CHAR *pattern)

This function encodes a variable of the XSD integer type using a pattern to specify the format of the integer value.

Table 2.97. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	XML namespace information (prefix and URI).
pattern	Pattern of the encoded value.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncIntPatternValue (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *pattern)

EXTERNXML int rtXmlEncUIntPattern (OSCTXT *pctxt, OSUINT32 value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS, const OSUTF8CHAR *pattern)

EXTERNXML int rtXmlEncUIntPatternValue (OSCTXT *pctxt, OSUINT32 value, const OSUTF8CHAR *pattern)

EXTERNXML int rtXmlEncInt64 (OSCTXT *pctxt, OSINT64 value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a variable of the XSD integer type.

This version of the function is used for 64-bit integer values.

Table 2.98. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	XML namespace information (prefix and URI).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncInt64Pattern (OSCTXT *pctxt, OSINT64 value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS, const OSUTF8CHAR *pattern)

EXTERNXML int rtXmlEncInt64Value (OSCTXT *pctxt, OSINT64 value)

This function encodes a variable of the XSD integer type.

This version of the function is used for 64-bit integer values. It just puts the encoded value in the destination buffer or stream without any tags.

Table 2.99. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncInt64PatternValue (OSCTXT *pctxt, OSINT64 value, const OSUTF8CHAR *pattern)

EXTERNXML int rtXmlEncInt64Attr (OSCTXT *pctxt, OSINT64 value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

This function encodes a variable of the XSD integer type as an attribute (name="value").

This version of the function is used for 64-bit integer values.

Table 2.100. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
attrName	XML attribute name.
attrNameLen	Length (in bytes) of the attribute name.

Returns: . Completion status of operation:

- 0 = success,

- negative return value is error.

EXTERNXML int rtXmlEncNamedBits (OSCTXT *pctxt, const OSBitMapItem *pBitMap, OSSIZE nbits, const OSOCTET *pvalue, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

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

In this case, a set of named bits was provided in the schema for the bit values. The encoding is a list of the named bit identifers corresponding to each set bit in the bit string value.

Table 2.101. Parameters

pctxt	Pointer to context block structure.
pBitMap	Bit map equating symbolic bit names to bit numbers.
nbits	Number of bits in the sit string value.
pvalue	Bit string value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	Pointer to namespace structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncNamedBitsValue (OSCTXT *pctxt, const OSBitMapItem *pBitMap, OSSIZE nbits, const OSOCTET *pvalue)

EXTERNXML int rtXmlEncNSAttrs (OSCTXT *pctxt, OSRTDList *pNSAttrs)

This function encodes namespace declaration attributes at the beginning of an XML document.

The attributes to be encoded are stored in the namespace list provided, or within the context if a NULL pointer is passed for pNSAttrs. Namespaces are added to this list by using the namespace utility functions.

Table 2.102. Parameters

pctxt	Pointer to context block structure.
pNSAttrs	Pointer to list of namespace attributes.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlPrintNSAttrs (const char *name, const OSRTDList *data)

This function prints a list of namespace attributes.

Table 2.103. Parameters

name	Name to print.
data	List of namespace attributes.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncReal10 (OSCTXT *pctxt, const OSUTF8CHAR *pvalue, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a variable of the ASN.1 REAL base 10 type.

Table 2.104. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to an REAL base 10 value.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	XML namespace information (prefix and URI).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncSoapArrayTypeAttr (OSCTXT *pctxt, const OSUTF8CHAR *name, const OSUTF8CHAR *value, OSSIZE itemCount)

This function encodes the special SOAP encoding attrType attribute which specifies the number and type of elements in a SOAP array.

The form of this attribute is 'attrType="\<type\>[count]"'.

Table 2.105. Parameters

pctxt	Pointer to context block structure.

name	Attribute name (NS prefix + arrayType)
value	UTF-8 string value to be encoded.
itemCount	Count of the number of elements in the array.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncSoapArrayTypeAttr2 (OSCTXT *pctxt, const OSUTF8CHAR *name, OSSIZE nameLen, const OSUTF8CHAR *value, OSSIZE valueLen, OSSIZE itemCount)

EXTERNXML int rtXmlEncStartDocument (OSCTXT *pctxt)

This function encodes the XML header text at the beginning of an XML document.

This is normally the following:

<?xml version="1.0" encoding="UTF-8"?>

Table 2.106. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncBOM (OSCTXT *pctxt)

This function encodes the Unicode byte order mark header at the start of the document.

It is called by rtXmlEncStartDocument and does not need to be called manually.

Table 2.107. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncStartElement (OSCTXT *pctxt, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS, OSRTDList *pNSAttrs, OSBOOL terminate)

This function encodes a start element tag value (\<elemName\>).

Table 2.108. Parameters

pctxt	Pointer to context block structure.
elemName	XML element name. Empty string and null are treated equivalently.
pNS	XML namespace information (prefix and URI). If the prefix is NULL, this method will search the context's namespace stack for a prefix to use.
pNSAttrs	List of namespace attributes to be added to element.
terminate	Add closing '>' character.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncStartSoapEnv (OSCTXT *pctxt, OSRTDList *pNSAttrs)

This function encodes a SOAP envelope start element tag.

This includes all of the standard SOAP namespace attributes.

Table 2.109. Parameters

pctxt	Pointer to context block structure.
pNSAttrs	List of namespace attributes to be added to SOAP envelope.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncStartSoapElems (OSCTXT *pctxt, OSXMLSOAPMsgType msgtype)

This function encodes a SOAP envelope start element tag and an optional SOAP body or fault tag.

This includes all of the standard SOAP namespace attributes.

Table 2.110. Parameters

pctxt	Pointer to context block structure.
msgtype	SOAP message type (body, fault, or none)

Returns: . Completion status of operation:

- 0 = success,

- negative return value is error.

EXTERNXML int rtXmlEncString (OSCTXT *pctxt, OSXMLSTRING *pxmlstr, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a variable of the XSD string type.

Table 2.111. Parameters

pctxt	Pointer to context block structure.
pxmlstr	XML string value to be encoded.
elemName	XML element name. If either null or empty string is passed, no element tag is added to the encoded value.
pNS	XML namespace information (prefix and URI).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncStringValue (OSCTXT *pctxt, const OSUTF8CHAR *value)

This function encodes a variable of the XSD string type.

Table 2.112. Parameters

pctxt	Pointer to context block structure.
value	XML string value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncStringValue2 (OSCTXT *pctxt, const OSUTF8CHAR *value, OSSIZE valueLen)

This function encodes a variable of the XSD string type.

Table 2.113. Parameters

pctxt	Pointer to context block structure.
value	XML string value to be encoded.

valueLen	UTF-8 string value length (in octets).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncTermStartElement (OSCTXT *pctxt)

This function terminates a currently open XML start element by adding either a '>' or '/>' (if empty) terminator.

It will also add XSI attributes to the element.

Table 2.114. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncUnicodeStr (OSCTXT *pctxt, const OSUNICHAR *value, OSSIZE nchars, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a Unicode string value.

Table 2.115. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded. This is a pointer to an array of 16-bit integer values.
nchars	Number of characters in value array.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	XML namespace information (prefix and URI).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncUTF8Attr (OSCTXT *pctxt, const OSUTF8CHAR *name, const OSUTF8CHAR *value)

This function encodes an attribute in which the name and value are given as a null-terminated UTF-8 strings.

Table 2.116. Parameters

pctxt	Pointer to context block structure.
name	Attribute name.
value	UTF-8 string value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncUTF8Attr2 (OSCTXT *pctxt, const OSUTF8CHAR *name, OSSIZE nameLen, const OSUTF8CHAR *value, OSSIZE valueLen)

This function encodes an attribute in which the name and value are given as a UTF-8 strings with lengths.

Table 2.117. Parameters

pctxt	Pointer to context block structure.
name	Attribute name.
nameLen	Attribute name length (in octets).
value	UTF-8 string value to be encoded.
valueLen	UTF-8 string value length (in octets).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncUTF8Str (OSCTXT *pctxt, const OSUTF8CHAR *value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a UTF-8 string value.

Table 2.118. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	XML namespace information (prefix and URI).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncUInt (OSCTXT *pctxt, OSUINT32 value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a variable of the XSD unsigned integer type.

Table 2.119. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	XML namespace information (prefix and URI).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncUIntValue (OSCTXT *pctxt, OSUINT32 value)

This function encodes a variable of the XSD unsigned integer type.

It just puts the encoded value in the destination buffer or stream without any tags.

Table 2.120. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncUIntAttr (OSCTXT *pctxt, OSUINT32 value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

This function encodes a variable of the XSD unsigned integer type as an attribute (name="value").

Table 2.121. Parameters

pctxt	Pointer to context block structure.

value	Value to be encoded.
attrName	XML attribute name.
attrNameLen	Length (in bytes) of the attribute name.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncUInt64 (OSCTXT *pctxt, OSUINT64 value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a variable of the XSD integer type.

This version of the function is used when constraints cause an unsigned 64-bit integer variable to be used.

Table 2.122. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
elemName	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
pNS	XML namespace information (prefix and URI).

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncUInt64Pattern (OSCTXT *pctxt, OSUINT64 value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS, const OSUTF8CHAR *pattern)

EXTERNXML int rtXmlEncUInt64Value (OSCTXT *pctxt, OSUINT64 value)

This function encodes a variable of the XSD integer type.

This version of the function is used when constraints cause an unsigned 64-bit integer variable to be used. It writes the encoded value to the destination buffer or stream without any tags.

Table 2.123. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncUInt64PatternValue (OSCTXT *pctxt, OSUINT64 value, const OSUTF8CHAR *pattern)

EXTERNXML int rtXmlEncUInt64Attr (OSCTXT *pctxt, OSUINT64 value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

This function encodes a variable of the XSD integer type as an attribute (name="value").

This version of the function is used for unsigned 64-bit integer values.

Table 2.124. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
attrName	XML attribute name.
attrNameLen	Length (in bytes) of the attribute name.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncXSIAttrs (OSCTXT *pctxt, OSBOOL needXSI)

This function encodes XML schema instance (XSI) attributes at the beginning of an XML document.

The encoded attributes include the XSI namespace declaration, the XSD schema location attribute, and the XSD no namespace schema location attribute.

The XSI namespace declaration will only be added to the document if schema location attributes exist or the 'needXSI' flag (see below) is set.

Table 2.125. Parameters

pctxt	Pointer to context block structure.
needXSI	This flag is set to true to indicate the XSI namespace declaration is required to support XML items in the main document body such as xsi:type or xsi:nil.

Returns: . Completion status of operation:

- 0 = success,

- negative return value is error.

EXTERNXML int rtXmlEncXSITypeAttr (OSCTXT *pctxt, const OSUTF8CHAR *value)

This function encodes an XML schema instance (XSI) type attribute value (xsi:type="value").

Table 2.126. Parameters

pctxt	Pointer to context block structure.
value	XSI type attribute value.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncXSITypeAttr2 (OSCTXT *pctxt, const OSUTF8CHAR *typeNsUri, const OSUTF8CHAR *typeName)

This function encodes an XML schema instance (XSI) type attribute value (xsi:type="pfx:value").

This will encode namespace declarations for the xsi namespace and for the typeNsUri namespace, if needed.

Table 2.127. Parameters

pctxt	Pointer to context block structure.
typeNsUri	The type's namespace URI. Either null or empty if none.
typeName	The type's name.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlEncXSINilAttr (OSCTXT *pctxt)

This function encodes an XML nil attribute (xsi:nil="true").

Table 2.128. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlFreeInputSource (OSCTXT *pctxt)

This function closes an input source that was previously created with one of the create input source functions such as 'rtXmlCreateFileInputSource'.

Table 2.129. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

EXTERNXML OSBOOL rtXmlStrCmpAsc (const OSUTF8CHAR *text1, const char *text2)

EXTERNXML OSBOOL rtXmlStrnCmpAsc (const OSUTF8CHAR *text1, const char *text2, OSSIZE len)

EXTERNXML int rtXmlSetEncBufPtr (OSCTXT *pctxt, OSOCTET *bufaddr, OSSIZE bufsiz)

This function is used to set the internal buffer within the run-time library encoding context.

It must be called after the context variable is initialized by the rtxInitContext function and before any other compiler generated or run-time library encode function.

This function should not be called with a context that has an associated stream open, but if it is, the stream may be automatically closed.

Table 2.130. Parameters

pctxt	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
bufaddr	A pointer to a memory buffer to use to encode a message. The buffer should be declared as an array of unsigned characters (OCTETs). This parameter can be set to NULL to specify dynamic encoding (i.e., the encode functions will dynamically allocate a buffer to hold the encoded message).
bufsiz	The length of the memory buffer in bytes. Should be set to zero if NULL was specified for bufaddr (i.e. dynamic encoding was selected).

EXTERNXML int rtXmlGetIndent (OSCTXT *pctxt)

This function returns current XML output indent value.

Table 2.131. Parameters

pctxt	Pointer to OSCTXT structure

Returns: . Current indent value (>= 0) if OK, negative status code if error.

EXTERNXML OSBOOL rtXmlGetWriteBOM (OSCTXT *pctxt)

This function returns whether the Unicode byte order mark will be encoded.

Table 2.132. Parameters

pctxt	Pointer to OSCTXT structure.

Returns: . TRUE if BOM is to be encoded, FALSE if not or if context uninitialized.

EXTERNXML int rtXmlGetIndentChar (OSCTXT *pctxt)

This function returns current XML output indent character value (default is space).

Table 2.133. Parameters

pctxt	Pointer to OSCTXT structure

Returns: . Current indent character (> 0) if OK, negative status code if error.

Macro Definition Documentation

#define rtXmlGetEncBufPtr

If a static buffer was used, this is simply the start address of the buffer. If dynamic encoding was done, this will return the start address of the dynamic buffer allocated by the encoder.

Table 2.134. Parameters

pctxt	Pointer to a context structure.

Definition at line 2650 of file osrtxml.h

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

- osrtxml.h

#define rtXmlGetEncBufLen

Table 2.135. Parameters

pctxt	Pointer to a context structure.

Definition at line 2657 of file osrtxml.h

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

- osrtxml.h

XML utility functions.

Detailed Description

Functions

- EXTERNXML int rtXmlWriteToFile (OSCTXT * pctxt, const char * filename)

  This function writes the encoded XML message stored in the context message buffer out to a file.

- EXTERNXML int rtXmlWriteUTF16ToFile (OSCTXT * pctxt, const char * filename)
- EXTERNXML void rtXmlTreatWhitespaces (OSCTXT * pctxt, int whiteSpaceType)
- EXTERNXML int rtXmlCheckBuffer (OSCTXT * pctxt, OSSIZE byte_count)

Function Documentation

EXTERNXML int rtXmlWriteToFile (OSCTXT *pctxt, const char *filename)

This function writes the encoded XML message stored in the context message buffer out to a file.

Table 2.136. Parameters

pctxt	Pointer to OSCTXT structure.
filename	Full path to file to which XML is to be written.

Returns: . 0 - if success, negative value if error.

EXTERNXML int rtXmlWriteUTF16ToFile (OSCTXT *pctxt, const char *filename)

EXTERNXML void rtXmlTreatWhitespaces (OSCTXT *pctxt, int whiteSpaceType)

EXTERNXML int rtXmlCheckBuffer (OSCTXT *pctxt, OSSIZE byte_count)

XML pull-parser decode functions.

Detailed Description

Functions

- EXTERNXML int rtXmlpDecAny (OSCTXT * pctxt, const OSUTF8CHAR ** pvalue)

This function decodes an arbitrary XML section of code as defined by the XSD any type (xsd:any).

- EXTERNXML int rtXmlpDecAny2 (OSCTXT * pctxt, OSUTF8CHAR ** pvalue)

  This version of the rtXmlpDecAny function returns the string in a mutable buffer.

- EXTERNXML int rtXmlpDecAnyAttrStr (OSCTXT * pctxt, const OSUTF8CHAR ** ppAttrStr, OSSIZE attrIndex)

  This function decodes an any attribute string.

- EXTERNXML int rtXmlpDecAnyElem (OSCTXT * pctxt, const OSUTF8CHAR ** pvalue)

  This function decodes an arbitrary XML section of code as defined by the XSD any type (xsd:any).

- EXTERNXML int rtXmlpDecBase64Str (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocts, OSSIZE bufsize)

  This function decodes a contents of a Base64-encode binary string into a static memory structure.

- EXTERNXML int rtXmlpDecBase64Str64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnocts, OSSIZE bufsize)

  This function is identical to rtXmlpDecBase64Str except that it supports 64-bit integer lengths on 64-bit systems.

- EXTERNXML int rtXmlpDecBigInt (OSCTXT * pctxt, const OSUTF8CHAR ** pvalue)

  This function will decode a variable of the XSD integer type.

- EXTERNXML int rtXmlpDecBitString (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnbits, OSUINT32 bufsize)

  This function decodes a bit string value.

- EXTERNXML int rtXmlpDecBitString64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnbits, OSSIZE bufsize)

  This function is identical to rtXmlpDecBitString except that it supports lengths up to 64-bits in size on 64-bit machines.

- EXTERNXML int rtXmlpDecBitStringExt (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnbits, OSOCTET ** ppextdata, OSUINT32 bufsize)

  This function decodes a bit string value.

- EXTERNXML int rtXmlpDecBitStringExt64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnbits, OSOCTET ** ppextdata, OSSIZE bufsize)

  This function is identical to rtXmlpDecBitStringExt except that it supports lengths up to 64-bits in size on 64-bit machines.

- EXTERNXML int rtXmlpDecBool (OSCTXT * pctxt, OSBOOL * pvalue)

  This function decodes a variable of the boolean type.

- EXTERNXML int rtXmlpDecDate (OSCTXT * pctxt, OSXSDDateTime * pvalue)

This function decodes a variable of the XSD 'date' type.

- EXTERNXML int rtXmlpDecDateTime (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'dateTime' type.

- EXTERNXML int rtXmlpDecDecimal (OSCTXT * pctxt, OSREAL * pvalue, int totalDigits, int fractionDigits)

  This function decodes the contents of a decimal data type.

- EXTERNXML int rtXmlpDecDouble (OSCTXT * pctxt, OSREAL * pvalue)

  This function decodes the contents of a float or double data type.

- EXTERNXML int rtXmlpDecDoubleExt (OSCTXT * pctxt, OSUINT8 flags, OSREAL * pvalue)

  This function decodes the contents of a float or double data type.

- EXTERNXML int rtXmlpDecDynBase64Str (OSCTXT * pctxt, OSDynOctStr * pvalue)

  This function decodes a contents of a Base64-encode binary string.

- EXTERNXML int rtXmlpDecDynBase64Str64 (OSCTXT * pctxt, OSDynOctStr64 * pvalue)

  This function is identical to rtXmlpDecDynBase64Str except that it supports 64-bit integer lengths on 64-bit systems.

- EXTERNXML int rtXmlpDecDynBitString (OSCTXT * pctxt, OSDynOctStr * pvalue)

  This function decodes a bit string value.

- EXTERNXML int rtXmlpDecDynHexStr (OSCTXT * pctxt, OSDynOctStr * pvalue)

  This function decodes a contents of a hexBinary string.

- EXTERNXML int rtXmlpDecDynHexStr64 (OSCTXT * pctxt, OSDynOctStr64 * pvalue)

  This function is identical to the rtXmlpDecDynHexStr except that it supports lengths up to 64 bits in size on 64-bit systems.

- EXTERNXML int rtXmlpDecDynUnicodeStr (OSCTXT * pctxt, const OSUNICHAR ** ppdata, OSSIZE * pnchars)

  This function decodes a Unicode string data type.

- EXTERNXML int rtXmlpDecDynUTF8Str (OSCTXT * pctxt, const OSUTF8CHAR ** outdata)

  This function decodes the contents of a UTF-8 string data type.

- EXTERNXML int rtXmlpDecUTF8Str (OSCTXT * pctxt, OSUTF8CHAR * out, OSSIZE max_len)

  This function decodes the contents of a UTF-8 string data type.

- EXTERNXML int rtXmlpDecGDay (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gDay' type.

- EXTERNXML int rtXmlpDecGMonth (OSCTXT * pctxt, OSXSDDateTime * pvalue)

This function decodes a variable of the XSD 'gMonth' type.

- EXTERNXML int rtXmlpDecGMonthDay (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gMonthDay' type.

- EXTERNXML int rtXmlpDecGYear (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gYear' type.

- EXTERNXML int rtXmlpDecGYearMonth (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gYearMonth' type.

- EXTERNXML int rtXmlpDecHexStr (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocts, OSSIZE bufsize)

  This function decodes the contents of a hexBinary string into a static memory structure.

- EXTERNXML int rtXmlpDecHexStr64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnocts, OSSIZE bufsize)

  This function is identical to rtXmlpDecHexStr except that it supports lengths up to 64-bits in size on 64-bit machines.

- EXTERNXML int rtXmlpDecInt (OSCTXT * pctxt, OSINT32 * pvalue)

  This function decodes the contents of a 32-bit integer data type.

- EXTERNXML int rtXmlpDecInt8 (OSCTXT * pctxt, OSINT8 * pvalue)

  This function decodes the contents of an 8-bit integer data type (i.e.

- EXTERNXML int rtXmlpDecInt16 (OSCTXT * pctxt, OSINT16 * pvalue)

  This function decodes the contents of a 16-bit integer data type.

- EXTERNXML int rtXmlpDecInt64 (OSCTXT * pctxt, OSINT64 * pvalue)

  This function decodes the contents of a 64-bit integer data type.

- EXTERNXML int rtXmlpDecNamedBits (OSCTXT * pctxt, const OSBitMapItem * pBitMap, OSOCTET * pvalue, OSUINT32 * pnbits, OSUINT32 bufsize)

  This function decodes the contents of a named bit field.

- EXTERNXML int rtXmlpDecNamedBits64 (OSCTXT * pctxt, const OSBitMapItem * pBitMap, OSOCTET * pvalue, OSSIZE * pnbits, OSSIZE bufsize)

  This function decodes the contents of a named bit field.

- EXTERNXML int rtXmlpDecStrList (OSCTXT * pctxt, OSRTDList * plist)

  This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.

- EXTERNXML int rtXmlpDecTime (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'time' type.

- EXTERNXML int rtXmlpDecUInt (OSCTXT * pctxt, OSUINT32 * pvalue)

  This function decodes the contents of an unsigned 32-bit integer data type.

- EXTERNXML int rtXmlpDecUInt8 (OSCTXT * pctxt, OSOCTET * pvalue)

  This function decodes the contents of an unsigned 8-bit integer data type (i.e.

- EXTERNXML int rtXmlpDecUInt16 (OSCTXT * pctxt, OSUINT16 * pvalue)

  This function decodes the contents of an unsigned 16-bit integer data type.

- EXTERNXML int rtXmlpDecUInt64 (OSCTXT * pctxt, OSUINT64 * pvalue)

  This function decodes the contents of an unsigned 64-bit integer data type.

- EXTERNXML int rtXmlpDecXmlStr (OSCTXT * pctxt, OSXMLSTRING * outdata)

  This function decodes the contents of an XML string data type.

- EXTERNXML int rtXmlpDecXmlStrList (OSCTXT * pctxt, OSRTDList * plist)

  This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.

- EXTERNXML int rtXmlpDecXSIAttr (OSCTXT * pctxt, const OSXMLNameFragments * attrName)

  This function decodes XSI (XML Schema Instance) attributes that may be present in any arbitrary XML element within a document.

- EXTERNXML int rtXmlpDecXSITypeAttr (OSCTXT * pctxt, const OSXMLNameFragments * attrName, const OSUTF8CHAR ** ppAttrValue)

  This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).

- EXTERNXML int rtXmlpGetAttributeID (const OSXMLStrFragment * attrName, OSINT16 nsidx, OSSIZE nAttr, const OSXMLAttrDescr attrNames, OSUINT32 attrPresent)

  This function finds an attribute in the descriptor table.

- EXTERNXML int rtXmlpGetNextElem (OSCTXT * pctxt, OSXMLElemDescr * pElem, OSINT32 level)

  This function parse the next element start tag.

- EXTERNXML int rtXmlpGetNextElemID (OSCTXT * pctxt, const OSXMLElemIDRec * tab, OSSIZE nrows, OSINT32 level, OSBOOL continueParse)

  This function parses the next start tag and finds the index of the element name in the descriptor table.

- EXTERNXML int rtXmlpMarkLastEventActive (OSCTXT * pctxt)

  This function marks current tag as unprocessed.

- EXTERNXML int rtXmlpMatchStartTag (OSCTXT * pctxt, const OSUTF8CHAR * elemLocalName, OSINT16 nsidx)

  This function parses the next start tag that matches with given name.

- EXTERNXML int rtXmlpMatchEndTag (OSCTXT * pctxt, OSINT32 level)

This function parse next end tag that matches with given name.

- EXTERNXML OSBOOL rtXmlpHasAttributes (OSCTXT * pctxt)

  This function checks accessibility of attributes.

- EXTERNXML int rtXmlpGetAttributeCount (OSCTXT * pctxt)

  This function returns number of attributes in last processed start tag.

- EXTERNXML int rtXmlpSelectAttribute (OSCTXT * pctxt, OSXMLNameFragments * pAttr, OSINT16 * nsidx, OSSIZE attrIndex)

  This function selects attribute to decode.

- EXTERNXML OSINT32 rtXmlpGetCurrentLevel (OSCTXT * pctxt)

  This function returns current nesting level.

- EXTERNXML void rtXmlpSetWhiteSpaceMode (OSCTXT * pctxt, OSXMLWhiteSpaceMode whiteSpaceMode)

  Sets the whitespace treatment mode.

- EXTERNXML OSBOOL rtXmlpSetMixedContentMode (OSCTXT * pctxt, OSBOOL mixedContentMode)

  Sets mixed content mode.

- EXTERNXML void rtXmlpSetListMode (OSCTXT * pctxt)

  Sets list mode.

- EXTERNXML OSBOOL rtXmlpListHasItem (OSCTXT * pctxt)

  Check for end of decoded token list.

- EXTERNXML void rtXmlpCountListItems (OSCTXT * pctxt, OSSIZE * itemCnt)

  Count tokens in list.

- EXTERNXML int rtXmlpGetNextSeqElemID2 (OSCTXT * pctxt, const OSXMLElemIDRec * tab, const OSXMLGroupDesc * pGroup, int groups, int curID, int lastMandatoryID, OSBOOL groupMode, OSBOOL checkRepeat)

  This function parses the next start tag and finds index of element name in descriptor table.

- EXTERNXML int rtXmlpGetNextSeqElemID (OSCTXT * pctxt, const OSXMLElemIDRec * tab, const OSXMLGroupDesc * pGroup, int curID, int lastMandatoryID, OSBOOL groupMode)

  This function parses the next start tag and finds index of element name in descriptor table.

- EXTERNXML int rtXmlpGetNextSeqElemIDExt (OSCTXT * pctxt, const OSXMLElemIDRec * tab, const OSXMLGroupDesc * ppGroup, const OSBOOL * extRequired, int postExtRootID, int curID, int lastMandatoryID, OSBOOL groupMode)

  This is an ASN.1 extension-supporting version of rtXmlpGetNextSeqElemID.

- EXTERNXML int rtXmlpGetNextAllElemID (OSCTXT * pctxt, const OSXMLElemIDRec * tab, OSSIZE nrows, const OSUINT8 * pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)

This function parses the next start tag and finds index of element name in descriptor table.

- EXTERNXML int rtXmlpGetNextAllElemID16 (OSCTXT * pctxt, const OSXMLElemIDRec * tab, OSSIZE nrows, const OSUINT16 * pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)

  This function parses the next start tag and finds index of element name in descriptor table.

- EXTERNXML int rtXmlpGetNextAllElemID32 (OSCTXT * pctxt, const OSXMLElemIDRec * tab, OSSIZE nrows, const OSUINT32 * pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)

  This function parses the next start tag and finds index of element name in descriptor table.

- EXTERNXML void rtXmlpSetNamespaceTable (OSCTXT * pctxt, const OSUTF8CHAR * namespaceTable, OSSIZE nmNamespaces)

  Sets user namespace table.

- EXTERNXML int rtXmlpCreateReader (OSCTXT * pctxt)

  Creates pull parser reader structure within the context.

- EXTERNXML void rtXmlpHideAttributes (OSCTXT * pctxt)

  Disable access to attributes.

- EXTERNXML OSBOOL rtXmlpNeedDecodeAttributes (OSCTXT * pctxt)

  This function checks if attributes were previously decoded.

- EXTERNXML void rtXmlpMarkPos (OSCTXT * pctxt)

  Save current decode position.

- EXTERNXML void rtXmlpRewindToMarkedPos (OSCTXT * pctxt)

  Rewind to saved decode position.

- EXTERNXML void rtXmlpResetMarkedPos (OSCTXT * pctxt)

  Reset saved decode position.

- EXTERNXML int rtXmlpGetXSITypeAttr (OSCTXT * pctxt, const OSUTF8CHAR ** ppAttrValue, OSINT16 * nsidx, OSSIZE * pLocalOffs)

  This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).

- EXTERNXML int rtXmlpGetXmlnsAttrs (OSCTXT * pctxt, OSRTDList * pNSAttrs)

  This function decodes namespace attributes from start tag and adds them to the given list.

- EXTERNXML int rtXmlpDecXSIAttrs (OSCTXT * pctxt)

  This function decodes XSI (XML Schema Instance) that may be present in any arbitrary XML element within a document.

- EXTERNXML OSBOOL rtXmlpIsEmptyElement (OSCTXT * pctxt)

  Check element content: empty or not.

- EXTERNXML int rtXmlEncAttrC14N (OSCTXT * pctxt)

  This function used only in C14 mode.

- EXTERNXML struct OSXMLReader * rtXmlpGetReader (OSCTXT * pctxt)

  This function fetches the XML reader structure from the context for use in low-level pull parser calls.

- EXTERNXML OSBOOL rtXmlpIsLastEventDone (OSCTXT * pctxt)

  Check processing status of current tag.

- EXTERNXML int rtXmlpGetXSITypeIndex (OSCTXT * pctxt, const OSXMLItemDescr typetab, OSSIZE typetabsiz)

  This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type) and find type index in descriptor table.

- EXTERNXML int rtXmlpLookupXSITypeIndex (OSCTXT * pctxt, const OSUTF8CHAR * pXsiType, OSINT16 xsiTypeIdx, const OSXMLItemDescr typetab, OSSIZE typetabsiz)

  This function find index of XSI (XML Schema Instance) type in descriptor table.

- EXTERNXML void rtXmlpForceDecodeAsGroup (OSCTXT * pctxt)

  Disable skipping of unknown elements in optional sequence tail.

- EXTERNXML OSBOOL rtXmlpIsDecodeAsGroup (OSCTXT * pctxt)

  This function checks if "decode as group" mode was forced.

- EXTERNXML OSBOOL rtXmlpIsUTF8Encoding (OSCTXT * pctxt)

  This function checks if the encoding specified in XML header is UTF-8.

- EXTERNXML int rtXmlpReadBytes (OSCTXT * pctxt, OSOCTET * pbuf, OSSIZE nbytes)

  This function reads the specified number of bytes directly from the underlying XML parser stream.

Macros

- #define OSXMLREALENC_OBJSYS 0x1F /* Obj-Sys XML encoding rules */
- #define OSXMLREALENC_BXER 0x10 /* basic-XER */
- #define OSXMLREALENC_EXERMODS 0x1B /* extended-XER with MODIFIED-ENCODINGS */
- #define OSXMLREALENC_EXERDECIMAL 0x03 /* extended-XER with DECIMAL */

Function Documentation

EXTERNXML int rtXmlpDecAny (OSCTXT *pctxt, const OSUTF8CHAR **pvalue)

This function decodes an arbitrary XML section of code as defined by the XSD any type (xsd:any).

The decoded XML fragment is returned as a string in the form as it appears in the document. Memory is allocated for the string using the rtxMemAlloc function.

Table 2.137. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to UTF8 character string pointer to receive decoded XML fragment. Memory is allocated for the string using the run-time memory manager.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecAny2 (OSCTXT *pctxt, OSUTF8CHAR **pvalue)

This version of the rtXmlpDecAny function returns the string in a mutable buffer.

Table 2.138. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to UTF8 character string pointer to receive decoded XML fragment. Memory is allocated for the string using the run-time memory manager.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecAnyAttrStr (OSCTXT *pctxt, const OSUTF8CHAR **ppAttrStr, OSSIZE attrIndex)

This function decodes an any attribute string.

The full attribute string (name="value") is decoded and returned on the string output argument. Memory is allocated for the string using the rtxMemAlloc function.

Table 2.139. Parameters

pctxt	Pointer to context block structure.
ppAttrStr	Pointer to UTF8 character string pointer to receive decoded attribute string. Memory is allocated for the string using the run-time memory manager.
attrIndex	Index of attribute.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecAnyElem (OSCTXT *pctxt, const OSUTF8CHAR **pvalue)

This function decodes an arbitrary XML section of code as defined by the XSD any type (xsd:any).

The decoded XML fragment is returned as a string in the form as it appears in the document. Memory is allocated for the string using the rtxMemAlloc function. The difference between this function and rtXmlpDecAny is that this function preserves the full encoded XML fragment including the start and end elements tags and attributes. rtXmlpDecAny decodes the contents within the start and end tags.

Table 2.140. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to UTF8 character string pointer to receive decoded XML fragment. Memory is allocated for the string using the run-time memory manager.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecBase64Str (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSSIZE bufsize)

This function decodes a contents of a Base64-encode binary string into a static memory structure.

The octet string must be Base64 encoded. This function call is used to decode a sized base64Binary string production. Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.141. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.
pnocts	A pointer to an integer value to receive the decoded number of octets.
bufsize	The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecBase64Str64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnocts, OSSIZE bufsize)

This function is identical to rtXmlpDecBase64Str except that it supports 64-bit integer lengths on 64-bit systems.

Table 2.142. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.
pnocts	A pointer to an integer value to receive the decoded number of octets.
bufsize	The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecBigInt (OSCTXT *pctxt, const OSUTF8CHAR **pvalue)

This function will decode a variable of the XSD integer type.

In this case, the integer is assumed to be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits). For example, parameters used to calculate security values are typically larger than these sizes.

These variables are stored in character string constant variables. The radix should be 10. If it is necessary to convert to another radix, then use rtxBigIntSetStr or rtxBigIntToString functions. Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.143. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to a pointer to receive decoded UTF-8 string. Dynamic memory is allocated for the variable using the rtMemAlloc function. The decoded variable is represented as a string starting with appropriate prefix.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecBitString (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnbits, OSUINT32 bufsize)

This function decodes a bit string value.

The string consists of a series of '1' and '0' characters. This is the static version in which the user provides a pre-allocated memory buffer to receive the decoded data. One byte in a memory buffer can hold 8 characters of encoded data. Bits are stored from MSB to LSB order.

Table 2.144. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to a variable to receive the decoded boolean value.
pnbits	Pointer to hold decoded number of bits.
bufsize	Size of buffer passed in pvalue argument.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecBitString64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnbits, OSSIZE bufsize)

This function is identical to rtXmlpDecBitString except that it supports lengths up to 64-bits in size on 64-bit machines.

This function decodes a bit string value. The string consists of a series of '1' and '0' characters. This is the static version in which the user provides a pre-allocated memory buffer to receive the decoded data. One byte in a memory buffer can hold 8 characters of encoded data. Bits are stored from MSB to LSB order.

Table 2.145. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to a variable to receive the decoded boolean value.
pnbits	Pointer to hold decoded number of bits.
bufsize	Size of buffer passed in pvalue argument.

Returns: . Completion status of operation:

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

See also: . rtXmlpDecBitString

EXTERNXML int rtXmlpDecBitStringExt (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnbits, OSOCTET **ppextdata, OSUINT32 bufsize)

This function decodes a bit string value.

The string consists of a series of '1' and '0' characters. This is the static version in which the user provides a pre-allocated memory buffer to receive the decoded data. One byte in a memory buffer can hold 8 characters of encoded data. Bits are stored from MSB to LSB order. This version supports BIT STRINGs with the extdata member present.

Table 2.146. Parameters

pctxt	Pointer to context block structure.

pvalue	Pointer to a variable to receive the decoded boolean value.
pnbits	Pointer to hold decoded number of bits.
ppextdata	Pointer to byte array containing extension data.
bufsize	Size of buffer passed in pvalue argument.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecBitStringExt64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnbits, OSOCTET **ppextdata, OSSIZE bufsize)

This function is identical to rtXmlpDecBitStringExt except that it supports lengths up to 64-bits in size on 64-bit machines.

This function decodes a bit string value. The string consists of a series of '1' and '0' characters. This is the static version in which the user provides a pre-allocated memory buffer to receive the decoded data. One byte in a memory buffer can hold 8 characters of encoded data. Bits are stored from MSB to LSB order. This version supports BIT STRINGs with the extdata member present.

Table 2.147. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to a variable to receive the decoded boolean value.
pnbits	Pointer to hold decoded number of bits.
ppextdata	Pointer to byte array containing extension data.
bufsize	Size of buffer passed in pvalue argument.

Returns: . Completion status of operation:

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

See also: . rtXmlpDecBitStringExt

EXTERNXML int rtXmlpDecBool (OSCTXT *pctxt, OSBOOL *pvalue)

This function decodes a variable of the boolean type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.148. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to a variable to receive the decoded boolean value.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecDate (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'date' type.

Input is expected to be a string of characters returned by an XML pull parser. The string should have CCYY-MM-DD format.

Table 2.149. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecDateTime (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'dateTime' type.

Input is expected to be a string of characters returned by an XML pull parser.

Table 2.150. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecDecimal (OSCTXT *pctxt, OSREAL *pvalue, int totalDigits, int fractionDigits)

This function decodes the contents of a decimal data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.151. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to 64-bit double value to receive decoded result.
totalDigits	Number of total digits in the decimal number from XSD totalDigits facet value. Argument should be set to -1 if facet not given.
fractionDigits	Number of fraction digits in the decimal number from XSD fractionDigits facet value. Argument should be set to -1 if facet not given.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecDouble (OSCTXT *pctxt, OSREAL *pvalue)

This function decodes the contents of a float or double data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.152. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to 64-bit double value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecDoubleExt (OSCTXT *pctxt, OSUINT8 flags, OSREAL *pvalue)

This function decodes the contents of a float or double data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.153. Parameters

pctxt	Pointer to context block structure.
flags	Specifies alternatives allowed in the lexical value. See OSXMLREALENC* constants. bit 0 (rightmost) set: recognize INF, -INF, NaN text values bit 1 set: recognize leading '+' on normal reals bit 2 set: recognize leading '+' on INF bit 3 set: recognize leading zeros in exponent bit 4 set: recognize exponents

pvalue	Pointer to 64-bit double value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecDynBase64Str (OSCTXT *pctxt, OSDynOctStr *pvalue)

This function decodes a contents of a Base64-encode binary string.

The octet string must be Base64 encoded. This function will allocate dynamic memory to store the decoded result. Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.154. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated for the string using the ::rtxMemAlloc function.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecDynBase64Str64 (OSCTXT *pctxt, OSDynOctStr64 *pvalue)

This function is identical to rtXmlpDecDynBase64Str except that it supports 64-bit integer lengths on 64-bit systems.

Table 2.155. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated for the string using the ::rtxMemAlloc function.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecDynBitString (OSCTXT *pctxt, OSDynOctStr *pvalue)

This function decodes a bit string value.

The string consists of a series of '1' and '0' characters. This is the dynamic version in which memory is allocated for the returned binary string variable. Bits are stored from MSB to LSB order.

Table 2.156. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to a variable to receive the decoded boolean value.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecDynHexStr (OSCTXT *pctxt, OSDynOctStr *pvalue)

This function decodes a contents of a hexBinary string.

This function will allocate dynamic memory to store the decoded result. Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.157. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the ::rtxMemAlloc function.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecDynHexStr64 (OSCTXT *pctxt, OSDynOctStr64 *pvalue)

This function is identical to the rtXmlpDecDynHexStr except that it supports lengths up to 64 bits in size on 64-bit systems.

Table 2.158. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the ::rtxMemAlloc function.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecDynUnicodeStr (OSCTXT *pctxt, const OSUNICHAR **ppdata, OSSIZE *pnchars)

This function decodes a Unicode string data type.

The input is assumed to be in UTF-8 format. This function reads each character and converts it into its Unicode equivalent.

Table 2.159. Parameters

pctxt	Pointer to context block structure.
ppdata	Pointer to Unicode character string. A Unicode character string is represented as an array of unsigned 16-bit integers in C. Memory is allocated for the string using the run-time memory manager.
pnchars	Pointer to integer variables to receive the number of characters in the string.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecDynUTF8Str (OSCTXT *pctxt, const OSUTF8CHAR **outdata)

This function decodes the contents of a UTF-8 string data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.160. Parameters

pctxt	Pointer to context block structure.
outdata	Pointer to a pointer to receive decoded UTF-8 string. Memory is allocated for this string using the run-time memory manager.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecUTF8Str (OSCTXT *pctxt, OSUTF8CHAR *out, OSSIZE max_len)

This function decodes the contents of a UTF-8 string data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.161. Parameters

pctxt	Pointer to context block structure.
out	Pointer to an array of OSUTF8CHAR to receive decoded UTF-8 string.
max_len	Length of out array.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecGDay (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'gDay' type.

Input is expected to be a string of characters returned by an XML pull parser. The string should have —DD[-+hh:mm|Z] format.

Table 2.162. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecGMonth (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'gMonth' type.

Input is expected to be a string of characters returned by an XML pull parser. The string should have –MM[-+hh:mm|Z] format.

Table 2.163. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

Returns: . Completion status of operation:

- 0 = success,

- negative return value is error.

EXTERNXML int rtXmlpDecGMonthDay (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'gMonthDay' type.

Input is expected to be a string of characters returned by an XML pull parser. The string should have –MM-DD[-+hh:mm|Z] format.

Table 2.164. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecGYear (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'gYear' type.

Input is expected to be a string of characters returned by an XML pull parser. The string should have CCYY[-+hh:mm|Z] format.

Table 2.165. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecGYearMonth (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'gYearMonth' type.

Input is expected to be a string of characters returned by an XML pull parser. The string should have CCYY-MM[-+hh:mm|Z] format.

Table 2.166. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecHexStr (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSSIZE bufsize)

This function decodes the contents of a hexBinary string into a static memory structure.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.167. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.
pnocts	A pointer to an integer value to receive the decoded number of octets.
bufsize	The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecHexStr64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnocts, OSSIZE bufsize)

This function is identical to rtXmlpDecHexStr except that it supports lengths up to 64-bits in size on 64-bit machines.

Table 2.168. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.
pnocts	A pointer to an integer value to receive the decoded number of octets.
bufsize	The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

Returns: . Completion status of operation:

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

See also: . rtXmlpDecHexStr

EXTERNXML int rtXmlpDecInt (OSCTXT *pctxt, OSINT32 *pvalue)

This function decodes the contents of a 32-bit integer data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.169. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to 32-bit integer value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecInt8 (OSCTXT *pctxt, OSINT8 *pvalue)

This function decodes the contents of an 8-bit integer data type (i.e.

a signed byte type in the range of -128 to 127). Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.170. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to 8-bit integer value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecInt16 (OSCTXT *pctxt, OSINT16 *pvalue)

This function decodes the contents of a 16-bit integer data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.171. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to 16-bit integer value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecInt64 (OSCTXT *pctxt, OSINT64 *pvalue)

This function decodes the contents of a 64-bit integer data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.172. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to 64-bit integer value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecNamedBits (OSCTXT *pctxt, const OSBitMapItem *pBitMap, OSOCTET *pvalue, OSUINT32 *pnbits, OSUINT32 bufsize)

This function decodes the contents of a named bit field.

This is a space-separated list of token values in which each token corresponds to a bit field in a bit map.

Table 2.173. Parameters

pctxt	Pointer to context block structure.
pBitMap	Pointer to bit map structure that defined token to bit mappings.
pvalue	Pointer to buffer to recieve decoded bit map.
pnbits	Number of bits in decoded bit map.
bufsize	Size of buffer passed in to received decoded bit values.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecNamedBits64 (OSCTXT *pctxt, const OSBitMapItem *pBitMap, OSOCTET *pvalue, OSSIZE *pnbits, OSSIZE bufsize)

This function decodes the contents of a named bit field.

This is a space-separated list of token values in which each token corresponds to a bit field in a bit map.

Table 2.174. Parameters

pctxt	Pointer to context block structure.
pBitMap	Pointer to bit map structure that defined token to bit mappings.
pvalue	Pointer to buffer to recieve decoded bit map.
pnbits	Number of bits in decoded bit map.
bufsize	Size of buffer passed in to received decoded bit values.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecStrList (OSCTXT *pctxt, OSRTDList *plist)

This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.

Memory is allocated for the list nodes and token values using the rtx memory management functions.

Table 2.175. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
plist	A pointer to a linked list structure to which the parsed token values will be added.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecTime (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'time' type.

Input is expected to be a string of characters returned by an XML pull parser. The string should have one of following formats:

(1) hh-mm-ss.ss used if tz_flag = false
(2) hh-mm-ss.ssZ used if tz_flag = false and tzo = 0
(3) hh-mm-ss.ss+HH:MM if tz_flag = false and tzo > 0
(4) hh-mm-ss.ss-HH:MM-HH:MM
if tz_flag = false and tzo < 0

Table 2.176. Parameters

pctxt	Pointer to context block structure.
pvalue	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecUInt (OSCTXT *pctxt, OSUINT32 *pvalue)

This function decodes the contents of an unsigned 32-bit integer data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.177. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to unsigned 32-bit integer value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecUInt8 (OSCTXT *pctxt, OSOCTET *pvalue)

This function decodes the contents of an unsigned 8-bit integer data type (i.e.

a signed byte type in the range of 0 to 255). Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.178. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to unsigned 8-bit integer value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecUInt16 (OSCTXT *pctxt, OSUINT16 *pvalue)

This function decodes the contents of an unsigned 16-bit integer data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.179. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to unsigned 16-bit integer value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecUInt64 (OSCTXT *pctxt, OSUINT64 *pvalue)

This function decodes the contents of an unsigned 64-bit integer data type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Table 2.180. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to unsigned 64-bit integer value to receive decoded result.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecXmlStr (OSCTXT *pctxt, OSXMLSTRING *outdata)

This function decodes the contents of an XML string data type.

This type contains a pointer to a UTF-8 characer string plus flags that can be set to alter the encoding of the string (for example, the cdata flag allows the string to be encoded in a CDATA section). Input is expected to be a string of UTF-8 characters returned by an XML parser.

Table 2.181. Parameters

pctxt	Pointer to context block structure.
outdata	Pointer to an XML string structure to receive the decoded string. Memory is allocated for the string using the run-time memory manager.

Returns: . Completion status of operation:

- 0 = success,

- negative return value is error.

EXTERNXML int rtXmlpDecXmlStrList (OSCTXT *pctxt, OSRTDList *plist)

This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.

Memory is allocated for the list nodes and token values using the rtx memory management functions. List contains OSXMLSTRING structures.

Table 2.182. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
plist	A pointer to a linked list structure to which the parsed token values will be added.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecXSIAttr (OSCTXT *pctxt, const OSXMLNameFragments *attrName)

This function decodes XSI (XML Schema Instance) attributes that may be present in any arbitrary XML element within a document.

Table 2.183. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
attrName	A pointer to a structure holding various parts of an attribute name. The parts are in the form of string fragments meaning they are not null terminated. The user must be careful to use the value and length when working with them.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecXSITypeAttr (OSCTXT *pctxt, const OSXMLNameFragments *attrName, const OSUTF8CHAR **ppAttrValue)

This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).

Prior to calling this function, use rtXmlpSelectAttribute to select the attribute. After calling this function, if you want to read the same attribute again, you will need to call rtXmlpSelectAttribute again.

Table 2.184. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
attrName	A pointer to a structure holding various parts of an attribute name. The parts are in the form of string fragments meaning they are not null terminated. The user must be careful to use the value and length when working with them.
ppAttrValue	A pointer to a pointer to a UTF8 character string to received the decoded XSI type name.

Returns: . Completion status of operation:

- 0 = success,
- 1 = OK, but attrName was not xsi:type (i.e. no attribute match)
- negative return value is error.

EXTERNXML int rtXmlpGetAttributeID (const OSXMLStrFragment *attrName, OSINT16 nsidx, OSSIZE nAttr, const OSXMLAttrDescr attrNames[], OSUINT32 attrPresent[])

This function finds an attribute in the descriptor table.

Table 2.185. Parameters

attrName	A pointer to a structure holding various parts of an attribute name. The parts are in the form of string fragments meaning they are not null terminated. The user must be careful to use the value and length when working with them.
nsidx	Namespace index: • 0 = attribute is unqualified, • positive value is user namespace from generated namespace table, • negative value is predefined namespace like XSD instance and ect.
nAttr	Number of descriptors in table.
attrNames	Attributes descriptor table.
attrPresent	Bit array to mark already decoded attributes. It is used to identify duplicate attributes.

Returns: . Completion status of operation:

- positive or zero return value is attribute index in descriptor table,
- negative return value is error.

EXTERNXML int rtXmlpGetNextElem (OSCTXT *pctxt, OSXMLElemDescr *pElem, OSINT32 level)

This function parse the next element start tag.

Table 2.186. Parameters

pctxt	Pointer to context block structure.
pElem	Pointer to a structure to receive the decoded element descriptor.
level	Nesting level of parsed start tag. When value equal -1 parsed next start tag.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpGetNextElemID (OSCTXT *pctxt, const OSXMLElemIDRec *tab, OSSIZE nrows, OSINT32 level, OSBOOL continueParse)

This function parses the next start tag and finds the index of the element name in the descriptor table.

If this reaches the closing tag for the current level, it returns XML_OK_EOB. If this encounters an unexpected element and !continueParse, it returns RTERR_UNEXPELEM (without logging it).

Table 2.187. Parameters

pctxt	Pointer to context block structure.
tab	Elements descriptor table.
nrows	Number of descriptors in table.
level	Nesting level of parsed start tag. When value equal -1 function parse next start tag.
continueParse	When value equals TRUE function skips unrecognized elements; skipped elements are logged, but an error is not returned.

Returns: . Completion status of operation:

- positive or zero value is element identifier,
- negative return value is error.

EXTERNXML int rtXmlpMarkLastEventActive (OSCTXT *pctxt)

This function marks current tag as unprocessed.

This will cause the element to be processed again in the next pull-parser function.

Table 2.188. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpMatchStartTag (OSCTXT *pctxt, const OSUTF8CHAR *elemLocalName, OSINT16 nsidx)

This function parses the next start tag that matches with given name.

Table 2.189. Parameters

pctxt	Pointer to context block structure.
elemLocalName	Name of parsed element.
nsidx	Namespace index: • 0 = attribute is unqualified, • positive value is user namespace from generated namespace table, • negative value is predefined namespace like XSD instance and ect.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpMatchEndTag (OSCTXT *pctxt, OSINT32 level)

This function parse next end tag that matches with given name.

Table 2.190. Parameters

pctxt	Pointer to context block structure.
level	Nesting level of parsed start tag. When value equal -1 function parse next end tag with current level.

Returns: . Completion status of operation:

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

EXTERNXML OSBOOL rtXmlpHasAttributes (OSCTXT *pctxt)

This function checks accessibility of attributes.

Table 2.191. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

- TRUE = attributes are present and access is enabled,

- FALSE = attributes are absent or access is disabled.

EXTERNXML int rtXmlpGetAttributeCount (OSCTXT *pctxt)

This function returns number of attributes in last processed start tag.

Table 2.192. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

- zero or positive value is atributes number,
- negative return value is error.

EXTERNXML int rtXmlpSelectAttribute (OSCTXT *pctxt, OSXML-NameFragments *pAttr, OSINT16 *nsidx, OSSIZE attrIndex)

This function selects attribute to decode.

Table 2.193. Parameters

pctxt	Pointer to context block structure.
pAttr	Pointer to structure to receive the various parts of an attribute name.
nsidx	Pointer to value to receive namespace index: • 0 = attribute is unqualified, • positive value is user namespace from generated namespace table, • negative value is predefined namespace like XSD instance and ect (see enum OSXMLNsIndex)
attrIndex	Index of selected attribute.

Returns: . Completion status of operation:

- positive or zero return value is attribute index in descriptor table,
- negative return value is error.

EXTERNXML OSINT32 rtXmlpGetCurrentLevel (OSCTXT *pctxt)

This function returns current nesting level.

Table 2.194. Parameters

pctxt	Pointer to context block structure.

Returns: . Current nesting level.

EXTERNXML void rtXmlpSetWhiteSpaceMode (OSCTXT *pctxt, OSXMLWhiteSpaceMode whiteSpaceMode)

Sets the whitespace treatment mode.

This mode affects the content and attribute values reading. For example, if OSXMLWSM_COLLAPSE mode is set then all spaces in returned data will be already collapsed.

Table 2.195. Parameters

pctxt	Pointer to context block structure.
whiteSpaceMode	White space mode.

Returns: . Previously set whitespace mode.

EXTERNXML OSBOOL rtXmlpSetMixedContentMode (OSCTXT *pctxt, OSBOOL mixedContentMode)

Sets mixed content mode.

Table 2.196. Parameters

pctxt	Pointer to context block structure.
mixedContentMode	Enable/disable mixed mode.

Returns: . Previously state of mixed mode.

EXTERNXML void rtXmlpSetListMode (OSCTXT *pctxt)

Sets list mode.

Attribute value or element content is decoded by tokens.

Table 2.197. Parameters

pctxt	Pointer to context block structure.

EXTERNXML OSBOOL rtXmlpListHasItem (OSCTXT *pctxt)

Check for end of decoded token list.

Table 2.198. Parameters

pctxt	Pointer to context block structure.

Returns: . State of decoded list:

- TRUE = list is not finished,

- FALSE = all tokens was decoded.

EXTERNXML void rtXmlpCountListItems (OSCTXT *pctxt, OSSIZE *itemCnt)

Count tokens in list.

Table 2.199. Parameters

pctxt	Pointer to context block structure.
itemCnt	Pointer to number of elements in list.

Returns: . Token number. For element content function check accessed part of content only. Returned value may be below then real token number.

EXTERNXML int rtXmlpGetNextSeqElemID2 (OSCTXT *pctxt, const OSXMLElemIDRec *tab, const OSXMLGroupDesc *pGroup, int groups, int curID, int lastMandatoryID, OSBOOL groupMode, OSBOOL checkRepeat)

This function parses the next start tag and finds index of element name in descriptor table.

It is used to decode sequences in strict mode.

Table 2.200. Parameters

pctxt	Pointer to context block structure.
tab	Elements descriptor table.
pGroup	Decode groups table.
groups	Number of groups in groups table. If checkRepeat is FALSE, you can pass 0 for this if the value is unknown.
curID	Current decode group.
lastMandatoryID	Identifier of last mandatory element.
groupMode	This parameter must be setted to TRUE when decode groups or base types.
checkRepeat	If true, this method is being called to check for a repeat of curID. In this case, we do not treat curID as being required. Otherwise, curID is required if curId <= lastMandatoryID.

Returns: . Completion status of operation:

- positive or zero value is element identifier,
- negative return value is error.

EXTERNXML int rtXmlpGetNextSeqElemID (OSCTXT *pctxt, const OSXMLElemIDRec *tab, const OSXMLGroupDesc *pGroup, int curID, int lastMandatoryID, OSBOOL groupMode)

This function parses the next start tag and finds index of element name in descriptor table.

It is used to decode sequences in strict mode.

Handling of unexpected elements:

- If the current decode group includes an any case, unexpected elements will be matched against it (the any case id will be returned).
- Otherwise, if groupMode is true, and the element identified by lastMandatoryID has been reached, XML_OK_EOB is returned. The unexpected element may belong to something following the elements in tab.
- If neither of the above applies, unknown elements will be logged and skipped over, with this method continuing as if the unexpected element were not present. Handling of the case of reaching closing tag for current level:
- If the last mandatory element is reached, return XML_OK_EOB.
- Otherwise, log and return XML_E_ELEMSMISRQ.

This function is equivalent to: rtXmlpGetNextSeqElemID2(pctxt, tab, pGroup, curID, lastMandatoryID, groupMode, FALSE);

Table 2.201. Parameters

pctxt	Pointer to context block structure.
tab	Elements descriptor table.
pGroup	Decode groups table.
curID	Current decode group.
lastMandatoryID	Identifier of last mandatory element.
groupMode	This parameter must be setted to TRUE when decoding groups or base types.

Returns: . Completion status of operation:

- positive or zero value is element identifier,
- negative return value is error.

EXTERNXML int rtXmlpGetNextSeqElemIDExt (OSCTXT *pctxt, const OSXMLElemIDRec *tab, const OSXMLGroupDesc *ppGroup, const OSBOOL *extRequired, int postExtRootID, int curID, int lastMandatoryID, OSBOOL groupMode)

This is an ASN.1 extension-supporting version of rtXmlpGetNextSeqElemID.

It allows required extension elements to be absent, except that when an extension group is present, all required extension elements in that group must be present. It otherwise behaves the same as rtXmlpGetNextSeqElemID.

Table 2.202. Parameters

pctxt	Pointer to context block structure.

tab	Elements descriptor table.
pGroup	Decode groups table.
extRequired	extRequired[i] corresponds to pGroup[i]. This is TRUE if and only if the decode group ends with a non-optional, non-default extension element that must be present in the encoding (because it is inside an extension group for which some element has already been decoded).
postExtRootID	This is the group id corresponding to the decode group that begins with the first root element that follows the extension additions. If there is no such element, this is -1.
curID	Current decode group.
lastMandatoryID	Identifier of last mandatory element.
groupMode	This parameter must be setted to TRUE when decoding groups or base types.

Returns: . Completion status of operation:

- positive or zero value is element identifier,
- negative return value is error.

EXTERNXML int rtXmlpGetNextAllElemID (OSCTXT *pctxt, const OSXMLElemIDRec *tab, OSSIZE nrows, const OSUINT8 *pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)

This function parses the next start tag and finds index of element name in descriptor table.

It used for decode "all" content model in strict mode.

Table 2.203. Parameters

pctxt	Pointer to context block structure.
tab	Elements descriptor table.
nrows	Number of descriptors in table.
pOrder	Pointer to array to receive elements order.
nOrder	Last element's index in order array.
maxOrder	Size of order array.
anyID	Identifier of xsd:any element.

Returns: . Completion status of operation:

- positive or zero value is element identifier,
- negative return value is error.

EXTERNXML int rtXmlpGetNextAllElemID16 (OSCTXT *pctxt, const OSXMLElemIDRec *tab, OSSIZE nrows, const OSUINT16 *pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)

This function parses the next start tag and finds index of element name in descriptor table.

It used for decode "all" content model in strict mode. This variant used when xsd:all has above 256 elements.

Table 2.204. Parameters

pctxt	Pointer to context block structure.
tab	Elements descriptor table.
nrows	Number of descriptors in table.
pOrder	Pointer to array to receive elements order.
nOrder	Last element's index in order array.
maxOrder	Size of order array.
anyID	Identifier of xsd:any element.

Returns: . Completion status of operation:

- positive or zero value is element identifier,
- negative return value is error.

EXTERNXML int rtXmlpGetNextAllElemID32 (OSCTXT *pctxt, const OSXMLElemIDRec *tab, OSSIZE nrows, const OSUINT32 *pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)

This function parses the next start tag and finds index of element name in descriptor table.

It used for decode "all" content model in strict mode.

Table 2.205. Parameters

pctxt	Pointer to context block structure.
tab	Elements descriptor table.
nrows	Number of descriptors in table.
pOrder	Pointer to array to receive elements order.
nOrder	Last element's index in order array.
maxOrder	Size of order array.
anyID	Identifier of xsd:any element.

Returns: . Completion status of operation:

- positive or zero value is element identifier,
- negative return value is error.

EXTERNXML void rtXmlpSetNamespaceTable (OSCTXT *pctxt, const OSUTF8CHAR *namespaceTable[], OSSIZE nmNamespaces)

Sets user namespace table.

Table 2.206. Parameters

pctxt	Pointer to context block structure.
namespaceTable	Array of namespace URI strings.
nmNamespaces	Number of namespaces in table.

EXTERNXML int rtXmlpCreateReader (OSCTXT *pctxt)

Creates pull parser reader structure within the context.

Table 2.207. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

EXTERNXML void rtXmlpHideAttributes (OSCTXT *pctxt)

Disable access to attributes.

Function used in derived types to disable repeated decode in base type.

Table 2.208. Parameters

pctxt	Pointer to context block structure.

EXTERNXML OSBOOL rtXmlpNeedDecodeAttributes (OSCTXT *pctxt)

This function checks if attributes were previously decoded.

Table 2.209. Parameters

pctxt	Pointer to context block structure.

Returns: . State of attributes:

- TRUE = attributes was not decoded,
- FALSE = attributes had been decoded.

EXTERNXML void rtXmlpMarkPos (OSCTXT *pctxt)

Save current decode position.

Table 2.210. Parameters

pctxt	Pointer to context block structure.

EXTERNXML void rtXmlpRewindToMarkedPos (OSCTXT *pctxt)

Rewind to saved decode position.

Table 2.211. Parameters

pctxt	Pointer to context block structure.

EXTERNXML void rtXmlpResetMarkedPos (OSCTXT *pctxt)

Reset saved decode position.

Table 2.212. Parameters

pctxt	Pointer to context block structure.

EXTERNXML int rtXmlpGetXSITypeAttr (OSCTXT *pctxt, const OSUTF8CHAR **ppAttrValue, OSINT16 *nsidx, OSSIZE *pLocalOffs)

This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).

Table 2.213. Parameters

pctxt	Pointer to context block structure.
ppAttrValue	A pointer to a pointer to a UTF8 character string to received the decoded XSI type QName.
nsidx	A pointer to OSINT16 value to received the decoded XSI type namespace index.
pLocalOffs	A pointer to size_t value to received the local name offset in ppAttrValue. When pLocalOffs value equal NULL function return in ppAttrValue local name only.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpGetXmlnsAttrs (OSCTXT *pctxt, OSRTDList *pNSAttrs)

This function decodes namespace attributes from start tag and adds them to the given list.

Table 2.214. Parameters

pctxt	Pointer to context block structure.

pNSAttrs	A pointer to a linked list of OSXMLNamespace structures.

Returns: . Completion status of operation:

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

EXTERNXML int rtXmlpDecXSIAttrs (OSCTXT *pctxt)

This function decodes XSI (XML Schema Instance) that may be present in any arbitrary XML element within a document.

Table 2.215. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

EXTERNXML OSBOOL rtXmlpIsEmptyElement (OSCTXT *pctxt)

Check element content: empty or not.

Table 2.216. Parameters

pctxt	Pointer to context block structure.

Returns: . Status of element content:

- TRUE = element is empty,
- FALSE = element has content.

EXTERNXML int rtXmlEncAttrC14N (OSCTXT *pctxt)

This function used only in C14 mode.

It provide atributes sorting.

Table 2.217. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

EXTERNXML struct OSXMLReader* rtXmlpGetReader (OSCTXT *pctxt)

This function fetches the XML reader structure from the context for use in low-level pull parser calls.

If the reader structure does not exist, it is created and initialized.

Table 2.218. Parameters

pctxt	Pointer to context block structure.

Returns: . Pointer to XML reader structure or NULL if an error occurred.

EXTERNXML OSBOOL rtXmlpIsLastEventDone (OSCTXT *pctxt)

Check processing status of current tag.

Table 2.219. Parameters

pctxt	Pointer to context block structure.

Returns: . Status of element content:

- TRUE = tag marked as processed,
- FALSE = tag will be processed again.

EXTERNXML int rtXmlpGetXSITypeIndex (OSCTXT *pctxt, const OSXMLItemDescr typetab[], OSSIZE typetabsiz)

This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type) and find type index in descriptor table.

Table 2.220. Parameters

pctxt	Pointer to context block structure.
typetab	XSI types descriptor table.
typetabsiz	Number of descriptors in table.

Returns: . Completion status of operation:

- positive or zero value is type index,
- negative return value is error.

EXTERNXML int rtXmlpLookupXSITypeIndex (OSCTXT *pctxt, const OSUTF8CHAR *pXsiType, OSINT16 xsiTypeIdx, const OSXMLItemDescr typetab[], OSSIZE typetabsiz)

This function find index of XSI (XML Schema Instance) type in descriptor table.

Table 2.221. Parameters

pctxt	Pointer to context block structure.
pXsiType	A pointer to XSI type name.
xsiTypeIdx	A XSI type namespace index.
typetab	XSI types descriptor table.
typetabsiz	Number of descriptors in table.

Returns: . Completion status of operation:

- positive or zero value is type index,
- negative return value is error.

EXTERNXML void rtXmlpForceDecodeAsGroup (OSCTXT *pctxt)

Disable skipping of unknown elements in optional sequence tail.

Function used in outer types to break decode on first unknown element after decoding mandatory sequence part.

Table 2.222. Parameters

pctxt	Pointer to context block structure.

EXTERNXML OSBOOL rtXmlpIsDecodeAsGroup (OSCTXT *pctxt)

This function checks if "decode as group" mode was forced.

Table 2.223. Parameters

pctxt	Pointer to context block structure.

Returns: . State of mode:

- TRUE = need decode as group,
- FALSE = normal sequence decoding.

EXTERNXML OSBOOL rtXmlpIsUTF8Encoding (OSCTXT *pctxt)

This function checks if the encoding specified in XML header is UTF-8.

Table 2.224. Parameters

pctxt	Pointer to context block structure.

Returns: . State of mode:

- TRUE = is in UTF-8 encoding,
- FALSE = is not in UTF-8 encoding.

EXTERNXML int rtXmlpReadBytes (OSCTXT *pctxt, OSOCTET *pbuf, OSSIZE nbytes)

This function reads the specified number of bytes directly from the underlying XML parser stream.

It has no effect on any of the parser state variables.

Table 2.225. Parameters

pctxt	Pointer to context block structure.
pbuf	Pointer to static buffer (byte array) to receive data. The buffer must be at least large enough to hold the requested number of bytes.
nbytes	Number of bytes to read.

Returns: . State of mode:

- TRUE = is in UTF-8 encoding,
- FALSE = is not in UTF-8 encoding.

Macro Definition Documentation

Chapter 3. Class Documentation

OSXMLDefaultHandler::ErrorInfo struct Reference

Public Attributes

- int stat
- const char * file
- int line

- ErrorInfo ( )

Member Data Documentation

OSXMLDefaultHandler::ErrorInfo::ErrorInfo ()

OSIntegerFmt struct Reference

Public Attributes

- OSINT8 integerMaxDigits
- OSBOOL signPresent

Member Data Documentation

OSRTMessageBuffer class Reference

OSXMLAnyHandler class Reference

Private Attributes

- OSRTContext mEncCtxt
- OSRTXMLString * mpAnyMsgData
- OSXSDAnyTypeClass * mpAnyTypeMsgData

- EXTXMLMETHOD void localInit (OSRTContext * pContext)

- EXTXMLMETHOD OSBOOL isEmptyElement (const OSUTF8CHAR * qname)
- EXTXMLMETHOD OSXMLAnyHandler & operator= (const OSXMLAnyHandler &)
- EXTXMLMETHOD OSXMLAnyHandler (OSXSDAnyTypeClass & msgData, OSRTContext * pContext, int level)
- EXTXMLMETHOD OSXMLAnyHandler (OSXSDAnyTypeClass & msgData, OSRTContext * pContext, const OSUTF8CHAR * elemName)
- EXTXMLMETHOD OSXMLAnyHandler (OSRTXMLString & msgData, OSRTContext * pContext, int level)
- EXTXMLMETHOD OSXMLAnyHandler (OSRTXMLString & msgData, OSRTContext * pContext, const OSUTF8CHAR * elemName)
- EXTXMLMETHOD ~OSXMLAnyHandler ( )
- virtual EXTXMLMETHOD int startElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const * attrs)

  Receive notification of the beginning of an element.
- virtual EXTXMLMETHOD int characters (const OSUTF8CHAR *const chars, OSUINT32 length)
- virtual EXTXMLMETHOD int endElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)

  Receive notification of the end of an element.

Member Data Documentation

EXTXMLMETHOD void OSXMLAnyHandler::localInit (OSRTContext *pContext)

EXTXMLMETHOD OSBOOL OSXMLAnyHandler::isEmptyElement (const OSUTF8CHAR *qname)

EXTXMLMETHOD OSXMLAnyHandler& OSXMLAnyHandler::operator= (const OSXMLAnyHandler &)

EXTXMLMETHOD OSXMLAnyHandler::OSXMLAnyHandler (OSXSDAnyTypeClass &msgData, OSRTContext *pContext, int level=0)

EXTXMLMETHOD OSXMLAnyHandler::OSXMLAnyHandler (OSXSDAnyTypeClass &msgData, OSRTContext *pContext, const OSUTF8CHAR *elemName)

EXTXMLMETHOD OSXMLAnyHandler::OSXMLAnyHandler (OSRTXMLString &msgData, OSRTContext *pContext, int level=0)

EXTXMLMETHOD OSXMLAnyHandler::OSXMLAnyHandler (OSRTXMLString &msgData, OSRTContext *pContext, const OSUTF8CHAR *elemName)

EXTXMLMETHOD OSXMLAnyHandler::~OSXMLAnyHandler ()

virtual EXTXMLMETHOD int OSXMLAnyHandler::startElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const *attrs)

Receive notification of the beginning of an element.

The Parser will invoke this method at the beginning of every element in the XML document; there will be a corresponding endElement() event for every startElement() event (even when the element is empty). All of the element's content will be reported, in order, before the corresponding endElement() event.

Table 3.1. Parameters

uri	The URI of the asscioated namespace for this element
localname	The local part of the element name
qname	The QName of this element
attrs	The attributes name/value pairs attached to the element, if any.

See also: . endElement

virtual EXTXMLMETHOD int OSXMLAnyHandler::characters (const OSUTF8CHAR *const chars, OSUINT32 length)

virtual EXTXMLMETHOD int OSXMLAnyHandler::endElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)

Receive notification of the end of an element.

The SAX parser will invoke this method at the end of every element in the XML document; there will be a corresponding startElement() event for every endElement() event (even when the element is empty).

Table 3.2. Parameters

uri	The URI of the asscioated namespace for this element
localname	The local part of the element name
qname	The QName of this element

OSXMLAnyTypeHandler class Reference

Private Attributes

- OSRTContext mEncCtxt
- OSXSDAnyTypeClass * mpAnyTypeMsgData
- void localInit (OSRTContext * pContext)

- OSBOOL isEmptyElement (const OSUTF8CHAR * qname)
- OSXMLAnyTypeHandler & operator= (const OSXMLAnyTypeHandler &)

- OSXMLAnyTypeHandler (OSXSDAnyTypeClass & msgData, OSRTContext * pContext, int level)
- OSXMLAnyTypeHandler (OSXSDAnyTypeClass & msgData, OSRTContext * pContext, const OSUTF8CHAR * elemName)
- ~OSXMLAnyTypeHandler ( )
- virtual int startElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const * attrs)

  Receive notification of the beginning of an element.
- virtual int characters (const OSUTF8CHAR *const chars, OSUINT32 length)
- virtual int endElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)

  Receive notification of the end of an element.

Member Data Documentation

void OSXMLAnyTypeHandler::localInit (OSRTContext *pContext)

OSBOOL OSXMLAnyTypeHandler::isEmptyElement (const OSUTF8CHAR *qname)

OSXMLAnyTypeHandler& OSXMLAnyTypeHandler::operator= (const OSXMLAnyTypeHandler &)

OSXMLAnyTypeHandler::OSXMLAnyTypeHandler (OSXSDAnyTypeClass &msgData, OSRTContext *pContext, int level=0)

OSXMLAnyTypeHandler::OSXMLAnyTypeHandler (OSXSDAnyTypeClass &msgData, OSRTContext *pContext, const OSUTF8CHAR *elemName)

OSXMLAnyTypeHandler::~OSXMLAnyTypeHandler ()

virtual int OSXMLAnyTypeHandler::startElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const *attrs)

Receive notification of the beginning of an element.

The Parser will invoke this method at the beginning of every element in the XML document; there will be a corresponding endElement() event for every startElement() event (even when the element is empty). All of the element's content will be reported, in order, before the corresponding endElement() event.

Table 3.3. Parameters

uri	The URI of the asscioated namespace for this element
localname	The local part of the element name
qname	The QName of this element
attrs	The attributes name/value pairs attached to the element, if any.

See also: . endElement

virtual int OSXMLAnyTypeHandler::characters (const OSUTF8CHAR *const chars, OSUINT32 length)

virtual int OSXMLAnyTypeHandler::endElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)

Receive notification of the end of an element.

The SAX parser will invoke this method at the end of every element in the XML document; there will be a corresponding startElement() event for every endElement() event (even when the element is empty).

Table 3.4. Parameters

uri	The URI of the asscioated namespace for this element
localname	The local part of the element name
qname	The QName of this element

OSXMLBase class Reference

- OSXMLBase ( )
- virtual ~OSXMLBase ( )
- virtual void release ( )

OSXMLBase::OSXMLBase ()

virtual OSXMLBase::~OSXMLBase ()

virtual void OSXMLBase::release ()=0

OSXMLBasePtr class Reference

Private Attributes

- OSXMLBase * mPtr
- OSXMLBasePtr ( )
- OSXMLBasePtr (OSXMLBase * ptr)
- ~OSXMLBasePtr ( )
- operator OSXMLBase * ( )

- OSXMLBase * operator= (OSXMLBase * ptr)

Member Data Documentation

OSXMLBasePtr::OSXMLBasePtr ()

OSXMLBasePtr::OSXMLBasePtr (OSXMLBase *ptr)

OSXMLBasePtr::~OSXMLBasePtr ()

OSXMLBasePtr::operator OSXMLBase * () const

OSXMLBase* OSXMLBasePtr::operator= (OSXMLBase *ptr)

OSXMLContentHandler class Reference

`#include <rtSaxCppParserIF.h>`

The virtual document handler interface

- virtual int characters (const OSUTF8CHAR *const chars, unsigned int length)

  Receive notification of character data.

- virtual int endElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)

  Receive notification of the end of an element.

- virtual int startElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const * attrs)

  Receive notification of the beginning of an element.

- virtual ~OSXMLContentHandler ( )

Receive notification of general document events.

virtual OSXMLContentHandler::~OSXMLContentHandler ()

OSXMLCtxtInfo struct Reference

Public Attributes

- OSFreeCtxtAppInfoPtr pFreeFunc

- OSResetCtxtAppInfoPtr pResetFunc
- OSUTF8CHAR * schemaLocation
- OSUTF8CHAR * noNSSchemaLoc
- OSUTF8CHAR * xsiTypeAttr
- OSXMLEncoding encoding
- OSRTDList namespaceList
- OSRTDList encodedNSList
- OSRTDList sortedAttrList
- OSXMLNSPfxLinkStack nsPfxLinkStack
- OSXMLNSURITable nsURITable
- OSRTMEMBUF memBuf
- OSINT32 mSaxLevel
- OSINT32 mSkipLevel
- OSUINT32 maxSaxErrors
- OSUINT32 errorsCnt
- OSUINT8 indent
- OSBOOL mbCdataProcessed
- char indentChar
- OSUINT8 soapVersion
- OSXMLFacets facets
- const OSUTF8CHAR * encodingStr
- OSXMLBOM byteOrderMark
- struct OSXMLReader * pXmlPPReader
- OSRTBuffer savedBuffer
- OSRTFLAGS savedFlags
- OSOCTET * attrsBuff
- OSSIZE attrsBuffSize
- OSSIZE attrStartPos

Member Data Documentation

OSXMLDecodeBuffer class Reference

`#include <OSXMLDecodeBuffer.h>`

Protected Attributes

- OSRTInputStream * mpInputStream

  Input source for message to be decoded.

- OSBOOL mbOwnStream

  This is set to true if this object creates the underlying stream object.

- OSXMLDecodeBuffer (const char * xmlFile)

  This version of the OSXMLDecodeBuffer constructor takes a name of a file that contains XML data to be decoded and constructs a buffer.

- OSXMLDecodeBuffer (const OSOCTET * msgbuf, size_t bufsiz)

  This version of the OSXMLDecodeBuffer constructor takes parameters describing a message in memory to be decoded and constructs a buffer.

- OSXMLDecodeBuffer (OSRTInputStream & inputStream)

  This version of the OSXMLDecodeBuffer constructor takes a reference to the OSInputStream object.

- ~OSXMLDecodeBuffer ( )

- EXTXMLMETHOD int decodeXML (OSXMLReaderClass * pReader)

  This method decodes an XML message associated with this buffer.

- virtual EXTXMLMETHOD int init ( )

  This method initializes the decode message buffer.

- EXTXMLMETHOD OSBOOL isWellFormed ( )

  This method determines if an XML fragment is well-formed.

- EXTXMLMETHOD int parseElementName (OSUTF8CHAR ** ppName)

  This method parses the initial tag from an XML message.

- EXTXMLMETHOD int parseElemQName (OSXMLQName * pQName)

  This method parses the initial tag from an XML message.

- EXTXMLMETHOD OSUINT32 setMaxErrors (OSUINT32 maxErrors)

This method sets the maximum number of errors returned by the SAX parser.

- virtual OSBOOL isA (Type bufferType)

  This is a virtual method that must be overridden by derived classes to allow identification of the class.

The OSXMLDecodeBuffer class is derived from the OSXMLMessageBuffer base class.

Detailed Description

It contains variables and methods specific to decoding XML messages. It is used to manage an input buffer or stream containing a message to be decoded.

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

Definition at line 45 of file OSXMLDecodeBuffer.h

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

- OSXMLDecodeBuffer.h

Member Data Documentation

OSBOOL OSXMLDecodeBuffer::mbOwnStream

In this case, the stream will be deleted in the object's destructor.

Definition at line 57 of file OSXMLDecodeBuffer.h

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

- OSXMLDecodeBuffer.h

OSXMLDecodeBuffer::OSXMLDecodeBuffer (const char *xmlFile)

This version of the OSXMLDecodeBuffer constructor takes a name of a file that contains XML data to be decoded and constructs a buffer.

Table 3.5. Parameters

xmlFile	A pointer to name of file to be decoded.

OSXMLDecodeBuffer::OSXMLDecodeBuffer (const OSOCTET *msgbuf, size_t bufsiz)

This version of the OSXMLDecodeBuffer constructor takes parameters describing a message in memory to be decoded and constructs a buffer.

Table 3.6. Parameters

msgbuf	A pointer to a buffer containing an XML message.
bufsiz	Size of the message buffer.

OSXMLDecodeBuffer::OSXMLDecodeBuffer (OSRTInputStream &inputStream)

This version of the OSXMLDecodeBuffer constructor takes a reference to the OSInputStream object.

The stream is assumed to have been previuously initialized to point at an encoded XML message.

Table 3.7. Parameters

inputStream	reference to the OSInputStream object

OSXMLDecodeBuffer::~OSXMLDecodeBuffer ()

EXTXMLMETHOD int OSXMLDecodeBuffer::decodeXML (OSXMLReaderClass *pReader)

This method decodes an XML message associated with this buffer.

Returns: . stat Status of the operation. Possible values are 0 if successful or one of the negative error status codes defined in Appendix A of the C/C++ runtime Common Functions Reference Manual.

Table 3.8. Parameters

pReader	Pointer to OSXMLReaderClass object.

Returns: . Completion status.

virtual EXTXMLMETHOD int OSXMLDecodeBuffer::init ()

This method initializes the decode message buffer.

Returns: . Completion status of operation:

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

EXTXMLMETHOD OSBOOL OSXMLDecodeBuffer::isWellFormed ()

This method determines if an XML fragment is well-formed.

The stream is reset to the start position following the test.

Returns: . Boolean result true if fragment well-formed; false otherwise.

EXTXMLMETHOD int OSXMLDecodeBuffer::parseElementName (OSUTF8CHAR **ppName)

This method parses the initial tag from an XML message.

If the tag is a QName, only the local part of the name is returned.

Table 3.9. Parameters

ppName	Pointer to a pointer to receive decoded UTF-8 string. Dynamic memory is allocated for the variable using the rtxMemAlloc function.

Returns: . Completion status of operation:

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

EXTXMLMETHOD int OSXMLDecodeBuffer::parseElemQName (OSXMLQName *pQName)

This method parses the initial tag from an XML message.

Table 3.10. Parameters

pQName	Pointer to a QName structure to receive parsed name prefix and local name. Dynamic memory is allocated for both name parts using the rtxMemAlloc function.

Returns: . Completion status of operation:

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

EXTXMLMETHOD OSUINT32 OSXMLDecodeBuffer::setMaxErrors (OSUINT32 maxErrors)

This method sets the maximum number of errors returned by the SAX parser.

Table 3.11. Parameters

maxErrors	The desired number of maximum errors.

Returns: . The previously set maximum number of errors.

virtual OSBOOL OSXMLDecodeBuffer::isA (Type bufferType)

This is a virtual method that must be overridden by derived classes to allow identification of the class.

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

Table 3.12. Parameters

bufferType	Enumerated identifier specifying a derived class. This type is defined as a public access type in the OSRTMessageBufferIF base interface. Possible values include BEREncode, BERDecode, PEREncode, PERDecode, XMLEncode, and XMLDecode.

Returns: . Boolean result of the match operation. True if the `bufferType` argument is `XMLDecode`. argument.

OSXMLDefaultHandler class Reference

`#include <rtSaxCppParser.h>`

Classes

- struct OSXMLDefaultHandler::ErrorInfo

Protected Attributes

- OSRTCtxtPtr mpContext
- const OSUTF8CHAR * mpElemName
- OSINT32 mLevel
- OSINT16 mStartLevel
- OSINT16 mReqElemCnt
- OSINT16 mCurrElemIdx
- OSINT16 mState
- struct EXTXMLCLASS OSXMLDefaultHandler::ErrorInfo errorInfo

- OSXMLDefaultHandler (OSRTContext * pContext, const OSUTF8CHAR * elemName, OSINT32 level)

- virtual ~OSXMLDefaultHandler ( )
- virtual EXTXMLMETHOD int startElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const * attrs)

  Receive notification of the beginning of an element.
- virtual EXTXMLMETHOD int characters (const OSUTF8CHAR *const chars, unsigned int length)

  Receive notification of character data.
- virtual EXTXMLMETHOD int endElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)

  Receive notification of the end of an element.
- virtual EXTXMLMETHOD void startDocument ( )
- virtual EXTXMLMETHOD void endDocument ( )
- virtual EXTXMLMETHOD int finalize ( )
- virtual EXTXMLMETHOD void resetErrorInfo ( )
- virtual EXTXMLMETHOD void setErrorInfo (int status, const char * file, int line)
- virtual EXTXMLMETHOD int getErrorInfo (int * status, const char ** file, int * line)
- OSINT16 getState ( )

  This method returns the current state of the decoding process.
- virtual void init (int level)
- void setElemName (const OSUTF8CHAR * elemName)
- OSBOOL isComplete ( )
- EXTXMLMETHOD void traceStartElement (const char * funcName, const OSUTF8CHAR * localName)
- EXTXMLMETHOD void traceEndElement (const char * funcName, const OSUTF8CHAR * localName)

This class is derived from the SAX class DefaultHandler base class.

Detailed Description

It contains variables and methods specific to decoding XML messages. It is used to intercept standard SAX parser events, such as startElement, characters, endElement. This class is used as a base class for XBinder generated global element control classes (<elem>_CC).

Definition at line 59 of file rtSaxCppParser.h

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

- rtSaxCppParser.h

Member Data Documentation

OSXMLDefaultHandler::OSXMLDefaultHandler (OSRTContext *pContext, const OSUTF8CHAR *elemName=0, OSINT32 level=0)

virtual OSXMLDefaultHandler::~OSXMLDefaultHandler ()

virtual EXTXMLMETHOD int OSXMLDefaultHandler::startElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const *attrs)

Receive notification of the beginning of an element.

The Parser will invoke this method at the beginning of every element in the XML document; there will be a corresponding endElement() event for every startElement() event (even when the element is empty). All of the element's content will be reported, in order, before the corresponding endElement() event.

Table 3.13. Parameters

uri	The URI of the asscioated namespace for this element
localname	The local part of the element name
qname	The QName of this element
attrs	The attributes name/value pairs attached to the element, if any.

See also: . endElement

virtual EXTXMLMETHOD int OSXMLDefaultHandler::characters (const OSUTF8CHAR *const chars, unsigned int length)

Receive notification of character data.

The Parser will call this method to report each chunk of character data. SAX parsers may return all contiguous character data in a single chunk, or they may split it into several chunks; however, all of the characters in any single event must come from the same external entity, so that the Locator provides useful information.

Table 3.14. Parameters

chars	The characters from the XML document.

length	The length of chars.

virtual EXTXMLMETHOD int OSXMLDefaultHandler::endElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)

Receive notification of the end of an element.

The SAX parser will invoke this method at the end of every element in the XML document; there will be a corresponding startElement() event for every endElement() event (even when the element is empty).

Table 3.15. Parameters

uri	The URI of the asscioated namespace for this element
localname	The local part of the element name
qname	The QName of this element

virtual EXTXMLMETHOD void OSXMLDefaultHandler::startDocument ()

virtual EXTXMLMETHOD void OSXMLDefaultHandler::endDocument ()

virtual EXTXMLMETHOD int OSXMLDefaultHandler::finalize ()

virtual EXTXMLMETHOD void OSXMLDefaultHandler::resetErrorInfo ()

virtual EXTXMLMETHOD void OSXMLDefaultHandler::setErrorInfo (int status, const char *file=0, int line=0)

virtual EXTXMLMETHOD int OSXMLDefaultHandler::getErrorInfo (int *status, const char **file, int *line)

OSINT16 OSXMLDefaultHandler::getState ()

This method returns the current state of the decoding process.

Returns: . The state of the decoding process as type OSXMLState. Can be XMLINIT, XMLSTART, XMLDATA, or XMLEND.

virtual void OSXMLDefaultHandler::init (int level=0)

void OSXMLDefaultHandler::setElemName (const OSUTF8CHAR *elemName)

OSBOOL OSXMLDefaultHandler::isComplete ()

EXTXMLMETHOD void OSXMLDefaultHandler::traceStartElement (const char *funcName, const OSUTF8CHAR *localName)

EXTXMLMETHOD void OSXMLDefaultHandler::traceEndElement (const char *funcName, const OSUTF8CHAR *localName)

OSXMLDefaultHandlerIF class Reference

```
#include <rtSaxCppParserIF.h>
```

- virtual ~OSXMLDefaultHandlerIF ( )
- virtual void startDocument ( )
- virtual void endDocument ( )
- virtual int finalize ( )

This class is derived from the SAX class DefaultHandler base class.

Detailed Description

It contains variables and methods specific to decoding XML messages. It is used to intercept standard SAX parser events, such as startElement, characters, endElement. This class is used as a base class for XBinder generated global element control classes (<elem>_CC).

Definition at line 262 of file rtSaxCppParserIF.h

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

- rtSaxCppParserIF.h

virtual OSXMLDefaultHandlerIF::~OSXMLDefaultHandlerIF ()

virtual void OSXMLDefaultHandlerIF::startDocument ()=0

virtual void OSXMLDefaultHandlerIF::endDocument ()=0

virtual int OSXMLDefaultHandlerIF::finalize ()=0

OSXMLDefaultHandlerPtr class Reference

Private Attributes

- OSXMLDefaultHandler * mPtr

- OSXMLDefaultHandlerPtr ( )
- OSXMLDefaultHandlerPtr (OSXMLDefaultHandler * ptr)
- ~OSXMLDefaultHandlerPtr ( )
- operator OSXMLDefaultHandler * ( )
- operator const OSXMLDefaultHandler * ( )
- OSXMLDefaultHandler * operator-> ( )
- OSXMLDefaultHandler * operator= (OSXMLDefaultHandler * ptr)
- int operator== (const OSXMLDefaultHandler * ptr)
- int operator!= (const OSXMLDefaultHandler * ptr)
- int operator! ( )

-
-

Member Data Documentation

OSXMLDefaultHandlerPtr::OSXMLDefaultHandlerPtr ()

OSXMLDefaultHandlerPtr::OSXMLDefaultHandlerPtr (OSXMLDefaultHandler *ptr)

OSXMLDefaultHandlerPtr::~OSXMLDefaultHandlerPtr ()

OSXMLDefaultHandlerPtr::operator OSXMLDefault-Handler * ()

OSXMLDefaultHandlerPtr::operator const OSXMLDe-faultHandler * () const

OSXMLDefaultHandler* OSXMLDefaultHandlerPtr::operator-> () const

OSXMLDefaultHandler* OSXMLDefaultHandlerPtr::operator= (OSXMLDefault-Handler *ptr)

int OSXMLDefaultHandlerPtr::operator== (const OSXMLDefaultHandler *ptr) const

int OSXMLDefaultHandlerPtr::operator!= (const OSXMLDefaultHandler *ptr) const

int OSXMLDefaultHandlerPtr::operator! () const

OSXMLElemIDRec struct Reference

Public Attributes

- OSXMLElemDescr descr
- OSUINT16 id

Member Data Documentation

OSXMLEncodeBase class Reference

`#include <OSXMLMessageBuffer.h>`

- EXTXMLMETHOD OSXMLEncodeBase (OSRTContext * pContext)

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

- EXTXMLMETHOD int encodeAttr (const OSUTF8CHAR * name, const OSUTF8CHAR * value)

  This function encodes an attribute in which the name and value are given as null-terminated UTF-8 strings.

- EXTXMLMETHOD int encodeText (const OSUTF8CHAR * value)

  This method encodes XML textual content.

- EXTXMLMETHOD int endDocument ( )

  This method ends an XML document by flushing any remaining data to the stream.

- EXTXMLMETHOD int endElement (const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This method encodes an end element tag value (\</elemName\>).

- EXTXMLMETHOD int startDocument ( )

  This method writes information to start an XML document to the encode stream.

- EXTXMLMETHOD int startElement (const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, OSRTDList * pNSAttrs, OSBOOL terminate)

  This method writes information to start an XML element to the encode stream.

- EXTXMLMETHOD int termStartElement ( )

  This metod terminates a currently open XML start element by adding either a '>' or '/>' (if empty) terminator.

OSXMLEncodeBase is a base class for the XML encode buffer and stream classes, OSXMLEncodeBuffer and OSXMLEncodeStream.

EXTXMLMETHOD OSXMLEncodeBase::OSXMLEncodeBase (OSRTContext *pContext=0)

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

Table 3.16. Parameters

pContext	Pointer to a context to use. If NULL, new context will be allocated.

EXTXMLMETHOD int OSXMLEncodeBase::encodeAttr (const OSUTF8CHAR *name, const OSUTF8CHAR *value)

This function encodes an attribute in which the name and value are given as null-terminated UTF-8 strings.

Table 3.17. Parameters

name	Attribute name.
value	UTF-8 string value to be encoded.

Returns: . Completion status of operation:

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

EXTXMLMETHOD int OSXMLEncodeBase::encodeText (const OSUTF8CHAR *value)

This method encodes XML textual content.

XML metadata characters such as '<' are escaped. The input value is specified in UTF-8 character format but may be transformed if a different character encoding is enabled.

Table 3.18. Parameters

value	UTF-8 string value to be encoded.

Returns: . Completion status of operation:

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

EXTXMLMETHOD int OSXMLEncodeBase::endDocument ()

This method ends an XML document by flushing any remaining data to the stream.

EXTXMLMETHOD int OSXMLEncodeBase::endElement (const OSUTF8CHAR *elemName, OSXMLNamespace *pNS=0)

This method encodes an end element tag value (\</elemName\>).

Table 3.19. Parameters

elemName	XML element name.
pNS	XML namespace information (prefix and URI).

Returns: . Completion status of operation:

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

EXTXMLMETHOD int OSXMLEncodeBase::startDocument ()

This method writes information to start an XML document to the encode stream.

This includes the XML header declaration.

EXTXMLMETHOD int OSXMLEncodeBase::startElement (const OSUTF8CHAR *elemName, OSXMLNamespace *pNS=0, OSRTDList *pNSAttrs=0, OSBOOL terminate=FALSE)

This method writes information to start an XML element to the encode stream.

It can leave the element open so that attributes can be added.

Table 3.20. Parameters

elemName	XML element name.
pNS	XML namespace information (prefix and URI). If the prefix is NULL, this method will search the context's namespace stack for a prefix to use.
pNSAttrs	List of namespace attributes to be added to element.
terminate	Add closing '>' character.

Returns: . Completion status of operation:

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

EXTXMLMETHOD int OSXMLEncodeBase::termStartElement ()

This metod terminates a currently open XML start element by adding either a '>' or '/>' (if empty) terminator.

It will also add XSI attributes to the element. Note that is important to use this method to terminate the element rather than writng a closing angle bracket text to the stream directly due to the way state is maintained in the context.

Returns: . Completion status of operation:

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

OSXMLEncodeBuffer class Reference

`#include <OSXMLEncodeBuffer.h>`

- OSXMLEncodeBuffer (OSRTContext * pContext)

- EXTXMLMETHOD OSXMLEncodeBuffer ( )

  Default constructor.

- EXTXMLMETHOD OSXMLEncodeBuffer (OSOCTET * pMsgBuf, size_t msgBufLen)

  This constructor allows a static message buffer to be specified to receive the encoded message.

- int addXMLHeader (const OSUTF8CHAR * version, const OSUTF8CHAR * encoding, OSBOOL newLine)

  This method adds XML header text to the output buffer with the given version number and encoding attributes.

- EXTXMLMETHOD int addXMLText (const OSUTF8CHAR * text)

  This method adds encoded XML text to the encode buffer.

- virtual size_t getMsgLen ( )

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

- virtual EXTXMLMETHOD int init ( )

  This method reinitializes the encode buffer to allow a new message to be encoded.

- virtual OSBOOL isA (Type bufferType)

  This is a virtual method that must be overridden by derived classes to allow identification of the class.

- void nullTerminate ( )

  This method adds a null-terminator character ('\0') at the current buffer position.

- EXTXMLMETHOD void setFragment (OSBOOL value)

  This method sets a flag indicating that the data is to be encoded as ax XML fragment instead of as a complete XML document (i.e.

- virtual EXTXMLMETHOD long write (const char * filename)

  This method writes the encoded message to the given file.

- virtual EXTXMLMETHOD long write (FILE * fp)

  This version of the write method writes to a file that is specified by a FILE pointer.

The OSXMLEncodeBuffer class is derived from the OSXMLEncodeBase class.

Detailed Description

It contains variables and methods specific to encoding XML messages. It is used to manage the buffer into which a message is to be encoded.

Definition at line 38 of file OSXMLEncodeBuffer.h

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

- OSXMLEncodeBuffer.h

OSXMLEncodeBuffer::OSXMLEncodeBuffer (OSRTContext *pContext)

EXTXMLMETHOD OSXMLEncodeBuffer::OSXMLEncodeBuffer ()

Default constructor.

EXTXMLMETHOD OSXMLEncodeBuffer::OSXMLEncodeBuffer (OSOCTET *pMsgBuf, size_t msgBufLen)

This constructor allows a static message buffer to be specified to receive the encoded message.

Table 3.21. Parameters

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

int OSXMLEncodeBuffer::addXMLHeader (const OSUTF8CHAR *version=OSUTF8("1.0"), const OSUTF8CHAR *encoding=OSUTF8(OSXMLHDRUTF8), OSBOOL newLine=TRUE)

This method adds XML header text to the output buffer with the given version number and encoding attributes.

Table 3.22. Parameters

version	XML version (default is 1.0)
encoding	Character encoding (default is UTF-8)

newLine	Add newline char at end of header

Returns: . Zero if success or negative error code.

EXTXMLMETHOD int OSXMLEncodeBuffer::addXMLText (const OSUTF8CHAR *text)

This method adds encoded XML text to the encode buffer.

It is assumed that the user has already processed the text to do character escaping, etc.. The text is copied directly to the buffer as-is.

Table 3.23. Parameters

text	Encoded XML text to be added to the buffer.

Returns: . Zero if success or negative error code.

virtual size_t OSXMLEncodeBuffer::getMsgLen ()

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

Returns: . Length of the XML message encapsulated within this buffer object.

virtual EXTXMLMETHOD int OSXMLEncodeBuffer::init ()

This method reinitializes the encode buffer to allow a new message to be encoded.

This makes it possible to reuse one message buffer object in a loop to encode multiple messages. After this method is called, any previously encoded message in the buffer will be overwritten on the next encode call.

virtual OSBOOL OSXMLEncodeBuffer::isA (Type bufferType)

This is a virtual method that must be overridden by derived classes to allow identification of the class.

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

Table 3.24. Parameters

bufferType	Enumerated identifier specifying a derived class. This type is defined as a public access type in the OSRTMessageBufferIF base interface. Possible values include BEREncode, BERDecode, PEREncode, PERDecode, XMLEncode, and XMLDecode.

Returns: . Boolean result of the match operation. True if the `bufferType` argument is `XMLEncode`. argument.

void OSXMLEncodeBuffer::nullTerminate ()

This method adds a null-terminator character ('\0') at the current buffer position.

EXTXMLMETHOD void OSXMLEncodeBuffer::setFragment (OSBOOL value=TRUE)

This method sets a flag indicating that the data is to be encoded as ax XML fragment instead of as a complete XML document (i.e.

an XML header will not be added).

virtual EXTXMLMETHOD long OSXMLEncodeBuffer::write (const char *filename)

This method writes the encoded message to the given file.

Table 3.25. Parameters

filename	The name of file to which the encoded message will be written.

Returns: . Number of octets actually written. This value may be less than the actual message length if an error occurs.

virtual EXTXMLMETHOD long OSXMLEncodeBuffer::write (FILE *fp)

This version of the write method writes to a file that is specified by a FILE pointer.

Table 3.26. Parameters

fp	Pointer to FILE structure to which the encoded message will be written.

Returns: . Number of octets actually written. This value may be less than the actual message length if an error occurs.

OSXMLEncodeStream class Reference

`#include <OSXMLEncodeStream.h>`

Protected Attributes

- OSRTOutputStream * mpStream

  A pointer to an OSRTOutputStream object.

- OSBOOL mbOwnStream

  TRUE if the OSXMLEncodeStream object will close and free the stream in the destructor.

- EXTXMLMETHOD OSXMLEncodeStream (OSRTOutputStream & outputStream)

This version of the OSXMLEncodeStream constructor takes a reference to the OSOutputStream object.

- OSXMLEncodeStream (OSRTOutputStream * pOutputStream, OSBOOL ownStream)

  This version of the OSXMLEncodeStream constructor takes a pointer to the OSRTOutputStream object.

- ~OSXMLEncodeStream ( )
- virtual EXTXMLMETHOD int init ( )

  This method reinitializes the encode stream to allow a new message to be encoded.

- virtual OSBOOL isA (Type bufferType)

  This is a virtual method that must be overridden by derived classes to allow identification of the class.

- virtual const OSOCTET * getMsgPtr ( )

  This is a virtual method that must be overridden by derived classes to allow access to the stored message.

- OSRTOutputStream * getStream ( )

  This method returns the output stream associated with the object.

The OSXMLEncodeStream class is derived from the OSXMLEncodeBase class.

Detailed Description

It contains variables and methods specific to streaming encoding XML messages. It is used to manage the stream into which a message is to be encoded.

Definition at line 40 of file OSXMLEncodeStream.h

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

- OSXMLEncodeStream.h

Member Data Documentation

OSRTOutputStream* OSXMLEncodeStream::mpStream

Definition at line 43 of file OSXMLEncodeStream.h

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

- OSXMLEncodeStream.h

OSBOOL OSXMLEncodeStream::mbOwnStream

Definition at line 47 of file OSXMLEncodeStream.h

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

- OSXMLEncodeStream.h

EXTXMLMETHOD OSXMLEncodeStream::OSXMLEncodeStream (OSRTOutputStream &outputStream)

This version of the OSXMLEncodeStream constructor takes a reference to the OSOutputStream object.

The stream is assumed to have been previously initialized.

Table 3.27. Parameters

outputStream	reference to the OSOutputStream object

OSXMLEncodeStream::OSXMLEncodeStream (OSRTOutputStream *pOutputStream, OSBOOL ownStream=TRUE)

This version of the OSXMLEncodeStream constructor takes a pointer to the OSRTOutputStream object.

The stream is assumed to have been previously initialized. If ownStream is set to TRUE, then stream will be closed and freed in the destructor.

Table 3.28. Parameters

pOutputStream	reference to the OSOutputStream object
ownStream	set ownership for the passed stream object.

OSXMLEncodeStream::~OSXMLEncodeStream ()

virtual EXTXMLMETHOD int OSXMLEncodeStream::init ()

This method reinitializes the encode stream to allow a new message to be encoded.

This makes it possible to reuse one stream object in a loop to encode multiple messages.

virtual OSBOOL OSXMLEncodeStream::isA (Type buffer-Type)

This is a virtual method that must be overridden by derived classes to allow identification of the class.

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

Table 3.29. Parameters

bufferType	Enumerated identifier specifying a derived class. This type is defined as a public access type in the OSRTMessageBufferIF base interface. Possible values include BEREncode, BERDecode, PEREncode, PERDecode, XMLEncode, and XMLDecode.

Returns: . Boolean result of the match operation. True if the `bufferType` argument is `XMLEncode` or `Stream`.

virtual const OSOCTET* OSXMLEncodeStream::getMsgPtr ()

This is a virtual method that must be overridden by derived classes to allow access to the stored message.

The base class implementation returns a null value.

Returns: . A pointer to the stored message.

OSRTOutputStream* OSXMLEncodeStream::getStream () const

This method returns the output stream associated with the object.

Returns: . A pointer to the output stream.

OSXMLErrorHandler class Reference

The error handler interface

- virtual void warning ( )

  Receive notification of a warning.

- virtual void error ( )

  Receive notification of a recoverable error.

- virtual void fatalError ( )

  Receive notification of a non-recoverable error.

- virtual void resetErrors ( )

  Reset the Error handler object on its reuse.

- virtual ~OSXMLErrorHandler ( )

virtual OSXMLErrorHandler::~OSXMLErrorHandler ()

OSXMLErrorInfo class Reference

- virtual ~OSXMLErrorInfo ( )
- virtual void resetErrorInfo ( )

- virtual void setErrorInfo (int status, const char * file, int line)
- virtual int getErrorInfo (int * status, const char ** file, int * line)

virtual OSXMLErrorInfo::~OSXMLErrorInfo ()

virtual void OSXMLErrorInfo::resetErrorInfo ()=0

virtual void OSXMLErrorInfo::setErrorInfo (int status, const char *file=0, int line=0)=0

virtual int OSXMLErrorInfo::getErrorInfo (int *status, const char **file, int *line)=0

OSXMLFacets struct Reference

Public Attributes

- int totalDigits
- int fractionDigits

Member Data Documentation

OSXMLGroupDesc struct Reference

```
#include <osrtxml.h>
```

Public Attributes

- int row
- int num
- int anyCase

OSXMLGroupDesc describes how entries in an OSXMLElemIDRec array make up a group.

Detailed Description

Here, "group" means a set of elements, any of which may be matched next. This does not correspond directly to an XSD group.

For example, if elementA is optional and followed by non-optional elementB, then there will be a group that contains both elements. There will also be a group that contains only elementB; this will be the group of interest after elementA is matched.

Definition at line 140 of file osrtxml.h

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

- osrtxml.h

Member Data Documentation

OSXMLItemDescr struct Reference

Public Attributes

- OSXMLStrFragment localName
- OSINT16 nsidx

Member Data Documentation

OSXMLMessageBuffer class Reference

```
#include <OSXMLMessageBuffer.h>
```

- EXTXMLMETHOD OSXMLMessageBuffer (Type bufferType, OSRTContext * pContext)

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

- virtual EXTXMLMETHOD void * getAppInfo ( )

  The getAppInfo method returns the pointer to application context information.

- EXTXMLMETHOD int getIndent ( )

  This method returns current XML output indent value.

- EXTXMLMETHOD int getIndentChar ( )

  This method returns current XML output indent character value (default is space).

- EXTXMLMETHOD OSBOOL getWriteBOM ( )

  This function returns whether writing the Unicode BOM is currently enabled or disabled.

- virtual EXTXMLMETHOD void setNamespace (const OSUTF8CHAR * prefix, const OSUTF8CHAR * uri, OSRTDList * pNSAttrs)

  This method sets a namespace in the context namespace list.

- virtual EXTXMLMETHOD void setAppInfo (void * pXMLInfo)

  This method sets application specific context information within the common context structure.

- EXTXMLMETHOD void setFormatting (OSBOOL doFormatting)

  This method sets XML output formatting to the given value.

- EXTXMLMETHOD void setIndent (OSUINT8 indent)

This method sets XML output indent to the given value.

- EXTXMLMETHOD void setIndentChar (char indentChar)

  This method sets XML output indent character to the given value.

- EXTXMLMETHOD void setWriteBOM (OSBOOL write)

  This method sets whether to write the Unicode byte order mark before the XML header.

The XML message buffer class is derived from the OSMessageBuffer base class.

Detailed Description

It is the base class for the OSXMLEncodeBuffer and OSXMLDecodeBuffer classes. It contains variables and methods specific to encoding or decoding XML messages. It is used to manage the buffer into which a message is to be encoded or decoded.

Definition at line 42 of file OSXMLMessageBuffer.h

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

- OSXMLMessageBuffer.h

EXTXMLMETHOD OSXMLMessageBuffer::OSXMLMessageBuffer (Type bufferType, OSRTContext *pContext=0)

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

Table 3.30. Parameters

bufferType	Type of message buffer that is being created (for example, XMLEncode or XMLDecode).
pContext	Pointer to a context to use. If NULL, new context will be allocated.

virtual EXTXMLMETHOD void* OSXMLMessageBuffer::getAppInfo ()

The getAppInfo method returns the pointer to application context information.

EXTXMLMETHOD int OSXMLMessageBuffer::getIndent ()

This method returns current XML output indent value.

Returns: . Current indent value (>= 0) if OK, negative status code if error.

EXTXMLMETHOD int OSXMLMessageBuffer::getIndentChar ()

This method returns current XML output indent character value (default is space).

Returns: . Current indent character (> 0) if OK, negative status code if error.

EXTXMLMETHOD OSBOOL OSXMLMessageBuffer::getWriteBOM ()

This function returns whether writing the Unicode BOM is currently enabled or disabled.

Returns: . TRUE if writing BOM is enabled, FALSE otherwise.

virtual EXTXMLMETHOD void OSXMLMessageBuffer::setNamespace (const OSUTF8CHAR *prefix, const OSUTF8CHAR *uri, OSRTDList *pNSAttrs=0)

This method sets a namespace in the context namespace list.

If the given namespace URI does not exist in the list, the namespace is added. If the URI is found, the value of the namespace prefix will be changed to the given prefix.

Table 3.31. Parameters

prefix	Namespace prefix
uri	Namespace URI
pNSAttrs	Namespace list to which namespace is to be added

virtual EXTXMLMETHOD void OSXMLMessageBuffer::setAppInfo (void *pXMLInfo)

This method sets application specific context information within the common context structure.

For XML encoding/decoding, this is a structure of type *OSXMLCtxtInfo*.

Table 3.32. Parameters

pXMLInfo	Pointer to context information.

EXTXMLMETHOD void OSXMLMessageBuffer::setFormatting (OSBOOL doFormatting)

This method sets XML output formatting to the given value.

If TRUE (the default), the XML document is formatted with indentation and newlines. If FALSE, all whitespace between elements is suppressed. Turning formatting off can provide more compressed documents and also a more canonical representation which is important for security applications.

Table 3.33. Parameters

doFormatting	Boolean value indicating if formatting is to be done

Returns: . Status of operation: 0 if OK, negative status code if error.

EXTXMLMETHOD void OSXMLMessageBuffer::setIndent (OSUINT8 indent)

This method sets XML output indent to the given value.

Table 3.34. Parameters

indent	Number of spaces per indent. Default is 3.

EXTXMLMETHOD void OSXMLMessageBuffer::setIndentChar (char indentChar)

This method sets XML output indent character to the given value.

Table 3.35. Parameters

indentChar	Indent character. Default is space.

EXTXMLMETHOD void OSXMLMessageBuffer::setWriteBOM (OSBOOL write)

This method sets whether to write the Unicode byte order mark before the XML header.

Table 3.36. Parameters

write	TRUE if BOM should be written, FALSE otherwise.

OSXMLNameFragments struct Reference

Public Attributes

- OSXMLStrFragment mQName
- OSXMLStrFragment mLocalName
- OSXMLStrFragment mPrefix

Member Data Documentation

OSXMLNamespace class Reference

OSXMLNamespaceClass class Reference

```
#include <rtXmlCppNamespace.h>
```

- OSXMLNamespaceClass ( )

  The default constructor sets the namespace prefix and URI values to empty values.

- ~OSXMLNamespaceClass ( )

  The destructor deletes the prefix and uri string variables.

- OSXMLNamespaceClass (const OSUTF8CHAR * nsPrefix, const OSUTF8CHAR * nsURI)

  The parameterized constructor sets the namespace prefix and URI values to the given values.

- OSXMLNamespaceClass (const OSUTF8CHAR * nsPrefix, size_t nsPrefixBytes, const OSUTF8CHAR * nsURI, size_t nsURIBytes)

  The parameterized constructor sets the namespace prefix and URI values to the given values.

- OSXMLNamespaceClass (const OSXMLNamespaceClass & o)

  The copy constructor make a deep-copy of the prefix and URI values.

- const OSUTF8CHAR * getPrefix ( )

  This method is used to get the namespace prefix value.

- const OSUTF8CHAR * getURI ( )

  This method is used to get the namespace URI value.

- void setPrefix (const OSUTF8CHAR * nsPrefix)

  This method is used to set the namespace prefix value.

- void setURI (const OSUTF8CHAR * nsURI)

  This method is used to set the namespace URI value.

This class is used to hold an XML namespace prefix to URI mapping.

OSXMLNamespaceClass::OSXMLNamespaceClass ()

The default constructor sets the namespace prefix and URI values to empty values.

OSXMLNamespaceClass::~OSXMLNamespaceClass ()

The destructor deletes the prefix and uri string variables.

OSXMLNamespaceClass::OSXMLNamespaceClass (const OSUTF8CHAR *nsPrefix, const OSUTF8CHAR *nsURI)

The parameterized constructor sets the namespace prefix and URI values to the given values.

A deep copy of the values is done.

Table 3.37. Parameters

nsPrefix	Namespace prefix value.

nsURI	Namespace URI value.

OSXMLNamespaceClass::OSXMLNamespaceClass (const OSUTF8CHAR *nsPrefix, size_t nsPrefixBytes, const OSUTF8CHAR *nsURI, size_t nsURIBytes)

The parameterized constructor sets the namespace prefix and URI values to the given values.

A deep copy of the values is done.

Table 3.38. Parameters

nsPrefix	Namespace prefix value.
nsPrefixBytes	Namespace prefix value size in bytes.
nsURI	Namespace URI value.
nsURIBytes	Namespace URI value size in bytes.

OSXMLNamespaceClass::OSXMLNamespaceClass (const OSXMLNamespaceClass &o)

The copy constructor make a deep-copy of the prefix and URI values.

const OSUTF8CHAR* OSXMLNamespaceClass::getPrefix () const

This method is used to get the namespace prefix value.

const OSUTF8CHAR* OSXMLNamespaceClass::getURI () const

This method is used to get the namespace URI value.

void OSXMLNamespaceClass::setPrefix (const OSUTF8CHAR *nsPrefix)

This method is used to set the namespace prefix value.

void OSXMLNamespaceClass::setURI (const OSUTF8CHAR *nsURI)

This method is used to set the namespace URI value.

OSXMLParserCtxt class Reference

Private Attributes

- OSRTCtxtPtr mpContext

- OSXMLParserCtxt (OSRTContext * pContext)
- virtual EXTXMLMETHOD OSRTInputStreamIF * createInputStream ( )
- virtual EXTXMLMETHOD OSRTInputStreamIF * createFileInputStream (const char *const filename)
- virtual EXTXMLMETHOD OSRTInputStreamIF * createMemoryInputStream (OSOCTET * pMemBuf, size_t bufSize)
- virtual EXTXMLMETHOD OSCTXT * getContext ( )
- virtual EXTXMLMETHOD const OSUTF8CHAR * parseQName (const OSUTF8CHAR *const qname)

Member Data Documentation

OSXMLParserCtxt::OSXMLParserCtxt (OSRTContext *pContext)

virtual EXTXMLMETHOD OSRTInputStreamIF* OSXMLParserCtxt::createInputStream ()

virtual EXTXMLMETHOD OSRTInputStreamIF* OSXMLParserCtxt::createFileInputStream (const char *const filename)

virtual EXTXMLMETHOD OSRTInputStreamIF* OSXMLParserCtxt::createMemoryInputStream (OSOCTET *pMemBuf, size_t bufSize)

virtual EXTXMLMETHOD OSCTXT* OSXMLParserCtxt::getContext ()

virtual EXTXMLMETHOD const OSUTF8CHAR* OSXMLParserCtxt::parseQName (const OSUTF8CHAR *const qname)

OSXMLParserCtxtIF class Reference

- virtual ~OSXMLParserCtxtIF ( )
- virtual OSRTInputStreamIF * createInputStream ( )
- virtual OSRTInputStreamIF * createFileInputStream (const char *const filename)

- virtual OSRTInputStreamIF * createMemoryInputStream (OSOCTET * pMemBuf, size_t bufSize)
- virtual OSCTXT * getContext ( )
- virtual const OSUTF8CHAR * parseQName (const OSUTF8CHAR *const qname)

virtual OSXMLParserCtxtIF::~OSXMLParserCtxtIF ()

virtual OSRTInputStreamIF* OSXMLParserCtxtIF::createInputStream ()=0

virtual OSRTInputStreamIF* OSXMLParserCtxtIF::createFileInputStream (const char *const filename)=0

virtual OSRTInputStreamIF* OSXMLParserCtxtIF::createMemoryInputStream (OSOCTET *pMemBuf, size_t bufSize)=0

virtual OSCTXT* OSXMLParserCtxtIF::getContext ()=0

virtual const OSUTF8CHAR* OSXMLParserCtxtIF::parseQName (const OSUTF8CHAR *const qname)=0

OSXMLQName struct Reference

Public Attributes

- const OSUTF8CHAR * nsPrefix
- const OSUTF8CHAR * ncName

Member Data Documentation

OSXMLReaderClass class Reference

- virtual int parse ( )
- virtual int parse (OSRTInputStreamIF & source)

  Parse an XML document.

- virtual int parse (const char *const pBuffer, size_t bufSize)

  Parse an XML document from memory buffer.

- virtual int parse (const char *const systemId)

  Parse an XML document from a system identifier (URI).

virtual int OSXMLReaderClass::parse ()=0

virtual int OSXMLReaderClass::parse (OSRTInputStreamIF &source)=0

Parse an XML document.

The application can use this method to instruct the SAX parser to begin parsing an XML document from any valid input source (a character stream, a byte stream, or a URI).

Applications may not invoke this method while a parse is in progress (they should create a new Parser instead for each additional XML document). Once a parse is complete, an application may reuse the same Parser object, possibly with a different input source.

Table 3.39. Parameters

source	The input source for the top-level of the XML document.

virtual int OSXMLReaderClass::parse (const char *const pBuffer, size_t bufSize)=0

Parse an XML document from memory buffer.

Table 3.40. Parameters

pBuffer	Buffer containg the XML data to be parsed.
bufSize	Buffer size, in octets.

virtual int OSXMLReaderClass::parse (const char *const systemId)=0

Parse an XML document from a system identifier (URI).

This method is a shortcut for the common case of reading a document from a system identifier. It is the exact equivalent of the following:

parse(new URLInputSource(systemId));

If the system identifier is a URL, it must be fully resolved by the application before it is passed to the parser.

Table 3.41. Parameters

systemId	The system identifier (URI).

OSXMLSimpleTypeHandler class Reference

Private Attributes

- OSRTMEMBUF mCurrElemValue

- EXTXMLMETHOD OSXMLSimpleTypeHandler (OSRTContext * pContext, const OSUTF8CHAR * elemName)
- EXTXMLMETHOD ~OSXMLSimpleTypeHandler ( )
- virtual EXTXMLMETHOD int startElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const * attrs)

  Receive notification of the beginning of an element.
- virtual EXTXMLMETHOD int characters (const OSUTF8CHAR *const chars, unsigned int length)

  Receive notification of character data.
- virtual EXTXMLMETHOD int endElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)

  Receive notification of the end of an element.

Member Data Documentation

EXTXMLMETHOD OSXMLSimpleTypeHandler::OSXMLSimpleTypeHandler (OSRTContext *pContext, const OSUTF8CHAR *elemName)

EXTXMLMETHOD OSXMLSimpleTypeHandler::~OSXMLSimpleTypeHandler ()

virtual EXTXMLMETHOD int OSXMLSimpleTypeHandler::startElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const *attrs)

Receive notification of the beginning of an element.

The Parser will invoke this method at the beginning of every element in the XML document; there will be a corresponding endElement() event for every startElement() event (even when the element is empty). All of the element's content will be reported, in order, before the corresponding endElement() event.

Table 3.42. Parameters

uri	The URI of the asscioated namespace for this element
localname	The local part of the element name
qname	The QName of this element
attrs	The attributes name/value pairs attached to the element, if any.

See also: . endElement

virtual EXTXMLMETHOD int OSXMLSimpleTypeHandler::characters (const OSUTF8CHAR *const chars, unsigned int length)

Receive notification of character data.

The Parser will call this method to report each chunk of character data. SAX parsers may return all contiguous character data in a single chunk, or they may split it into several chunks; however, all of the characters in any single event must come from the same external entity, so that the Locator provides useful information.

Table 3.43. Parameters

chars	The characters from the XML document.
length	The length of chars.

virtual EXTXMLMETHOD int OSXMLSimpleTypeHandler::endElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)

Receive notification of the end of an element.

The SAX parser will invoke this method at the end of every element in the XML document; there will be a corresponding startElement() event for every endElement() event (even when the element is empty).

Table 3.44. Parameters

uri	The URI of the asscioated namespace for this element
localname	The local part of the element name
qname	The QName of this element

OSXMLSoapHandler class Reference

Private Attributes

- OSXMLDefaultHandler * mpMsgSaxHandler
- OSXMLDefaultHandler * mpFaultMsgSaxHandler
- OSRTMemBuf mCurrElemValue
- OSBOOL mbEnvelopeParsed
- OSBOOL mbFault
- OSBOOL mbProcessingDetailFault
- OSBOOL mbParsingHeader

- OSXMLSoapHandler (OSXMLDefaultHandler * msgSaxHandler, OSRTContext * pContext, OSXMLDefault-Handler * faultSaxHandler)
- ~OSXMLSoapHandler ( )
- virtual int startElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const * attrs)

  Receive notification of the beginning of an element.
- virtual int characters (const OSUTF8CHAR *const chars, unsigned int length)

  Receive notification of character data.
- virtual int endElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)

  Receive notification of the end of an element.
- OSBOOL isSoapEnv (const OSUTF8CHAR * uri)

Member Data Documentation

OSXMLSoapHandler::OSXMLSoapHandler (OSXMLDefaultHandler *msgSaxHandler, OSRTContext *pContext, OSXMLDefaultHandler *faultSaxHandler=0)

OSXMLSoapHandler::~OSXMLSoapHandler ()

virtual int OSXMLSoapHandler::startElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname, const OSUTF8CHAR *const *attrs)

Receive notification of the beginning of an element.

The Parser will invoke this method at the beginning of every element in the XML document; there will be a corresponding endElement() event for every startElement() event (even when the element is empty). All of the element's content will be reported, in order, before the corresponding endElement() event.

Table 3.45. Parameters

uri	The URI of the asscioated namespace for this element
localname	The local part of the element name
qname	The QName of this element
attrs	The attributes name/value pairs attached to the element, if any.

See also: . endElement

virtual int OSXMLSoapHandler::characters (const OSUTF8CHAR *const chars, unsigned int length)

Receive notification of character data.

The Parser will call this method to report each chunk of character data. SAX parsers may return all contiguous character data in a single chunk, or they may split it into several chunks; however, all of the characters in any single event must come from the same external entity, so that the Locator provides useful information.

Table 3.46. Parameters

chars	The characters from the XML document.
length	The length of chars.

virtual int OSXMLSoapHandler::endElement (const OSUTF8CHAR *const uri, const OSUTF8CHAR *const localname, const OSUTF8CHAR *const qname)

Receive notification of the end of an element.

The SAX parser will invoke this method at the end of every element in the XML document; there will be a corresponding startElement() event for every endElement() event (even when the element is empty).

Table 3.47. Parameters

uri	The URI of the asscioated namespace for this element
localname	The local part of the element name
qname	The QName of this element

OSBOOL OSXMLSoapHandler::isSoapEnv (const OSUTF8CHAR *uri)

OSXMLSortedAttrOffset struct Reference

Public Attributes

- OSSIZE offset
- OSSIZE length
- OSSIZE prefixLength
- OSSIZE nameLength

Member Data Documentation

OSXMLStrFragment struct Reference

Public Attributes

- const OSUTF8CHAR * value
- OSSIZE length

Member Data Documentation

OSXMLStringListParser class Reference

```
#include <rtXmlpCppDecFuncs.h>
```

Private Attributes

- OSXMLDataCtxt dataCtxt
- OSCTXT * mpctxt
- OSBOOL mbInit
- const OSUTF8CHAR * inpdata

- OSXMLStringListParser (OSCTXT * pctxt)

  Create a parser on the given context.

- int next (OSRTXMLString & value)

  Assign the next string from the XML schema list to value.

Class enabling parsing of an XML Schema list into strings.

Member Data Documentation

OSXMLStringListParser::OSXMLStringListParser (OSCTXT *pctxt)

Create a parser on the given context.

The OSXMLReader associated with the context is used to read the XML value.

Table 3.48. Parameters

pctxt	Holds state information and provides access to the input that is to be parsed.

int OSXMLStringListParser::next (OSRTXMLString &value)

Assign the next string from the XML schema list to value.

To get all of the values from the list, call this function repeatedly until it returns zero or a negative value. After that, this object can no longer be used.

Table 3.49. Parameters

value	Receives the next string value or empty string if there isn't one.

Returns: . Return value indicates the result as follows: return < 0: error return == 0: no more strings; value is assigned empty return == 1: value is assigned the next string

OSXMLStrListHandler class Reference

```
#include <rtSaxCppStrList.h>
```

- OSXMLStrListHandler ( )

- static EXTXMLMETHOD int parse (OSCTXT * pctxt, OSRTMEMBUF * pMemBuf, OSRTDList * pStrList)
- static EXTXMLMETHOD int parse (OSCTXT * pctxt, OSRTMEMBUF * pMemBuf, OSRTObjListClass * pStrList, OSBOOL useSTL)
- static int parseSTL (OSCTXT * pctxt, OSRTMEMBUF * pMemBuf, OSRTObjListClass * pStrList)
- static int match (OSCTXT *)

OSXMLStrListHandler.

OSXMLStrListHandler::OSXMLStrListHandler ()

static EXTXMLMETHOD int OSXMLStrListHandler::parse (OSCTXT *pctxt, OSRTMEMBUF *pMemBuf, OSRTDList *pStrList)

static EXTXMLMETHOD int OSXMLStrListHandler::parse (OSCTXT *pctxt, OSRTMEMBUF *pMemBuf, OSRTObjListClass *pStrList, OSBOOL useSTL=FALSE)

static int OSXMLStrListHandler::parseSTL (OSCTXT *pctxt, OSRTMEMBUF *pMemBuf, OSRTObjListClass *pStrList)

static int OSXMLStrListHandler::match (OSCTXT *)

OSXSDAnyType struct Reference

Public Attributes

- OSXMLSTRING value
- OSRTDList attrs

Member Data Documentation

OSXSDGlobalElement class Reference

```
#include <rtXmlCppXSDElement.h>
```

Protected Attributes

- OSRTCtxtPtr mpContext

  The mpContext member variable holds a reference-counted C runtime variable.

- OSRTMessageBufferIF * mpMsgBuf

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

- OSXSDGlobalElement ( )

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

- OSXSDGlobalElement (OSRTContext & ctxt)

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

- void setMsgBuf (OSRTMessageBufferIF & msgBuf)

  The setMsgBuf method is used to set the internal message buffer pointer to point at the given message buffer or stream object.

- OSXSDGlobalElement (OSRTMessageBufferIF & msgBuf)

  This constructor sets the internal message buffer pointer to point at the given message buffer or stream object.

- OSXSDGlobalElement (const OSXSDGlobalElement & o)

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

- virtual ~OSXSDGlobalElement ( )

  The virtual destructor does nothing.

- int decode ( )

  The \c decode method decodes the message described by the encapsulated message buffer object.

- virtual int decodeFrom (OSRTMessageBufferIF &)

  The \c decodeFrom method decodes a message from the given message buffer or stream argument.

- int encode ( )

  The \c encode method encodes a message using the encoding rules specified by the derived message buffer object.

- virtual int encodeTo (OSRTMessageBufferIF &)

  The \c encodeTo method encodes a message into the given message buffer or stream argument.

- OSCTXT * getCtxtPtr ( )

  The getCtxtPtr method returns the underlying C runtime context.

- void * memAlloc (size_t numocts)

  The memAlloc method allocates memory using the C runtime memory management functions.

- void memFreePtr (void * ptr)

  The memFreePtr method frees the memory at a specific location.

- void setDefaultNamespace (const OSUTF8CHAR * uri)

  The setDefaultNamespace method sets the default namespace for the element to the given value.

- void setDiag (OSBOOL value)

  The setDiag method turns diagnostic tracing on or off.

- void setEncXSINamespace (OSBOOL value)

  The setEncXSINamespace method sets a flag in the internal context that indicates the xsi namespace attribute must be encoded.

- void setNamespace (const OSUTF8CHAR * prefix, const OSUTF8CHAR * uri)

  The setNamespace method adds or modifies the namespace with the given URI in the namespace list to contain the given prefix.

- void setNoNSSchemaLocation (const OSUTF8CHAR * uri)

  The setNoNSSchemaLocation method adds an xsi:noNamespaceSchemaLocation attribute to the document.

- void setSchemaLocation (const OSUTF8CHAR * uri)

  The setSchemaLocation method adds an xsi:schemaLocation attribute to the document.

- void setXSIType (const OSUTF8CHAR * typeName)

  The setXSIType method sets a type name to be used in the xsi:type attribute in the top-level module element declaration.

- int validate ( )

  The \c validate method validates the message described by the encapsulated message buffer object.

- virtual int validateFrom (OSRTMessageBufferIF &)

  The \c validateFrom method validates a message from the given message buffer or stream argument.

XSD global element base class.

Detailed Description

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

Definition at line 57 of file rtXmlCppXSDElement.h

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

- rtXmlCppXSDElement.h

Member Data Documentation

OSRTCtxtPtr OSXSDGlobalElement::mpContext

This context is used in calls to all C run-time functions. The context pointed at by this smart-pointer object is shared with the message buffer object contained within this class.

Definition at line 65 of file rtXmlCppXSDElement.h

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

- rtXmlCppXSDElement.h

OSXSDGlobalElement::OSXSDGlobalElement ()

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

OSXSDGlobalElement::OSXSDGlobalElement (OSRT-Context &ctxt)

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

Table 3.50. Parameters

ctxt	- Reference to a context object.

void OSXSDGlobalElement::setMsgBuf (OSRTMessage-BufferIF &msgBuf)

The setMsgBuf method is used to set the internal message buffer pointer to point at the given message buffer or stream object.

Table 3.51. Parameters

msgBuf	- Reference to a message buffer or stream object.

OSXSDGlobalElement::OSXSDGlobalElement (OSRTMessageBufferIF &msgBuf)

This constructor sets the internal message buffer pointer to point at the given message buffer or stream object.

The context is set to point at the context contained within the message buffer object. Thus, the message buffer and control class object share the context. It will not be released until both objects are destroyed.

Table 3.52. Parameters

msgBuf	- Reference to a message buffer or stream object.

OSXSDGlobalElement::OSXSDGlobalElement (const OSXSDGlobalElement &o)

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

Table 3.53. Parameters

o	- Reference to a global element object.

virtual OSXSDGlobalElement::~OSXSDGlobalElement ()

The virtual destructor does nothing.

It is overridden by derived versions of this class.

int OSXSDGlobalElement::decode ()

The \c decode method decodes the message described by the encapsulated message buffer object.

virtual int OSXSDGlobalElement::decodeFrom (OSRTMessageBufferIF &)

The \c decodeFrom method decodes a message from the given message buffer or stream argument.

Table 3.54. Parameters

-	Message buffer or stream containing message to decode.

int OSXSDGlobalElement::encode ()

The \c encode method encodes a message using the encoding rules specified by the derived message buffer object.

virtual int OSXSDGlobalElement::encodeTo (OSRTMessageBufferIF &)

The \c encodeTo method encodes a message into the given message buffer or stream argument.

Table 3.55. Parameters

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

OSCTXT* OSXSDGlobalElement::getCtxtPtr ()

The getCtxtPtr method returns the underlying C runtime context.

This context can be used in calls to C runtime functions.

void* OSXSDGlobalElement::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 3.56. Parameters

numocts	- Number of bytes of memory to allocate

void OSXSDGlobalElement::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 3.57. Parameters

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

void OSXSDGlobalElement::setDefaultNamespace (const OSUTF8CHAR *uri)

The setDefaultNamespace method sets the default namespace for the element to the given value.

Table 3.58. Parameters

uri	- Default namespace URI

void OSXSDGlobalElement::setDiag (OSBOOL value=TRUE)

The setDiag method turns diagnostic tracing on or off.

Table 3.59. Parameters

value	- Boolean on/off value (default = on)

void OSXSDGlobalElement::setEncXSINamespace (OSBOOL value=TRUE)

The setEncXSINamespace method sets a flag in the internal context that indicates the xsi namespace attribute must be encoded.

Table 3.60. Parameters

value	- Boolean on/off value (default = on)

void OSXSDGlobalElement::setNamespace (const OSUTF8CHAR *prefix, const OSUTF8CHAR *uri)

The setNamespace method adds or modifies the namespace with the given URI in the namespace list to contain the given prefix.

Table 3.61. Parameters

prefix	- Namespace prefix
uri	- Namespace URI

void OSXSDGlobalElement::setNoNSSchemaLocation (const OSUTF8CHAR *uri)

The setNoNSSchemaLocation method adds an xsi:noNamespaceSchemaLocation attribute to the document.

Table 3.62. Parameters

uri	- URI for noNamespaceSchemaLocation.

void OSXSDGlobalElement::setSchemaLocation (const OSUTF8CHAR *uri)

The setSchemaLocation method adds an xsi:schemaLocation attribute to the document.

Table 3.63. Parameters

uri	- URI for schemaLocation.

void OSXSDGlobalElement::setXSIType (const OSUTF8CHAR *typeName)

The setXSIType method sets a type name to be used in the xsi:type attribute in the top-level module element declaration.

Table 3.64. Parameters

typeName	- XSI type name

int OSXSDGlobalElement::validate ()

The \c validate method validates the message described by the encapsulated message buffer object.

virtual int OSXSDGlobalElement::validateFrom (OSRTMessageBufferIF &)

The \c validateFrom method validates a message from the given message buffer or stream argument.

Table 3.65. Parameters

-	Message buffer or stream containing message to validate.

Chapter 4. File Documentation

osrtxml.h File Reference

```
#include "rtxsrc/rtxCommon.h"
#include "rtxmlsrc/rtSaxDefs.h"
#include "rtxsrc/rtxDList.h"
#include "rtxsrc/rtxMemBuf.h"
#include "rtxmlsrc/rtXmlExternDefs.h"
#include "rtxmlsrc/rtXmlErrCodes.h"
#include "rtxmlsrc/rtXmlNamespace.h"
```

Classes

- struct OSXMLFacets
- struct OSXMLStrFragment
- struct OSXMLNameFragments
- struct OSXMLItemDescr
- struct OSXMLElemIDRec
- struct OSXMLGroupDesc

  OSXMLGroupDesc describes how entries in an OSXMLElemIDRec array make up a group.
- struct OSXSDAnyType
- struct OSXMLCtxtInfo
- struct OSXMLQName
- struct OSIntegerFmt
- struct OSXMLSortedAttrOffset

Macros

- #define OSXMLNS12
- #define OSUPCASE 0x00008000 /* convert characters to upper case */
- #define OSTERMSTART 0x00004000 /* term for start elem (>) needed */
- #define OSEMPTYELEM 0x00002000 /* element is empty (no content) */
- #define OSQUALATTR 0x00001000 /* qualified attribute */

- #define OSXMLFRAG 0x00000800 /* XML fragment (not full doc) */
- #define OSXMLNSSET 0x00000400 /* Indicates namespaces are set */
- #define OSXMLC14N 0x00000200 /* Flag used to indicate XML canonical encode when OSASN1XER is not set, or canonical XER when OSASN1XER is set. */
- #define OSXSIATTR 0x00000100 /* add xsi ns decl to encoded msg */
- #define OSXMLNOCMPNS 0x00000080 /* match local names only */
- #define OSXSINIL 0x00000040 /* add xsi:nil decl to encoded msg */
- #define OSHASDEFAULT 0x00000010 /* decode should accept values which are empty after whitespace processing because the element has a default value */
- #define OSASN1XER 0x00000008 /* if true, -xml code and runtime should produce XER encodings instead of Obj-Sys encodings */
- #define OSXMLFRAGSEQUAL (frag1.length==frag2.length && ! memcmp(frag1.value,frag2.value,frag1.length))
- #define OSXMLQNAMEEQUALS rtxUTF8StrnEqual \ (xnamefrag.mQName.value, OSUTF8(qnametext), xnamefrag.mQName.length)
- #define OSXMLSETUTF8DECPTR rtxInitContextBuffer (pctxt, OSRTSAFECONSTCAST (OSOCTET*, str), \ OSUTF8LEN (str))
- #define IS_XMLNSATTR ((OSUTF8LEN(name) >= 5) && name[0] == 'x' && name[1] == 'm' && \ name[2] == 'l' && name[3] == 'n' && name[4] == 's')
- #define IS_XSIATTR ((OSUTF8LEN(name) >= 4) && name[0] == 'x' && name[1] == 's' && \ name[2] == 'i' && name[3] == ':')
- #define OSXMLINDENT 3
- #define rtXmlErrAddStrParm rtxErrAddStrParm
- #define rtxPrintNSAttrs rtXmlPrintNSAttrs(name,&data)
- #define rtXmlFinalizeMemBuf do { \ (pMemBuf)->pctxt->buffer.data = (pMemBuf)->buffer + (pMemBuf)->startidx; \ (pMemBuf)->pctxt->buffer.size = \ ((pMemBuf)->usedcnt - (pMemBuf)->startidx); \ (pMemBuf)->pctxt->buffer.dynamic = FALSE; \ (pMemBuf)->pctxt->buffer.byteIndex = 0; \ rtxMemBufReset (pMemBuf); \ } while(0)
- #define rtXmlGetEncBufPtr (pctxt)->buffer.data

  This macro returns the start address of the encoded XML message.
- #define rtXmlGetEncBufLen (pctxt)->buffer.byteIndex

  This macro returns the length of the encoded XML message.
- #define OSXMLREALENC_OBJSYS 0x1F /* Obj-Sys XML encoding rules */
- #define OSXMLREALENC_BXER 0x10 /* basic-XER */
- #define OSXMLREALENC_EXERMODS 0x1B /* extended-XER with MODIFIED-ENCODINGS */

- #define OSXMLREALENC_EXERDECIMAL 0x03 /* extended-XER with DECIMAL */

Enumerations

- enum OSXMLEncoding {
  OSXMLUTF8,
  OSXMLUTF16,
  OSXMLUTF16BE,
  OSXMLUTF16LE,
  OSXMLLATIN1
  }

- enum OSXMLSOAPMsgType {
  OSSOAPNONE,
  OSSOAPHEADER,
  OSSOAPBODY,
  OSSOAPFAULT
  }

- enum OSXMLBOM {
  OSXMLBOM_NO_BOM,
  OSXMLBOM_UTF32_BE,
  OSXMLBOM_UTF32_LE,
  OSXMLBOM_UTF16_BE,
  OSXMLBOM_UTF16_LE,
  OSXMLBOM_UTF8,
  OSXMLBOM_CHECK
  }

- enum OSXMLNsIndex {
  OSXMLNSI_UNQUALIFIED= 0,
  OSXMLNSI_UNKNOWN= -1,
  OSXMLNSI_UNCHECKED= -2,
  OSXMLNSI_XSI= -3,
  OSXMLNSI_XMLNS= -4,
  OSXMLNSI_XML= -5,
  OSXMLNSI_SOAP_ENVELOPE= -6,
  OSXMLNSI_XSD= -7
  }

- enum OSXMLREALEncoding {
  OSXMLREALOBJSYS,
  OSXMLREALBXER,
  OSXMLREALEXERMODS,
  OSXMLREALEXERDEC
  }

- enum OSXMLState {
  OSXMLINIT,
  OSXMLHEADER,
  OSXMLSTART,
  OSXMLATTR,
  OSXMLDATA,
  OSXMLEND,

OSXMLCOMMENT
}

- enum OSXMLWhiteSpaceMode {
  OSXMLWSM_PRESERVE= 0,
  OSXMLWSM_REPLACE,
  OSXMLWSM_COLLAPSE
  }

  Whitespace treatment options.

Typedefs

- typedef struct OSXMLFacets OSXMLFacets
- typedef struct OSXMLItemDescr OSXMLItemDescr
- typedef OSXMLItemDescr OSXMLAttrDescr
- typedef OSXMLItemDescr OSXMLElemDescr
- typedef struct OSXMLElemIDRec OSXMLElemIDRec
- typedef struct OSXMLGroupDesc OSXMLGroupDesc

  OSXMLGroupDesc describes how entries in an OSXMLElemIDRec array make up a group.
- typedef struct OSXSDAnyType OSXSDAnyType
- typedef struct OSXMLQName OSXMLQName
- typedef struct OSIntegerFmt OSIntegerFmt

Variables

- static const char OSXMLHDRUTF8
- static const char OSXMLHDRUTF16
- static const char OSXMLHDRUTF16BE
- static const char OSXMLHDRUTF16LE
- static const char OSXMLHDRLATIN1

Functions

- EXTERNXML int rtXmlInitContext (OSCTXT * pctxt)

  This function initializes a context variable for XML encoding or decoding.
- EXTERNXML int rtXmlInitContextUsingKey (OSCTXT * pctxt, const OSOCTET * key, OSSIZE keylen)

  This function initializes a context using a run-time key.
- EXTERNXML int rtXmlInitCtxtAppInfo (OSCTXT * pctxt)

This function initializes the XML application info section of the given context.

- EXTERNXML int rtXmlCreateFileInputSource (OSCTXT * pctxt, const char * filepath)

  This function creates an XML document file input source.

- EXTERNXML OSBOOL rtXmlCmpQName (const OSUTF8CHAR * qname1, const OSUTF8CHAR * name2, const OSUTF8CHAR * nsPrefix2)

- EXTERNXML int rtXmlGetBase64StrDecodedLen (const OSUTF8CHAR * inpdata, OSSIZE srcDataSize, OSSIZE * pNumOcts, OSSIZE * pSrcDataLen)

- EXTERNXML void rtXmlMemFreeAnyAttrs (OSCTXT * pctxt, OSRTDList * pAnyAttrList)

  This function frees a list of anyAttribute that is a member of OSXSDAnyType structure.

- EXTERNXML int rtXmlDecBase64Binary (OSRTMEMBUF * pMemBuf, const OSUTF8CHAR * inpdata, OSSIZE length)

  This function decodes the contents of a Base64-encoded binary data type into a memory buffer.

- EXTERNXML int rtXmlDecBase64Str (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocts, OSINT32 bufsize)

  This function decodes a contents of a Base64-encode binary string into a static memory structure.

- EXTERNXML int rtXmlDecBase64Str64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnocts, OSSIZE bufsize)

  This function is identical to rtXmlDecBase64Str except that is supports a 64-bit integer length on 64-bit systems.

- EXTERNXML int rtXmlDecBase64StrValue (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocts, OSSIZE bufSize, OSSIZE srcDataLen)

  This function decodes a contents of a Base64-encode binary string into the specified octet array.

- EXTERNXML int rtXmlDecBase64StrValue64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnocts, OSSIZE bufSize, OSSIZE srcDataLen)

  This function decodes is identical to rtXmlDecBase64StrValue except that it supports a 64-bit integer length on 64-bit systems.

- EXTERNXML int rtXmlDecBigInt (OSCTXT * pctxt, const OSUTF8CHAR ** ppvalue)

  This function will decode a variable of the XSD integer type.

- EXTERNXML int rtXmlDecBool (OSCTXT * pctxt, OSBOOL * pvalue)

  This function decodes a variable of the boolean type.

- EXTERNXML int rtXmlDecDate (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'date' type.

- EXTERNXML int rtXmlDecTime (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'time' type.

- EXTERNXML int rtXmlDecDateTime (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'dateTime' type.

- EXTERNXML int rtXmlDecDecimal (OSCTXT * pctxt, OSREAL * pvalue)

  This function decodes the contents of a decimal data type.

- EXTERNXML int rtXmlDecDouble (OSCTXT * pctxt, OSREAL * pvalue)

  This function decodes the contents of a float or double data type.

- EXTERNXML int rtXmlDecDynBase64Str (OSCTXT * pctxt, OSDynOctStr * pvalue)

  This function decodes a contents of a Base64-encode binary string.

- EXTERNXML int rtXmlDecDynBase64Str64 (OSCTXT * pctxt, OSDynOctStr64 * pvalue)

  This function is identical to rtXmlDecDynBase64Str except that it supports a 64-bit integer length on 64-bit systems.

- EXTERNXML int rtXmlDecDynHexStr (OSCTXT * pctxt, OSDynOctStr * pvalue)

  This function decodes a contents of a hexBinary string.

- EXTERNXML int rtXmlDecDynHexStr64 (OSCTXT * pctxt, OSDynOctStr64 * pvalue)

  This function is identical to rtXmlDecDynHexStr except that it supports a 64-bit integer length on 64-bit systems.

- EXTERNXML int rtXmlDecEmptyElement (OSCTXT * pctxt)

  This function is used to enforce a requirement that an element be empty.

- EXTERNXML int rtXmlDecUTF8Str (OSCTXT * pctxt, OSUTF8CHAR * outdata, OSSIZE max_len)

  This function decodes the contents of a UTF-8 string data type.

- EXTERNXML int rtXmlDecDynUTF8Str (OSCTXT * pctxt, const OSUTF8CHAR ** outdata)

  This function decodes the contents of a UTF-8 string data type.

- EXTERNXML int rtXmlDecHexBinary (OSRTMEMBUF * pMemBuf, const OSUTF8CHAR * inpdata, OSSIZE length)

  This function decodes the contents of a hex-encoded binary data type into a memory buffer.

- EXTERNXML int rtXmlDecHexStr (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocts, OSINT32 bufsize)

  This function decodes the contents of a hexBinary string into a static memory structure.

- EXTERNXML int rtXmlDecHexStr64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnocts, OSSIZE bufsize)

  This function is identical to rtXmlDecHexStr except that it supports a 64-bit integer length on 64-bit systems.

- EXTERNXML int rtXmlDecHexStrValue (OSCTXT * pctxt, const OSUTF8CHAR *const inpdata, OSSIZE nbytes, OSOCTET * pvalue, OSUINT32 * pnbits, OSINT32 bufsize)

- EXTERNXML int rtXmlDecHexStrValue64 (OSCTXT * pctxt, const OSUTF8CHAR *const inpdata, OSSIZE nbytes, OSOCTET * pvalue, OSSIZE * pnbits, OSSIZE bufsize)

- EXTERNXML int rtXmlDecGYear (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gYear' type.

- EXTERNXML int rtXmlDecGYearMonth (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gYearMonth' type.

- EXTERNXML int rtXmlDecGMonth (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gMonth' type.

- EXTERNXML int rtXmlDecGMonthDay (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gMonthDay' type.

- EXTERNXML int rtXmlDecGDay (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gDay' type.

- EXTERNXML int rtXmlDecInt (OSCTXT * pctxt, OSINT32 * pvalue)

  This function decodes the contents of a 32-bit integer data type.

- EXTERNXML int rtXmlDecInt8 (OSCTXT * pctxt, OSINT8 * pvalue)

  This function decodes the contents of an 8-bit integer data type (i.e.

- EXTERNXML int rtXmlDecInt16 (OSCTXT * pctxt, OSINT16 * pvalue)

  This function decodes the contents of a 16-bit integer data type.

- EXTERNXML int rtXmlDecInt64 (OSCTXT * pctxt, OSINT64 * pvalue)

  This function decodes the contents of a 64-bit integer data type.

- EXTERNXML int rtXmlDecUInt (OSCTXT * pctxt, OSUINT32 * pvalue)

  This function decodes the contents of an unsigned 32-bit integer data type.

- EXTERNXML int rtXmlDecUInt8 (OSCTXT * pctxt, OSUINT8 * pvalue)

  This function decodes the contents of an unsigned 8-bit integer data type (i.e.

- EXTERNXML int rtXmlDecUInt16 (OSCTXT * pctxt, OSUINT16 * pvalue)

  This function decodes the contents of an unsigned 16-bit integer data type.

- EXTERNXML int rtXmlDecUInt64 (OSCTXT * pctxt, OSUINT64 * pvalue)

  This function decodes the contents of an unsigned 64-bit integer data type.

- EXTERNXML int rtXmlDecNSAttr (OSCTXT * pctxt, const OSUTF8CHAR * attrName, const OSUTF8CHAR * attrValue, OSRTDList * pNSAttrs, const OSUTF8CHAR * nsTable, OSUINT32 nsTableRowCount)

  This function decodes an XML namespac attribute (xmlns).

- EXTERNXML const OSUTF8CHAR * rtXmlDecQName (OSCTXT * pctxt, const OSUTF8CHAR * qname, const OSUTF8CHAR ** prefix)

This function decodes an XML qualified name string (QName) type.

- EXTERNXML int rtXmlDecXSIAttr (OSCTXT * pctxt, const OSUTF8CHAR * attrName, const OSUTF8CHAR * attrValue)

  This function decodes XML schema instance (XSI) attribute.

- EXTERNXML int rtXmlDecXSIAttrs (OSCTXT * pctxt, const OSUTF8CHAR *const * attrs, const char * typeName)

  This function decodes XML schema instance (XSI) attributes.

- EXTERNXML int rtXmlDecXmlStr (OSCTXT * pctxt, OSXMLSTRING * outdata)

  This function decodes the contents of an XML string data type.

- EXTERNXML int rtXmlParseElementName (OSCTXT * pctxt, OSUTF8CHAR ** ppName)

  This function parses the initial tag from an XML message.

- EXTERNXML int rtXmlParseElemQName (OSCTXT * pctxt, OSXMLQName * pQName)

  This function parses the initial tag from an XML message.

- EXTERNXML int rtXmlEncAny (OSCTXT * pctxt, OSXMLSTRING * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD any type.

- EXTERNXML int rtXmlEncAnyStr (OSCTXT * pctxt, const OSUTF8CHAR * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

- EXTERNXML int rtXmlEncAnyTypeValue (OSCTXT * pctxt, const OSUTF8CHAR * pvalue)

  This function encodes a variable of the XSD anyType type.

- EXTERNXML int rtXmlEncAnyAttr (OSCTXT * pctxt, OSRTDList * pAnyAttrList)

  This function encodes a list of OSAnyAttr attributes in which the name and value are given as a UTF-8 string.

- EXTERNXML int rtXmlEncBase64Binary (OSCTXT * pctxt, OSSIZE nocts, const OSOCTET * value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD base64Binary type.

- EXTERNXML int rtXmlEncBase64BinaryAttr (OSCTXT * pctxt, OSUINT32 nocts, const OSOCTET * value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen)

  This function encodes a variable of the XSD base64Binary type as an attribute.

- EXTERNXML int rtXmlEncBase64StrValue (OSCTXT * pctxt, OSSIZE nocts, const OSOCTET * value)

  This function encodes a variable of the XSD base64Binary type.

- EXTERNXML int rtXmlEncBigInt (OSCTXT * pctxt, const OSUTF8CHAR * value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD integer type.

- EXTERNXML int rtXmlEncBigIntAttr (OSCTXT * pctxt, const OSUTF8CHAR * value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen)

  This function encodes an XSD integer attribute value.

- EXTERNXML int rtXmlEncBigIntValue (OSCTXT * pctxt, const OSUTF8CHAR * value)

  This function encodes an XSD integer attribute value.

- EXTERNXML int rtXmlEncBitString (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

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

- EXTERNXML int rtXmlEncBitStringExt (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * value, OSSIZE dataSize, const OSOCTET * extValue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

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

- EXTERNXML int rtXmlEncBinStrValue (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data)

  This function encodes a binary string value as a sequence of '1's and '0's.

- EXTERNXML int rtXmlEncBool (OSCTXT * pctxt, OSBOOL value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD boolean type.

- EXTERNXML int rtXmlEncBoolValue (OSCTXT * pctxt, OSBOOL value)

  This function encodes a variable of the XSD boolean type.

- EXTERNXML int rtXmlEncBoolAttr (OSCTXT * pctxt, OSBOOL value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen)

  This function encodes an XSD boolean attribute value.

- EXTERNXML int rtXmlEncCanonicalSort (OSCTXT * pctxt, OSCTXT * pBufCtxt, OSRTSList * pList)

  Sort (as for CXER, Canonical-XER) and encode previously encoded SET OF components.

- EXTERNXML int rtXmlEncComment (OSCTXT * pctxt, const OSUTF8CHAR * comment)

  This function encodes an XML comment.

- EXTERNXML int rtXmlEncDate (OSCTXT * pctxt, const OSXSDDateTime * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD 'date' type as a string.

- EXTERNXML int rtXmlEncDateValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)

  This function encodes a variable of the XSD 'date' type as a string.

- EXTERNXML int rtXmlEncTime (OSCTXT * pctxt, const OSXSDDateTime * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD 'time' type as an string.

- EXTERNXML int rtXmlEncTimeValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)

  This function encodes a variable of the XSD 'time' type as an string.

- EXTERNXML int rtXmlEncDateTime (OSCTXT * pctxt, const OSXSDDateTime * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a numeric date/time value into an XML string representation.

- EXTERNXML int rtXmlEncDateTimeValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)

  This function encodes a numeric date/time value into an XML string representation.

- EXTERNXML int rtXmlEncDecimal (OSCTXT * pctxt, OSREAL value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, const OSDecimalFmt * pFmtSpec)

  This function encodes a variable of the XSD decimal type.

- EXTERNXML int rtXmlEncDecimalAttr (OSCTXT * pctxt, OSREAL value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen, const OSDecimalFmt * pFmtSpec)

  This function encodes a variable of the XSD decimal type as an attribute.

- EXTERNXML int rtXmlEncDecimalValue (OSCTXT * pctxt, OSREAL value, const OSDecimalFmt * pFmtSpec, char * pDestBuf, OSSIZE destBufSize)

  This function encodes a value of the XSD decimal type.

- EXTERNXML int rtXmlEncDouble (OSCTXT * pctxt, OSREAL value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, const OSDoubleFmt * pFmtSpec)

  This function encodes a variable of the XSD double type.

- EXTERNXML int rtXmlEncDoubleAttr (OSCTXT * pctxt, OSREAL value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen, const OSDoubleFmt * pFmtSpec)

  This function encodes a variable of the XSD double type as an attribute.

- EXTERNXML int rtXmlEncDoubleNormalValue (OSCTXT * pctxt, OSREAL value, const OSDoubleFmt * pFmtSpec, int defaultPrecision)

  This function encodes a normal (not +/-INF or NaN) value of the XSD double or float type.

- EXTERNXML int rtXmlEncDoubleValue (OSCTXT * pctxt, OSREAL value, const OSDoubleFmt * pFmtSpec, int defaultPrecision)

  This function encodes a value of the XSD double or float type.

- EXTERNXML int rtXmlEncEmptyElement (OSCTXT * pctxt, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, OSRTDList * pNSAttrs, OSBOOL terminate)

  This function encodes an enpty element tag value (\<elemName/\>).

- EXTERNXML int rtXmlEncEndDocument (OSCTXT * pctxt)

  This function adds trailor information and a null terminator at the end of the XML document being encoded.

- EXTERNXML int rtXmlEncEndElement (OSCTXT * pctxt, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

This function encodes an end element tag value (\</elemName\>).

- EXTERNXML int rtXmlEncEndSoapEnv (OSCTXT * pctxt)

  This function encodes a SOAP envelope end element tag (\<SOAP-ENV:Envelope/\>).

- EXTERNXML int rtXmlEncEndSoapElems (OSCTXT * pctxt, OSXMLSOAPMsgType msgtype)

  This function encodes SOAP end element tags.

- EXTERNXML int rtXmlEncFloat (OSCTXT * pctxt, OSREAL value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, const OSDoubleFmt * pFmtSpec)

  This function encodes a variable of the XSD float type.

- EXTERNXML int rtXmlEncFloatAttr (OSCTXT * pctxt, OSREAL value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen, const OSDoubleFmt * pFmtSpec)

  This function encodes a variable of the XSD float type as an attribute.

- EXTERNXML int rtXmlEncGYear (OSCTXT * pctxt, const OSXSDDateTime * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a numeric gYear element into an XML string representation.

- EXTERNXML int rtXmlEncGYearMonth (OSCTXT * pctxt, const OSXSDDateTime * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a numeric gYearMonth element into an XML string representation.

- EXTERNXML int rtXmlEncGMonth (OSCTXT * pctxt, const OSXSDDateTime * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a numeric gMonth element into an XML string representation.

- EXTERNXML int rtXmlEncGMonthDay (OSCTXT * pctxt, const OSXSDDateTime * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a numeric gMonthDay element into an XML string representation.

- EXTERNXML int rtXmlEncGDay (OSCTXT * pctxt, const OSXSDDateTime * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a numeric gDay element into an XML string representation.

- EXTERNXML int rtXmlEncGYearValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)

  This function encodes a numeric gYear value into an XML string representation.

- EXTERNXML int rtXmlEncGYearMonthValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)

  This function encodes a numeric gYearMonth value into an XML string representation.

- EXTERNXML int rtXmlEncGMonthValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)

  This function encodes a numeric gMonth value into an XML string representation.

- EXTERNXML int rtXmlEncGMonthDayValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)

This function encodes a numeric gMonthDay value into an XML string representation.

- EXTERNXML int rtXmlEncGDayValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)

  This function encodes a numeric gDay value into an XML string representation.

- EXTERNXML int rtXmlEncHexBinary (OSCTXT * pctxt, OSSIZE nocts, const OSOCTET * value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD hexBinary type.

- EXTERNXML int rtXmlEncHexBinaryAttr (OSCTXT * pctxt, OSUINT32 nocts, const OSOCTET * value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen)

  This function encodes a variable of the XSD hexBinary type as an attribute.

- EXTERNXML int rtXmlEncHexStrValue (OSCTXT * pctxt, OSSIZE nocts, const OSOCTET * data)

  This function encodes a variable of the XSD hexBinary type.

- EXTERNXML int rtXmlEncIndent (OSCTXT * pctxt)

  This function adds indentation whitespace to the output stream.

- EXTERNXML int rtXmlEncInt (OSCTXT * pctxt, OSINT32 value, const OSUTF8CHAR * elemName, OSXML-Namespace * pNS)

  This function encodes a variable of the XSD integer type.

- EXTERNXML int rtXmlEncIntValue (OSCTXT * pctxt, OSINT32 value)

  This function encodes a variable of the XSD integer type.

- EXTERNXML int rtXmlEncIntAttr (OSCTXT * pctxt, OSINT32 value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen)

  This function encodes a variable of the XSD integer type as an attribute (name="value").

- EXTERNXML int rtXmlEncIntPattern (OSCTXT * pctxt, OSINT32 value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, const OSUTF8CHAR * pattern)

  This function encodes a variable of the XSD integer type using a pattern to specify the format of the integer value.

- EXTERNXML int rtXmlEncIntPatternValue (OSCTXT * pctxt, OSINT32 value, const OSUTF8CHAR * pattern)

- EXTERNXML int rtXmlEncUIntPattern (OSCTXT * pctxt, OSUINT32 value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, const OSUTF8CHAR * pattern)

- EXTERNXML int rtXmlEncUIntPatternValue (OSCTXT * pctxt, OSUINT32 value, const OSUTF8CHAR * pattern)

- EXTERNXML int rtXmlEncInt64 (OSCTXT * pctxt, OSINT64 value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD integer type.

- EXTERNXML int rtXmlEncInt64Pattern (OSCTXT * pctxt, OSINT64 value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, const OSUTF8CHAR * pattern)

- EXTERNXML int rtXmlEncInt64Value (OSCTXT * pctxt, OSINT64 value)

  This function encodes a variable of the XSD integer type.

- EXTERNXML int rtXmlEncInt64PatternValue (OSCTXT * pctxt, OSINT64 value, const OSUTF8CHAR * pattern)

- EXTERNXML int rtXmlEncInt64Attr (OSCTXT * pctxt, OSINT64 value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen)

  This function encodes a variable of the XSD integer type as an attribute (name="value").

- EXTERNXML int rtXmlEncNamedBits (OSCTXT * pctxt, const OSBitMapItem * pBitMap, OSSIZE nbits, const OSOCTET * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

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

- EXTERNXML int rtXmlEncNamedBitsValue (OSCTXT * pctxt, const OSBitMapItem * pBitMap, OSSIZE nbits, const OSOCTET * pvalue)

- EXTERNXML int rtXmlEncNSAttrs (OSCTXT * pctxt, OSRTDList * pNSAttrs)

  This function encodes namespace declaration attributes at the beginning of an XML document.

- EXTERNXML int rtXmlPrintNSAttrs (const char * name, const OSRTDList * data)

  This function prints a list of namespace attributes.

- EXTERNXML int rtXmlEncReal10 (OSCTXT * pctxt, const OSUTF8CHAR * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the ASN.1 REAL base 10 type.

- EXTERNXML int rtXmlEncSoapArrayTypeAttr (OSCTXT * pctxt, const OSUTF8CHAR * name, const OSUTF8CHAR * value, OSSIZE itemCount)

  This function encodes the special SOAP encoding attrType attribute which specifies the number and type of elements in a SOAP array.

- EXTERNXML int rtXmlEncSoapArrayTypeAttr2 (OSCTXT * pctxt, const OSUTF8CHAR * name, OSSIZE nameLen, const OSUTF8CHAR * value, OSSIZE valueLen, OSSIZE itemCount)

- EXTERNXML int rtXmlEncStartDocument (OSCTXT * pctxt)

  This function encodes the XML header text at the beginning of an XML document.

- EXTERNXML int rtXmlEncBOM (OSCTXT * pctxt)

  This function encodes the Unicode byte order mark header at the start of the document.

- EXTERNXML int rtXmlEncStartElement (OSCTXT * pctxt, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, OSRTDList * pNSAttrs, OSBOOL terminate)

  This function encodes a start element tag value (\<elemName\>).

- EXTERNXML int rtXmlEncStartSoapEnv (OSCTXT * pctxt, OSRTDList * pNSAttrs)

  This function encodes a SOAP envelope start element tag.

- EXTERNXML int rtXmlEncStartSoapElems (OSCTXT * pctxt, OSXMLSOAPMsgType msgtype)

  This function encodes a SOAP envelope start element tag and an optional SOAP body or fault tag.

- EXTERNXML int rtXmlEncString (OSCTXT * pctxt, OSXMLSTRING * pxmlstr, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD string type.

- EXTERNXML int rtXmlEncStringValue (OSCTXT * pctxt, const OSUTF8CHAR * value)

  This function encodes a variable of the XSD string type.

- EXTERNXML int rtXmlEncStringValue2 (OSCTXT * pctxt, const OSUTF8CHAR * value, OSSIZE valueLen)

  This function encodes a variable of the XSD string type.

- EXTERNXML int rtXmlEncTermStartElement (OSCTXT * pctxt)

  This function terminates a currently open XML start element by adding either a '>' or '/>' (if empty) terminator.

- EXTERNXML int rtXmlEncUnicodeStr (OSCTXT * pctxt, const OSUNICHAR * value, OSSIZE nchars, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a Unicode string value.

- EXTERNXML int rtXmlEncUTF8Attr (OSCTXT * pctxt, const OSUTF8CHAR * name, const OSUTF8CHAR * value)

  This function encodes an attribute in which the name and value are given as a null-terminated UTF-8 strings.

- EXTERNXML int rtXmlEncUTF8Attr2 (OSCTXT * pctxt, const OSUTF8CHAR * name, OSSIZE nameLen, const OSUTF8CHAR * value, OSSIZE valueLen)

  This function encodes an attribute in which the name and value are given as a UTF-8 strings with lengths.

- EXTERNXML int rtXmlEncUTF8Str (OSCTXT * pctxt, const OSUTF8CHAR * value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a UTF-8 string value.

- EXTERNXML int rtXmlEncUInt (OSCTXT * pctxt, OSUINT32 value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD unsigned integer type.

- EXTERNXML int rtXmlEncUIntValue (OSCTXT * pctxt, OSUINT32 value)

  This function encodes a variable of the XSD unsigned integer type.

- EXTERNXML int rtXmlEncUIntAttr (OSCTXT * pctxt, OSUINT32 value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen)

  This function encodes a variable of the XSD unsigned integer type as an attribute (name="value").

- EXTERNXML int rtXmlEncUInt64 (OSCTXT * pctxt, OSUINT64 value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD integer type.

- EXTERNXML int rtXmlEncUInt64Pattern (OSCTXT * pctxt, OSUINT64 value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, const OSUTF8CHAR * pattern)
- EXTERNXML int rtXmlEncUInt64Value (OSCTXT * pctxt, OSUINT64 value)

  This function encodes a variable of the XSD integer type.
- EXTERNXML int rtXmlEncUInt64PatternValue (OSCTXT * pctxt, OSUINT64 value, const OSUTF8CHAR * pattern)
- EXTERNXML int rtXmlEncUInt64Attr (OSCTXT * pctxt, OSUINT64 value, const OSUTF8CHAR * attrName, OSSIZE attrNameLen)

  This function encodes a variable of the XSD integer type as an attribute (name="value").
- EXTERNXML int rtXmlEncXSIAttrs (OSCTXT * pctxt, OSBOOL needXSI)

  This function encodes XML schema instance (XSI) attributes at the beginning of an XML document.
- EXTERNXML int rtXmlEncXSITypeAttr (OSCTXT * pctxt, const OSUTF8CHAR * value)

  This function encodes an XML schema instance (XSI) type attribute value (xsi:type="value").
- EXTERNXML int rtXmlEncXSITypeAttr2 (OSCTXT * pctxt, const OSUTF8CHAR * typeNsUri, const OSUTF8CHAR * typeName)

  This function encodes an XML schema instance (XSI) type attribute value (xsi:type="pfx:value").
- EXTERNXML int rtXmlEncXSINilAttr (OSCTXT * pctxt)

  This function encodes an XML nil attribute (xsi:nil="true").
- EXTERNXML int rtXmlFreeInputSource (OSCTXT * pctxt)

  This function closes an input source that was previously created with one of the create input source functions such as 'rtXmlCreateFileInputSource'.
- EXTERNXML OSBOOL rtXmlStrCmpAsc (const OSUTF8CHAR * text1, const char * text2)
- EXTERNXML OSBOOL rtXmlStrnCmpAsc (const OSUTF8CHAR * text1, const char * text2, OSSIZE len)
- EXTERNXML int rtXmlSetEncBufPtr (OSCTXT * pctxt, OSOCTET * bufaddr, OSSIZE bufsiz)

  This function is used to set the internal buffer within the run-time library encoding context.
- EXTERNXML int rtXmlGetIndent (OSCTXT * pctxt)

  This function returns current XML output indent value.
- EXTERNXML OSBOOL rtXmlGetWriteBOM (OSCTXT * pctxt)

  This function returns whether the Unicode byte order mark will be encoded.
- EXTERNXML int rtXmlGetIndentChar (OSCTXT * pctxt)

  This function returns current XML output indent character value (default is space).
- EXTERNXML int rtXmlPrepareContext (OSCTXT * pctxt)

This function prepares the context for another encode by setting the state back to OSXMLINIT and moving the buffer's cursor back to the beginning of the buffer.

- EXTERNXML int rtXmlSetEncC14N (OSCTXT * pctxt, OSBOOL value)

  This function sets the option to encode in C14N mode.

- EXTERNXML int rtXmlSetEncXSINamespace (OSCTXT * pctxt, OSBOOL value)

  This function sets a flag in the context that indicates the XSI namespace declaration (xmlns:xsi) should be added to the encoded XML instance.

- EXTERNXML int rtXmlSetEncXSINilAttr (OSCTXT * pctxt, OSBOOL value)

  This function sets a flag in the context that indicates the XSI attribute declaration (xmlns:xsi) should be added to the encoded XML instance.

- EXTERNXML int rtXmlSetDigitsFacets (OSCTXT * pctxt, int totalDigits, int fractionDigits)

- EXTERNXML int rtXmlSetEncDocHdr (OSCTXT * pctxt, OSBOOL value)

  This function sets the option to add the XML document header (i.e.

- EXTERNXML int rtXmlSetEncodingStr (OSCTXT * pctxt, const OSUTF8CHAR * encodingStr)

  This function sets the XML output encoding to the given value.

- EXTERNXML int rtXmlSetFormatting (OSCTXT * pctxt, OSBOOL doFormatting)

  This function sets XML output formatting to the given value.

- EXTERNXML int rtXmlSetIndent (OSCTXT * pctxt, OSUINT8 indent)

  This function sets XML output indent to the given value.

- EXTERNXML int rtXmlSetIndentChar (OSCTXT * pctxt, char indentChar)

  This function sets XML output indent character to the given value.

- EXTERNXML void rtXmlSetNamespacesSet (OSCTXT * pctxt, OSBOOL value)

  This function sets the context 'namespaces are set' flag.

- EXTERNXML int rtXmlSetNSPrefixLinks (OSCTXT * pctxt, OSRTDList * pNSAttrs)

  This function sets namespace prefix/URI links in the namspace prefix stack in the context structure.

- EXTERNXML int rtXmlSetSchemaLocation (OSCTXT * pctxt, const OSUTF8CHAR * schemaLocation)

  This function sets the XML Schema Instance (xsi) schema location attribute to be added to an encoded document.

- EXTERNXML int rtXmlSetNoNSSchemaLocation (OSCTXT * pctxt, const OSUTF8CHAR * schemaLocation)

  This function sets the XML Schema Instance (xsi) no namespace schema location attribute to be added to an encoded document.

- EXTERNXML void rtXmlSetSoapVersion (OSCTXT * pctxt, OSUINT8 version)

This function sets the SOAP version number.

- EXTERNXML int rtXmlSetXSITypeAttr (OSCTXT * pctxt, const OSUTF8CHAR * xsiType)

  This function sets the XML Schema Instance (xsi) type attribute value.

- EXTERNXML int rtXmlSetWriteBOM (OSCTXT * pctxt, OSBOOL write)

  This function sets whether the Unicode byte order mark is encoded.

- EXTERNXML int rtXmlMatchHexStr (OSCTXT * pctxt, OSSIZE minLength, OSSIZE maxLength)

  This function tests the context buffer for containing a correct hexadecimal string.

- EXTERNXML int rtXmlMatchBase64Str (OSCTXT * pctxt, OSSIZE minLength, OSSIZE maxLength)

  This function tests the context buffer for containing a correct base64 string.

- EXTERNXML int rtXmlMatchDate (OSCTXT * pctxt)

  This function tests the context buffer for containing a correct date string.

- EXTERNXML int rtXmlMatchTime (OSCTXT * pctxt)

  This function tests the context buffer for containing a correct time string.

- EXTERNXML int rtXmlMatchDateTime (OSCTXT * pctxt)

  This function tests the context buffer for containing a correct dateTime string.

- EXTERNXML int rtXmlMatchGYear (OSCTXT * pctxt)

  This function tests the context buffer for containing a correct gYear string.

- EXTERNXML int rtXmlMatchGYearMonth (OSCTXT * pctxt)

  This function tests the context buffer for containing a correct gYearMonth string.

- EXTERNXML int rtXmlMatchGMonth (OSCTXT * pctxt)

  This function tests the context buffer for containing a correct gMonth string.

- EXTERNXML int rtXmlMatchGMonthDay (OSCTXT * pctxt)

  This function tests the context buffer for containing a correct gMonthDay string.

- EXTERNXML int rtXmlMatchGDay (OSCTXT * pctxt)

  This function tests the context buffer for containing a correct gDay string.

- EXTERNXML OSUTF8CHAR * rtXmlNewQName (OSCTXT * pctxt, const OSUTF8CHAR * localName, const OSUTF8CHAR * prefix)

  This function creates a new QName given the localName and prefix parts.

- EXTERNXML OSBOOL rtXmlCmpBase64Str (OSUINT32 nocts1, const OSOCTET * data1, const OSUTF8CHAR * data2)

This function compares an array of octets to a base64 string.

- EXTERNXML OSBOOL rtXmlCmpHexStr (OSUINT32 nocts1, const OSOCTET * data1, const OSUTF8CHAR * data2)

  This function compares an array of octets to a hex string.

- EXTERNXML OSBOOL rtXmlCmpHexChar (OSUTF8CHAR ch, OSOCTET hexval)

- EXTERNXML int rtSaxGetAttributeID (const OSUTF8CHAR * attrName, OSSIZE nAttr, const OSUTF8CHAR * attrNames, OSUINT32 attrPresent)

- EXTERNXML const OSUTF8CHAR * rtSaxGetAttrValue (const OSUTF8CHAR * attrName, const OSUTF8CHAR *const * attrs)

  This function looks up an attribute in the attribute array returned by SAX to the startElement function.

- EXTERNXML OSINT16 rtSaxGetElemID (OSINT16 * pState, OSINT16 prevElemIdx, const OSUTF8CHAR * localName, OSINT32 nsidx, const OSSAXElemTableRec idtab, const OSINT16 * fstab, OSINT16 fstabRows, OSINT16 fstabCols)

  This function looks up a sequence element name in the given element info array.

- EXTERNXML OSINT16 rtSaxGetElemID8 (OSINT16 * pState, OSINT16 prevElemIdx, const OSUTF8CHAR * localName, OSINT32 nsidx, const OSSAXElemTableRec idtab, const OSINT8 * fstab, OSINT16 fstabRows, OSINT16 fstabCols)

  This function is a space optimized version of \c rtSaxGetElemID.

- EXTERNXML OSINT16 rtSaxFindElemID (OSINT16 * pState, OSINT16 prevElemIdx, const OSUTF8CHAR * localName, OSINT32 nsidx, const OSSAXElemTableRec idtab, const OSINT16 * fstab, OSINT16 fstabRows, OSINT16 fstabCols)

- EXTERNXML OSINT16 rtSaxFindElemID8 (OSINT16 * pState, OSINT16 prevElemIdx, const OSUTF8CHAR * localName, OSINT32 nsidx, const OSSAXElemTableRec idtab, const OSINT8 * fstab, OSINT16 fstabRows, OSINT16 fstabCols)

- EXTERNXML OSBOOL rtSaxHasXMLNSAttrs (const OSUTF8CHAR *const * attrs)

  This function checks if the given attribute list contains one or more XML namespace attributes (xmlns).

- EXTERNXML OSBOOL rtSaxIsEmptyBuffer (OSCTXT * pctxt)

  This function checks if the buffer in the context is empty or not.

- EXTERNXML OSINT16 rtSaxLookupElemID (OSCTXT * pctxt, OSINT16 * pState, OSINT16 prevElemIdx, const OSUTF8CHAR * localName, const OSUTF8CHAR * qName, OSINT32 nsidx, const OSSAXElemTableRec idtab, const OSINT16 * fstab, OSINT16 fstabRows, OSINT16 fstabCols)

- EXTERNXML OSINT16 rtSaxLookupElemID8 (OSCTXT * pctxt, OSINT16 * pState, OSINT16 prevElemIdx, const OSUTF8CHAR * localName, const OSUTF8CHAR * qName, OSINT32 nsidx, const OSSAXElemTableRec idtab, const OSINT8 * fstab, OSINT16 fstabRows, OSINT16 fstabCols)

- EXTERNXML int rtSaxStrListParse (OSCTXT * pctxt, OSRTMEMBUF * pMemBuf, OSRTDList * pvalue)

  This function parses the list of strings.

- EXTERNXML int rtSaxSortAttrs (OSCTXT * pctxt, const OSUTF8CHAR *const * attrs, OSUINT16 ** order)

  This function sorts a SAX attribute list in ascending order based on attribute name.

- EXTERNXML int rtSaxStrListMatch (OSCTXT * pctxt)

  This function mathes the list of strings.

- EXTERNXML OSBOOL rtSaxTestFinal (OSINT16 state, OSINT16 currElemIdx, const int * fstab, int fstabRows, int fstabCols)

- EXTERNXML OSBOOL rtSaxTestFinal8 (OSINT16 state, OSINT16 currElemIdx, const OSINT8 * fstab, int fstabRows, int fstabCols)

- EXTERNXML int rtSaxSetSkipLevelToCurrent (OSCTXT * pctxt, int stat)

- EXTERNXML OSUINT32 rtSaxSetMaxErrors (OSCTXT * pctxt, OSUINT32 maxErrors)

- EXTERNXML OSUINT32 rtSaxGetMaxErrors (OSCTXT * pctxt)

- EXTERNXML int rtSaxTestAttributesPresent (OSCTXT * pctxt, const OSUINT32 * attrPresent, const OSUINT32 * reqAttrMask, const OSUTF8CHAR *const * attrNames, OSSIZE numOfAttrs, const char * parentTypeName)

- EXTERNXML OSBOOL rtSaxIncErrors (OSCTXT * pctxt)

- EXTERNXML int rtSaxReportUnexpAttrs (OSCTXT * pctxt, const OSUTF8CHAR *const * attrs, const char * typeName)

- EXTERNXML int rtXmlWriteToFile (OSCTXT * pctxt, const char * filename)

  This function writes the encoded XML message stored in the context message buffer out to a file.

- EXTERNXML int rtXmlWriteUTF16ToFile (OSCTXT * pctxt, const char * filename)

- EXTERNXML void rtXmlTreatWhitespaces (OSCTXT * pctxt, int whiteSpaceType)

- EXTERNXML int rtXmlCheckBuffer (OSCTXT * pctxt, OSSIZE byte_count)

- EXTERNXML void rtErrXmlInit (OSVOIDARG)

- EXTERNXML int rtXmlPutChar (OSCTXT * pctxt, const OSUTF8CHAR value)

- EXTERNXML int rtXmlWriteChars (OSCTXT * pctxt, const OSUTF8CHAR * value, OSSIZE len)

- EXTERNXML int rtXmlpDecAny (OSCTXT * pctxt, const OSUTF8CHAR ** pvalue)

  This function decodes an arbitrary XML section of code as defined by the XSD any type (xsd:any).

- EXTERNXML int rtXmlpDecAny2 (OSCTXT * pctxt, OSUTF8CHAR ** pvalue)

  This version of the rtXmlpDecAny function returns the string in a mutable buffer.

- EXTERNXML int rtXmlpDecAnyAttrStr (OSCTXT * pctxt, const OSUTF8CHAR ** ppAttrStr, OSSIZE attrIndex)

  This function decodes an any attribute string.

- EXTERNXML int rtXmlpDecAnyElem (OSCTXT * pctxt, const OSUTF8CHAR ** pvalue)

  This function decodes an arbitrary XML section of code as defined by the XSD any type (xsd:any).

- EXTERNXML int rtXmlpDecBase64Str (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocts, OSSIZE bufsize)

  This function decodes a contents of a Base64-encode binary string into a static memory structure.

- EXTERNXML int rtXmlpDecBase64Str64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnocts, OSSIZE bufsize)

  This function is identical to rtXmlpDecBase64Str except that it supports 64-bit integer lengths on 64-bit systems.

- EXTERNXML int rtXmlpDecBigInt (OSCTXT * pctxt, const OSUTF8CHAR ** pvalue)

  This function will decode a variable of the XSD integer type.

- EXTERNXML int rtXmlpDecBitString (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnbits, OSUINT32 bufsize)

  This function decodes a bit string value.

- EXTERNXML int rtXmlpDecBitString64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnbits, OSSIZE bufsize)

  This function is identical to rtXmlpDecBitString except that it supports lengths up to 64-bits in size on 64-bit machines.

- EXTERNXML int rtXmlpDecBitStringExt (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnbits, OSOCTET ** ppextdata, OSUINT32 bufsize)

  This function decodes a bit string value.

- EXTERNXML int rtXmlpDecBitStringExt64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnbits, OSOCTET ** ppextdata, OSSIZE bufsize)

  This function is identical to rtXmlpDecBitStringExt except that it supports lengths up to 64-bits in size on 64-bit machines.

- EXTERNXML int rtXmlpDecBool (OSCTXT * pctxt, OSBOOL * pvalue)

  This function decodes a variable of the boolean type.

- EXTERNXML int rtXmlpDecDate (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'date' type.

- EXTERNXML int rtXmlpDecDateTime (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'dateTime' type.

- EXTERNXML int rtXmlpDecDecimal (OSCTXT * pctxt, OSREAL * pvalue, int totalDigits, int fractionDigits)

  This function decodes the contents of a decimal data type.

- EXTERNXML int rtXmlpDecDouble (OSCTXT * pctxt, OSREAL * pvalue)

  This function decodes the contents of a float or double data type.

- EXTERNXML int rtXmlpDecDoubleExt (OSCTXT * pctxt, OSUINT8 flags, OSREAL * pvalue)

  This function decodes the contents of a float or double data type.

- EXTERNXML int rtXmlpDecDynBase64Str (OSCTXT * pctxt, OSDynOctStr * pvalue)

  This function decodes a contents of a Base64-encode binary string.

- EXTERNXML int rtXmlpDecDynBase64Str64 (OSCTXT * pctxt, OSDynOctStr64 * pvalue)

  This function is identical to rtXmlpDecDynBase64Str except that it supports 64-bit integer lengths on 64-bit systems.

- EXTERNXML int rtXmlpDecDynBitString (OSCTXT * pctxt, OSDynOctStr * pvalue)

  This function decodes a bit string value.

- EXTERNXML int rtXmlpDecDynHexStr (OSCTXT * pctxt, OSDynOctStr * pvalue)

  This function decodes a contents of a hexBinary string.

- EXTERNXML int rtXmlpDecDynHexStr64 (OSCTXT * pctxt, OSDynOctStr64 * pvalue)

  This function is identical to the rtXmlpDecDynHexStr except that it supports lengths up to 64 bits in size on 64-bit systems.

- EXTERNXML int rtXmlpDecDynUnicodeStr (OSCTXT * pctxt, const OSUNICHAR ** ppdata, OSSIZE * pnchars)

  This function decodes a Unicode string data type.

- EXTERNXML int rtXmlpDecDynUTF8Str (OSCTXT * pctxt, const OSUTF8CHAR ** outdata)

  This function decodes the contents of a UTF-8 string data type.

- EXTERNXML int rtXmlpDecUTF8Str (OSCTXT * pctxt, OSUTF8CHAR * out, OSSIZE max_len)

  This function decodes the contents of a UTF-8 string data type.

- EXTERNXML int rtXmlpDecGDay (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gDay' type.

- EXTERNXML int rtXmlpDecGMonth (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gMonth' type.

- EXTERNXML int rtXmlpDecGMonthDay (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gMonthDay' type.

- EXTERNXML int rtXmlpDecGYear (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gYear' type.

- EXTERNXML int rtXmlpDecGYearMonth (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'gYearMonth' type.

- EXTERNXML int rtXmlpDecHexStr (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocts, OSSIZE bufsize)

  This function decodes the contents of a hexBinary string into a static memory structure.

- EXTERNXML int rtXmlpDecHexStr64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnocts, OSSIZE bufsize)

  This function is identical to rtXmlpDecHexStr except that it supports lengths up to 64-bits in size on 64-bit machines.

- EXTERNXML int rtXmlpDecInt (OSCTXT * pctxt, OSINT32 * pvalue)

  This function decodes the contents of a 32-bit integer data type.

- EXTERNXML int rtXmlpDecInt8 (OSCTXT * pctxt, OSINT8 * pvalue)

  This function decodes the contents of an 8-bit integer data type (i.e.

- EXTERNXML int rtXmlpDecInt16 (OSCTXT * pctxt, OSINT16 * pvalue)

  This function decodes the contents of a 16-bit integer data type.

- EXTERNXML int rtXmlpDecInt64 (OSCTXT * pctxt, OSINT64 * pvalue)

  This function decodes the contents of a 64-bit integer data type.

- EXTERNXML int rtXmlpDecNamedBits (OSCTXT * pctxt, const OSBitMapItem * pBitMap, OSOCTET * pvalue, OSUINT32 * pnbits, OSUINT32 bufsize)

  This function decodes the contents of a named bit field.

- EXTERNXML int rtXmlpDecNamedBits64 (OSCTXT * pctxt, const OSBitMapItem * pBitMap, OSOCTET * pvalue, OSSIZE * pnbits, OSSIZE bufsize)

  This function decodes the contents of a named bit field.

- EXTERNXML int rtXmlpDecStrList (OSCTXT * pctxt, OSRTDList * plist)

  This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.

- EXTERNXML int rtXmlpDecTime (OSCTXT * pctxt, OSXSDDateTime * pvalue)

  This function decodes a variable of the XSD 'time' type.

- EXTERNXML int rtXmlpDecUInt (OSCTXT * pctxt, OSUINT32 * pvalue)

  This function decodes the contents of an unsigned 32-bit integer data type.

- EXTERNXML int rtXmlpDecUInt8 (OSCTXT * pctxt, OSOCTET * pvalue)

  This function decodes the contents of an unsigned 8-bit integer data type (i.e.

- EXTERNXML int rtXmlpDecUInt16 (OSCTXT * pctxt, OSUINT16 * pvalue)

  This function decodes the contents of an unsigned 16-bit integer data type.

- EXTERNXML int rtXmlpDecUInt64 (OSCTXT * pctxt, OSUINT64 * pvalue)

  This function decodes the contents of an unsigned 64-bit integer data type.

- EXTERNXML int rtXmlpDecXmlStr (OSCTXT * pctxt, OSXMLSTRING * outdata)

  This function decodes the contents of an XML string data type.

- EXTERNXML int rtXmlpDecXmlStrList (OSCTXT * pctxt, OSRTDList * plist)

  This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.

- EXTERNXML int rtXmlpDecXSIAttr (OSCTXT * pctxt, const OSXMLNameFragments * attrName)

  This function decodes XSI (XML Schema Instance) attributes that may be present in any arbitrary XML element within a document.

- EXTERNXML int rtXmlpDecXSITypeAttr (OSCTXT * pctxt, const OSXMLNameFragments * attrName, const OSUTF8CHAR ** ppAttrValue)

  This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).

- EXTERNXML int rtXmlpGetAttributeID (const OSXMLStrFragment * attrName, OSINT16 nsidx, OSSIZE nAttr, const OSXMLAttrDescr attrNames, OSUINT32 attrPresent)

  This function finds an attribute in the descriptor table.

- EXTERNXML int rtXmlpGetNextElem (OSCTXT * pctxt, OSXMLElemDescr * pElem, OSINT32 level)

  This function parse the next element start tag.

- EXTERNXML int rtXmlpGetNextElemID (OSCTXT * pctxt, const OSXMLElemIDRec * tab, OSSIZE nrows, OSINT32 level, OSBOOL continueParse)

  This function parses the next start tag and finds the index of the element name in the descriptor table.

- EXTERNXML int rtXmlpMarkLastEventActive (OSCTXT * pctxt)

  This function marks current tag as unprocessed.

- EXTERNXML int rtXmlpMatchStartTag (OSCTXT * pctxt, const OSUTF8CHAR * elemLocalName, OSINT16 nsidx)

  This function parses the next start tag that matches with given name.

- EXTERNXML int rtXmlpMatchEndTag (OSCTXT * pctxt, OSINT32 level)

  This function parse next end tag that matches with given name.

- EXTERNXML OSBOOL rtXmlpHasAttributes (OSCTXT * pctxt)

  This function checks accessibility of attributes.

- EXTERNXML int rtXmlpGetAttributeCount (OSCTXT * pctxt)

  This function returns number of attributes in last processed start tag.

- EXTERNXML int rtXmlpSelectAttribute (OSCTXT * pctxt, OSXMLNameFragments * pAttr, OSINT16 * nsidx, OSSIZE attrIndex)

  This function selects attribute to decode.

- EXTERNXML OSINT32 rtXmlpGetCurrentLevel (OSCTXT * pctxt)

  This function returns current nesting level.

- EXTERNXML void rtXmlpSetWhiteSpaceMode (OSCTXT * pctxt, OSXMLWhiteSpaceMode whiteSpaceMode)

Sets the whitespace treatment mode.

- EXTERNXML OSBOOL rtXmlpSetMixedContentMode (OSCTXT * pctxt, OSBOOL mixedContentMode)

  Sets mixed content mode.

- EXTERNXML void rtXmlpSetListMode (OSCTXT * pctxt)

  Sets list mode.

- EXTERNXML OSBOOL rtXmlpListHasItem (OSCTXT * pctxt)

  Check for end of decoded token list.

- EXTERNXML void rtXmlpCountListItems (OSCTXT * pctxt, OSSIZE * itemCnt)

  Count tokens in list.

- EXTERNXML int rtXmlpGetNextSeqElemID2 (OSCTXT * pctxt, const OSXMLElemIDRec * tab, const OSXMLGroupDesc * pGroup, int groups, int curID, int lastMandatoryID, OSBOOL groupMode, OSBOOL checkRepeat)

  This function parses the next start tag and finds index of element name in descriptor table.

- EXTERNXML int rtXmlpGetNextSeqElemID (OSCTXT * pctxt, const OSXMLElemIDRec * tab, const OSXMLGroupDesc * pGroup, int curID, int lastMandatoryID, OSBOOL groupMode)

  This function parses the next start tag and finds index of element name in descriptor table.

- EXTERNXML int rtXmlpGetNextSeqElemIDExt (OSCTXT * pctxt, const OSXMLElemIDRec * tab, const OSXMLGroupDesc * ppGroup, const OSBOOL * extRequired, int postExtRootID, int curID, int lastMandatoryID, OSBOOL groupMode)

  This is an ASN.1 extension-supporting version of rtXmlpGetNextSeqElemID.

- EXTERNXML int rtXmlpGetNextAllElemID (OSCTXT * pctxt, const OSXMLElemIDRec * tab, OSSIZE nrows, const OSUINT8 * pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)

  This function parses the next start tag and finds index of element name in descriptor table.

- EXTERNXML int rtXmlpGetNextAllElemID16 (OSCTXT * pctxt, const OSXMLElemIDRec * tab, OSSIZE nrows, const OSUINT16 * pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)

  This function parses the next start tag and finds index of element name in descriptor table.

- EXTERNXML int rtXmlpGetNextAllElemID32 (OSCTXT * pctxt, const OSXMLElemIDRec * tab, OSSIZE nrows, const OSUINT32 * pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)

  This function parses the next start tag and finds index of element name in descriptor table.

- EXTERNXML void rtXmlpSetNamespaceTable (OSCTXT * pctxt, const OSUTF8CHAR * namespaceTable, OSSIZE nmNamespaces)

  Sets user namespace table.

- EXTERNXML int rtXmlpCreateReader (OSCTXT * pctxt)

  Creates pull parser reader structure within the context.

- EXTERNXML void rtXmlpHideAttributes (OSCTXT * pctxt)

  Disable access to attributes.

- EXTERNXML OSBOOL rtXmlpNeedDecodeAttributes (OSCTXT * pctxt)

  This function checks if attributes were previously decoded.

- EXTERNXML void rtXmlpMarkPos (OSCTXT * pctxt)

  Save current decode position.

- EXTERNXML void rtXmlpRewindToMarkedPos (OSCTXT * pctxt)

  Rewind to saved decode position.

- EXTERNXML void rtXmlpResetMarkedPos (OSCTXT * pctxt)

  Reset saved decode position.

- EXTERNXML int rtXmlpGetXSITypeAttr (OSCTXT * pctxt, const OSUTF8CHAR ** ppAttrValue, OSINT16 * nsidx, OSSIZE * pLocalOffs)

  This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).

- EXTERNXML int rtXmlpGetXmlnsAttrs (OSCTXT * pctxt, OSRTDList * pNSAttrs)

  This function decodes namespace attributes from start tag and adds them to the given list.

- EXTERNXML int rtXmlpDecXSIAttrs (OSCTXT * pctxt)

  This function decodes XSI (XML Schema Instance) that may be present in any arbitrary XML element within a document.

- EXTERNXML OSBOOL rtXmlpIsEmptyElement (OSCTXT * pctxt)

  Check element content: empty or not.

- EXTERNXML int rtXmlEncAttrC14N (OSCTXT * pctxt)

  This function used only in C14 mode.

- EXTERNXML struct OSXMLReader * rtXmlpGetReader (OSCTXT * pctxt)

  This function fetches the XML reader structure from the context for use in low-level pull parser calls.

- EXTERNXML OSBOOL rtXmlpIsLastEventDone (OSCTXT * pctxt)

  Check processing status of current tag.

- EXTERNXML int rtXmlpGetXSITypeIndex (OSCTXT * pctxt, const OSXMLItemDescr typetab, OSSIZE typetabsiz)

  This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type) and find type index in descriptor table.

- EXTERNXML int rtXmlpLookupXSITypeIndex (OSCTXT * pctxt, const OSUTF8CHAR * pXsiType, OSINT16 xsiTypeIdx, const OSXMLItemDescr typetab, OSSIZE typetabsiz)

This function find index of XSI (XML Schema Instance) type in descriptor table.

- EXTERNXML void rtXmlpForceDecodeAsGroup (OSCTXT * pctxt)

  Disable skipping of unknown elements in optional sequence tail.

- EXTERNXML OSBOOL rtXmlpIsDecodeAsGroup (OSCTXT * pctxt)

  This function checks if "decode as group" mode was forced.

- EXTERNXML OSBOOL rtXmlpIsUTF8Encoding (OSCTXT * pctxt)

  This function checks if the encoding specified in XML header is UTF-8.

- EXTERNXML int rtXmlpReadBytes (OSCTXT * pctxt, OSOCTET * pbuf, OSSIZE nbytes)

  This function reads the specified number of bytes directly from the underlying XML parser stream.

Detailed Description

XML low-level C encode/decode functions.

Definition in file osrtxml.h

OSXMLDecodeBuffer.h File Reference

```
#include "rtxsrc/OSRTInputStream.h"
#include "rtxmlsrc/OSXMLMessageBuffer.h"
#include "rtxmlsrc/rtSaxCppParserIF.h"
```

Classes

- struct OSXMLDecodeBuffer

  The OSXMLDecodeBuffer class is derived from the OSXMLMessageBuffer base class.

Detailed Description

XML decode buffer or stream class definition.

Definition in file OSXMLDecodeBuffer.h

OSXMLEncodeBuffer.h File Reference

```
#include "rtxmlsrc/OSXMLMessageBuffer.h"
```

Classes

- struct OSXMLEncodeBuffer

  The OSXMLEncodeBuffer class is derived from the OSXMLEncodeBase class.

Detailed Description

XML encode message buffer class definition.

Definition in file OSXMLEncodeBuffer.h

OSXMLEncodeStream.h File Reference

```
#include "rtxsrc/OSRTOutputStream.h"
```

```
#include "rtxmlsrc/OSXMLMessageBuffer.h"
```

Classes

- struct OSXMLEncodeStream

  The OSXMLEncodeStream class is derived from the OSXMLEncodeBase class.

Detailed Description

XML encode stream class definition.

Definition in file OSXMLEncodeStream.h

OSXMLMessageBuffer.h File Reference

```
#include "rtxsrc/OSRTMsgBuf.h"
```

```
#include "rtxmlsrc/osrtxml.h"
```

Classes

- struct OSXMLMessageBuffer

  The XML message buffer class is derived from the OSMessageBuffer base class.

- struct OSXMLEncodeBase

  OSXMLEncodeBase is a base class for the XML encode buffer and stream classes, OSXMLEncodeBuffer and OSXMLEncodeStream.

Detailed Description

XML encode/decode buffer and stream base class.

Definition in file OSXMLMessageBuffer.h

rtSaxCppAny.h File Reference

```
#include "rtxsrc/OSRTContext.h"
```

```
#include "rtxmlsrc/osrtxml.h"
```

```
#include "rtxmlsrc/rtSaxCppParser.h"

#include "rtxmlsrc/rtXmlCppMsgBuf.h"

#include "rtxsrc/rtxCppXmlString.h"

#include "rtxmlsrc/OSXSDAnyTypeClass.h"

#include "rtxmlsrc/rtSaxCppAnyType.h"
```

Classes

- struct OSXMLAnyHandler

Detailed Description

Definition in file rtSaxCppAny.h

rtSaxCppAnyType.h File Reference

```
#include "rtxsrc/OSRTContext.h"

#include "rtxmlsrc/osrtxml.h"

#include "rtxmlsrc/rtSaxCppParser.h"

#include "rtxmlsrc/rtXmlCppMsgBuf.h"

#include "rtxsrc/rtxCppXmlString.h"

#include "rtxmlsrc/OSXSDAnyTypeClass.h"
```

Classes

- struct OSXMLAnyTypeHandler

Detailed Description

Definition in file rtSaxCppAnyType.h

rtSaxCppParser.h File Reference

```
#include <string.h>

#include "rtxsrc/rtxErrCodes.h"

#include "rtxsrc/OSRTContext.h"

#include "rtxmlsrc/rtSaxCppParserIF.h"

#include "rtxmlsrc/osrtxml.h"

#include "rtxmlsrc/rtSaxDefs.h"
```

Classes

- struct OSXMLBasePtr
- struct OSXMLDefaultHandler

  This class is derived from the SAX class DefaultHandler base class.
- struct OSXMLDefaultHandler::ErrorInfo
- struct OSXMLDefaultHandlerPtr
- struct OSXMLParserCtxt

Macros

- #define OSCPPSAXDIAGSTART traceStartElement (funcName, localName)
- #define OSCPPSAXDIAGEND traceEndElement (funcName, localName)

Functions

- int operator!= (const void * ptr, const OSXMLDefaultHandlerPtr & ptr2)
- int operator== (const void * ptr, const OSXMLDefaultHandlerPtr & ptr2)

Detailed Description

Definition in file rtSaxCppParser.h

rtSaxCppParserIF.h File Reference

```
#include "rtxsrc/rtxErrCodes.h"
#include "rtxsrc/OSRTInputStreamIF.h"
#include "rtxmlsrc/osrtxml.h"
#include "rtxmlsrc/rtSaxDefs.h"
```

Classes

- struct OSXMLErrorInfo
- struct OSXMLErrorHandler
- struct OSXMLContentHandler

  Receive notification of general document events.
- struct OSXMLBase
- struct OSXMLReaderClass
- struct OSXMLDefaultHandlerIF

This class is derived from the SAX class DefaultHandler base class.

- struct OSXMLParserCtxtIF

Macros

- #define DECLARE_XMLBASEIMP public: void release () { delete this; }

Functions

- OSXMLReaderClass * rtSaxCppCreateXmlReader (OSXMLParserCtxtIF * pContext, OSXMLDefaultHandlerIF * pSaxHandler)

  This function creates XML reader instance.

- int rtSaxCppEnableThreadSafety ( )

  Enables the parser's thread-safe mode (if applicable).

- void rtSaxCppLockXmlLibrary ( )

  This function locks XML library.

- void rtSaxCppUnlockXmlLibrary ( )

  This function unlocks XML library.

Detailed Description

Definition in file rtSaxCppParserIF.h

rtSaxCppSimpleType.h File Reference

```
#include "rtxmlsrc/osrtxml.h"
```

```
#include "rtxmlsrc/rtSaxCppParser.h"
```

Classes

- struct OSXMLSimpleTypeHandler

Detailed Description

Definition in file rtSaxCppSimpleType.h

rtSaxCppSoap.h File Reference

```
#include "rtxsrc/rtxCppDynOctStr.h"
```

```
#include "rtxsrc/OSRTContext.h"
```

```
#include "rtxsrc/OSRTMemBuf.h"
```

```
#include "rtxsrc/rtxCppXmlString.h"

#include "rtxmlsrc/osrtxml.h"

#include "rtxmlsrc/rtSaxCppParser.h"

#include "rtxmlsrc/rtXmlCppMsgBuf.h"
```

Classes

- struct OSXMLSoapHandler

Detailed Description

Definition in file rtSaxCppSoap.h

rtSaxCppStrList.h File Reference

```
#include "rtxsrc/rtxCppDList.h"

#include "rtxmlsrc/osrtxml.h"

#include "rtxmlsrc/rtSaxCppParser.h"
```

Classes

- struct OSXMLStrListHandler

  OSXMLStrListHandler.

Detailed Description

Definition in file rtSaxCppStrList.h

rtXmlCppEncFuncs.h File Reference

```
#include "rtxmlsrc/osrtxml.h"

#include "rtxsrc/rtxCppDList.h"
```

Functions

- EXTERNXML int rtXmlCppEncAnyAttr (OSCTXT * pctxt, OSRTObjListClass * pAnyAttrList)

  This function encodes a variable of the XSD any attribute type.

- EXTERNXML int rtXmlEncAny (OSCTXT * pctxt, OSRTXMLString * pxmlstr, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD any type.

- EXTERNXML int rtXmlCppEncAnyTypeValue (OSCTXT * pctxt, OSXSDAnyTypeClass * pvalue)

This function encodes a variable of the XSD anyType type.

- EXTERNXML int rtXmlCppEncStartElement (OSCTXT * pctxt, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS, OSRTDListClass * pNSAttrs, OSBOOL terminate)

  This function encodes a start element tag value (\<elemName\>).

- EXTERNXML int rtXmlEncString (OSCTXT * pctxt, OSRTXMLString * pxmlstr, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)

  This function encodes a variable of the XSD string type.

Detailed Description

XML low-level C++ encode functions.

These are overloaded versions of C XML encode functions for use with C++.

Definition in file rtXmlCppEncFuncs.h

rtXmlCppMsgBuf.h File Reference

```
#include "rtxmlsrc/OSXMLEncodeBuffer.h"
#include "rtxmlsrc/OSXMLEncodeStream.h"
#include "rtxmlsrc/OSXMLDecodeBuffer.h"
```

Detailed Description

This file is deprecated.

Users should use one or more of the individual headers files defined in the #include statements below.

Definition in file rtXmlCppMsgBuf.h

rtXmlCppNamespace.h File Reference

```
#include "rtxmlsrc/osrtxml.h"
#include "rtxsrc/OSRTBaseType.h"
#include "rtxsrc/OSRTString.h"
```

Classes

- struct OSXMLNamespaceClass

  This class is used to hold an XML namespace prefix to URI mapping.

Detailed Description

XML namespace handling structures and function definitions.

Definition in file rtXmlCppNamespace.h

rtXmlCppXSDElement.h File Reference

```
#include "rtxsrc/OSRTContext.h"
#include "rtxsrc/OSRTMsgBufIF.h"
#include "rtxsrc/rtxDiag.h"
#include "rtxsrc/rtxErrCodes.h"
#include "rtxmlsrc/osrtxml.h"
```

Classes

- struct OSXSDGlobalElement

  XSD global element base class.

Detailed Description

C++ run-time XML schema global element class definition.

Definition in file rtXmlCppXSDElement.h

rtXmlpCppDecFuncs.h File Reference

```
#include "rtxmlsrc/osrtxml.h"
#include "rtxmlsrc/rtXmlPull.h"
#include "rtxmlsrc/OSXSDComplexType.h"
```

Classes

- struct OSXMLStringListParser

  Class enabling parsing of an XML Schema list into strings.

Functions

- EXTERNXML int rtXmlpCppDecStrList (OSCTXT * pctxt, OSRTObjListClass * plist)
- EXTERNXML int rtXmlpCppDecXmlStr (OSCTXT * pctxt, OSRTXMLString * outdata)
- EXTERNXML int rtXmlpCppGetXmlnsAttrs (OSCTXT * pctxt, OSXSDComplexType * ptype)

Detailed Description

XML low-level C++ decode functions.

These are overloaded versions of C XML encode functions for use with C++.

Definition in file rtXmlpCppDecFuncs.h