

ASN1C

ASN.1 Compiler
Version 6.0
XML Runtime
Reference Manual

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

Copyright Notice

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

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

Author's Contact Information

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

Contents

1	ASN1C XML Runtime Module Index	1
1.1	ASN1C XML Runtime Modules	1
2	ASN1C XML Runtime Hierarchical Index	2
2.1	ASN1C XML Runtime Class Hierarchy	2
3	ASN1C XML Runtime Class Index	3
3.1	ASN1C XML Runtime Class List	3
4	ASN1C XML Runtime File Index	4
4.1	ASN1C XML Runtime File List	4
5	ASN1C XML Runtime Module Documentation	5
5.1	Run-time error status codes.	5
5.2	ASN.1-XML encode/decode functions.	8
5.3	XML decode functions.	14
5.4	XML encode functions.	27
5.5	XML utility functions.	57
5.6	XML pull-parser decode functions.	58
6	ASN1C XML Runtime Class Documentation	73
6.1	OSXMLDecodeBuffer Class Reference	73
6.2	OSXMLEncodeBuffer Class Reference	78
6.3	OSXMLEncodeStream Class Reference	80
6.4	OSXMLMessageBuffer Class Reference	82
6.5	OSXSDGlobalElement Class Reference	85
7	ASN1C XML Runtime File Documentation	89
7.1	osrtxml.h File Reference	89
7.2	rtXmlCppMsgBuf.h File Reference	104
7.3	rtXmlCppXSDElement.h File Reference	105

7.4	rtXmlErrCodes.h File Reference	106
-----	--	-----

Chapter 1

ASN1C XML Runtime Module Index

1.1 ASN1C XML Runtime Modules

Here is a list of all modules:

Run-time error status codes.	5
ASN.1-XML encode/decode functions.	8
XML decode functions.	14
XML encode functions.	27
XML utility functions.	57
XML pull-parser decode functions.	58

Chapter 2

ASN1C XML Runtime Hierarchical Index

2.1 ASN1C XML Runtime Class Hierarchy

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

OSXMLMessageBuffer	82
OSXMLDecodeBuffer	73
OSXMLEncodeBuffer	78
OSXMLEncodeStream	80
OSXSDGlobalElement	85

Chapter 3

ASN1C XML Runtime Class Index

3.1 ASN1C XML Runtime Class List

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

OSXMLDecodeBuffer	73
OSXMLEncodeBuffer	78
OSXMLEncodeStream	80
OSXMLMessageBuffer	82
OSXSDGlobalElement	85

Chapter 4

ASN1C XML Runtime File Index

4.1 ASN1C XML Runtime File List

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

asn1xml.h	??
osrtxml.h	89
rtXmlCppMsgBuf.h	104
rtXmlCppXSDElement.h	105
rtXmlErrCodes.h	106

Chapter 5

ASN1C XML Runtime Module Documentation

5.1 Run-time error status codes.

Defines

- #define XML_OK_EOB 0x7ffffff
- #define XML_OK_FRAG XML_OK_EOB
- #define XML_E_BASE -200
- #define XML_E_GENERR (XML_E_BASE)
- #define XML_E_INVSYMBOL (XML_E_BASE-1)
- #define XML_E_TAGMISMATCH (XML_E_BASE-2)
- #define XML_E_DUPLATTR (XML_E_BASE-3)
- #define XML_E_BADCHARREF (XML_E_BASE-4)
- #define XML_E_INVMODE (XML_E_BASE-5)
- #define XML_E_UNEXPEOF (XML_E_BASE-6)
- #define XML_E_NOMATCH (XML_E_BASE-7)
- #define XML_E_ELEMMISRQ (XML_E_BASE-8)
- #define XML_E_ELEMSISRQ (XML_E_BASE-9)
- #define XML_E_TOOFWELEMS (XML_E_BASE-10)
- #define XML_E_UNEXPSTARTTAG (XML_E_BASE-11)
- #define XML_E_UNEXPENDTAG (XML_E_BASE-12)

5.1.1 Detailed Description

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

5.1.2 Define Documentation

5.1.2.1 #define XML_E_BADCHARREF (XML_E_BASE-4)

Bad character reference found.

5.1.2.2 #define XML_E_BASE -200

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

5.1.2.3 #define XML_E_DUPLATTR (XML_E_BASE-3)

Duplicate attribute found.

5.1.2.4 #define XML_E_ELEMMISRQ (XML_E_BASE-8)

Missing required element. This status code is returned by the decoder when the decoder knows exactly which element is absent.

5.1.2.5 #define XML_E_ELEMSMISRQ (XML_E_BASE-9)

Missing required elements. This status code is returned by the decoder when the number of elements decoded for a given content model group is less than the required number of elements as specified in the schema.

5.1.2.6 #define XML_E_GENERR (XML_E_BASE)

General error

5.1.2.7 #define XML_E_INVMODE (XML_E_BASE-5)

Invalid mode.

5.1.2.8 #define XML_E_INVSYMBOL (XML_E_BASE-1)

Invalid symbol is detected.

5.1.2.9 #define XML_E_NOMATCH (XML_E_BASE-7)

Current tag is not matched to specified one. Informational code.

5.1.2.10 #define XML_E_TAGMISMATCH (XML_E_BASE-2)

Start-end XML tags mismatch. Fatal error.

5.1.2.11 #define XML_E_TOOFWELEMS (XML_E_BASE-10)

minOccurs is not reached.

5.1.2.12 #define XML_E_UNEXPENDTAG (XML_E_BASE-12)

Unexpected end tag.

5.1.2.13 #define XML_E_UNEXPEOF (XML_E_BASE-6)

Unexpected end of file (document).

5.1.2.14 #define XML_E_UNEXPSTARTTAG (XML_E_BASE-11)

Unexpected start tag.

5.1.2.15 #define XML_OK_EOB 0x7fffffff

End of block marker.

5.1.2.16 #define XML_OK_FRAG XML_OK_EOB

Maintained for backward compatibility.

5.2 ASN.1-XML encode/decode functions.

Functions

- EXTERNXML int [rtAsn1XmlpDecObjId](#) (OSCTXT *pctx, ASN1OBJID *pvalue)
- EXTERNXML int [rtAsn1XmlpDecUnivStr](#) (OSCTXT *pctx, const OS32BITCHAR **ppdata, OSUINT32 *pnchars)
- EXTERNXML int [rtAsn1XmlEncGenTime](#) (OSCTXT *pctx, const char *value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtAsn1XmlEncUTCTime](#) (OSCTXT *pctx, const char *value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtAsn1XmlEncObjId](#) (OSCTXT *pctx, const ASN1OBJID *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtAsn1XmlEncRelOID](#) (OSCTXT *pctx, const ASN1OBJID *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtAsn1XmlEncOpenType](#) (OSCTXT *pctx, const OSOCTET *data, OSUINT32 nocts, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtAsn1XmlEncOpenTypeExt](#) (OSCTXT *pctx, OSRTDList *pElemList)
- EXTERNXML int [rtAsn1XmlEncUnivStr](#) (OSCTXT *pctx, const OS32BITCHAR *value, OSUINT32 nchars, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtAsn1XmlFmtAttrStr](#) (OSCTXT *pctx, const OSUTF8CHAR *name, const OSUTF8CHAR *value, OSUTF8CHAR **pAttrStr)
- EXTERNXML int [rtAsn1XmlParseAttrStr](#) (OSCTXT *pctx, const OSUTF8CHAR *pAttrStr, OSUTF8NVP *pNVPair)
- EXTERNXML int [rtAsn1XmlAddAnyAttr](#) (OSCTXT *pctx, const OSUTF8CHAR *name, const OSUTF8CHAR *value, OSRTDList *plist)
- EXTERNXML int [rtAsn1XmlpDecDynBitStr](#) (OSCTXT *pctx, ASN1DynBitStr *pvalue)
- EXTERNXML int [rtXmlpDecListOfASN1DynBitStr](#) (OSCTXT *pctx, OSRTDList *plist)
- EXTERNXML int [rtAsn1XmlpDecRelOID](#) (OSCTXT *pctx, ASN1OBJID *pvalue)

5.2.1 Function Documentation

5.2.1.1 EXTERNXML int [rtAsn1XmlAddAnyAttr](#) (OSCTXT * *pctx*, const OSUTF8CHAR * *name*, const OSUTF8CHAR * *value*, OSRTDList * *plist*)

This function formats an attribute string and adds it to the attribute list.

Parameters:

pctx 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.

name The name of the new attribute.

value The value of the new attribute.

plist The attribute list.

5.2.1.2 EXTERNXML int [rtAsn1XmlEncGenTime](#) (OSCTXT * *pctx*, const char * *value*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

This function encodes a variable of the ASN.1 GeneralizedTime type. It performs conversion from ASN.1 time format into the XML dateTime format.

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.
- value* A pointer to a null-terminated C character string to be encoded. It should contain UTCTime in ASN.1 format.
- elemName* XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
- nsPrefix* XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

Returns:

Completion status of operation:

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

5.2.1.3 EXTERNXML int rtAsn1XmlEncObjId (OSCTXT * pctxt, const ASN1OBJID * pvalue, const OSUTF8CHAR * elemName, const OSUTF8CHAR * nsPrefix)

This function encodes a variable of the ASN.1 OBJECT IDENTIFIER type.

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 object identifier structure. This structure contains an integer to hold the number of subidentifiers in the object and an array to hold the subidentifier values.
- elemName* XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
- nsPrefix* XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

Returns:

Completion status of operation:

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

5.2.1.4 EXTERNXML int rtAsn1XmlEncOpenType (OSCTXT * pctxt, const OSOCTET * data, OSUINT32 nocts, const OSUTF8CHAR * elemName, const OSUTF8CHAR * nsPrefix)

This function encodes a variable of the ASN.1 open type. It copies the data as it exists in the structure to the encode buffer or stream.

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.
- data* A pointer to a buffer containing the open type data.
- nocts* Number of bytes in the data buffer to encode.
- elemName* XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
- nsPrefix* XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

Returns:

Completion status of operation:

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

5.2.1.5 EXTERNXML int rtAsn1XmlEncOpenTypeExt (OSCTXT * *pctxt*, OSRTDList * *pElemList*)

This function encodes an ASN.1 open type extension. This occurs in a SEQUENCE or SET type when a ... is present. The type is represented as a list of open type structures.

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.

pElemList Linked list of ASN.1 open type structures.

Returns:

Completion status of operation:

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

5.2.1.6 EXTERNXML int rtAsn1XmlEncRelOID (OSCTXT * *pctxt*, const ASN1OBJID * *pvalue*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

This function encodes a variable of the ASN.1 RELATIVE-OID type.

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 object identifier structure. This structure contains an integer to hold the number of subidentifiers in the object and an array to hold the subidentifier values.

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

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

Returns:

Completion status of operation:

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

5.2.1.7 EXTERNXML int rtAsn1XmlEncUnivStr (OSCTXT * *pctxt*, const OS32BITCHAR * *value*, OSUINT32 *nchars*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

This function encodes a variable of the ASN.1 UNIVERSAL string type.

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.

value Array of universal characters to be encoded. Each character is represented as a 32-bit integer.

nchars Number of characters to encode.

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

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

Returns:

Completion status of operation:

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

5.2.1.8 EXTERNXML int rtAsn1XmlEncUTCTime (OSCTXT * *pctxt*, const char * *value*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

This function encodes a variable of the ASN.1 UTCTime type. It performs conversion from ASN.1 time format into the XML dateTime format.

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.

value A pointer to a null-terminated C character string to be encoded. It should contain UTCTime in ASN.1 format.

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

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

Returns:

Completion status of operation:

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

5.2.1.9 EXTERNXML int rtAsn1XmlFmtAttrStr (OSCTXT * *pctxt*, const OSUTF8CHAR * *name*, const OSUTF8CHAR * *value*, OSUTF8CHAR ** *pAttrStr*)

This function formats a name-value XML pair into a name="value" attribute string.

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.

name A pointer to an XML element name. A name must be provided.

value A pointer to the corresponding element value.

pAttrStr The resulting name="value" string.

Returns:

Completion status of operation:

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

5.2.1.10 EXTERNXML int rtAsn1XmlParseAttrStr (OSCTXT * *pctxt*, const OSUTF8CHAR * *pAttrStr*, OSUTF8NVP * *pNVPair*)

This function parses an XML name-value pair from an attribute string.

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.

pAttrStr The name-value string to be parsed.

pNVPair A pointer to an XML name-value pair structure filled in by invoking this method.

Returns:

Completion status of operation:

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

5.2.1.11 EXTERNXML int rtAsn1XmlpDecDynBitStr (OSCTXT * *pctxt*, ASN1DynBitStr * *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.

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.

5.2.1.12 EXTERNXML int rtAsn1XmlpDecObjId (OSCTXT * *pctxt*, ASN1OBJID * *pvalue*)

This function decodes the contents of an ASN.1 OBJECT IDENTIFIER type. Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

Parameters:

pctxt Pointer to context block structure.

pvalue Pointer to ASN.1 object identifier value to receive decoded result.

Returns:

Completion status of operation:

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

5.2.1.13 EXTERNXML int rtAsn1XmlpDecRelOID (OSCTXT * *pctxt*, ASN1OBJID * *pvalue*)

This function decodes the contents of an ASN.1 RELATIVE-OID type. Input is expected to be a string of OS-UTF8CHAR characters returned by an XML pull parser.

Parameters:

pctxt Pointer to context block structure.

pvalue Pointer to ASN.1 object identifier value to receive decoded result.

Returns:

Completion status of operation:

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

5.2.1.14 EXTERNXML int rtAsn1XmlpDecUnivStr (OSCTXT * *pctxt*, const OS32BITCHAR ** *ppdata*, OSUINT32 * *pchars*)

This function decodes the contents of an ASN.1 UNIVERSAL string type. Input is expected to be a string of OS-UTF8CHAR characters returned by an XML pull parser.

Parameters:

pctxt Pointer to context block structure.

ppdata Pointer to 32-bit character string value to receive decoded result.

pchars Pointer to length value to receive decoded length in characters.

Returns:

Completion status of operation:

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

5.2.1.15 EXTERNXML int rtXmlpDecListOfASN1DynBitStr (OSCTXT * *pctxt*, OSRTDList * *plist*)

This function decodes a list of space-separated bit strings and returns each bit string as a separate item on the given list. The string consists of a series of '1' and '0' characters. Memory is allocated for the list nodes and token values using the rtx memory management functions. Bits are stored from MSB to LSB order.

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 bit string values will be added.

Returns:

Completion status of operation:

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

5.3 XML decode functions.

Functions

- EXTERNXML int [rtXmlDecBase64Binary](#) (OSRTMEMBUF *pMemBuf, const OSUTF8CHAR *inpdata, int length)
- EXTERNXML int [rtXmlDecBase64Str](#) (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT16 *pnocts, OSINT32 bufsize)
- EXTERNXML int [rtXmlDecBase64StrValue](#) (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, size_t bufSize, size_t srcDataLen)
- EXTERNXML int [rtXmlDecBigInt](#) (OSCTXT *pctxt, const OSUTF8CHAR **ppvalue)
- EXTERNXML int [rtXmlDecBool](#) (OSCTXT *pctxt, OSBOOL *pvalue)
- EXTERNXML int [rtXmlDecDate](#) (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlDecTime](#) (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlDecDateTime](#) (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlDecDecimal](#) (OSCTXT *pctxt, OSREAL *pvalue, int totalDigits, int fractionDigits)
- EXTERNXML int [rtXmlDecDouble](#) (OSCTXT *pctxt, OSREAL *pvalue, int totalDigits, int fractionDigits)
- EXTERNXML int [rtXmlDecDynBase64Str](#) (OSCTXT *pctxt, OSDynOctStr *pvalue)
- EXTERNXML int [rtXmlDecDynHexStr](#) (OSCTXT *pctxt, OSDynOctStr *pvalue)
- EXTERNXML int [rtXmlDecDynUTF8Str](#) (OSCTXT *pctxt, const OSUTF8CHAR **outdata)
- EXTERNXML int [rtXmlDecHexBinary](#) (OSRTMEMBUF *pMemBuf, const OSUTF8CHAR *inpdata, int length)
- EXTERNXML int [rtXmlDecHexStr](#) (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT16 *pnocts, OSINT32 bufsize)
- EXTERNXML int [rtXmlDecHexStrValue](#) (OSCTXT *pctxt, const OSUTF8CHAR *const inpdata, size_t nbytes, OSOCTET *pvalue, OSUINT32 *pnbits, OSINT32 bufsize)
- EXTERNXML int [rtXmlDecGYear](#) (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlDecGYearMonth](#) (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlDecGMonth](#) (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlDecGMonthDay](#) (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlDecGDay](#) (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlDecInt](#) (OSCTXT *pctxt, OSINT32 *pvalue)
- EXTERNXML int [rtXmlDecInt8](#) (OSCTXT *pctxt, OSINT8 *pvalue)
- EXTERNXML int [rtXmlDecInt16](#) (OSCTXT *pctxt, OSINT16 *pvalue)
- EXTERNXML int [rtXmlDecInt64](#) (OSCTXT *pctxt, OSINT64 *pvalue)
- EXTERNXML int [rtXmlDecUInt](#) (OSCTXT *pctxt, OSUINT32 *pvalue)
- EXTERNXML int [rtXmlDecUInt8](#) (OSCTXT *pctxt, OSUINT8 *pvalue)
- EXTERNXML int [rtXmlDecUInt16](#) (OSCTXT *pctxt, OSUINT16 *pvalue)
- EXTERNXML int [rtXmlDecUInt64](#) (OSCTXT *pctxt, OSUINT64 *pvalue)
- EXTERNXML const OSUTF8CHAR * [rtXmlDecQName](#) (OSCTXT *pctxt, const OSUTF8CHAR *qname, const OSUTF8CHAR **prefix)
- EXTERNXML int [rtXmlDecXSIAttr](#) (OSCTXT *pctxt, const OSUTF8CHAR *attrName, const OSUTF8CHAR *attrValue)
- EXTERNXML int [rtXmlDecXSIAttrs](#) (OSCTXT *pctxt, const OSUTF8CHAR *const *attrs, const char *typeName)
- EXTERNXML int [rtXmlDecXmlStr](#) (OSCTXT *pctxt, OSXMLSTRING *outdata)
- EXTERNXML int [rtXmlParseElementName](#) (OSCTXT *pctxt, OSUTF8CHAR **ppName)
- EXTERNXML int [rtXmlParseElemQName](#) (OSCTXT *pctxt, OSXMLQName *pQName)

5.3.1 Function Documentation

5.3.1.1 EXTERNXML int rtXmlDecBase64Binary (OSRTMEMBUF * *pMemBuf*, const OSUTF8CHAR * *inpdata*, int *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.

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.

5.3.1.2 EXTERNXML int rtXmlDecBase64Str (OSCTXT * *pctxt*, OSOCTET * *pvalue*, OSUINT16 * *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.

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.

5.3.1.3 EXTERNXML int rtXmlDecBase64StrValue (OSCTXT * *pctxt*, OSOCTET * *pvalue*, OSUINT32 * *pnocts*, size_t *bufSize*, size_t *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.

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.

5.3.1.4 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.

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.

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

This function decodes a variable of the boolean type.

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.

5.3.1.6 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.

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.

5.3.1.7 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.

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.

5.3.1.8 EXTERNXML int rtXmlDecDecimal (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 characters returned by an XML parser.

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.

fractionDigits Number of fraction digits in the decimal number from XSD fractionDigits facet value.

Returns:

Completion status of operation:

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

5.3.1.9 EXTERNXML int rtXmlDecDouble (OSCTXT * *pctxt*, OSREAL * *pvalue*, int *totalDigits*, int *fractionDigits*)

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.

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.

5.3.1.10 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.

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.

5.3.1.11 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.

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.

5.3.1.12 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.

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.

5.3.1.13 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.

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.

5.3.1.14 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.

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.

5.3.1.15 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.

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.

5.3.1.16 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.

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.

5.3.1.17 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.

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.

5.3.1.18 EXTERNXML int rtXmlDecHexBinary (OSRTMEMBUF * *pMemBuf*, const OSUTF8CHAR * *inpdata*, int *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.

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.

5.3.1.19 EXTERNXML int rtXmlDecHexStr (OSCTXT * *pctxt*, OSOCTET * *pvalue*, OSUINT16 * *pnocets*, OSINT32 *bufsize*)

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

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.
- pnocets* 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.

5.3.1.20 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.

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.

5.3.1.21 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.

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.

5.3.1.22 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.

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.

5.3.1.23 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.

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.

5.3.1.24 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

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.

5.3.1.25 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`

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.

5.3.1.26 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 OS-UTF8CHAR characters returned by an XML parser.

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.

5.3.1.27 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 OS-UTF8CHAR characters returned by an XML parser.

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.

5.3.1.28 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 OS-UTF8CHAR characters returned by an XML parser.

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.

5.3.1.29 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.

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.

5.3.1.30 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 character 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.

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.

5.3.1.31 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.

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.

5.3.1.32 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.

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.

5.3.1.33 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.

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.

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

This function parses the initial tag from an XML message.

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.

5.4 XML encode functions.

Defines

- #define **rtXmlFinalizeMemBuf**(pMemBuf)
- #define **rtXmlGetEncBufPtr**(pctx) (pctx) → buffer.data
- #define **rtXmlGetEncBufLen**(pctx) (pctx) → buffer.byteIndex

Functions

- EXTERNXML int **rtXmlEncAny** (OSCTXT *pctx, OSXMLSTRING *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int **rtXmlEncAnyStr** (OSCTXT *pctx, const OSUTF8CHAR *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int **rtXmlEncAnyAttr** (OSCTXT *pctx, OSRTDList *pAnyAttrList)
- EXTERNXML int **rtXmlEncBase64Binary** (OSCTXT *pctx, OSUINT32 nocts, const OSOCTET *value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int **rtXmlEncBase64BinaryAttr** (OSCTXT *pctx, OSUINT32 nocts, const OSOCTET *value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen)
- EXTERNXML int **rtXmlEncBase64StrValue** (OSCTXT *pctx, OSUINT32 nocts, const OSOCTET *value)
- EXTERNXML int **rtXmlEncBigInt** (OSCTXT *pctx, const OSUTF8CHAR *value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int **rtXmlEncBigIntAttr** (OSCTXT *pctx, const OSUTF8CHAR *value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen)
- EXTERNXML int **rtXmlEncBigIntValue** (OSCTXT *pctx, const OSUTF8CHAR *value)
- EXTERNXML int **rtXmlEncBitString** (OSCTXT *pctx, OSUINT32 nbits, const OSOCTET *value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int **rtXmlEncBinStrValue** (OSCTXT *pctx, OSUINT32 nbits, const OSOCTET *data)
- EXTERNXML int **rtXmlEncBool** (OSCTXT *pctx, OSBOOL value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int **rtXmlEncBoolValue** (OSCTXT *pctx, OSBOOL value)
- EXTERNXML int **rtXmlEncBoolAttr** (OSCTXT *pctx, OSBOOL value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen)
- EXTERNXML int **rtXmlEncDate** (OSCTXT *pctx, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int **rtXmlEncDateValue** (OSCTXT *pctx, const OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlEncTime** (OSCTXT *pctx, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int **rtXmlEncTimeValue** (OSCTXT *pctx, const OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlEncDateTime** (OSCTXT *pctx, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int **rtXmlEncDateTimeValue** (OSCTXT *pctx, const OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlEncDecimal** (OSCTXT *pctx, OSREAL value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix, const OSDecimalFmt *pFmtSpec)
- EXTERNXML int **rtXmlEncDecimalAttr** (OSCTXT *pctx, OSREAL value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen, const OSDecimalFmt *pFmtSpec)
- EXTERNXML int **rtXmlEncDecimalValue** (OSCTXT *pctx, OSREAL value, const OSDecimalFmt *pFmtSpec, char *pDestBuf, size_t destBufSize)
- EXTERNXML int **rtXmlEncDouble** (OSCTXT *pctx, OSREAL value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix, const OSDoubleFmt *pFmtSpec)

- EXTERNXML int [rtXmlEncDoubleAttr](#) (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen, const OSDoubleFmt *pFmtSpec)
- EXTERNXML int [rtXmlEncDoubleValue](#) (OSCTXT *pctxt, OSREAL value, const OSDoubleFmt *pFmtSpec, int defaultPrecision)
- EXTERNXML int [rtXmlEncEmptyElement](#) (OSCTXT *pctxt, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix, OSBOOL terminate)
- EXTERNXML int [rtXmlEncEmptyElement2](#) (OSCTXT *pctxt, const OSUTF8CHAR *elemName, size_t elemLen, const OSUTF8CHAR *nsPrefix, size_t nsPrefixLen, OSBOOL terminate)
- EXTERNXML int [rtXmlEncEndDocument](#) (OSCTXT *pctxt)
- EXTERNXML int [rtXmlEncEndElement](#) (OSCTXT *pctxt, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncEndElement2](#) (OSCTXT *pctxt, const OSUTF8CHAR *elemName, size_t elemLen, const OSUTF8CHAR *nsPrefix, size_t nsPrefixLen)
- EXTERNXML int [rtXmlEncEndSoapEnv](#) (OSCTXT *pctxt)
- EXTERNXML int [rtXmlEncFloat](#) (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix, const OSDoubleFmt *pFmtSpec)
- EXTERNXML int [rtXmlEncFloatAttr](#) (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen, const OSDoubleFmt *pFmtSpec)
- EXTERNXML int [rtXmlEncGYear](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncGYearMonth](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncGMonth](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncGMonthDay](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncGDay](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncGYearValue](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlEncGYearMonthValue](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlEncGMonthValue](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlEncGMonthDayValue](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlEncGDayValue](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlEncHexBinary](#) (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncHexBinaryAttr](#) (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen)
- EXTERNXML int [rtXmlEncHexStrValue](#) (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *data)
- EXTERNXML int [rtXmlEncIndent](#) (OSCTXT *pctxt)
- EXTERNXML int [rtXmlEncInt](#) (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncIntValue](#) (OSCTXT *pctxt, OSINT32 value)
- EXTERNXML int [rtXmlEncIntAttr](#) (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen)
- EXTERNXML int [rtXmlEncIntPattern](#) (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix, const OSUTF8CHAR *pattern)
- EXTERNXML int [rtXmlEncIntPatternValue](#) (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *pattern)
- EXTERNXML int [rtXmlEncInt64](#) (OSCTXT *pctxt, OSINT64 value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncInt64Value](#) (OSCTXT *pctxt, OSINT64 value)

- EXTERNXML int [rtXmlEncInt64Attr](#) (OSCTXT *pctxt, OSINT64 value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen)
- EXTERNXML int [rtXmlEncNamedBits](#) (OSCTXT *pctxt, const OSBitMapItem *pBitMap, OSUINT32 nbits, const OSOCTET *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncNamedBitsValue](#) (OSCTXT *pctxt, const OSBitMapItem *pBitMap, OSUINT32 nbits, const OSOCTET *pvalue)
- EXTERNXML int [rtXmlEncNSAttrs](#) (OSCTXT *pctxt)
- EXTERNXML int [rtXmlEncSoapArrayTypeAttr](#) (OSCTXT *pctxt, const OSUTF8CHAR *name, const OSUTF8CHAR *value, size_t itemCount)
- EXTERNXML int [rtXmlEncSoapArrayTypeAttr2](#) (OSCTXT *pctxt, const OSUTF8CHAR *name, size_t nameLen, const OSUTF8CHAR *value, size_t valueLen, size_t itemCount)
- EXTERNXML int [rtXmlEncStartDocument](#) (OSCTXT *pctxt)
- EXTERNXML int [rtXmlEncStartElement](#) (OSCTXT *pctxt, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix, OSBOOL terminate)
- EXTERNXML int [rtXmlEncStartElement2](#) (OSCTXT *pctxt, const OSUTF8CHAR *elemName, size_t elemLen, const OSUTF8CHAR *nsPrefix, size_t nsPrefixLen, OSBOOL terminate)
- EXTERNXML int [rtXmlEncStartSoapEnv](#) (OSCTXT *pctxt)
- EXTERNXML int [rtXmlEncString](#) (OSCTXT *pctxt, OSXMLSTRING *pxmlstr, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncStringValue](#) (OSCTXT *pctxt, const OSUTF8CHAR *value)
- EXTERNXML int [rtXmlEncStringValue2](#) (OSCTXT *pctxt, const OSUTF8CHAR *value, size_t valueLen)
- EXTERNXML int [rtXmlEncUnicodeStr](#) (OSCTXT *pctxt, const OSUNICHAR *value, OSUINT32 nchars, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncUTF8Attr](#) (OSCTXT *pctxt, const OSUTF8CHAR *name, const OSUTF8CHAR *value)
- EXTERNXML int [rtXmlEncUTF8Attr2](#) (OSCTXT *pctxt, const OSUTF8CHAR *name, size_t nameLen, const OSUTF8CHAR *value, size_t valueLen)
- EXTERNXML int [rtXmlEncUTF8Str](#) (OSCTXT *pctxt, const OSUTF8CHAR *value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncUInt](#) (OSCTXT *pctxt, OSUINT32 value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncUIntValue](#) (OSCTXT *pctxt, OSUINT32 value)
- EXTERNXML int [rtXmlEncUIntAttr](#) (OSCTXT *pctxt, OSUINT32 value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen)
- EXTERNXML int [rtXmlEncUInt64](#) (OSCTXT *pctxt, OSUINT64 value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncUInt64Value](#) (OSCTXT *pctxt, OSUINT64 value)
- EXTERNXML int [rtXmlEncUInt64Attr](#) (OSCTXT *pctxt, OSUINT64 value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen)
- EXTERNXML int [rtXmlEncXSIAttrs](#) (OSCTXT *pctxt, OSBOOL needXSI)
- EXTERNXML int [rtXmlFreeInputSource](#) (OSCTXT *pctxt)
- EXTERNXML OSBOOL [rtXmlStrCmpAsc](#) (const OSUTF8CHAR *text1, const char *text2)
- EXTERNXML OSBOOL [rtXmlStrnCmpAsc](#) (const OSUTF8CHAR *text1, const char *text2, size_t len)
- EXTERNXML int [rtXmlSetEncBufPtr](#) (OSCTXT *pctxt, OSOCTET *bufaddr, size_t bufsiz)
- EXTERNXML int [rtXmlGetIndent](#) (OSCTXT *pctxt)
- EXTERNXML int [rtXmlGetIndentChar](#) (OSCTXT *pctxt)

5.4.1 Define Documentation

5.4.1.1 #define rtXmlFinalizeMemBuf(pMemBuf)

Value:

```
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)
```

5.4.1.2 #define rtXmlGetEncBufLen(pctxt) (pctxt) → buffer.byteIndex

This macro returns the length of the encoded XML message.

Parameters:

pctxt Pointer to a context structure.

5.4.1.3 #define rtXmlGetEncBufPtr(pctxt) (pctxt) → buffer.data

This macro returns the start address of the encoded XML message. 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.

Parameters:

pctxt Pointer to a context structure.

5.4.2 Function Documentation

5.4.2.1 EXTERNXML int rtXmlEncAny (OSCTXT *pctxt, OSXMLSTRING *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)

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>)

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.

nsPrefix XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.2 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.

Parameters:

pctxt Pointer to context block structure.

pAnyAttrList List of attributes.

Returns:

Completion status of operation:

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

5.4.2.3 EXTERNXML int rtXmlEncBase64Binary (OSCTXT * *pctxt*, OSUINT32 *nocts*, const OSOCTET * *value*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

This function encodes a variable of the XSD base64Binary type.

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.

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.4 EXTERNXML int rtXmlEncBase64BinaryAttr (OSCTXT * *pctxt*, OSUINT32 *nocts*, const OSOCTET * *value*, const OSUTF8CHAR * *attrName*, OSUINT16 *attrNameLen*)

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

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.

5.4.2.5 EXTERNXML int rtXmlEncBase64StrValue (OSCTXT * *pctxt*, OSUINT32 *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.

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.

5.4.2.6 EXTERNXML int rtXmlEncBigInt (OSCTXT * *pctxt*, const OSUTF8CHAR * *value*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

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 radices are currently not supported.

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.
- nsPrefix* XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.7 EXTERNXML int rtXmlEncBigIntAttr (OSCTXT * *pctxt*, const OSUTF8CHAR * *value*, const OSUTF8CHAR * *attrName*, OSUINT16 *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).

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.

5.4.2.8 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.

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.

5.4.2.9 EXTERNXML int rtXmlEncBinStrValue (OSCTXT * *pctxt*, OSUINT32 *nbits*, const OSOCTET * *data*)

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

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.

5.4.2.10 EXTERNXML int rtXmlEncBitString (OSCTXT * *pctxt*, OSUINT32 *nbits*, const OSOCTET * *value*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

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](#)).

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.

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.11 EXTERNXML int rtXmlEncBool (OSCTXT * *pctxt*, OSBOOL *value*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

This function encodes a variable of the XSD boolean type.

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.

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.12 EXTERNXML int rtXmlEncBoolAttr (OSCTXT * *pctxt*, OSBOOL *value*, const OSUTF8CHAR * *attrName*, OSUINT16 *attrNameLen*)

This function encodes an XSD boolean attribute value.

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.

5.4.2.13 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.

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.

5.4.2.14 EXTERNXML int rtXmlEncDate (OSCTXT * *pctxt*, const OSXSDDateTime * *pvalue*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

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.

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.
- nsPrefix* XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.15 EXTERNXML int rtXmlEncDateTime (OSCTXT * *pctxt*, const OSXSDDateTime * *pvalue*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

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

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.
- nsPrefix* XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.16 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.

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.

5.4.2.17 EXTERNXML int rtXmlEncDateValue (OSCTXT * *pctxt*, const OSXSDDateTime * *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.

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.

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

This function encodes a variable of the XSD decimal type.

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.

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

pFmtSpec Pointer to format specification structure.

Returns:

Completion status of operation:

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

5.4.2.19 EXTERNXML int rtXmlEncDecimalAttr (OSCTXT * *pctxt*, OSREAL *value*, const OSUTF8CHAR * *attrName*, OSUINT16 *attrNameLen*, const OSDecimalFmt * *pFmtSpec*)

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

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.

5.4.2.20 EXTERNXML int rtXmlEncDecimalValue (OSCTXT * *pctxt*, OSREAL *value*, const OSDecimalFmt * *pFmtSpec*, char * *pDestBuf*, size_t *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.

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.

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

This function encodes a variable of the XSD double type.

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.

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

pFmtSpec Pointer to format specification structure.

Returns:

Completion status of operation:

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

5.4.2.22 EXTERNXML int rtXmlEncDoubleAttr (OSCTXT * *pctxt*, OSREAL *value*, const OSUTF8CHAR * *attrName*, OSUINT16 *attrNameLen*, const OSDoubleFmt * *pFmtSpec*)

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

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.

5.4.2.23 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.

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.

5.4.2.24 EXTERNXML int rtXmlEncEmptyElement (OSCTXT * *pctxt*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*, OSBOOL *terminate*)

This function encodes an empty element tag value (<elemName>).

Parameters:

pctxt Pointer to context block structure.

elemName XML element name.

nsPrefix XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

terminate Add closing '>' character.

Returns:

Completion status of operation:

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

5.4.2.25 EXTERNXML int rtXmlEncEndDocument (OSCTXT * *pctxt*)

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

Parameters:

pctxt Pointer to context block structure.

Returns:

Completion status of operation:

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

5.4.2.26 EXTERNXML int rtXmlEncEndElement (OSCTXT * *pctxt*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

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

Parameters:

pctxt Pointer to context block structure.

elemName XML element name.

nsPrefix XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.27 EXTERNXML int rtXmlEncEndSoapEnv (OSCTXT * *pctxt*)

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

Parameters:

pctxt Pointer to context block structure.

Returns:

Completion status of operation:

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

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

This function encodes a variable of the XSD float type.

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.

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

pFmtSpec Pointer to format specification structure.

Returns:

Completion status of operation:

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

5.4.2.29 EXTERNXML int rtXmlEncFloatAttr (OSCTXT * *pctxt*, OSREAL *value*, const OSUTF8CHAR * *attrName*, OSUINT16 *attrNameLen*, const OSDoubleFmt * *pFmtSpec*)

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

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.

5.4.2.30 EXTERNXML int rtXmlEncGDay (OSCTXT * *pctxt*, const OSXSDDateTime * *pvalue*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

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

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.

nsPrefix XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.31 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.

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.

5.4.2.32 EXTERNXML int rtXmlEncGMonth (OSCTXT * *pctxt*, const OSXSDDateTime * *pvalue*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

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

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.

nsPrefix XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.33 EXTERNXML int rtXmlEncGMonthDay (OSCTXT * *pctxt*, const OSXSDDateTime * *pvalue*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

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

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.

nsPrefix XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.34 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.

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.

5.4.2.35 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.

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.

5.4.2.36 EXTERNXML int rtXmlEncGYear (OSCTXT * *pctxt*, const OSXSDDateTime * *pvalue*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

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

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.

nsPrefix XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.37 EXTERNXML int rtXmlEncGYearMonth (OSCTXT * *pctxt*, const OSXSDDateTime * *pvalue*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

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

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.

nsPrefix XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.38 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.

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.

5.4.2.39 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.

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.

5.4.2.40 EXTERNXML int rtXmlEncHexBinary (OSCTXT * *pctxt*, OSUINT32 *nocts*, const OSOCTET * *value*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

This function encodes a variable of the XSD hexBinary type.

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.
- nsPrefix* XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.41 EXTERNXML int rtXmlEncHexBinaryAttr (OSCTXT * *pctxt*, OSUINT32 *nocts*, const OSOCTET * *value*, const OSUTF8CHAR * *attrName*, OSUINT16 *attrNameLen*)

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

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.

5.4.2.42 EXTERNXML int rtXmlEncHexStrValue (OSCTXT * *pctxt*, OSUINT32 *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.

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.

5.4.2.43 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.

Parameters:

- pctxt* Pointer to context block structure.

Returns:

Completion status of operation:

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

5.4.2.44 EXTERNXML int rtXmlEncInt (OSCTXT * *pctxt*, OSINT32 *value*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

This function encodes a variable of the XSD integer type.

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.
- nsPrefix* XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.45 EXTERNXML int rtXmlEncInt64 (OSCTXT * *pctxt*, OSINT64 *value*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

This function encodes a variable of the XSD integer type. This version of the function is used for 64-bit integer values.

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.

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.46 EXTERNXML int rtXmlEncInt64Attr (OSCTXT * *pctxt*, OSINT64 *value*, const OSUTF8CHAR * *attrName*, OSUINT16 *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.

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.

5.4.2.47 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.

Parameters:

pctxt Pointer to context block structure.

value Value to be encoded.

Returns:

Completion status of operation:

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

5.4.2.48 EXTERNXML int rtXmlEncIntAttr (OSCTXT * *pctxt*, OSINT32 *value*, const OSUTF8CHAR * *attrName*, OSUINT16 *attrNameLen*)

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

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.

5.4.2.49 EXTERNXML int rtXmlEncIntPattern (OSCTXT * *pctxt*, OSINT32 *value*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*, const OSUTF8CHAR * *pattern*)

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

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.
- nsPrefix* XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.
- pattern* Pattern of the encoded value.

Returns:

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

5.4.2.50 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.

Parameters:

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

Returns:

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

5.4.2.51 EXTERNXML int rtXmlEncNamedBits (OSCTXT * *pctxt*, const OSBitMapItem * *pBitMap*, OSUINT32 *nbits*, const OSOCTET * *pvalue*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

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 identifiers corresponding to each set bit in the bit string value.

Parameters:

pctxt Pointer to context block structure.

pBitMap Bit map equating symbolic bit names to bit numbers.

nbits Number of bits in the bit 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.

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.52 EXTERNXML int rtXmlEncNSAttrs (OSCTXT * *pctxt*)

This function encodes namespace declaration attributes at the beginning of an XML document. The attributes to be encoded are stored in the namespace list within the context. Namespaces are added to this list by using the namespace utility functions.

Parameters:

pctxt Pointer to context block structure.

Returns:

Completion status of operation:

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

5.4.2.53 EXTERNXML int rtXmlEncSoapArrayTypeAttr (OSCTXT * *pctxt*, const OSUTF8CHAR * *name*, const OSUTF8CHAR * *value*, size_t *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]"'.

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.

5.4.2.54 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"?>
```

Parameters:

pctxt Pointer to context block structure.

Returns:

Completion status of operation:

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

5.4.2.55 EXTERNXML int rtXmlEncStartElement (OSCTXT * *pctxt*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*, OSBOOL *terminate*)

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

Parameters:

pctxt Pointer to context block structure.

elemName XML element name.

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

terminate Add closing '>' character.

Returns:

Completion status of operation:

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

5.4.2.56 EXTERNXML int rtXmlEncStartElement2 (OSCTXT * *pctxt*, const OSUTF8CHAR * *elemName*, size_t *elemLen*, const OSUTF8CHAR * *nsPrefix*, size_t *nsPrefixLen*, OSBOOL *terminate*)

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

Parameters:

pctxt Pointer to context block structure.

elemName XML element name.

elemLen Length of XML element name.

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

nsPrefixLen Length of XML namespace prefix.

terminate Add closing '>' character.

Returns:

Completion status of operation:

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

5.4.2.57 EXTERNXML int rtXmlEncStartSoapEnv (OSCTXT * *pctxt*)

This function encodes a SOAP envelope start element tag. This includes all of the standard SOAP namespace attributes.

Parameters:

pctxt Pointer to context block structure.

Returns:

Completion status of operation:

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

5.4.2.58 EXTERNXML int rtXmlEncString (OSCTXT * *pctxt*, OSXMLSTRING * *pxmlstr*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

This function encodes a variable of the XSD string type.

Parameters:

pctxt Pointer to context block structure.

pxmlstr XML 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.

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

Returns:

Completion status of operation:

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

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

This function encodes a variable of the XSD string type.

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.

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

This function encodes a variable of the XSD string type.

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.

5.4.2.61 EXTERNXML int rtXmlEncTime (OSCTXT * *pctxt*, const OSXSDDateTime * *pvalue*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

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

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.
- nsPrefix* XML namespace prefix. If not null, will be prepended to elemName to form a qualified name.

Returns:

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

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

This function encodes a variable of the XSD 'time' type as a string. This version of the function just puts the encoded value in the destination buffer or stream without any tags.

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.

5.4.2.63 EXTERNXML int rtXmlEncUInt (OSCTXT * *pctxt*, OSUINT32 *value*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

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

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.

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.64 EXTERNXML int rtXmlEncUInt64 (OSCTXT * *pctxt*, OSUINT64 *value*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

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.

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.

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.65 EXTERNXML int rtXmlEncUInt64Attr (OSCTXT * *pctxt*, OSUINT64 *value*, const OSUTF8CHAR * *attrName*, OSUINT16 *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.

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.

5.4.2.66 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.

Parameters:

pctxt Pointer to context block structure.

value Value to be encoded.

Returns:

Completion status of operation:

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

5.4.2.67 EXTERNXML int rtXmlEncUIntAttr (OSCTXT * *pctxt*, OSUINT32 *value*, const OSUTF8CHAR * *attrName*, OSUINT16 *attrNameLen*)

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

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.

5.4.2.68 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.

Parameters:

pctxt Pointer to context block structure.

value Value to be encoded.

Returns:

Completion status of operation:

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

5.4.2.69 EXTERNXML int rtXmlEncUnicodeStr (OSCTXT * *pctxt*, const OSUNICHAR * *value*, OSUINT32 *nchars*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)

This function encodes a Unicode string value.

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.

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.70 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.

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.

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

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

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.

5.4.2.72 **EXTERNXML int rtXmlEncUTF8Str (OSCTXT * *pctxt*, const OSUTF8CHAR * *value*, const OSUTF8CHAR * *elemName*, const OSUTF8CHAR * *nsPrefix*)**

This function encodes a UTF-8 string value.

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.

nsPrefix XML namespace prefix. If not null, will be prepended to *elemName* to form a qualified name.

Returns:

Completion status of operation:

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

5.4.2.73 **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.

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`.

5.4.2.74 **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'.

Parameters:

pctxt Pointer to context block structure.

Returns:

Completion status of operation:

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

5.4.2.75 EXTERNXML int rtXmlGetIndent (OSCTXT * *pctxt*)

This function returns current XML output indent value.

Parameters:

pctxt Pointer to OSCTXT structure

Returns:

Current indent value (≥ 0) if OK, negative status code if error.

5.4.2.76 EXTERNXML int rtXmlGetIndentChar (OSCTXT * *pctxt*)

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

Parameters:

pctxt Pointer to OSCTXT structure

Returns:

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

5.4.2.77 EXTERNXML int rtXmlSetEncBufPtr (OSCTXT * *pctxt*, OSOCTET * *bufaddr*, size_t *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.

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).

5.5 XML utility functions.

Functions

- EXTERNXML int `rtXmlWriteToFile` (OSCTXT *pctx, const char *filename)
- EXTERNXML void `rtXmlTreatWhitespaces` (OSCTXT *pctx, int whiteSpaceType)

5.5.1 Function Documentation

5.5.1.1 EXTERNXML int `rtXmlWriteToFile` (OSCTXT * *pctx*, const char * *filename*)

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

Parameters:

pctx Pointer to OSCTXT structure.

filename Full path to file to which XML is to be written.

Returns:

0 - if success, negative value if error.

5.6 XML pull-parser decode functions.

Enumerations

- enum `OSXMLWhiteSpaceMode` { `OSXMLWSM_PRESERVE` = 0, `OSXMLWSM_REPLACE`, `OSXMLWSM_COLLAPSE` }

Functions

- EXTERNXML int `rtXmlpDecAny` (OSCTXT *pctx, const OSUTF8CHAR **pvalue)
- EXTERNXML int `rtXmlpDecAnyAttrStr` (OSCTXT *pctx, const OSUTF8CHAR **ppAttrStr, size_t index)
- EXTERNXML int `rtXmlpDecAnyElem` (OSCTXT *pctx, const OSUTF8CHAR **pvalue)
- EXTERNXML int `rtXmlpDecBase64Str` (OSCTXT *pctx, OSOCTET *pvalue, OSUINT32 *pnoc, OSINT32 bufsize)
- EXTERNXML int `rtXmlpDecBigInt` (OSCTXT *pctx, const OSUTF8CHAR **pvalue)
- EXTERNXML int `rtXmlpDecBitString` (OSCTXT *pctx, OSOCTET *pvalue, OSUINT32 *pnbits, OSUINT32 bufsize)
- EXTERNXML int `rtXmlpDecBool` (OSCTXT *pctx, OSBOOL *pvalue)
- EXTERNXML int `rtXmlpDecDate` (OSCTXT *pctx, OSXSDDateTime *pvalue)
- EXTERNXML int `rtXmlpDecDateTime` (OSCTXT *pctx, OSXSDDateTime *pvalue)
- EXTERNXML int `rtXmlpDecDecimal` (OSCTXT *pctx, OSREAL *pvalue, int totalDigits, int fractionDigits)
- EXTERNXML int `rtXmlpDecDouble` (OSCTXT *pctx, OSREAL *pvalue, int totalDigits, int fractionDigits)
- EXTERNXML int `rtXmlpDecDynBase64Str` (OSCTXT *pctx, OSDynOctStr *pvalue)
- EXTERNXML int `rtXmlpDecDynBitString` (OSCTXT *pctx, OSDynOctStr *pvalue)
- EXTERNXML int `rtXmlpDecDynHexStr` (OSCTXT *pctx, OSDynOctStr *pvalue)
- EXTERNXML int `rtXmlpDecDynUnicodeStr` (OSCTXT *pctx, const OSUNICHAR **ppdata, OSUINT32 *pnchars)
- EXTERNXML int `rtXmlpDecDynUTF8Str` (OSCTXT *pctx, const OSUTF8CHAR **outdata)
- EXTERNXML int `rtXmlpDecGDay` (OSCTXT *pctx, OSXSDDateTime *pvalue)
- EXTERNXML int `rtXmlpDecGMonth` (OSCTXT *pctx, OSXSDDateTime *pvalue)
- EXTERNXML int `rtXmlpDecGMonthDay` (OSCTXT *pctx, OSXSDDateTime *pvalue)
- EXTERNXML int `rtXmlpDecGYear` (OSCTXT *pctx, OSXSDDateTime *pvalue)
- EXTERNXML int `rtXmlpDecGYearMonth` (OSCTXT *pctx, OSXSDDateTime *pvalue)
- EXTERNXML int `rtXmlpDecHexStr` (OSCTXT *pctx, OSOCTET *pvalue, OSUINT32 *pnoc, OSINT32 bufsize)
- EXTERNXML int `rtXmlpDecInt` (OSCTXT *pctx, OSINT32 *pvalue)
- EXTERNXML int `rtXmlpDecInt8` (OSCTXT *pctx, OSINT8 *pvalue)
- EXTERNXML int `rtXmlpDecInt16` (OSCTXT *pctx, OSINT16 *pvalue)
- EXTERNXML int `rtXmlpDecInt64` (OSCTXT *pctx, OSINT64 *pvalue)
- EXTERNXML int `rtXmlpDecNamedBits` (OSCTXT *pctx, const OSBitMapItem *pBitMap, OSOCTET *pvalue, OSUINT32 *pnbits, OSUINT32 bufsize)
- EXTERNXML int `rtXmlpDecStrList` (OSCTXT *pctx, OSRTDList *plist)
- EXTERNXML int `rtXmlpDecTime` (OSCTXT *pctx, OSXSDDateTime *pvalue)
- EXTERNXML int `rtXmlpDecUInt` (OSCTXT *pctx, OSUINT32 *pvalue)
- EXTERNXML int `rtXmlpDecUInt8` (OSCTXT *pctx, OSOCTET *pvalue)
- EXTERNXML int `rtXmlpDecUInt16` (OSCTXT *pctx, OSUINT16 *pvalue)
- EXTERNXML int `rtXmlpDecUInt64` (OSCTXT *pctx, OSUINT64 *pvalue)
- EXTERNXML int `rtXmlpDecXmlStr` (OSCTXT *pctx, OSXMLSTRING *outdata)
- EXTERNXML int `rtXmlpDecXSIAAttr` (OSCTXT *pctx, const OSXMLNameFragments *attrName)
- EXTERNXML int `rtXmlpDecXSITypeAttr` (OSCTXT *pctx, const OSXMLNameFragments *attrName, const OSUTF8CHAR **ppAttrValue)

- EXTERNXML int **rtXmlpGetAttributeID** (OSCTXT *pctxt, const OSXMLStrFragment *attrName, size_t nAttr, const OSXMLAttrDescr attrNames[], OSUINT32 attrPresent[])
- EXTERNXML int **rtXmlpGetNextElem** (OSCTXT *pctxt, OSXMLElemDescr *pElem, OSINT32 level)
- EXTERNXML int **rtXmlpGetNextElemID** (OSCTXT *pctxt, const OSXMLElemIDRec *tab, size_t nrows, OSINT32 level, OSBOOL continueParse)
- EXTERNXML OSBOOL **rtXmlpIsInGroup** (int elemID, int grpId, const OSBOOL *grpTab, int nElems)
- EXTERNXML int **rtXmlpMarkLastEventActive** (OSCTXT *pctxt)
- EXTERNXML int **rtXmlpMatchStartTag** (OSCTXT *pctxt, const OSUTF8CHAR *elemLocalName, const OSUTF8CHAR *elemURI)
- EXTERNXML int **rtXmlpMatchEndTag** (OSCTXT *pctxt, OSINT32 level)
- EXTERNXML OSBOOL **rtXmlpMatchElemId** (OSCTXT *pctxt, int elemID, int matchingID)
- EXTERNXML OSBOOL **rtXmlpHasAttributes** (OSCTXT *pctxt)
- EXTERNXML int **rtXmlpGetAttributeCount** (OSCTXT *pctxt)
- EXTERNXML void **rtXmlpGetContent** (OSCTXT *pctxt, int level)
- EXTERNXML int **rtXmlpSelectAttribute** (OSCTXT *pctxt, OSXMLNameFragments *pAttr, size_t index)
- EXTERNXML int **rtXmlpCreateReader** (OSCTXT *pctxt)
- EXTERNXML OSINT32 **rtXmlpGetCurrentLevel** (OSCTXT *pctxt)
- EXTERNXML void **rtXmlpSetWhiteSpaceMode** (OSCTXT *pctxt, [OSXMLWhiteSpaceMode](#) whiteSpaceMode)
- EXTERNXML void **rtXmlpSetMixedContentMode** (OSCTXT *pctxt, OSBOOL mixedContentMode)
- EXTERNXML OSBOOL **rtXmlpIsContentMode** (OSCTXT *pctxt)
- EXTERNXML void **rtXmlpSetListMode** (OSCTXT *pctxt)
- EXTERNXML OSBOOL **rtXmlpListHasItem** (OSCTXT *pctxt)
- EXTERNXML void **rtXmlpCountListItems** (OSCTXT *pctxt, OSUINT32 *itemCnt)
- EXTERNXML int **rtXmlpGetNextSeqElemID** (OSCTXT *pctxt, const OSXMLElemIDRec *tab, const OSXMLGroupDesc *ppGroup, int curID, int lastMandatoryID)
- EXTERNXML int **rtXmlpGetNextAllElemID** (OSCTXT *pctxt, const OSXMLElemIDRec *tab, size_t nrows, const OSUINT8 *pOrder, OSUINT32 nOrder, OSUINT32 maxOrder, int anyID)

5.6.1 Enumeration Type Documentation

5.6.1.1 enum [OSXMLWhiteSpaceMode](#)

Whitespace treatment mode.

5.6.2 Function Documentation

5.6.2.1 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.

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.

5.6.2.2 EXTERNXML int rtXmlpDecAnyAttrStr (OSCTXT * *pctxt*, const OSUTF8CHAR ** *ppAttrStr*, size_t *index*)

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.

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.

index Index of attribute.

Returns:

Completion status of operation:

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

5.6.2.3 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.

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.

5.6.2.4 EXTERNXML int rtXmlpDecBase64Str (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. Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

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.

5.6.2.5 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.

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.

5.6.2.6 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.

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.

5.6.2.7 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.

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.

5.6.2.8 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.

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.

5.6.2.9 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.

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.

5.6.2.10 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.

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.

fractionDigits Number of fraction digits in the decimal number from XSD fractionDigits facet value.

Returns:

Completion status of operation:

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

5.6.2.11 EXTERNXML int rtXmlpDecDouble (OSCTXT * *pctxt*, OSREAL * *pvalue*, int *totalDigits*, int *fractionDigits*)

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.

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.

5.6.2.12 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.

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.

5.6.2.13 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.

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.

5.6.2.14 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.

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.

5.6.2.15 EXTERNXML int rtXmlpDecDynUnicodeStr (OSCTXT * *pctxt*, const OSUNICHAR ** *ppdata*, OSUINT32 * *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.

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.

5.6.2.16 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.

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.

5.6.2.17 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.

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.

5.6.2.18 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.

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.

5.6.2.19 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.

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.

5.6.2.20 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.

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.

5.6.2.21 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.

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.

5.6.2.22 EXTERNXML int rtXmlpDecHexStr (OSCTXT * *pctxt*, OSOCTET * *pvalue*, OSUINT32 * *pnocts*, OSINT32 *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.

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.

5.6.2.23 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.

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.

5.6.2.24 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.

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.

5.6.2.25 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.

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.

5.6.2.26 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.

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.

5.6.2.27 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.

Parameters:

- pctxt* Pointer to context block structure.
- pBitMap* Pointer to bit map structure that defined token to bit mappings.
- pvalue* Pointer to buffer to receive decoded bit map.
- pnbits* Number of bits in decoded bit map.
- bufsize* Size of buffer passed in to receive decoded bit values.

Returns:

Completion status of operation:

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

5.6.2.28 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.

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.

5.6.2.29 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

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.

5.6.2.30 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 OS-UTF8CHAR characters returned by an XML pull parser.

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.

5.6.2.31 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 OS-UTF8CHAR characters returned by an XML pull parser.

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.

5.6.2.32 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 OS-UTF8CHAR characters returned by an XML pull parser.

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.

5.6.2.33 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.

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.

5.6.2.34 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 character 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.

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.

5.6.2.35 EXTERNXML int rtXmlpDecXSIAttr (OSCTXT * *pctxt*, const OSXMLNameFragments * *attrName*)

This function decodes XSI (XML Schema Instance) and XML namespace attributes that may be present in any arbitrary XML element within a document.

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.

5.6.2.36 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).

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.

5.6.2.37 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.

Returns:

Previously set whitespace mode.

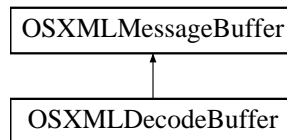
Chapter 6

ASN1C XML Runtime Class Documentation

6.1 OSXMLDecodeBuffer Class Reference

```
#include <rtXmlCppMsgBuf.h>
```

Inheritance diagram for OSXMLDecodeBuffer::



Public Member Functions

- [OSXMLDecodeBuffer](#) (const char *xmlFile)
- [OSXMLDecodeBuffer](#) (const OSOCTET *msgbuf, size_t bufsiz)
- [OSXMLDecodeBuffer](#) (OSRTInputStream &inputStream)
- int [decodeXML](#) (OSXMLReaderClass *pReader)
- virtual int [init](#) ()
- int [parseElementName](#) (OSUTF8CHAR **ppName)
- int [parseElemQName](#) (OSXMLQName *pQName)
- OSUINT32 [setMaxErrors](#) (OSUINT32 maxErrors)
- const char * [getXmlFileName](#) ()
- virtual OSBOOL [isA](#) (int bufferType)

Protected Types

- enum { [INPUT_FILE](#), [INPUT_STREAM](#), [INPUT_STREAM_ATTACHED](#), [INPUT_MEMORY](#) }

Protected Member Functions

- void [initContextData](#) ()

Protected Attributes

- union {
 - const char * **fileName**
 - OSRTInputStream * **pInputStream**
 - struct {
 - const OSOCTET * **pMemBuf**
 - size_t **bufSize**
 - } **memBuf**
 - } **mInput**
- enum OSXMLDecodeBuffer:: { ... } **mInputId**

6.1.1 Detailed Description

The [OSXMLDecodeBuffer](#) class is derived from the [OSXMLMessageBuffer](#) base class. It contains variables and methods specific to decoding XML messages. It is used to manage the input buffer 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.

6.1.2 Member Enumeration Documentation

6.1.2.1 anonymous enum [protected]

This enumeration associates each possible input location with an ID to properly decode the input data.

6.1.3 Constructor & Destructor Documentation

6.1.3.1 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.

Parameters:

xmlFile A pointer to name of file to be decoded.

6.1.3.2 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.

Parameters:

msgbuf A pointer to a buffer containing an XML message.

bufsiz Size of the message buffer.

6.1.3.3 OSXMLDecodeBuffer::OSXMLDecodeBuffer (OSRTInputStream & *inputStream*)

This version of the [OSXMLDecodeBuffer](#) constructor takes a reference to the OSInputStream object. The stream is assumed to have been previously initialized to point at an encoded XML message.

Parameters:

inputStream reference to the OSInputStream object

6.1.4 Member Function Documentation

6.1.4.1 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.

Parameters:

pReader Pointer to OSXMLReaderClass object.

Returns:

Completion status.

6.1.4.2 const char* OSXMLDecodeBuffer::getXmlFileName () [inline]

This method returns the name of the XML file that is associated with the current buffer.

Returns:

Name of the XML file that is associated with this object.

6.1.4.3 virtual int OSXMLDecodeBuffer::init () [virtual]

This method initializes the decode message buffer.

Returns:

Completion status of operation:

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

6.1.4.4 virtual OSBOOL OSXMLDecodeBuffer::isA (int *bufferType*) [inline, virtual]

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.

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.

6.1.4.5 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.

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.

6.1.4.6 int OSXMLDecodeBuffer::parseElemQName (OSXMLQName * pQName)

This method parses the initial tag from an XML message.

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.

6.1.4.7 OSUINT32 OSXMLDecodeBuffer::setMaxErrors (OSUINT32 maxErrors)

This method sets the maximum number of errors returned by the SAX parser.

Parameters:

maxErrors The desired number of maximum errors.

Returns:

The previously set maximum number of errors.

6.1.5 Member Data Documentation**6.1.5.1 union { ... } OSXMLDecodeBuffer::mInput [protected]**

This structure stores the location of the input buffer. Possible inputs are a file on disk, an input stream, or a memory buffer.

6.1.5.2 `enum { ... } OSXMLDecodeBuffer::mInputId` [protected]

This enumeration associates each possible input location with an ID to properly decode the input data.

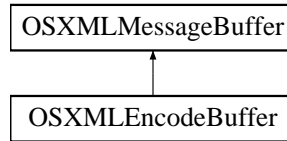
The documentation for this class was generated from the following file:

- [rtXmlCppMsgBuf.h](#)

6.2 OSXMLEncodeBuffer Class Reference

```
#include <rtXmlCppMsgBuf.h>
```

Inheritance diagram for OSXMLEncodeBuffer::



Public Member Functions

- [OSXMLEncodeBuffer](#) ()
- [OSXMLEncodeBuffer](#) (OSOCKET *pMsgBuf, size_t msgBufLen)
- virtual size_t [getMsgLen](#) ()
- virtual int [init](#) ()
- virtual OSBOOL [isA](#) (int bufferType)
- virtual long [write](#) (const char *filename)
- virtual long [write](#) (FILE *fp)

Protected Member Functions

- [OSXMLEncodeBuffer](#) (OSRTContext *pContext)

6.2.1 Detailed Description

The [OSXMLEncodeBuffer](#) class is derived from the [OSXMLMessageBuffer](#) base class. 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.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 OSXMLEncodeBuffer::OSXMLEncodeBuffer ()

Default constructor

6.2.2.2 OSXMLEncodeBuffer::OSXMLEncodeBuffer (OSOCKET *pMsgBuf, size_t msgBufLen)

This constructor allows a static message buffer to be specified to receive the encoded message.

Parameters:

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

6.2.3 Member Function Documentation

6.2.3.1 virtual size_t OSXMLEncodeBuffer::getMsgLen () [inline, virtual]

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

Returns:

Length of the XML message encapsulated within this buffer object.

6.2.3.2 virtual int OSXMLEncodeBuffer::init () [virtual]

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.

6.2.3.3 virtual OSBOOL OSXMLEncodeBuffer::isA (int *bufferType*) [inline, virtual]

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.

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.

6.2.3.4 virtual long OSXMLEncodeBuffer::write (FILE * *fp*) [virtual]

This version of the write method writes to a file that is specified by a FILE pointer.

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.

6.2.3.5 virtual long OSXMLEncodeBuffer::write (const char * *filename*) [virtual]

This method writes the encoded message to the given file.

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.

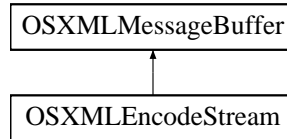
The documentation for this class was generated from the following file:

- [rtXmlCppMsgBuf.h](#)

6.3 OSXMLEncodeStream Class Reference

```
#include <rtXmlCppMsgBuf.h>
```

Inheritance diagram for OSXMLEncodeStream::



Public Member Functions

- [OSXMLEncodeStream](#) (OSRTOutputStream &outputStream)
- [OSXMLEncodeStream](#) (OSRTOutputStream *pOutputStream, OSBOOL ownStream=TRUE)
- virtual OSBOOL [isA](#) (int bufferType)
- virtual const OSOCTET * [getMsgPtr](#) ()

Protected Attributes

- OSRTOutputStream * [mpStream](#)
- OSBOOL [mbOwnStream](#)

6.3.1 Detailed Description

The [OSXMLEncodeStream](#) class is derived from the [OSXMLMessageBuffer](#) base class. 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.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 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.

Parameters:

outputStream reference to the OSOutputStream object

6.3.2.2 OSXMLEncodeStream::OSXMLEncodeStream (OSRTOutputStream * *pOutputStream*, OSBOOL *ownStream* = TRUE)

This version of the [OSXMLEncodeStream](#) constructor takes a pointer to the OSOutputStream 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.

Parameters:

pOutputStream reference to the OSOutputStream object

ownStream set ownership for the passed stream object.

6.3.3 Member Function Documentation

6.3.3.1 virtual const OSOCTET* OSXMLEncodeStream::getMsgPtr () [inline, virtual]

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.

6.3.3.2 virtual OSBOOL OSXMLEncodeStream::isA (int *bufferType*) [inline, virtual]

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.

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.

6.3.4 Member Data Documentation

6.3.4.1 OSBOOL OSXMLEncodeStream::mbOwnStream [protected]

TRUE if the [OSXMLEncodeStream](#) object will close and free the stream in the destructor.

6.3.4.2 OSRTOutputStream* OSXMLEncodeStream::mpStream [protected]

A pointer to an OSRTOutputStream object.

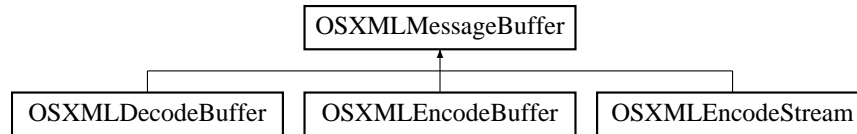
The documentation for this class was generated from the following file:

- [rtXmlCppMsgBuf.h](#)

6.4 OSXMLMessageBuffer Class Reference

```
#include <rtXmlCppMsgBuf.h>
```

Inheritance diagram for OSXMLMessageBuffer::



Public Member Functions

- virtual void * [getAppInfo](#) ()
- int [getIndent](#) ()
- int [getIndentChar](#) ()
- virtual void [setNamespace](#) (const OSUTF8CHAR *prefix, const OSUTF8CHAR *uri)
- virtual void [setAppInfo](#) (void *pXMLInfo)
- void [setFormatting](#) (OSBOOL doFormatting)
- void [setIndent](#) (OSUINT8 indent)
- void [setIndentChar](#) (char indentChar)

Protected Member Functions

- [OSXMLMessageBuffer](#) (Type bufferType, OSRTContext *pContext=0)

6.4.1 Detailed Description

The XML message buffer class is derived from the OSMessageBuffer base class. 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.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 OSXMLMessageBuffer::OSXMLMessageBuffer (Type *bufferType*, OSRTContext * *pContext* = 0) [protected]

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

Parameters:

bufferType Type of message buffer that is being created (for example, XMLEncode or XMLDecode).

pContext Pointer to a context to use. If NULL, new context will be allocated.

6.4.3 Member Function Documentation

6.4.3.1 virtual void* OSXMLMessageBuffer::getAppInfo () [virtual]

The getAppInfo method returns the pointer to application context information.

6.4.3.2 int OSXMLMessageBuffer::getIndent ()

This method returns current XML output indent value.

Returns:

Current indent value (≥ 0) if OK, negative status code if error.

6.4.3.3 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.

6.4.3.4 virtual void OSXMLMessageBuffer::setAppInfo (void * *pXMLInfo*) [virtual]

This method sets application specific context information within the common context structure. For XML encoding/decoding, this is a structure of type *OSXMLCtxtInfo*.

Parameters:

pXMLInfo Pointer to context information.

6.4.3.5 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.

Parameters:

doFormatting Boolean value indicating if formatting is to be done

Returns:

Status of operation: 0 if OK, negative status code if error.

6.4.3.6 void OSXMLMessageBuffer::setIndent (OSUINT8 *indent*)

This method sets XML output indent to the given value.

Parameters:

indent Number of spaces per indent. Default is 3.

6.4.3.7 void OSXMLMessageBuffer::setIndentChar (char *indentChar*)

This method sets XML output indent character to the given value.

Parameters:

indentChar Indent character. Default is space.

6.4.3.8 virtual void OSXMLMessageBuffer::setNamespace (const OSUTF8CHAR * *prefix*, const OSUTF8CHAR * *uri*) [virtual]

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.

Parameters:

prefix Namespace prefix

uri Namespace URI

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

- [rtXmlCppMsgBuf.h](#)

6.5 OSXSDGlobalElement Class Reference

```
#include <rtXmlCppXSDElement.h>
```

Public Member Functions

- [OSXSDGlobalElement](#) (OSRTMessageBufferIF &msgBuf)
- [OSXSDGlobalElement](#) (const [OSXSDGlobalElement](#) &o)
- virtual [~OSXSDGlobalElement](#) ()
- int [decode](#) ()
- virtual int [decodeFrom](#) (OSRTMessageBufferIF &msgBuf)
- int [encode](#) ()
- virtual int [encodeTo](#) (OSRTMessageBufferIF &msgBuf)
- OSCTXT * [getCtxtPtr](#) ()
- void * [memAlloc](#) (size_t numocts)
- void [memFreePtr](#) (void *ptr)
- void [setDefaultNamespace](#) (const OSUTF8CHAR *uri)
- void [setNamespace](#) (const OSUTF8CHAR *prefix, const OSUTF8CHAR *uri)
- void [setXSIType](#) (const OSUTF8CHAR *typeName)
- int [validate](#) ()
- virtual int [validateFrom](#) (OSRTMessageBufferIF &msgBuf)

Protected Member Functions

- [OSXSDGlobalElement](#) ()
- [OSXSDGlobalElement](#) (OSRTContext &ctxt)
- void [setMsgBuf](#) (OSRTMessageBufferIF &msgBuf)

Protected Attributes

- OSRTCtxtPtr [mpContext](#)
- OSRTMessageBufferIF * [mpMsgBuf](#)

6.5.1 Detailed Description

XSD global element base class. 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.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 [OSXSDGlobalElement::OSXSDGlobalElement](#) () [inline, protected]

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

6.5.2.2 OSXSDGlobalElement::OSXSDGlobalElement (OSRTContext & *ctxt*) [inline, protected]

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

Parameters:

ctxt - Reference to a context object.

6.5.2.3 OSXSDGlobalElement::OSXSDGlobalElement (OSRTMessageBufferIF & *msgBuf*) [inline]

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

Parameters:

msgBuf - Reference to a message buffer or stream object.

6.5.2.4 OSXSDGlobalElement::OSXSDGlobalElement (const OSXSDGlobalElement & *o*) [inline]

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

Parameters:

o - Reference to a global element object.

6.5.2.5 virtual OSXSDGlobalElement::~OSXSDGlobalElement () [inline, virtual]

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

6.5.3 Member Function Documentation

6.5.3.1 int OSXSDGlobalElement::decode () [inline]

The `decode` method decodes the message described by the encapsulated message buffer object.

6.5.3.2 virtual int OSXSDGlobalElement::decodeFrom (OSRTMessageBufferIF & *msgBuf*) [inline, virtual]

The `decodeFrom` method decodes a message from the given message buffer or stream argument.

Parameters:

msgBuf - Message buffer or stream containing message to decode.

6.5.3.3 int OSXSDGlobalElement::encode () [inline]

The `encode` method encodes a message using the encoding rules specified by the derived message buffer object.

6.5.3.4 virtual int OSXSDGlobalElement::encodeTo (OSRTMessageBufferIF & msgBuf) [inline, virtual]

The `encodeTo` method encodes a message into the given message buffer or stream argument.

Parameters:

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

6.5.3.5 OSCTXT* OSXSDGlobalElement::getCtxtPtr () [inline]

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

6.5.3.6 void* OSXSDGlobalElement::memAlloc (size_t numocts) [inline]

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

Parameters:

numocts - Number of bytes of memory to allocate

6.5.3.7 void OSXSDGlobalElement::memFreePtr (void * ptr) [inline]

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

Parameters:

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

6.5.3.8 void OSXSDGlobalElement::setDefaultNamespace (const OSUTF8CHAR * uri) [inline]

The `setDefaultNamespace` method sets the default namespace for the element to the given value.

Parameters:

uri - Default namespace URI

6.5.3.9 void OSXSDGlobalElement::setMsgBuf (OSRTMessageBufferIF & msgBuf) [protected]

The `setMsgBuf` method is used to set the internal message buffer pointer to point at the given message buffer or stream object.

Parameters:

msgBuf - Reference to a message buffer or stream object.

6.5.3.10 void OSXSDGlobalElement::setNamespace (const OSUTF8CHAR * *prefix*, const OSUTF8CHAR * *uri*) [inline]

The `setNamespace` method adds or modifies the namespace with the given URI in the namespace list to contain the given prefix.

Parameters:

prefix - Namespace prefix
uri - Namespace URI

6.5.3.11 void OSXSDGlobalElement::setXSIType (const OSUTF8CHAR * *typeName*) [inline]

The `setXSIType` method sets a type name to be used in the `xsi:type` attribute in the top-level module element declaration.

Parameters:

typeName - XSI type name

6.5.3.12 int OSXSDGlobalElement::validate () [inline]

The `validate` method validates the message described by the encapsulated message buffer object.

6.5.3.13 virtual int OSXSDGlobalElement::validateFrom (OSRTMessageBufferIF & *msgBuf*) [inline, virtual]

The `validateFrom` method validates a message from the given message buffer or stream argument.

Parameters:

msgBuf - Message buffer or stream containing message to validate.

6.5.4 Member Data Documentation

6.5.4.1 OSRTCtxtPtr OSXSDGlobalElement::mpContext [protected]

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

6.5.4.2 OSRTMessageBufferIF* OSXSDGlobalElement::mpMsgBuf [protected]

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

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

- [rtXmlCppXSDElement.h](#)

Chapter 7

ASN1C XML Runtime File Documentation

7.1 osrtxml.h File Reference

```
#include "rtxsrc/rtxCommon.h"
#include "rtxmlsrc/rtSaxDefs.h"
#include "rtxsrc/rtxDList.h"
#include "rtxsrc/rtxMemBuf.h"
#include "rtxmlsrc/rtXmlErrCodes.h"
#include "rtxmlsrc/rtXmlNamespace.h"
```

Classes

- struct **OSXMLFacets**
- struct **OSXMLStrFragment**
- struct **OSXMLNameFragments**
- struct **OSXMLItemDescr**
- struct **OSXMLElemIDRec**
- struct **OSXMLGroupDesc**
- struct **OSXMLCtxtInfo**
- struct **OSXMLQName**
- struct **OSIntegerFmt**
- struct **OSDecimalFmt**
- struct **OSDoubleFmt**

Defines

- #define **EXTERNXML**
- #define **OSUPCASE** 0x00008000
- #define **OSTERMSTART** 0x00004000
- #define **OSXMLFRAGSEQUAL**(frag1, frag2) (frag1.length==frag2.length && !memcmp(frag1.value,frag2.value,frag1.length))
- #define **OSXMLQNAMEEQUALS**(xnamefrag, qnametext)
- #define **OSXMLINDENT** 3

- #define **rtXmlErrAddStrParm** rtxErrAddStrParm
- #define **rtXmlFinalizeMemBuf**(pMemBuf)
- #define **rtXmlGetEncBufPtr**(pctx) (pctx) → buffer.data
- #define **rtXmlGetEncBufLen**(pctx) (pctx) → buffer.byteIndex

Typedefs

- typedef OSXMLItemDescr **OSXMLAttrDescr**
- typedef OSXMLItemDescr **OSXMLElemDescr**

Enumerations

- enum **OSXMLEncoding** { **OSXMLUTF8**, **OSXMLUTF16** }
- enum **OSXMLState** {
OSXMLINIT, **OSXMLHEADER**, **OSXMLSTART**, **OSXMLDATA**,
OSXMLEND }
- enum **OSXMLWhiteSpaceMode** { **OSXMLWSM_PRESERVE** = 0, **OSXMLWSM_REPLACE**,
OSXMLWSM_COLLAPSE }

Functions

- EXTERNXML int **rtXmlInitContext** (OSCTXT *pctx)
- EXTERNXML int **rtXmlInitCtxtAppInfo** (OSCTXT *pctx)
- EXTERNXML int **rtXmlCreateFileInputSource** (OSCTXT *pctx, const char *filepath)
- EXTERNXML OSBOOL **rtXmlCmpQName** (const OSUTF8CHAR *qname1, const OSUTF8CHAR *name2, const OSUTF8CHAR *nsPrefix2)
- EXTERNXML int **rtXmlGetBase64StrDecodedLen** (const OSUTF8CHAR *inpdata, size_t srcDataSize, size_t *pNumOcts, size_t *pSrcDataLen)
- EXTERNXML int **rtXmlDecBase64Binary** (OSRTMEMBUF *pMemBuf, const OSUTF8CHAR *inpdata, int length)
- EXTERNXML int **rtXmlDecBase64Str** (OSCTXT *pctx, OSOCTET *pvalue, OSUINT16 *pnocts, OSINT32 bufsize)
- EXTERNXML int **rtXmlDecBase64StrValue** (OSCTXT *pctx, OSOCTET *pvalue, OSUINT32 *pnocts, size_t bufSize, size_t srcDataLen)
- EXTERNXML int **rtXmlDecBigInt** (OSCTXT *pctx, const OSUTF8CHAR **ppvalue)
- EXTERNXML int **rtXmlDecBool** (OSCTXT *pctx, OSBOOL *pvalue)
- EXTERNXML int **rtXmlDecDate** (OSCTXT *pctx, OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlDecTime** (OSCTXT *pctx, OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlDecDateTime** (OSCTXT *pctx, OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlDecDecimal** (OSCTXT *pctx, OSREAL *pvalue, int totalDigits, int fractionDigits)
- EXTERNXML int **rtXmlDecDouble** (OSCTXT *pctx, OSREAL *pvalue, int totalDigits, int fractionDigits)
- EXTERNXML int **rtXmlDecDynBase64Str** (OSCTXT *pctx, OSDynOctStr *pvalue)
- EXTERNXML int **rtXmlDecDynHexStr** (OSCTXT *pctx, OSDynOctStr *pvalue)
- EXTERNXML int **rtXmlDecDynUTF8Str** (OSCTXT *pctx, const OSUTF8CHAR **outdata)
- EXTERNXML int **rtXmlDecHexBinary** (OSRTMEMBUF *pMemBuf, const OSUTF8CHAR *inpdata, int length)
- EXTERNXML int **rtXmlDecHexStr** (OSCTXT *pctx, OSOCTET *pvalue, OSUINT16 *pnocts, OSINT32 bufsize)

- EXTERNXML int **rtXmlDecHexStrValue** (OSCTXT *pctxt, const OSUTF8CHAR *const inpdata, size_t nbytes, OSOCTET *pvalue, OSUINT32 *pnbits, OSINT32 bufsize)
- EXTERNXML int **rtXmlDecGYear** (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlDecGYearMonth** (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlDecGMonth** (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlDecGMonthDay** (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlDecGDay** (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlDecInt** (OSCTXT *pctxt, OSINT32 *pvalue)
- EXTERNXML int **rtXmlDecInt8** (OSCTXT *pctxt, OSINT8 *pvalue)
- EXTERNXML int **rtXmlDecInt16** (OSCTXT *pctxt, OSINT16 *pvalue)
- EXTERNXML int **rtXmlDecInt64** (OSCTXT *pctxt, OSINT64 *pvalue)
- EXTERNXML int **rtXmlDecUInt** (OSCTXT *pctxt, OSUINT32 *pvalue)
- EXTERNXML int **rtXmlDecUInt8** (OSCTXT *pctxt, OSUINT8 *pvalue)
- EXTERNXML int **rtXmlDecUInt16** (OSCTXT *pctxt, OSUINT16 *pvalue)
- EXTERNXML int **rtXmlDecUInt64** (OSCTXT *pctxt, OSUINT64 *pvalue)
- EXTERNXML const OSUTF8CHAR * **rtXmlDecQName** (OSCTXT *pctxt, const OSUTF8CHAR *qname, const OSUTF8CHAR **prefix)
- EXTERNXML int **rtXmlDecXSIAttr** (OSCTXT *pctxt, const OSUTF8CHAR *attrName, const OSUTF8CHAR *attrValue)
- EXTERNXML int **rtXmlDecXSIAttrs** (OSCTXT *pctxt, const OSUTF8CHAR *const *attrs, const char *typeName)
- EXTERNXML int **rtXmlDecXmlStr** (OSCTXT *pctxt, OSXMLSTRING *outdata)
- EXTERNXML int **rtXmlParseElementName** (OSCTXT *pctxt, OSUTF8CHAR **ppName)
- EXTERNXML int **rtXmlParseElemQName** (OSCTXT *pctxt, OSXMLQName *pQName)
- EXTERNXML int **rtXmlEncAny** (OSCTXT *pctxt, OSXMLSTRING *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int **rtXmlEncAnyStr** (OSCTXT *pctxt, const OSUTF8CHAR *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int **rtXmlEncAnyAttr** (OSCTXT *pctxt, OSRTDList *pAnyAttrList)
- EXTERNXML int **rtXmlEncBase64Binary** (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int **rtXmlEncBase64BinaryAttr** (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen)
- EXTERNXML int **rtXmlEncBase64StrValue** (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *value)
- EXTERNXML int **rtXmlEncBigInt** (OSCTXT *pctxt, const OSUTF8CHAR *value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int **rtXmlEncBigIntAttr** (OSCTXT *pctxt, const OSUTF8CHAR *value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen)
- EXTERNXML int **rtXmlEncBigIntValue** (OSCTXT *pctxt, const OSUTF8CHAR *value)
- EXTERNXML int **rtXmlEncBitString** (OSCTXT *pctxt, OSUINT32 nbits, const OSOCTET *value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int **rtXmlEncBinStrValue** (OSCTXT *pctxt, OSUINT32 nbits, const OSOCTET *data)
- EXTERNXML int **rtXmlEncBool** (OSCTXT *pctxt, OSBOOL value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int **rtXmlEncBoolValue** (OSCTXT *pctxt, OSBOOL value)
- EXTERNXML int **rtXmlEncBoolAttr** (OSCTXT *pctxt, OSBOOL value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen)
- EXTERNXML int **rtXmlEncDate** (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int **rtXmlEncDateValue** (OSCTXT *pctxt, const OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlEncTime** (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)

- EXTERNXML int [rtXmlEncTimeValue](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlEncDateTime](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncDateTimeValue](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlEncDecimal](#) (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix, const OSDecimalFmt *pFmtSpec)
- EXTERNXML int [rtXmlEncDecimalAttr](#) (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen, const OSDecimalFmt *pFmtSpec)
- EXTERNXML int [rtXmlEncDecimalValue](#) (OSCTXT *pctxt, OSREAL value, const OSDecimalFmt *pFmtSpec, char *pDestBuf, size_t destBufSize)
- EXTERNXML int [rtXmlEncDouble](#) (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix, const OSDoubleFmt *pFmtSpec)
- EXTERNXML int [rtXmlEncDoubleAttr](#) (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen, const OSDoubleFmt *pFmtSpec)
- EXTERNXML int [rtXmlEncDoubleValue](#) (OSCTXT *pctxt, OSREAL value, const OSDoubleFmt *pFmtSpec, int defaultPrecision)
- EXTERNXML int [rtXmlEncEmptyElement](#) (OSCTXT *pctxt, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix, OSBOOL terminate)
- EXTERNXML int [rtXmlEncEmptyElement2](#) (OSCTXT *pctxt, const OSUTF8CHAR *elemName, size_t elemLen, const OSUTF8CHAR *nsPrefix, size_t nsPrefixLen, OSBOOL terminate)
- EXTERNXML int [rtXmlEncEndDocument](#) (OSCTXT *pctxt)
- EXTERNXML int [rtXmlEncEndElement](#) (OSCTXT *pctxt, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncEndElement2](#) (OSCTXT *pctxt, const OSUTF8CHAR *elemName, size_t elemLen, const OSUTF8CHAR *nsPrefix, size_t nsPrefixLen)
- EXTERNXML int [rtXmlEncEndSoapEnv](#) (OSCTXT *pctxt)
- EXTERNXML int [rtXmlEncFloat](#) (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix, const OSDoubleFmt *pFmtSpec)
- EXTERNXML int [rtXmlEncFloatAttr](#) (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen, const OSDoubleFmt *pFmtSpec)
- EXTERNXML int [rtXmlEncGYear](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncGYearMonth](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncGMonth](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncGMonthDay](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncGDay](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncGYearValue](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlEncGYearMonthValue](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlEncGMonthValue](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlEncGMonthDayValue](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlEncGDayValue](#) (OSCTXT *pctxt, const OSXSDDateTime *pvalue)
- EXTERNXML int [rtXmlEncHexBinary](#) (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncHexBinaryAttr](#) (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen)
- EXTERNXML int [rtXmlEncHexStrValue](#) (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *data)
- EXTERNXML int [rtXmlEncIndent](#) (OSCTXT *pctxt)

- EXTERNXML int [rtXmlEncInt](#) (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncIntValue](#) (OSCTXT *pctxt, OSINT32 value)
- EXTERNXML int [rtXmlEncIntAttr](#) (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen)
- EXTERNXML int [rtXmlEncIntPattern](#) (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix, const OSUTF8CHAR *pattern)
- EXTERNXML int [rtXmlEncIntPatternValue](#) (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *pattern)
- EXTERNXML int [rtXmlEncInt64](#) (OSCTXT *pctxt, OSINT64 value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncInt64Value](#) (OSCTXT *pctxt, OSINT64 value)
- EXTERNXML int [rtXmlEncInt64Attr](#) (OSCTXT *pctxt, OSINT64 value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen)
- EXTERNXML int [rtXmlEncNamedBits](#) (OSCTXT *pctxt, const OSBitMapItem *pBitMap, OSUINT32 nbits, const OSOCTET *pvalue, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncNamedBitsValue](#) (OSCTXT *pctxt, const OSBitMapItem *pBitMap, OSUINT32 nbits, const OSOCTET *pvalue)
- EXTERNXML int [rtXmlEncNSAttrs](#) (OSCTXT *pctxt)
- EXTERNXML int [rtXmlEncSoapArrayTypeAttr](#) (OSCTXT *pctxt, const OSUTF8CHAR *name, const OSUTF8CHAR *value, size_t itemCount)
- EXTERNXML int [rtXmlEncSoapArrayTypeAttr2](#) (OSCTXT *pctxt, const OSUTF8CHAR *name, size_t nameLen, const OSUTF8CHAR *value, size_t valueLen, size_t itemCount)
- EXTERNXML int [rtXmlEncStartDocument](#) (OSCTXT *pctxt)
- EXTERNXML int [rtXmlEncStartElement](#) (OSCTXT *pctxt, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix, OSBOOL terminate)
- EXTERNXML int [rtXmlEncStartElement2](#) (OSCTXT *pctxt, const OSUTF8CHAR *elemName, size_t elemLen, const OSUTF8CHAR *nsPrefix, size_t nsPrefixLen, OSBOOL terminate)
- EXTERNXML int [rtXmlEncStartSoapEnv](#) (OSCTXT *pctxt)
- EXTERNXML int [rtXmlEncString](#) (OSCTXT *pctxt, OSXMLSTRING *pxmlstr, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncStringValue](#) (OSCTXT *pctxt, const OSUTF8CHAR *value)
- EXTERNXML int [rtXmlEncStringValue2](#) (OSCTXT *pctxt, const OSUTF8CHAR *value, size_t valueLen)
- EXTERNXML int [rtXmlEncUnicodeStr](#) (OSCTXT *pctxt, const OSUNICHAR *value, OSUINT32 nchars, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncUTF8Attr](#) (OSCTXT *pctxt, const OSUTF8CHAR *name, const OSUTF8CHAR *value)
- EXTERNXML int [rtXmlEncUTF8Attr2](#) (OSCTXT *pctxt, const OSUTF8CHAR *name, size_t nameLen, const OSUTF8CHAR *value, size_t valueLen)
- EXTERNXML int [rtXmlEncUTF8Str](#) (OSCTXT *pctxt, const OSUTF8CHAR *value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncUInt](#) (OSCTXT *pctxt, OSUINT32 value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncUIntValue](#) (OSCTXT *pctxt, OSUINT32 value)
- EXTERNXML int [rtXmlEncUIntAttr](#) (OSCTXT *pctxt, OSUINT32 value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen)
- EXTERNXML int [rtXmlEncUInt64](#) (OSCTXT *pctxt, OSUINT64 value, const OSUTF8CHAR *elemName, const OSUTF8CHAR *nsPrefix)
- EXTERNXML int [rtXmlEncUInt64Value](#) (OSCTXT *pctxt, OSUINT64 value)
- EXTERNXML int [rtXmlEncUInt64Attr](#) (OSCTXT *pctxt, OSUINT64 value, const OSUTF8CHAR *attrName, OSUINT16 attrNameLen)
- EXTERNXML int [rtXmlEncXSIAttrs](#) (OSCTXT *pctxt, OSBOOL needXSI)

- EXTERNXML int **rtXmlFreeInputSource** (OSCTXT *pctx)
- EXTERNXML OSBOOL **rtXmlStrCmpAsc** (const OSUTF8CHAR *text1, const char *text2)
- EXTERNXML OSBOOL **rtXmlStrnCmpAsc** (const OSUTF8CHAR *text1, const char *text2, size_t len)
- EXTERNXML int **rtXmlSetEncBufPtr** (OSCTXT *pctx, OSOCTET *bufaddr, size_t bufsiz)
- EXTERNXML int **rtXmlGetIndent** (OSCTXT *pctx)
- EXTERNXML int **rtXmlGetIndentChar** (OSCTXT *pctx)
- EXTERNXML int **rtXmlSetDigitsFacets** (OSCTXT *pctx, int totalDigits, int fractionDigits)
- EXTERNXML int **rtXmlSetEncDocHdr** (OSCTXT *pctx, OSBOOL value)
- EXTERNXML int **rtXmlSetEncoding** (OSCTXT *pctx, OSXMLEncoding encoding)
- EXTERNXML int **rtXmlSetFormatting** (OSCTXT *pctx, OSBOOL doFormatting)
- EXTERNXML int **rtXmlSetIndent** (OSCTXT *pctx, OSUINT8 indent)
- EXTERNXML int **rtXmlSetIndentChar** (OSCTXT *pctx, char indentChar)
- EXTERNXML int **rtXmlSetSchemaLocation** (OSCTXT *pctx, const OSUTF8CHAR *schemaLocation)
- EXTERNXML int **rtXmlSetNoNSSchemaLocation** (OSCTXT *pctx, const OSUTF8CHAR *schemaLocation)
- EXTERNXML int **rtXmlSetXSITypeAttr** (OSCTXT *pctx, const OSUTF8CHAR * xsiType)
- EXTERNXML int **rtXmlMatchHexStr** (OSCTXT *pctx, size_t minLength, size_t maxLength)
- EXTERNXML int **rtXmlMatchBase64Str** (OSCTXT *pctx, size_t minLength, size_t maxLength)
- EXTERNXML int **rtXmlMatchDate** (OSCTXT *pctx)
- EXTERNXML int **rtXmlMatchTime** (OSCTXT *pctx)
- EXTERNXML int **rtXmlMatchDateTime** (OSCTXT *pctx)
- EXTERNXML int **rtXmlMatchGYear** (OSCTXT *pctx)
- EXTERNXML int **rtXmlMatchGYearMonth** (OSCTXT *pctx)
- EXTERNXML int **rtXmlMatchGMonth** (OSCTXT *pctx)
- EXTERNXML int **rtXmlMatchGMonthDay** (OSCTXT *pctx)
- EXTERNXML int **rtXmlMatchGDay** (OSCTXT *pctx)
- EXTERNXML OSBOOL **rtXmlCmpBase64Str** (OSCTXT *pctx, OSUINT32 noctx1, const OSOCTET *data1, const OSUTF8CHAR *data2)
- EXTERNXML OSBOOL **rtXmlCmpHexStr** (OSCTXT *pctx, OSUINT32 noctx1, const OSOCTET *data1, const OSUTF8CHAR *data2)
- EXTERNXML int **rtSaxGetAttributeID** (OSCTXT *pctx, const OSUTF8CHAR *attrName, size_t nAttr, const OSUTF8CHAR *attrNames[], OSUINT32 attrPresent[])
- EXTERNXML int **rtSaxGetElemID** (OSINT16 *pState, OSINT16 prevElemIdx, const OSUTF8CHAR *localName, const OSSAXElemTableRec idtab[], const int *fstab, int fstabRows, int fstabCols)
- EXTERNXML int **rtSaxGetElemID8** (OSINT16 *pState, OSINT16 prevElemIdx, const OSUTF8CHAR *localName, const OSSAXElemTableRec idtab[], const OSINT8 *fstab, int fstabRows, int fstabCols)
- EXTERNXML int **rtSaxFindElemID** (OSINT16 *pState, OSINT16 prevElemIdx, const OSUTF8CHAR *localName, const OSSAXElemTableRec idtab[], const int *fstab, int fstabRows, int fstabCols)
- EXTERNXML int **rtSaxFindElemID8** (OSINT16 *pState, OSINT16 prevElemIdx, const OSUTF8CHAR *localName, const OSSAXElemTableRec idtab[], const OSINT8 *fstab, int fstabRows, int fstabCols)
- EXTERNXML OSBOOL **rtSaxIsEmptyBuffer** (OSCTXT *pctx, int whitespace)
- EXTERNXML int **rtSaxLookupElemID** (OSCTXT *pctx, OSINT16 *pState, OSINT16 prevElemIdx, const OSUTF8CHAR *localName, const OSUTF8CHAR *qName, const OSSAXElemTableRec idtab[], const int *fstab, int fstabRows, int fstabCols)
- EXTERNXML int **rtSaxLookupElemID8** (OSCTXT *pctx, OSINT16 *pState, OSINT16 prevElemIdx, const OSUTF8CHAR *localName, const OSUTF8CHAR *qName, const OSSAXElemTableRec idtab[], const OSINT8 *fstab, int fstabRows, int fstabCols)
- EXTERNXML int **rtSaxStrListParse** (OSCTXT *pctx, OSRTMEMBUF *pMemBuf, OSRTDList *pvalue)
- EXTERNXML int **rtSaxStrListMatch** (OSCTXT *pctx)
- 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, size_t 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)
- EXTERNXML void **rtXmlTreatWhitespaces** (OSCTXT *pctxt, int whiteSpaceType)
- EXTERNXML void **rtErrXmlInit** (void)
- EXTERNXML int **rtXmlpDecAny** (OSCTXT *pctxt, const OSUTF8CHAR **pvalue)
- EXTERNXML int **rtXmlpDecAnyAttrStr** (OSCTXT *pctxt, const OSUTF8CHAR **ppAttrStr, size_t index)
- EXTERNXML int **rtXmlpDecAnyElem** (OSCTXT *pctxt, const OSUTF8CHAR **pvalue)
- EXTERNXML int **rtXmlpDecBase64Str** (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSINT32 bufsize)
- EXTERNXML int **rtXmlpDecBigInt** (OSCTXT *pctxt, const OSUTF8CHAR **pvalue)
- EXTERNXML int **rtXmlpDecBitString** (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnbits, OSUINT32 bufsize)
- EXTERNXML int **rtXmlpDecBool** (OSCTXT *pctxt, OSBOOL *pvalue)
- EXTERNXML int **rtXmlpDecDate** (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlpDecDateTime** (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlpDecDecimal** (OSCTXT *pctxt, OSREAL *pvalue, int totalDigits, int fractionDigits)
- EXTERNXML int **rtXmlpDecDouble** (OSCTXT *pctxt, OSREAL *pvalue, int totalDigits, int fractionDigits)
- EXTERNXML int **rtXmlpDecDynBase64Str** (OSCTXT *pctxt, OSDynOctStr *pvalue)
- EXTERNXML int **rtXmlpDecDynBitString** (OSCTXT *pctxt, OSDynOctStr *pvalue)
- EXTERNXML int **rtXmlpDecDynHexStr** (OSCTXT *pctxt, OSDynOctStr *pvalue)
- EXTERNXML int **rtXmlpDecDynUnicodeStr** (OSCTXT *pctxt, const OSUNICHAR **ppdata, OSUINT32 *pnchars)
- EXTERNXML int **rtXmlpDecDynUTF8Str** (OSCTXT *pctxt, const OSUTF8CHAR **outdata)
- EXTERNXML int **rtXmlpDecGDay** (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlpDecGMonth** (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlpDecGMonthDay** (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlpDecGYear** (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlpDecGYearMonth** (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlpDecHexStr** (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSINT32 bufsize)
- EXTERNXML int **rtXmlpDecInt** (OSCTXT *pctxt, OSINT32 *pvalue)
- EXTERNXML int **rtXmlpDecInt8** (OSCTXT *pctxt, OSINT8 *pvalue)
- EXTERNXML int **rtXmlpDecInt16** (OSCTXT *pctxt, OSINT16 *pvalue)
- EXTERNXML int **rtXmlpDecInt64** (OSCTXT *pctxt, OSINT64 *pvalue)
- EXTERNXML int **rtXmlpDecNamedBits** (OSCTXT *pctxt, const OSBitMapItem *pBitMap, OSOCTET *pvalue, OSUINT32 *pnbits, OSUINT32 bufsize)
- EXTERNXML int **rtXmlpDecStrList** (OSCTXT *pctxt, OSRTDList *plist)
- EXTERNXML int **rtXmlpDecTime** (OSCTXT *pctxt, OSXSDDateTime *pvalue)
- EXTERNXML int **rtXmlpDecUInt** (OSCTXT *pctxt, OSUINT32 *pvalue)
- EXTERNXML int **rtXmlpDecUInt8** (OSCTXT *pctxt, OSOCTET *pvalue)
- EXTERNXML int **rtXmlpDecUInt16** (OSCTXT *pctxt, OSUINT16 *pvalue)

- EXTERNXML int **rtXmlpDecUInt64** (OSCTXT *pctxt, OSUINT64 *pvalue)
- EXTERNXML int **rtXmlpDecXmlStr** (OSCTXT *pctxt, OSXMLSTRING *outdata)
- EXTERNXML int **rtXmlpDecXSIAAttr** (OSCTXT *pctxt, const OSXMLNameFragments *attrName)
- EXTERNXML int **rtXmlpDecXSITypeAttr** (OSCTXT *pctxt, const OSXMLNameFragments *attrName, const OSUTF8CHAR **ppAttrValue)
- EXTERNXML int **rtXmlpGetAttributeID** (OSCTXT *pctxt, const OSXMLStrFragment *attrName, size_t nAttr, const OSXMLAttrDescr attrNames[], OSUINT32 attrPresent[])
- EXTERNXML int **rtXmlpGetNextElem** (OSCTXT *pctxt, OSXMLElemDescr *pElem, OSINT32 level)
- EXTERNXML int **rtXmlpGetNextElemID** (OSCTXT *pctxt, const OSXMLElemIDRec *tab, size_t nrows, OSINT32 level, OSBOOL continueParse)
- EXTERNXML OSBOOL **rtXmlpIsInGroup** (int elemID, int grpId, const OSBOOL *grpTab, int nElems)
- EXTERNXML int **rtXmlpMarkLastEventActive** (OSCTXT *pctxt)
- EXTERNXML int **rtXmlpMatchStartTag** (OSCTXT *pctxt, const OSUTF8CHAR *elemLocalName, const OSUTF8CHAR *elemURI)
- EXTERNXML int **rtXmlpMatchEndTag** (OSCTXT *pctxt, OSINT32 level)
- EXTERNXML OSBOOL **rtXmlpMatchElemId** (OSCTXT *pctxt, int elemID, int matchingID)
- EXTERNXML OSBOOL **rtXmlpHasAttributes** (OSCTXT *pctxt)
- EXTERNXML int **rtXmlpGetAttributeCount** (OSCTXT *pctxt)
- EXTERNXML void **rtXmlpGetContent** (OSCTXT *pctxt, int level)
- EXTERNXML int **rtXmlpSelectAttribute** (OSCTXT *pctxt, OSXMLNameFragments *pAttr, size_t index)
- EXTERNXML int **rtXmlpCreateReader** (OSCTXT *pctxt)
- EXTERNXML OSINT32 **rtXmlpGetCurrentLevel** (OSCTXT *pctxt)
- EXTERNXML void **rtXmlpSetWhiteSpaceMode** (OSCTXT *pctxt, OSXMLWhiteSpaceMode whiteSpaceMode)
- EXTERNXML void **rtXmlpSetMixedContentMode** (OSCTXT *pctxt, OSBOOL mixedContentMode)
- EXTERNXML OSBOOL **rtXmlpIsContentMode** (OSCTXT *pctxt)
- EXTERNXML void **rtXmlpSetListMode** (OSCTXT *pctxt)
- EXTERNXML OSBOOL **rtXmlpListHasItem** (OSCTXT *pctxt)
- EXTERNXML void **rtXmlpCountListItems** (OSCTXT *pctxt, OSUINT32 *itemCnt)
- EXTERNXML int **rtXmlpGetNextSeqElemID** (OSCTXT *pctxt, const OSXMLElemIDRec *tab, const OSXMLGroupDesc *ppGroup, int curID, int lastMandatoryID)
- EXTERNXML int **rtXmlpGetNextAllElemID** (OSCTXT *pctxt, const OSXMLElemIDRec *tab, size_t nrows, const OSUINT8 *pOrder, OSUINT32 nOrder, OSUINT32 maxOrder, int anyID)

7.1.1 Detailed Description

XML low-level C encode/decode functions.

7.1.2 Define Documentation

7.1.2.1 #define OSXMLQNAMEEQUALS(xnamefrag, qnametext)

Value:

```
rtxUTF8StrnEqual \
(xnamefrag.mQName.value, OSUTF8(qnametext), xnamefrag.mQName.length)
```

7.1.3 Function Documentation

7.1.3.1 EXTERNXML int rtSaxGetElemID (OSINT16 * pState, OSINT16 prevElemIdx, const OSUTF8CHAR * localName, const OSSAXElemTableRec idtab[], const int * fstab, int fstabRows, int fstabCols)

This function looks up a sequence element name in the given element info array. It ensures elements are received in the correct order and also sets the required element count variable.

Parameters:

- pState* The pointer to state variable to be changed.
- prevElemIdx* Previous index of element. The search will be started from this element for better performance.
- localName* Local name of XML element
- idtab* Element ID table
- fstab* Finite state table
- fstabRows* Number of rows in *fstab*.
- fstabCols* Number of columns in *fstab*.

7.1.3.2 EXTERNXML int rtSaxGetElemID8 (OSINT16 * pState, OSINT16 prevElemIdx, const OSUTF8CHAR * localName, const OSSAXElemTableRec idtab[], const OSINT8 * fstab, int fstabRows, int fstabCols)

This function is a space optimized version of `rtSaxGetElemID`. It operates with array of 8-bit integers (OSINT8) instead of 32-bit integers (int).

Parameters:

- pState* The pointer to state variable to be changed.
- prevElemIdx* Previous index of element. The search will be started from this element + 1 for better performance.
- localName* Local name of XML element
- idtab* Element ID table
- fstab* Finite state table (array of 8-bit integers)
- fstabRows* Number of rows in *fstab*.
- fstabCols* Number of columns in *fstab*.

7.1.3.3 EXTERNXML OSBOOL rtSaxIsEmptyBuffer (OSCTXT * pctxt, int whitespace)

This function checks if the buffer in the context is empty or not. Testing is performed according to 'whitespace' rule.

Parameters:

- pctxt* Pointer to OSCTXT structure
- whitespace* One of the following constant:
 - OS_WHITESPACE_COLLAPSE
 - OS_WHITESPACE_PRESERVE
 - OS_WHITESPACE_REPLACE

Returns:

- TRUE, if the buffer contains empty string.

7.1.3.4 EXTERNXML int rtSaxStrListMatch (OSCTXT * *pctxt*)

This function matches the list of strings. It is used for matching NMTOKENS, IDREFS, NMENTITIES.

Parameters:

pctxt Pointer to OSCTXT structure

Returns:

0 - if success, negative value is error.

7.1.3.5 EXTERNXML int rtSaxStrListParse (OSCTXT * *pctxt*, OSRTMEMBUF * *pMemBuf*, OSRTDList * *pvalue*)

This function parses the list of strings. It is used for parsing NMTOKENS, IDREFS, NMENTITIES.

Parameters:

pctxt Pointer to OSCTXT structure. Can be NULL, if *pMemBuf* is not NULL.

pMemBuf Pointer to memory buffer structure. Can be NULL, if *pctxt* is not NULL.

pvalue Doubly-linked list for parsed strings.

Returns:

0 - if success, negative value is error.

7.1.3.6 EXTERNXML int rtXmlCreateFileInputSource (OSCTXT * *pctxt*, const char * *filepath*)

This function creates an XML document file input source. The document can then be decoded by invoking an XML decode function.

Parameters:

pctxt Pointer to context block structure.

filepath Full pathname of XML document file to open.

Returns:

Completion status of operation:

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

7.1.3.7 EXTERNXML int rtXmlInitContext (OSCTXT * *pctxt*)

This function initializes a context variable for XML encoding or decoding.

Parameters:

pctxt Pointer to OSCTXT structure

7.1.3.8 EXTERNXML int rtXmlInitCtxAppInfo (OSCTXT * *pctxt*)

This function initializes the XML application info section of the given context.

Parameters:

pctxt Pointer to OSCTXT structure

7.1.3.9 EXTERNXML int rtXmlMatchBase64Str (OSCTXT * *pctxt*, size_t *minLength*, size_t *maxLength*)

This function tests the context buffer for containing a correct base64 string. It does not decode the value.

Parameters:

pctxt Pointer to OSCTXT structure

minLength A minimal length of expected string.

maxLength A maximal length of expected string.

Returns:

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

7.1.3.10 EXTERNXML int rtXmlMatchDate (OSCTXT * *pctxt*)

This function tests the context buffer for containing a correct date string. It does not decode the value.

Parameters:

pctxt Pointer to OSCTXT structure

Returns:

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

7.1.3.11 EXTERNXML int rtXmlMatchDateTime (OSCTXT * *pctxt*)

This function tests the context buffer for containing a correct dateTime string. It does not decode the value.

Parameters:

pctxt Pointer to OSCTXT structure

Returns:

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

7.1.3.12 EXTERNXML int rtXmlMatchGDay (OSCTXT * *pctxt*)

This function tests the context buffer for containing a correct gDay string. It does not decode the value.

Parameters:

pctxt Pointer to OSCTXT structure

Returns:

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

7.1.3.13 EXTERNXML int rtXmlMatchGMonth (OSCTXT * *pctxt*)

This function tests the context buffer for containing a correct gMonth string. It does not decode the value.

Parameters:

pctxt Pointer to OSCTXT structure

Returns:

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

7.1.3.14 EXTERNXML int rtXmlMatchGMonthDay (OSCTXT * *pctxt*)

This function tests the context buffer for containing a correct gMonthDay string. It does not decode the value.

Parameters:

pctxt Pointer to OSCTXT structure

Returns:

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

7.1.3.15 EXTERNXML int rtXmlMatchGYear (OSCTXT * *pctxt*)

This function tests the context buffer for containing a correct gYear string. It does not decode the value.

Parameters:

pctxt Pointer to OSCTXT structure

Returns:

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

7.1.3.16 EXTERNXML int rtXmlMatchGYearMonth (OSCTXT * *pctxt*)

This function tests the context buffer for containing a correct gYearMonth string. It does not decode the value.

Parameters:

pctxt Pointer to OSCTXT structure

Returns:

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

7.1.3.17 EXTERNXML int rtXmlMatchHexStr (OSCTXT * *pctxt*, size_t *minLength*, size_t *maxLength*)

This function tests the context buffer for containing a correct hexadecimal string. It does not decode the value.

Parameters:

pctxt Pointer to OSCTXT structure

minLength A minimal length of expected string.

maxLength A maximal length of expected string.

Returns:

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

7.1.3.18 EXTERNXML int rtXmlMatchTime (OSCTXT * *pctxt*)

This function tests the context buffer for containing a correct time string. It does not decode the value.

Parameters:

pctxt Pointer to OSCTXT structure

Returns:

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

7.1.3.19 EXTERNXML int rtXmlSetEncDocHdr (OSCTXT * *pctxt*, OSBOOL *value*)

This function sets the option to add the XML document header (i.e. <?xml version="1.0" encoding="UTF-8"?>) to the XML output stream.

Parameters:

pctxt Pointer to OSCTXT structure

value Boolean value: true = add document header

Returns:

Status of operation: 0 if OK, negative status code if error.

7.1.3.20 EXTERNXML int rtXmlSetEncoding (OSCTXT * *pctxt*, OSXMLEncoding *encoding*)

This function sets the XML output encoding to the given value. Currently, only UTF-8 encoding is supported.

Parameters:

pctxt Pointer to OSCTXT structure

encoding XML output encoding format (UTF-8 or UTF-16)

Returns:

Status of operation: 0 if OK, negative status code if error.

7.1.3.21 EXTERNXML int rtXmlSetFormatting (OSCTXT * *pctxt*, OSBOOL *doFormatting*)

This function 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. Also the function 'rtXmlSetIndent' might be used to set the exact size of indentation.

Parameters:

pctxt Pointer to OSCTXT structure

doFormatting Boolean value indicating if formatting is to be done

Returns:

Status of operation: 0 if OK, negative status code if error.

7.1.3.22 EXTERNXML int rtXmlSetIndent (OSCTXT * *pctxt*, OSUINT8 *indent*)

This function sets XML output indent to the given value.

Parameters:

pctxt Pointer to OSCTXT structure

indent Number of spaces per indent. Default is 3.

Returns:

Status of operation: 0 if OK, negative status code if error.

7.1.3.23 EXTERNXML int rtXmlSetIndentChar (OSCTXT * *pctxt*, char *indentChar*)

This function sets XML output indent character to the given value.

Parameters:

pctxt Pointer to OSCTXT structure

indentChar Indent character. Default is space.

Returns:

Status of operation: 0 if OK, negative status code if error.

7.1.3.24 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. This attribute is optional: if not set, no xsi:noNamespaceSchemaLocation attribute will be added.

Parameters:

pctxt Pointer to OSCTXT structure
schemaLocation Schema location attribute value

Returns:

Status of operation: 0 if OK, negative status code if error.

7.1.3.25 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. This attribute is optional: if not set, no xsi:schemaLocation attribute will be added.

Parameters:

pctxt Pointer to OSCTXT structure
schemaLocation Schema location attribute value

Returns:

Status of operation: 0 if OK, negative status code if error.

7.1.3.26 EXTERNXML int rtXmlSetXSITypeAttr (OSCTXT * *pctxt*, const OSUTF8CHAR * *xsiType*)

This function sets the XML Schema Instance (xsi) type attribute value. This will cause an xsi:type attribute to be added to the top level element in an encoded XML instance.

Parameters:

pctxt Pointer to OSCTXT structure
xsiType xsi:type attribute value

Returns:

Status of operation: 0 if OK, negative status code if error.

7.2 rtXmlCppMsgBuf.h File Reference

```
#include "rtxsrc/OSRTMsgBuf.h"  
#include "rtxsrc/OSRTInputStream.h"  
#include "rtxsrc/OSRTOutputStream.h"  
#include "rtxmlsrc/osrtxml.h"  
#include "rtxmlsrc/rtSaxCppParser.h"
```

Classes

- class [OSXMLMessageBuffer](#)
- class [OSXMLEncodeBuffer](#)
- class [OSXMLEncodeStream](#)
- class [OSXMLDecodeBuffer](#)

7.2.1 Detailed Description

7.3 rtXmlCppXSDElement.h File Reference

```
#include "rtxsrc/OSRTContext.h"  
#include "rtxsrc/OSRTMsgBufIF.h"  
#include "rtxsrc/rtxErrCodes.h"  
#include "rtxmlsrc/osrtxml.h"
```

Classes

- class [OSXSDGlobalElement](#)

7.3.1 Detailed Description

C++ run-time XML schema global element class definition.

7.4 rtXmlErrCodes.h File Reference

```
#include "rtxsrc/rtxErrCodes.h"
```

Defines

- #define [XML_OK_EOB](#) 0x7fffffff
- #define [XML_OK_FRAG](#) XML_OK_EOB
- #define [XML_E_BASE](#) -200
- #define [XML_E_GENERR](#) (XML_E_BASE)
- #define [XML_E_INVSYMBOL](#) (XML_E_BASE-1)
- #define [XML_E_TAGMISMATCH](#) (XML_E_BASE-2)
- #define [XML_E_DUPLATTR](#) (XML_E_BASE-3)
- #define [XML_E_BADCHARREF](#) (XML_E_BASE-4)
- #define [XML_E_INVMODE](#) (XML_E_BASE-5)
- #define [XML_E_UNEXPEOF](#) (XML_E_BASE-6)
- #define [XML_E_NOMATCH](#) (XML_E_BASE-7)
- #define [XML_E_ELEMMISRQ](#) (XML_E_BASE-8)
- #define [XML_E_ELEMSMISRQ](#) (XML_E_BASE-9)
- #define [XML_E_TOOFWELEMS](#) (XML_E_BASE-10)
- #define [XML_E_UNEXPSTARTTAG](#) (XML_E_BASE-11)
- #define [XML_E_UNEXPENDTAG](#) (XML_E_BASE-12)

7.4.1 Detailed Description

List of numeric status codes that can be returned by ASN1C run-time functions and generated code.

Index

- ~OSXSDGlobalElement
 - OSXSDGlobalElement, 86
- ASN.1-XML encode/decode functions., 8
- asn1xml
 - rtAsn1XmlAddAnyAttr, 8
 - rtAsn1XmlEncGenTime, 8
 - rtAsn1XmlEncObjId, 9
 - rtAsn1XmlEncOpenType, 9
 - rtAsn1XmlEncOpenTypeExt, 10
 - rtAsn1XmlEncRelOID, 10
 - rtAsn1XmlEncUnivStr, 10
 - rtAsn1XmlEncUTCTime, 11
 - rtAsn1XmlFmtAttrStr, 11
 - rtAsn1XmlParseAttrStr, 11
 - rtAsn1XmlpDecDynBitStr, 12
 - rtAsn1XmlpDecObjId, 12
 - rtAsn1XmlpDecRelOID, 12
 - rtAsn1XmlpDecUnivStr, 13
 - rtXmlpDecListOfASN1DynBitStr, 13
- decode
 - OSXSDGlobalElement, 86
- decodeFrom
 - OSXSDGlobalElement, 86
- decodeXML
 - OSXMLDecodeBuffer, 75
- encode
 - OSXSDGlobalElement, 86
- encodeTo
 - OSXSDGlobalElement, 86
- getAppInfo
 - OSXMLMessageBuffer, 82
- getCtxtPtr
 - OSXSDGlobalElement, 87
- getIndent
 - OSXMLMessageBuffer, 82
- getIndentChar
 - OSXMLMessageBuffer, 83
- getMsgLen
 - OSXMLEncodeBuffer, 79
- getMsgPtr
 - OSXMLEncodeStream, 81
- getXmlFileName
 - OSXMLDecodeBuffer, 75
- init
 - OSXMLDecodeBuffer, 75
 - OSXMLEncodeBuffer, 79
- isA
 - OSXMLDecodeBuffer, 75
 - OSXMLEncodeBuffer, 79
 - OSXMLEncodeStream, 81
- mbOwnStream
 - OSXMLEncodeStream, 81
- memAlloc
 - OSXSDGlobalElement, 87
- memFreePtr
 - OSXSDGlobalElement, 87
- mInput
 - OSXMLDecodeBuffer, 76
- mInputId
 - OSXMLDecodeBuffer, 76
- mpContext
 - OSXSDGlobalElement, 88
- mpMsgBuf
 - OSXSDGlobalElement, 88
- mpStream
 - OSXMLEncodeStream, 81
- osrtxml.h, 89
 - OSXMLQNAMEEQUALS, 96
 - rtSaxGetElemID, 97
 - rtSaxGetElemID8, 97
 - rtSaxIsEmptyBuffer, 97
 - rtSaxStrListMatch, 97
 - rtSaxStrListParse, 98
 - rtXmlCreateFileInputSource, 98
 - rtXmlInitContext, 98
 - rtXmlInitCtxtAppInfo, 98
 - rtXmlMatchBase64Str, 99
 - rtXmlMatchDate, 99
 - rtXmlMatchDateTime, 99
 - rtXmlMatchGDay, 99
 - rtXmlMatchGMonth, 100
 - rtXmlMatchGMonthDay, 100
 - rtXmlMatchGYear, 100
 - rtXmlMatchGYearMonth, 100
 - rtXmlMatchHexStr, 101

- rtXmlMatchTime, 101
- rtXmlSetEncDocHdr, 101
- rtXmlSetEncoding, 101
- rtXmlSetFormatting, 102
- rtXmlSetIndent, 102
- rtXmlSetIndentChar, 102
- rtXmlSetNoNSSchemaLocation, 102
- rtXmlSetSchemaLocation, 103
- rtXmlSetXSITypeAttr, 103
- OSXMLDecodeBuffer, 73
 - OSXMLDecodeBuffer, 74
- OSXMLDecodeBuffer
 - decodeXML, 75
 - getXmlFileName, 75
 - init, 75
 - isA, 75
 - mInput, 76
 - mInputId, 76
 - OSXMLDecodeBuffer, 74
 - parseElementName, 76
 - parseElemQName, 76
 - setMaxErrors, 76
- OSXMLEncodeBuffer, 78
 - OSXMLEncodeBuffer, 78
- OSXMLEncodeBuffer
 - getMsgLen, 79
 - init, 79
 - isA, 79
 - OSXMLEncodeBuffer, 78
 - write, 79
- OSXMLEncodeStream, 80
 - OSXMLEncodeStream, 80
- OSXMLEncodeStream
 - getMsgPtr, 81
 - isA, 81
 - mbOwnStream, 81
 - mpStream, 81
 - OSXMLEncodeStream, 80
- OSXMLMessageBuffer, 82
 - OSXMLMessageBuffer, 82
- OSXMLMessageBuffer
 - getAppInfo, 82
 - getIndent, 82
 - getIndentChar, 83
 - OSXMLMessageBuffer, 82
 - setAppInfo, 83
 - setFormatting, 83
 - setIndent, 83
 - setIndentChar, 83
 - setNamespace, 83
- OSXMLQNAMEEQUALS
 - osrxml.h, 96
- OSXMLWhiteSpaceMode
 - rtXmlpDec, 59
- OSXSDGlobalElement, 85
 - OSXSDGlobalElement, 85, 86
- OSXSDGlobalElement
 - ~OSXSDGlobalElement, 86
 - decode, 86
 - decodeFrom, 86
 - encode, 86
 - encodeTo, 86
 - getCtxtPtr, 87
 - memAlloc, 87
 - memFreePtr, 87
 - mpContext, 88
 - mpMsgBuf, 88
 - OSXSDGlobalElement, 85, 86
 - setDefaultNamespace, 87
 - setMsgBuf, 87
 - setNamespace, 87
 - setXSIType, 88
 - validate, 88
 - validateFrom, 88
- parseElementName
 - OSXMLDecodeBuffer, 76
- parseElemQName
 - OSXMLDecodeBuffer, 76
- rtAsn1XmlAddAnyAttr
 - asn1xml, 8
- rtAsn1XmlEncGenTime
 - asn1xml, 8
- rtAsn1XmlEncObjId
 - asn1xml, 9
- rtAsn1XmlEncOpenType
 - asn1xml, 9
- rtAsn1XmlEncOpenTypeExt
 - asn1xml, 10
- rtAsn1XmlEncRelOID
 - asn1xml, 10
- rtAsn1XmlEncUnivStr
 - asn1xml, 10
- rtAsn1XmlEncUTCTime
 - asn1xml, 11
- rtAsn1XmlFmtAttrStr
 - asn1xml, 11
- rtAsn1XmlParseAttrStr
 - asn1xml, 11
- rtAsn1XmlpDecDynBitStr
 - asn1xml, 12
- rtAsn1XmlpDecObjId
 - asn1xml, 12
- rtAsn1XmlpDecRelOID
 - asn1xml, 12
- rtAsn1XmlpDecUnivStr
 - asn1xml, 13

- rtSaxGetElemID
 - osrtxml.h, [97](#)
- rtSaxGetElemID8
 - osrtxml.h, [97](#)
- rtSaxIsEmptyBuffer
 - osrtxml.h, [97](#)
- rtSaxStrListMatch
 - osrtxml.h, [97](#)
- rtSaxStrListParse
 - osrtxml.h, [98](#)
- rtXmlCppMsgBuf.h, [104](#)
- rtXmlCppXSDElement.h, [105](#)
- rtXmlCreateFileInputSource
 - osrtxml.h, [98](#)
- rtXmlDec
 - rtXmlDecBase64Binary, [15](#)
 - rtXmlDecBase64Str, [15](#)
 - rtXmlDecBase64StrValue, [15](#)
 - rtXmlDecBigInt, [16](#)
 - rtXmlDecBool, [16](#)
 - rtXmlDecDate, [16](#)
 - rtXmlDecDateTime, [17](#)
 - rtXmlDecDecimal, [17](#)
 - rtXmlDecDouble, [17](#)
 - rtXmlDecDynBase64Str, [18](#)
 - rtXmlDecDynHexStr, [18](#)
 - rtXmlDecDynUTF8Str, [18](#)
 - rtXmlDecGDay, [19](#)
 - rtXmlDecGMonth, [19](#)
 - rtXmlDecGMonthDay, [19](#)
 - rtXmlDecGYear, [20](#)
 - rtXmlDecGYearMonth, [20](#)
 - rtXmlDecHexBinary, [20](#)
 - rtXmlDecHexStr, [21](#)
 - rtXmlDecInt, [21](#)
 - rtXmlDecInt16, [21](#)
 - rtXmlDecInt64, [22](#)
 - rtXmlDecInt8, [22](#)
 - rtXmlDecQName, [22](#)
 - rtXmlDecTime, [23](#)
 - rtXmlDecUInt, [23](#)
 - rtXmlDecUInt16, [23](#)
 - rtXmlDecUInt64, [24](#)
 - rtXmlDecUInt8, [24](#)
 - rtXmlDecXmlStr, [24](#)
 - rtXmlDecXSIAAttr, [25](#)
 - rtXmlDecXSIAAttrs, [25](#)
 - rtXmlParseElementName, [25](#)
 - rtXmlParseElemQName, [26](#)
- rtXmlDecBase64Binary
 - rtXmlDec, [15](#)
- rtXmlDecBase64Str
 - rtXmlDec, [15](#)
- rtXmlDecBase64StrValue
 - rtXmlDec, [15](#)
- rtXmlDecBigInt
 - rtXmlDec, [16](#)
- rtXmlDecBool
 - rtXmlDec, [16](#)
- rtXmlDecDate
 - rtXmlDec, [16](#)
- rtXmlDecDateTime
 - rtXmlDec, [17](#)
- rtXmlDecDecimal
 - rtXmlDec, [17](#)
- rtXmlDecDouble
 - rtXmlDec, [17](#)
- rtXmlDecDynBase64Str
 - rtXmlDec, [18](#)
- rtXmlDecDynHexStr
 - rtXmlDec, [18](#)
- rtXmlDecDynUTF8Str
 - rtXmlDec, [18](#)
- rtXmlDecGDay
 - rtXmlDec, [19](#)
- rtXmlDecGMonth
 - rtXmlDec, [19](#)
- rtXmlDecGMonthDay
 - rtXmlDec, [19](#)
- rtXmlDecGYear
 - rtXmlDec, [20](#)
- rtXmlDecGYearMonth
 - rtXmlDec, [20](#)
- rtXmlDecHexBinary
 - rtXmlDec, [20](#)
- rtXmlDecHexStr
 - rtXmlDec, [21](#)
- rtXmlDecInt
 - rtXmlDec, [21](#)
- rtXmlDecInt16
 - rtXmlDec, [21](#)
- rtXmlDecInt64
 - rtXmlDec, [22](#)
- rtXmlDecInt8
 - rtXmlDec, [22](#)
- rtXmlDecQName
 - rtXmlDec, [22](#)
- rtXmlDecTime
 - rtXmlDec, [23](#)
- rtXmlDecUInt
 - rtXmlDec, [23](#)
- rtXmlDecUInt16
 - rtXmlDec, [23](#)
- rtXmlDecUInt64
 - rtXmlDec, [24](#)
- rtXmlDecUInt8
 - rtXmlDec, [24](#)
- rtXmlDecXmlStr

- rtXmlDec, 24
- rtXmlDecXSIAttr
 - rtXmlDec, 25
- rtXmlDecXSIAttrs
 - rtXmlDec, 25
- rtXmlEnc
 - rtXmlEncAny, 30
 - rtXmlEncAnyAttr, 30
 - rtXmlEncBase64Binary, 31
 - rtXmlEncBase64BinaryAttr, 31
 - rtXmlEncBase64StringValue, 31
 - rtXmlEncBigInt, 32
 - rtXmlEncBigIntAttr, 32
 - rtXmlEncBigIntValue, 33
 - rtXmlEncBinStringValue, 33
 - rtXmlEncBitString, 33
 - rtXmlEncBool, 34
 - rtXmlEncBoolAttr, 34
 - rtXmlEncBoolValue, 34
 - rtXmlEncDate, 35
 - rtXmlEncDateTime, 35
 - rtXmlEncDateTimeValue, 35
 - rtXmlEncDateValue, 36
 - rtXmlEncDecimal, 36
 - rtXmlEncDecimalAttr, 36
 - rtXmlEncDecimalValue, 37
 - rtXmlEncDouble, 37
 - rtXmlEncDoubleAttr, 38
 - rtXmlEncDoubleValue, 38
 - rtXmlEncEmptyElement, 38
 - rtXmlEncEndDocument, 39
 - rtXmlEncEndElement, 39
 - rtXmlEncEndSoapEnv, 39
 - rtXmlEncFloat, 40
 - rtXmlEncFloatAttr, 40
 - rtXmlEncGDay, 40
 - rtXmlEncGDayValue, 41
 - rtXmlEncGMonth, 41
 - rtXmlEncGMonthDay, 41
 - rtXmlEncGMonthDayValue, 42
 - rtXmlEncGMonthValue, 42
 - rtXmlEncGYear, 42
 - rtXmlEncGYearMonth, 43
 - rtXmlEncGYearMonthValue, 43
 - rtXmlEncGYearValue, 43
 - rtXmlEncHexBinary, 44
 - rtXmlEncHexBinaryAttr, 44
 - rtXmlEncHexStringValue, 44
 - rtXmlEncIndent, 45
 - rtXmlEncInt, 45
 - rtXmlEncInt64, 45
 - rtXmlEncInt64Attr, 46
 - rtXmlEncInt64Value, 46
 - rtXmlEncIntAttr, 46
 - rtXmlEncIntPattern, 47
 - rtXmlEncIntValue, 47
 - rtXmlEncNamedBits, 47
 - rtXmlEncNSAttrs, 48
 - rtXmlEncSoapArrayTypeAttr, 48
 - rtXmlEncStartDocument, 49
 - rtXmlEncStartElement, 49
 - rtXmlEncStartElement2, 49
 - rtXmlEncStartSoapEnv, 50
 - rtXmlEncString, 50
 - rtXmlEncStringValue, 50
 - rtXmlEncStringValue2, 50
 - rtXmlEncTime, 51
 - rtXmlEncTimeValue, 51
 - rtXmlEncUInt, 51
 - rtXmlEncUInt64, 52
 - rtXmlEncUInt64Attr, 52
 - rtXmlEncUInt64Value, 53
 - rtXmlEncUIntAttr, 53
 - rtXmlEncUIntValue, 53
 - rtXmlEncUnicodeStr, 53
 - rtXmlEncUTF8Attr, 54
 - rtXmlEncUTF8Attr2, 54
 - rtXmlEncUTF8Str, 54
 - rtXmlEncXSIAttrs, 55
 - rtXmlFinalizeMemBuf, 30
 - rtXmlFreeInputSource, 55
 - rtXmlGetEncBufLen, 30
 - rtXmlGetEncBufPtr, 30
 - rtXmlGetIndent, 55
 - rtXmlGetIndentChar, 56
 - rtXmlSetEncBufPtr, 56
- rtXmlEncAny
 - rtXmlEnc, 30
- rtXmlEncAnyAttr
 - rtXmlEnc, 30
- rtXmlEncBase64Binary
 - rtXmlEnc, 31
- rtXmlEncBase64BinaryAttr
 - rtXmlEnc, 31
- rtXmlEncBase64StringValue
 - rtXmlEnc, 31
- rtXmlEncBigInt
 - rtXmlEnc, 32
- rtXmlEncBigIntAttr
 - rtXmlEnc, 32
- rtXmlEncBigIntValue
 - rtXmlEnc, 33
- rtXmlEncBinStringValue
 - rtXmlEnc, 33
- rtXmlEncBitString
 - rtXmlEnc, 33
- rtXmlEncBool
 - rtXmlEnc, 34

rtXmlEncBoolAttr	rtXmlEncGYearValue
rtXmlEnc, 34	rtXmlEnc, 43
rtXmlEncBoolValue	rtXmlEncHexBinary
rtXmlEnc, 34	rtXmlEnc, 44
rtXmlEncDate	rtXmlEncHexBinaryAttr
rtXmlEnc, 35	rtXmlEnc, 44
rtXmlEncDateTime	rtXmlEncHexStrValue
rtXmlEnc, 35	rtXmlEnc, 44
rtXmlEncDateTimeValue	rtXmlEncIndent
rtXmlEnc, 35	rtXmlEnc, 45
rtXmlEncDateValue	rtXmlEncInt
rtXmlEnc, 36	rtXmlEnc, 45
rtXmlEncDecimal	rtXmlEncInt64
rtXmlEnc, 36	rtXmlEnc, 45
rtXmlEncDecimalAttr	rtXmlEncInt64Attr
rtXmlEnc, 36	rtXmlEnc, 46
rtXmlEncDecimalValue	rtXmlEncInt64Value
rtXmlEnc, 37	rtXmlEnc, 46
rtXmlEncDouble	rtXmlEncIntAttr
rtXmlEnc, 37	rtXmlEnc, 46
rtXmlEncDoubleAttr	rtXmlEncIntPattern
rtXmlEnc, 38	rtXmlEnc, 47
rtXmlEncDoubleValue	rtXmlEncIntValue
rtXmlEnc, 38	rtXmlEnc, 47
rtXmlEncEmptyElement	rtXmlEncNamedBits
rtXmlEnc, 38	rtXmlEnc, 47
rtXmlEncEndDocument	rtXmlEncNSAttrs
rtXmlEnc, 39	rtXmlEnc, 48
rtXmlEncEndElement	rtXmlEncSoapArrayTypeAttr
rtXmlEnc, 39	rtXmlEnc, 48
rtXmlEncEndSoapEnv	rtXmlEncStartDocument
rtXmlEnc, 39	rtXmlEnc, 49
rtXmlEncFloat	rtXmlEncStartElement
rtXmlEnc, 40	rtXmlEnc, 49
rtXmlEncFloatAttr	rtXmlEncStartElement2
rtXmlEnc, 40	rtXmlEnc, 49
rtXmlEncGDay	rtXmlEncStartSoapEnv
rtXmlEnc, 40	rtXmlEnc, 50
rtXmlEncGDayValue	rtXmlEncString
rtXmlEnc, 41	rtXmlEnc, 50
rtXmlEncGMonth	rtXmlEncStringValue
rtXmlEnc, 41	rtXmlEnc, 50
rtXmlEncGMonthDay	rtXmlEncStringValue2
rtXmlEnc, 41	rtXmlEnc, 50
rtXmlEncGMonthDayValue	rtXmlEncTime
rtXmlEnc, 42	rtXmlEnc, 51
rtXmlEncGMonthValue	rtXmlEncTimeValue
rtXmlEnc, 42	rtXmlEnc, 51
rtXmlEncGYear	rtXmlEncUInt
rtXmlEnc, 42	rtXmlEnc, 51
rtXmlEncGYearMonth	rtXmlEncUInt64
rtXmlEnc, 43	rtXmlEnc, 52
rtXmlEncGYearMonthValue	rtXmlEncUInt64Attr
rtXmlEnc, 43	rtXmlEnc, 52

- rtXmlEncUInt64Value
 - rtXmlEnc, 53
- rtXmlEncUIntAttr
 - rtXmlEnc, 53
- rtXmlEncUIntValue
 - rtXmlEnc, 53
- rtXmlEncUnicodeStr
 - rtXmlEnc, 53
- rtXmlEncUTF8Attr
 - rtXmlEnc, 54
- rtXmlEncUTF8Attr2
 - rtXmlEnc, 54
- rtXmlEncUTF8Str
 - rtXmlEnc, 54
- rtXmlEncXSIAttrs
 - rtXmlEnc, 55
- rtXmlErrCodes.h, 106
- rtXmlFinalizeMemBuf
 - rtXmlEnc, 30
- rtXmlFreeInputSource
 - rtXmlEnc, 55
- rtXmlGetEncBufLen
 - rtXmlEnc, 30
- rtXmlGetEncBufPtr
 - rtXmlEnc, 30
- rtXmlGetIndent
 - rtXmlEnc, 55
- rtXmlGetIndentChar
 - rtXmlEnc, 56
- rtXmlInitContext
 - osrtxml.h, 98
- rtXmlInitCtxtAppInfo
 - osrtxml.h, 98
- rtXmlMatchBase64Str
 - osrtxml.h, 99
- rtXmlMatchDate
 - osrtxml.h, 99
- rtXmlMatchDateTime
 - osrtxml.h, 99
- rtXmlMatchGDay
 - osrtxml.h, 99
- rtXmlMatchGMonth
 - osrtxml.h, 100
- rtXmlMatchGMonthDay
 - osrtxml.h, 100
- rtXmlMatchGYear
 - osrtxml.h, 100
- rtXmlMatchGYearMonth
 - osrtxml.h, 100
- rtXmlMatchHexStr
 - osrtxml.h, 101
- rtXmlMatchTime
 - osrtxml.h, 101
- rtXmlParseElementName
 - rtXmlDec, 25
- rtXmlParseElemQName
 - rtXmlDec, 26
- rtXmlpDec
 - OSXMLWhiteSpaceMode, 59
 - rtXmlpDecAny, 59
 - rtXmlpDecAnyAttrStr, 59
 - rtXmlpDecAnyElem, 60
 - rtXmlpDecBase64Str, 60
 - rtXmlpDecBigInt, 61
 - rtXmlpDecBitString, 61
 - rtXmlpDecBool, 61
 - rtXmlpDecDate, 62
 - rtXmlpDecDateTime, 62
 - rtXmlpDecDecimal, 62
 - rtXmlpDecDouble, 63
 - rtXmlpDecDynBase64Str, 63
 - rtXmlpDecDynBitString, 64
 - rtXmlpDecDynHexStr, 64
 - rtXmlpDecDynUnicodeStr, 64
 - rtXmlpDecDynUTF8Str, 65
 - rtXmlpDecGDay, 65
 - rtXmlpDecGMonth, 65
 - rtXmlpDecGMonthDay, 65
 - rtXmlpDecGYear, 66
 - rtXmlpDecGYearMonth, 66
 - rtXmlpDecHexStr, 66
 - rtXmlpDecInt, 67
 - rtXmlpDecInt16, 67
 - rtXmlpDecInt64, 67
 - rtXmlpDecInt8, 68
 - rtXmlpDecNamedBits, 68
 - rtXmlpDecStrList, 68
 - rtXmlpDecTime, 69
 - rtXmlpDecUInt, 69
 - rtXmlpDecUInt16, 69
 - rtXmlpDecUInt64, 70
 - rtXmlpDecUInt8, 70
 - rtXmlpDecXmlStr, 70
 - rtXmlpDecXSIAttr, 71
 - rtXmlpDecXSITypeAttr, 71
 - rtXmlpSetWhiteSpaceMode, 72
- rtXmlpDecAny
 - rtXmlpDec, 59
- rtXmlpDecAnyAttrStr
 - rtXmlpDec, 59
- rtXmlpDecAnyElem
 - rtXmlpDec, 60
- rtXmlpDecBase64Str
 - rtXmlpDec, 60
- rtXmlpDecBigInt
 - rtXmlpDec, 61
- rtXmlpDecBitString
 - rtXmlpDec, 61

rtXmlpDecBool
 rtXmlpDec, 61
 rtXmlpDecDate
 rtXmlpDec, 62
 rtXmlpDecDateTime
 rtXmlpDec, 62
 rtXmlpDecDecimal
 rtXmlpDec, 62
 rtXmlpDecDouble
 rtXmlpDec, 63
 rtXmlpDecDynBase64Str
 rtXmlpDec, 63
 rtXmlpDecDynBitString
 rtXmlpDec, 64
 rtXmlpDecDynHexStr
 rtXmlpDec, 64
 rtXmlpDecDynUnicodeStr
 rtXmlpDec, 64
 rtXmlpDecDynUTF8Str
 rtXmlpDec, 65
 rtXmlpDecGDay
 rtXmlpDec, 65
 rtXmlpDecGMonth
 rtXmlpDec, 65
 rtXmlpDecGMonthDay
 rtXmlpDec, 65
 rtXmlpDecGYear
 rtXmlpDec, 66
 rtXmlpDecGYearMonth
 rtXmlpDec, 66
 rtXmlpDecHexStr
 rtXmlpDec, 66
 rtXmlpDecInt
 rtXmlpDec, 67
 rtXmlpDecInt16
 rtXmlpDec, 67
 rtXmlpDecInt64
 rtXmlpDec, 67
 rtXmlpDecInt8
 rtXmlpDec, 68
 rtXmlpDecListOfASN1DynBitStr
 asn1xml, 13
 rtXmlpDecNamedBits
 rtXmlpDec, 68
 rtXmlpDecStrList
 rtXmlpDec, 68
 rtXmlpDecTime
 rtXmlpDec, 69
 rtXmlpDecUInt
 rtXmlpDec, 69
 rtXmlpDecUInt16
 rtXmlpDec, 69
 rtXmlpDecUInt64
 rtXmlpDec, 70
 rtXmlpDecUInt8
 rtXmlpDec, 70
 rtXmlpDecXmlStr
 rtXmlpDec, 70
 rtXmlpDecXSIAAttr
 rtXmlpDec, 71
 rtXmlpDecXSITypeAttr
 rtXmlpDec, 71
 rtXmlpSetWhiteSpaceMode
 rtXmlpDec, 72
 rtXmlSetEncBufPtr
 rtXmlEnc, 56
 rtXmlSetEncDocHdr
 osrtxml.h, 101
 rtXmlSetEncoding
 osrtxml.h, 101
 rtXmlSetFormatting
 osrtxml.h, 102
 rtXmlSetIndent
 osrtxml.h, 102
 rtXmlSetIndentChar
 osrtxml.h, 102
 rtXmlSetNoNSSchemaLocation
 osrtxml.h, 102
 rtXmlSetSchemaLocation
 osrtxml.h, 103
 rtXmlSetXSITypeAttr
 osrtxml.h, 103
 rtXmlUtil
 rtXmlWriteToFile, 57
 rtXmlWriteToFile
 rtXmlUtil, 57
 Run-time error status codes., 5

 setAppInfo
 OSXMLMessageBuffer, 83
 setDefaultNamespace
 OSXSDGlobalElement, 87
 setFormatting
 OSXMLMessageBuffer, 83
 setIndent
 OSXMLMessageBuffer, 83
 setIndentChar
 OSXMLMessageBuffer, 83
 setMaxErrors
 OSXMLDecodeBuffer, 76
 setMsgBuf
 OSXSDGlobalElement, 87
 setNamespace
 OSXMLMessageBuffer, 83
 OSXSDGlobalElement, 87
 setXSIType
 OSXSDGlobalElement, 88
 validate

- OSXSDGlobalElement, 88
- validateFrom
 - OSXSDGlobalElement, 88
- write
 - OSXMLEncodeBuffer, 79

- XML decode functions., 14
- XML encode functions., 27
- XML pull-parser decode functions., 58
- XML utility functions., 57

- XML_E_BADCHARREF
 - xmlErrCodes, 5

- XML_E_BASE
 - xmlErrCodes, 5

- XML_E_DUPLATTR
 - xmlErrCodes, 6

- XML_E_ELEMMISRQ
 - xmlErrCodes, 6

- XML_E_ELEMSMISRQ
 - xmlErrCodes, 6

- XML_E_GENERR
 - xmlErrCodes, 6

- XML_E_INVMODE
 - xmlErrCodes, 6

- XML_E_INVSYMBOL
 - xmlErrCodes, 6

- XML_E_NOMATCH
 - xmlErrCodes, 6

- XML_E_TAGMISMATCH
 - xmlErrCodes, 6

- XML_E_TOOFWELEMS
 - xmlErrCodes, 6

- XML_E_UNEXPENDTAG
 - xmlErrCodes, 6

- XML_E_UNEXPEOF
 - xmlErrCodes, 6

- XML_E_UNEXPSTARTTAG
 - xmlErrCodes, 7

- XML_OK_EOB
 - xmlErrCodes, 7

- XML_OK_FRAG
 - xmlErrCodes, 7

- xmlErrCodes

- XML_E_BADCHARREF, 5

- XML_E_BASE, 5

- XML_E_DUPLATTR, 6

- XML_E_ELEMMISRQ, 6

- XML_E_ELEMSMISRQ, 6

- XML_E_GENERR, 6

- XML_E_INVMODE, 6

- XML_E_INVSYMBOL, 6

- XML_E_NOMATCH, 6

- XML_E_TAGMISMATCH, 6

- XML_E_TOOFWELEMS, 6

- XML_E_UNEXPENDTAG, 6

- XML_E_UNEXPEOF, 6

- XML_E_UNEXPSTARTTAG, 7

- XML_OK_EOB, 7

- XML_OK_FRAG, 7