

# ASN1C

---

ASN.1 Compiler  
Version 7.1  
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 ©1997–2017 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](mailto:info@obj-sys.com).



# Contents

<b>1</b>	<b>Main Page</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>2</b>
2.1	Modules . . . . .	2
<b>3</b>	<b>Data Structure Index</b>	<b>3</b>
3.1	Class Hierarchy . . . . .	3
<b>4</b>	<b>Data Structure Index</b>	<b>4</b>
4.1	Data Structures . . . . .	4
<b>5</b>	<b>File Index</b>	<b>5</b>
5.1	File List . . . . .	5
<b>6</b>	<b>Module Documentation</b>	<b>6</b>
6.1	ASN.1-XML encode/decode functions. . . . .	6
6.1.1	Function Documentation . . . . .	7
6.1.1.1	rtAsn1XmlAddAnyAttr . . . . .	7
6.1.1.2	rtAsn1XmlEncGenTime . . . . .	8
6.1.1.3	rtAsn1XmlEncObjId . . . . .	8
6.1.1.4	rtAsn1XmlEncOpenType . . . . .	9
6.1.1.5	rtAsn1XmlEncOpenTypeExt . . . . .	9
6.1.1.6	rtAsn1XmlEncReal . . . . .	9
6.1.1.7	rtAsn1XmlEncRelOID . . . . .	10
6.1.1.8	rtAsn1XmlEncUnivStr . . . . .	10
6.1.1.9	rtAsn1XmlEncUTCTime . . . . .	11
6.1.1.10	rtAsn1XmlFmtAttrStr . . . . .	11
6.1.1.11	rtAsn1XmlParseAttrStr . . . . .	11
6.1.1.12	rtAsn1XmlpDecDynBitStr . . . . .	12
6.1.1.13	rtAsn1XmlpDecDynBitStr64 . . . . .	12

6.1.1.14	rtAsn1XmlpDecGenTime	13
6.1.1.15	rtAsn1XmlpDecObjId	13
6.1.1.16	rtAsn1XmlpDecOpenType	13
6.1.1.17	rtAsn1XmlpDecReal	14
6.1.1.18	rtAsn1XmlpDecRelOID	14
6.1.1.19	rtAsn1XmlpDecUnivStr	14
6.1.1.20	rtAsn1XmlpDecUTCTime	15
6.1.1.21	rtXmlpDecListOfASN1DynBitStr	15
6.2	XML decode functions	16
6.2.1	Function Documentation	18
6.2.1.1	rtXmlDecBase64Binary	18
6.2.1.2	rtXmlDecBase64Str	19
6.2.1.3	rtXmlDecBase64StrValue	19
6.2.1.4	rtXmlDecBigInt	20
6.2.1.5	rtXmlDecBool	20
6.2.1.6	rtXmlDecDate	20
6.2.1.7	rtXmlDecDateTime	21
6.2.1.8	rtXmlDecDecimal	21
6.2.1.9	rtXmlDecDouble	21
6.2.1.10	rtXmlDecDynBase64Str	22
6.2.1.11	rtXmlDecDynHexStr	22
6.2.1.12	rtXmlDecDynUTF8Str	22
6.2.1.13	rtXmlDecEmptyElement	23
6.2.1.14	rtXmlDecGDay	23
6.2.1.15	rtXmlDecGMonth	23
6.2.1.16	rtXmlDecGMonthDay	24
6.2.1.17	rtXmlDecGYear	24
6.2.1.18	rtXmlDecGYearMonth	24
6.2.1.19	rtXmlDecHexBinary	25
6.2.1.20	rtXmlDecHexStr	25
6.2.1.21	rtXmlDecInt	25
6.2.1.22	rtXmlDecInt16	26
6.2.1.23	rtXmlDecInt64	26
6.2.1.24	rtXmlDecInt8	26
6.2.1.25	rtXmlDecNSAttr	27
6.2.1.26	rtXmlDecQName	27
6.2.1.27	rtXmlDecTime	28

6.2.1.28	rtXmlDecUInt	28
6.2.1.29	rtXmlDecUInt16	28
6.2.1.30	rtXmlDecUInt64	29
6.2.1.31	rtXmlDecUInt8	29
6.2.1.32	rtXmlDecUTF8Str	29
6.2.1.33	rtXmlDecXmlStr	30
6.2.1.34	rtXmlDecXSIAAttr	30
6.2.1.35	rtXmlDecXSIAAttrs	31
6.2.1.36	rtXmlParseElementName	31
6.2.1.37	rtXmlParseElemQName	31
6.3	XML encode functions.	32
6.3.1	Define Documentation	39
6.3.1.1	rtXmlFinalizeMemBuf	39
6.3.1.2	rtXmlGetEncBufLen	39
6.3.1.3	rtXmlGetEncBufPtr	39
6.3.2	Function Documentation	39
6.3.2.1	rtXmlEncAny	39
6.3.2.2	rtXmlEncAnyAttr	40
6.3.2.3	rtXmlEncAnyTypeValue	40
6.3.2.4	rtXmlEncBase64Binary	40
6.3.2.5	rtXmlEncBase64BinaryAttr	41
6.3.2.6	rtXmlEncBase64StrValue	41
6.3.2.7	rtXmlEncBigInt	42
6.3.2.8	rtXmlEncBigIntAttr	42
6.3.2.9	rtXmlEncBigIntValue	42
6.3.2.10	rtXmlEncBinStrValue	43
6.3.2.11	rtXmlEncBitString	43
6.3.2.12	rtXmlEncBOM	44
6.3.2.13	rtXmlEncBool	44
6.3.2.14	rtXmlEncBoolAttr	44
6.3.2.15	rtXmlEncBoolValue	45
6.3.2.16	rtXmlEncComment	45
6.3.2.17	rtXmlEncDate	45
6.3.2.18	rtXmlEncDateTime	46
6.3.2.19	rtXmlEncDateTimeValue	46
6.3.2.20	rtXmlEncDateValue	46
6.3.2.21	rtXmlEncDecimal	47

6.3.2.22	rtXmlEncDecimalAttr	47
6.3.2.23	rtXmlEncDecimalValue	47
6.3.2.24	rtXmlEncDouble	48
6.3.2.25	rtXmlEncDoubleAttr	48
6.3.2.26	rtXmlEncDoubleNormalValue	49
6.3.2.27	rtXmlEncDoubleValue	49
6.3.2.28	rtXmlEncEmptyElement	49
6.3.2.29	rtXmlEncEndDocument	50
6.3.2.30	rtXmlEncEndElement	50
6.3.2.31	rtXmlEncEndSoapElems	50
6.3.2.32	rtXmlEncEndSoapEnv	51
6.3.2.33	rtXmlEncFloat	51
6.3.2.34	rtXmlEncFloatAttr	52
6.3.2.35	rtXmlEncGDay	52
6.3.2.36	rtXmlEncGDayValue	52
6.3.2.37	rtXmlEncGMonth	53
6.3.2.38	rtXmlEncGMonthDay	53
6.3.2.39	rtXmlEncGMonthDayValue	53
6.3.2.40	rtXmlEncGMonthValue	54
6.3.2.41	rtXmlEncGYear	54
6.3.2.42	rtXmlEncGYearMonth	54
6.3.2.43	rtXmlEncGYearMonthValue	55
6.3.2.44	rtXmlEncGYearValue	55
6.3.2.45	rtXmlEncHexBinary	55
6.3.2.46	rtXmlEncHexBinaryAttr	56
6.3.2.47	rtXmlEncHexStrValue	56
6.3.2.48	rtXmlEncIndent	56
6.3.2.49	rtXmlEncInt	57
6.3.2.50	rtXmlEncInt64	57
6.3.2.51	rtXmlEncInt64Attr	57
6.3.2.52	rtXmlEncInt64Value	58
6.3.2.53	rtXmlEncIntAttr	58
6.3.2.54	rtXmlEncIntPattern	59
6.3.2.55	rtXmlEncIntValue	59
6.3.2.56	rtXmlEncNamedBits	59
6.3.2.57	rtXmlEncNSAttrs	60
6.3.2.58	rtXmlEncReal10	60



6.3.2.59	rtXmlEncSoapArrayTypeAttr	61
6.3.2.60	rtXmlEncStartDocument	61
6.3.2.61	rtXmlEncStartElement	61
6.3.2.62	rtXmlEncStartSoapElems	62
6.3.2.63	rtXmlEncStartSoapEnv	62
6.3.2.64	rtXmlEncString	62
6.3.2.65	rtXmlEncStringValue	63
6.3.2.66	rtXmlEncStringValue2	63
6.3.2.67	rtXmlEncTermStartElement	64
6.3.2.68	rtXmlEncTime	64
6.3.2.69	rtXmlEncTimeValue	64
6.3.2.70	rtXmlEncUInt	65
6.3.2.71	rtXmlEncUInt64	65
6.3.2.72	rtXmlEncUInt64Attr	65
6.3.2.73	rtXmlEncUInt64Value	66
6.3.2.74	rtXmlEncUIntAttr	66
6.3.2.75	rtXmlEncUIntValue	67
6.3.2.76	rtXmlEncUnicodeStr	67
6.3.2.77	rtXmlEncUTF8Attr	67
6.3.2.78	rtXmlEncUTF8Attr2	68
6.3.2.79	rtXmlEncUTF8Str	68
6.3.2.80	rtXmlEncXSAttrs	68
6.3.2.81	rtXmlEncXSINilAttr	69
6.3.2.82	rtXmlEncXSITypeAttr	69
6.3.2.83	rtXmlEncXSITypeAttr2	69
6.3.2.84	rtXmlFreeInputSource	70
6.3.2.85	rtXmlGetIndent	70
6.3.2.86	rtXmlGetIndentChar	70
6.3.2.87	rtXmlGetWriteBOM	70
6.3.2.88	rtXmlPrintNSAttrs	71
6.3.2.89	rtXmlSetEncBufPtr	71
6.4	XML utility functions.	72
6.4.1	Function Documentation	72
6.4.1.1	rtXmlWriteToFile	72
6.5	XML pull-parser decode functions.	73
6.5.1	Function Documentation	78
6.5.1.1	rtXmlEncAttrC14N	78

6.5.1.2	<code>rtXmlpCountListItems</code>	79
6.5.1.3	<code>rtXmlpCreateReader</code>	79
6.5.1.4	<code>rtXmlpDecAny</code>	79
6.5.1.5	<code>rtXmlpDecAny2</code>	80
6.5.1.6	<code>rtXmlpDecAnyAttrStr</code>	80
6.5.1.7	<code>rtXmlpDecAnyElem</code>	80
6.5.1.8	<code>rtXmlpDecBase64Str</code>	81
6.5.1.9	<code>rtXmlpDecBigInt</code>	81
6.5.1.10	<code>rtXmlpDecBitString</code>	82
6.5.1.11	<code>rtXmlpDecBool</code>	82
6.5.1.12	<code>rtXmlpDecDate</code>	82
6.5.1.13	<code>rtXmlpDecDateTime</code>	83
6.5.1.14	<code>rtXmlpDecDecimal</code>	83
6.5.1.15	<code>rtXmlpDecDouble</code>	83
6.5.1.16	<code>rtXmlpDecDoubleExt</code>	84
6.5.1.17	<code>rtXmlpDecDynBase64Str</code>	84
6.5.1.18	<code>rtXmlpDecDynBitString</code>	84
6.5.1.19	<code>rtXmlpDecDynHexStr</code>	85
6.5.1.20	<code>rtXmlpDecDynUnicodeStr</code>	85
6.5.1.21	<code>rtXmlpDecDynUTF8Str</code>	86
6.5.1.22	<code>rtXmlpDecGDay</code>	86
6.5.1.23	<code>rtXmlpDecGMonth</code>	86
6.5.1.24	<code>rtXmlpDecGMonthDay</code>	87
6.5.1.25	<code>rtXmlpDecGYear</code>	87
6.5.1.26	<code>rtXmlpDecGYearMonth</code>	87
6.5.1.27	<code>rtXmlpDecHexStr</code>	88
6.5.1.28	<code>rtXmlpDecInt</code>	88
6.5.1.29	<code>rtXmlpDecInt16</code>	88
6.5.1.30	<code>rtXmlpDecInt64</code>	89
6.5.1.31	<code>rtXmlpDecInt8</code>	89
6.5.1.32	<code>rtXmlpDecNamedBits</code>	89
6.5.1.33	<code>rtXmlpDecStrList</code>	90
6.5.1.34	<code>rtXmlpDecTime</code>	90
6.5.1.35	<code>rtXmlpDecUInt</code>	91
6.5.1.36	<code>rtXmlpDecUInt16</code>	91
6.5.1.37	<code>rtXmlpDecUInt64</code>	91
6.5.1.38	<code>rtXmlpDecUInt8</code>	92

6.5.1.39	rtXmlpDecUTF8Str	92
6.5.1.40	rtXmlpDecXmlStr	92
6.5.1.41	rtXmlpDecXmlStrList	93
6.5.1.42	rtXmlpDecXSIAAttr	93
6.5.1.43	rtXmlpDecXSIAAttrs	93
6.5.1.44	rtXmlpDecXSITypeAttr	94
6.5.1.45	rtXmlpForceDecodeAsGroup	94
6.5.1.46	rtXmlpGetAttributeCount	94
6.5.1.47	rtXmlpGetAttributeID	94
6.5.1.48	rtXmlpGetCurrentLevel	95
6.5.1.49	rtXmlpGetNextAllElemID	95
6.5.1.50	rtXmlpGetNextAllElemID16	96
6.5.1.51	rtXmlpGetNextAllElemID32	96
6.5.1.52	rtXmlpGetNextElem	97
6.5.1.53	rtXmlpGetNextElemID	97
6.5.1.54	rtXmlpGetNextSeqElemID	97
6.5.1.55	rtXmlpGetNextSeqElemID2	98
6.5.1.56	rtXmlpGetNextSeqElemIDExt	99
6.5.1.57	rtXmlpGetReader	99
6.5.1.58	rtXmlpGetXmInsAttrs	99
6.5.1.59	rtXmlpGetXSITypeAttr	100
6.5.1.60	rtXmlpGetXSITypeIndex	100
6.5.1.61	rtXmlpHasAttributes	100
6.5.1.62	rtXmlpHideAttributes	101
6.5.1.63	rtXmlpIsDecodeAsGroup	101
6.5.1.64	rtXmlpIsEmptyElement	101
6.5.1.65	rtXmlpIsLastEventDone	101
6.5.1.66	rtXmlpIsUTF8Encoding	102
6.5.1.67	rtXmlpListHasItem	102
6.5.1.68	rtXmlpLookupXSITypeIndex	102
6.5.1.69	rtXmlpMarkLastEventActive	103
6.5.1.70	rtXmlpMarkPos	103
6.5.1.71	rtXmlpMatchEndTag	103
6.5.1.72	rtXmlpMatchStartTag	103
6.5.1.73	rtXmlpNeedDecodeAttributes	104
6.5.1.74	rtXmlpReadBytes	104
6.5.1.75	rtXmlpResetMarkedPos	104

6.5.1.76	rtXmIpRewindToMarkedPos	105
6.5.1.77	rtXmIpSelectAttribute	105
6.5.1.78	rtXmIpSetListMode	105
6.5.1.79	rtXmIpSetMixedContentMode	105
6.5.1.80	rtXmIpSetNamespaceTable	106
6.5.1.81	rtXmIpSetWhiteSpaceMode	106
6.6	XML run-time error status codes	107
6.6.1	Detailed Description	108
6.6.2	Define Documentation	108
6.6.2.1	XML_E_BASE	108
6.6.2.2	XML_E_ELEMMISRQ	109
6.6.2.3	XML_E_ELEMSISRQ	109
6.6.2.4	XML_E_FLDABSENT	109
6.6.2.5	XML_E_NOMATCH	109
6.6.2.6	XML_E_NSURINOTFOU	109
6.6.2.7	XML_E_TAGMISMATCH	109
6.6.2.8	XML_OK_EOB	109
6.6.2.9	XML_OK_FRAG	110
<b>7</b>	<b>Data Structure Documentation</b>	<b>111</b>
7.1	OSIntegerFmt Struct Reference	111
7.1.1	Detailed Description	111
7.2	OSXMLCtxtInfo Struct Reference	112
7.2.1	Detailed Description	112
7.3	OSXMLDecodeBuffer Class Reference	113
7.3.1	Detailed Description	114
7.3.2	Constructor & Destructor Documentation	114
7.3.2.1	OSXMLDecodeBuffer	114
7.3.2.2	OSXMLDecodeBuffer	114
7.3.2.3	OSXMLDecodeBuffer	114
7.3.3	Member Function Documentation	115
7.3.3.1	decodeXML	115
7.3.3.2	init	115
7.3.3.3	isA	115
7.3.3.4	isWellFormed	115
7.3.3.5	parseElementName	116
7.3.3.6	parseElemQName	116

7.3.3.7	setMaxErrors	116
7.3.4	Field Documentation	116
7.3.4.1	mbOwnStream	116
7.4	OSXMLElemIDRec Struct Reference	117
7.4.1	Detailed Description	117
7.5	OSXMLEncodeBuffer Class Reference	118
7.5.1	Detailed Description	119
7.5.2	Constructor & Destructor Documentation	119
7.5.2.1	OSXMLEncodeBuffer	119
7.5.3	Member Function Documentation	119
7.5.3.1	addXMLHeader	119
7.5.3.2	addXMLText	119
7.5.3.3	getMsgLen	120
7.5.3.4	init	120
7.5.3.5	isA	120
7.5.3.6	setFragment	120
7.5.3.7	write	120
7.5.3.8	write	121
7.6	OSXMLEncodeStream Class Reference	122
7.6.1	Detailed Description	123
7.6.2	Constructor & Destructor Documentation	123
7.6.2.1	OSXMLEncodeStream	123
7.6.2.2	OSXMLEncodeStream	123
7.6.3	Member Function Documentation	124
7.6.3.1	encodeAttr	124
7.6.3.2	encodeText	124
7.6.3.3	endElement	124
7.6.3.4	getMsgPtr	125
7.6.3.5	getStream	125
7.6.3.6	init	125
7.6.3.7	isA	125
7.6.3.8	startDocument	125
7.6.3.9	startElement	126
7.6.3.10	termStartElement	126
7.6.4	Field Documentation	126
7.6.4.1	mbOwnStream	126
7.6.4.2	mpCtxt	126

7.6.4.3	mpStream	127
7.7	OSXMLFacets Struct Reference	128
7.7.1	Detailed Description	128
7.8	OSXMLGroupDesc Struct Reference	129
7.8.1	Detailed Description	129
7.9	OSXMLItemDescr Struct Reference	130
7.9.1	Detailed Description	130
7.10	OSXMLMessageBuffer Class Reference	131
7.10.1	Detailed Description	132
7.10.2	Constructor & Destructor Documentation	132
7.10.2.1	OSXMLMessageBuffer	132
7.10.3	Member Function Documentation	132
7.10.3.1	getIndent	132
7.10.3.2	getIndentChar	132
7.10.3.3	getWriteBOM	132
7.10.3.4	setAppInfo	133
7.10.3.5	setFormatting	133
7.10.3.6	setIndent	133
7.10.3.7	setIndentChar	133
7.10.3.8	setNamespace	133
7.10.3.9	setWriteBOM	134
7.11	OSXMLNameFragments Struct Reference	135
7.11.1	Detailed Description	135
7.12	OSXMLQName Struct Reference	136
7.12.1	Detailed Description	136
7.13	OSXMLSortedAttrOffset Struct Reference	137
7.13.1	Detailed Description	137
7.14	OSXMLStrFragment Struct Reference	138
7.14.1	Detailed Description	138
7.15	OSXSAnyType Struct Reference	139
7.15.1	Detailed Description	139
<b>8</b>	<b>File Documentation</b>	<b>140</b>
8.1	osrxml.h File Reference	140
8.1.1	Detailed Description	160
8.1.2	Define Documentation	160
8.1.2.1	IS_XMLNSATTR	160

8.1.2.2	IS_XSIATTR	161
8.1.2.3	OSXMLQNAMEEQUALS	161
8.1.2.4	OSXMLSETUTF8DECPTR	161
8.1.3	Typedef Documentation	161
8.1.3.1	OSXMLGroupDesc	161
8.1.4	Function Documentation	161
8.1.4.1	rtSaxGetAttrValue	161
8.1.4.2	rtSaxGetElemID	162
8.1.4.3	rtSaxGetElemID8	162
8.1.4.4	rtSaxHasXMLNSAttrs	162
8.1.4.5	rtSaxIsEmptyBuffer	163
8.1.4.6	rtSaxSortAttrs	163
8.1.4.7	rtSaxStrListMatch	163
8.1.4.8	rtSaxStrListParse	163
8.1.4.9	rtXmlCmpBase64Str	164
8.1.4.10	rtXmlCmpHexStr	164
8.1.4.11	rtXmlCreateFileInputSource	164
8.1.4.12	rtXmlInitContext	165
8.1.4.13	rtXmlInitContextUsingKey	165
8.1.4.14	rtXmlInitCtxtAppInfo	165
8.1.4.15	rtXmlMatchBase64Str	165
8.1.4.16	rtXmlMatchDate	166
8.1.4.17	rtXmlMatchDateTime	166
8.1.4.18	rtXmlMatchGDay	166
8.1.4.19	rtXmlMatchGMonth	167
8.1.4.20	rtXmlMatchGMonthDay	167
8.1.4.21	rtXmlMatchGYear	167
8.1.4.22	rtXmlMatchGYearMonth	167
8.1.4.23	rtXmlMatchHexStr	168
8.1.4.24	rtXmlMatchTime	168
8.1.4.25	rtXmlMemFreeAnyAttrs	168
8.1.4.26	rtXmlNewQName	168
8.1.4.27	rtXmlPrepareContext	169
8.1.4.28	rtXmlSetEncC14N	169
8.1.4.29	rtXmlSetEncDocHdr	169
8.1.4.30	rtXmlSetEncodingStr	169
8.1.4.31	rtXmlSetEncXSINamespace	170

8.1.4.32	rtXmlSetEncXSINilAttr	170
8.1.4.33	rtXmlSetFormatting	170
8.1.4.34	rtXmlSetIndent	171
8.1.4.35	rtXmlSetIndentChar	171
8.1.4.36	rtXmlSetNamespacesSet	171
8.1.4.37	rtXmlSetNoNSSchemaLocation	171
8.1.4.38	rtXmlSetNSPrefixLinks	172
8.1.4.39	rtXmlSetSchemaLocation	172
8.1.4.40	rtXmlSetSoapVersion	172
8.1.4.41	rtXmlSetWriteBOM	172
8.1.4.42	rtXmlSetXSITypeAttr	173
8.2	OSXMLDecodeBuffer.h File Reference	174
8.2.1	Detailed Description	174
8.3	OSXMLEncodeBuffer.h File Reference	175
8.3.1	Detailed Description	175
8.4	OSXMLEncodeStream.h File Reference	176
8.4.1	Detailed Description	176
8.5	OSXMLMessageBuffer.h File Reference	177
8.5.1	Detailed Description	177
8.6	rtXmlErrCodes.h File Reference	178
8.6.1	Detailed Description	179



# Chapter 1

## Main Page

### C XML Runtime Library Functions

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

The categories of functions provided are as follows:

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

# Chapter 2

## Module Index

### 2.1 Modules

Here is a list of all modules:

ASN.1-XML encode/decode functions. . . . .	6
XML decode functions. . . . .	16
XML encode functions. . . . .	32
XML utility functions. . . . .	72
XML pull-parser decode functions. . . . .	73
XML run-time error status codes. . . . .	107

# Chapter 3

## Data Structure Index

### 3.1 Class Hierarchy

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

OSIntegerFmt . . . . .	111
OSXMLCtxtInfo . . . . .	112
OSXMLElemIDRec . . . . .	117
OSXMLFacets . . . . .	128
OSXMLGroupDesc . . . . .	129
OSXMLItemDescr . . . . .	130
OSXMLMessageBuffer . . . . .	131
OSXMLDecodeBuffer . . . . .	113
OSXMLEncodeBuffer . . . . .	118
OSXMLEncodeStream . . . . .	122
OSXMLNameFragments . . . . .	135
OSXMLQName . . . . .	136
OSXMLSortedAttrOffset . . . . .	137
OSXMLStrFragment . . . . .	138
OSXSAnyType . . . . .	139

# Chapter 4

## Data Structure Index

### 4.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">OSIntegerFmt</a> . . . . .	111
<a href="#">OSXMLCtxtInfo</a> . . . . .	112
<a href="#">OSXMLDecodeBuffer</a> (Derived from the <a href="#">OSXMLMessageBuffer</a> base class ) . . . . .	113
<a href="#">OSXMLElemIDRec</a> . . . . .	117
<a href="#">OSXMLEncodeBuffer</a> (Derived from the <a href="#">OSXMLMessageBuffer</a> base class ) . . . . .	118
<a href="#">OSXMLEncodeStream</a> (Derived from the <a href="#">OSXMLMessageBuffer</a> base class ) . . . . .	122
<a href="#">OSXMLFacets</a> . . . . .	128
<a href="#">OSXMLGroupDesc</a> ( <a href="#">OSXMLGroupDesc</a> describes how entries in an <a href="#">OSXMLElemIDRec</a> array make up a group ) . . . . .	129
<a href="#">OSXMLItemDescr</a> . . . . .	130
<a href="#">OSXMLMessageBuffer</a> (The XML message buffer class is derived from the <a href="#">OSMessageBuffer</a> base class ) . . . . .	131
<a href="#">OSXMLNameFragments</a> . . . . .	135
<a href="#">OSXMLQName</a> . . . . .	136
<a href="#">OSXMLSortedAttrOffset</a> . . . . .	137
<a href="#">OSXMLStrFragment</a> . . . . .	138
<a href="#">OSXSDAnyType</a> . . . . .	139

# Chapter 5

## File Index

### 5.1 File List

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

<b>asn1xml.h</b> . . . . .	<b>??</b>
<a href="#">osrxml.h</a> (XML low-level C encode/decode functions) . . . . .	140
<a href="#">OSXMLDecodeBuffer.h</a> (XML decode buffer or stream class definition) . . . . .	174
<a href="#">OSXMLEncodeBuffer.h</a> (XML encode message buffer class definition) . . . . .	175
<a href="#">OSXMLEncodeStream.h</a> (XML encode stream class definition) . . . . .	176
<a href="#">OSXMLMessageBuffer.h</a> (XML encode/decode buffer and stream base class) . . . . .	177
<a href="#">rtXmlErrCodes.h</a> (List of numeric status codes that can be returned by ASN1C run-time functions and generated code) . . . . .	178

# Chapter 6

## Module Documentation

### 6.1 ASN.1-XML encode/decode functions.

#### Functions

- EXTERNXML int [rtAsn1XmlpDecGenTime](#) (OSCTXT \*pctxt, const char \*\*outdata)  
*This function decodes the contents of an ASN.1 Generalized type.*
- EXTERNXML int [rtAsn1XmlpDecReal](#) (OSCTXT \*pctxt, OSREAL \*pvalue)  
*This function decodes the contents of an ASN.1 REAL type.*
- EXTERNXML int [rtAsn1XmlpDecObjId](#) (OSCTXT \*pctxt, ASN1OBJID \*pvalue)  
*This function decodes the contents of an ASN.1 OBJECT IDENTIFIER type.*
- EXTERNXML int [rtAsn1XmlpDecOpenType](#) (OSCTXT \*pctxt, ASN1OpenType \*pOpenType)  
*This function decodes a variable of the ASN.1 open type.*
- EXTERNXML int [rtAsn1XmlpDecUnivStr](#) (OSCTXT \*pctxt, const OS32BITCHAR \*\*ppdata, OSSIZE \*pnchars)  
*This function decodes the contents of an ASN.1 UNIVERSAL string type.*
- EXTERNXML int [rtAsn1XmlpDecUTCTime](#) (OSCTXT \*pctxt, const char \*\*outdata)  
*This function decodes the contents of an ASN.1 UTCTime type.*
- EXTERNXML int [rtAsn1XmlEncGenTime](#) (OSCTXT \*pctxt, const char \*value, const OSUTF8CHAR \*elemName, const OSUTF8CHAR \*nsPrefix)  
*This function encodes a variable of the ASN.1 GeneralizedTime type.*
- 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.*
- 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.*

- EXTERNXML int `rtAsn1XmlEncReal` (OSCTXT \*pctx, OSREAL value, const OSUTF8CHAR \*elemName, const OSUTF8CHAR \*nsPrefix)  
*This function encodes a variable of the ASN.1 REAL type.*
- EXTERNXML int `rtAsn1XmlEncRelOID` (OSCTXT \*pctx, const ASN1OBJID \*pvalue, const OSUTF8CHAR \*elemName, const OSUTF8CHAR \*nsPrefix)  
*This function encodes a variable of the ASN.1 RELATIVE-OID type.*
- EXTERNXML int `rtAsn1XmlEncOpenType` (OSCTXT \*pctx, const OSOCTET \*data, OSSIZE noctx, const OSUTF8CHAR \*elemName, const OSUTF8CHAR \*nsPrefix)  
*This function encodes a variable of the ASN.1 open type.*
- EXTERNXML int `rtAsn1XmlEncOpenTypeExt` (OSCTXT \*pctx, OSRTDList \*pElemList)  
*This function encodes an ASN.1 open type extension.*
- EXTERNXML int `rtAsn1XmlEncUnivStr` (OSCTXT \*pctx, const OS32BITCHAR \*value, OSSIZE nchars, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the ASN.1 UNIVERSAL string type.*
- EXTERNXML int `rtAsn1XmlFmtAttrStr` (OSCTXT \*pctx, const OSUTF8CHAR \*name, const OSUTF8CHAR \*value, OSUTF8CHAR \*\*pAttrStr)  
*This function formats a name-value XML pair into a name="value" attribute string.*
- EXTERNXML int `rtAsn1XmlParseAttrStr` (OSCTXT \*pctx, const OSUTF8CHAR \*pAttrStr, OSUTF8NVP \*pNVPair)  
*This function parses an XML name-value pair from an attribute string.*
- 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.*
- EXTERNXML int `rtAsn1XmlpDecDynBitStr` (OSCTXT \*pctx, ASN1DynBitStr \*pvalue)  
*This function decodes a bit string value.*
- EXTERNXML int `rtAsn1XmlpDecDynBitStr64` (OSCTXT \*pctx, ASN1DynBitStr64 \*pvalue)  
*This function decodes a bit string value.*
- EXTERNXML int `rtXmlpDecListOfASN1DynBitStr` (OSCTXT \*pctx, 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.*
- EXTERNXML int `rtAsn1XmlpDecRelOID` (OSCTXT \*pctx, ASN1OBJID \*pvalue)  
*This function decodes the contents of an ASN.1 RELATIVE-OID type.*

## 6.1.1 Function Documentation

### 6.1.1.1 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

- 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* The name of the new attribute.
- value* The value of the new attribute.
- plist* The attribute list.

### 6.1.1.2 EXTERNXML int rtAsn1XmlEncGenTime (OSCTXT \* *pctxt*, 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.

### 6.1.1.3 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.



#### 6.1.1.4 EXTERNXML int rtAsn1XmlEncOpenType (OSCTXT \* *pctxt*, const OSOCTET \* *data*, OSSIZE *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.

#### 6.1.1.5 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.

#### 6.1.1.6 EXTERNXML int rtAsn1XmlEncReal (OSCTXT \* *pctxt*, OSREAL *value*, const OSUTF8CHAR \* *elemName*, const OSUTF8CHAR \* *nsPrefix*)

This function encodes a variable of the ASN.1 REAL 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 OSREAL value.

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

*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.

#### 6.1.1.7 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.

#### 6.1.1.8 EXTERNXML int rtAsn1XmlEncUnivStr (OSCTXT \* *pctxt*, const OS32BITCHAR \* *value*, OSSIZE *nchars*, const OSUTF8CHAR \* *elemName*, OSXMLNamespace \* *pNS*)

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.

*pNS* XML namespace information (prefix and URI).

### Returns

Completion status of operation:

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

**6.1.1.9 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.

**6.1.1.10 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.

**6.1.1.11 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.

#### 6.1.1.12 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 value.

### Returns

Completion status of operation:

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

#### 6.1.1.13 EXTERNXML int rtAsn1XmlpDecDynBitStr64 (OSCTXT \* *pctxt*, ASN1DynBitStr64 \* *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.

This version of the function can hold strings with lengths up to 64 bits in size.

### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to a variable to receive the decoded value.

### Returns

Completion status of operation:

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

#### 6.1.1.14 EXTERNXML int rtAsn1XmlpDecGenTime (OSCTXT \* *pctxt*, const char \*\* *outdata*)

This function decodes the contents of an ASN.1 Generalized type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser. The encoding is expected to be in xsd:dateTime format (not in the XER format).

##### 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.

#### 6.1.1.15 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.

#### 6.1.1.16 EXTERNXML int rtAsn1XmlpDecOpenType (OSCTXT \* *pctxt*, ASN1OpenType \* *pOpenType*)

This function decodes a variable of the ASN.1 open type.

If the XML open type is in the form of an xmlhstring (see X.681, clause 14.6), it is converted to binary form and stored in the open type structure. Otherwise, the textual representation is saved as-is.

##### 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.

*pOpenType* A pointer to an open type structure to receive the decoded data.

##### Returns

Completion status of operation:

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

#### 6.1.1.17 EXTERNXML int rtAsn1XmlpDecReal (OSCTXT \* *pctxt*, OSREAL \* *pvalue*)

This function decodes the contents of an ASN.1 REAL type.

Input is expected to be returned by an XML pull parser.

This method recognizes the basic-XER encoding for a REAL, which consists of the elements <PLUS-INFINITY/>, <MINUS-INFINITY/>, and <NOT-A-NUMBER/>, and realnumber with an optional negative sign.

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to OSREAL to value to receive decoded result.

##### Returns

Completion status of operation:

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

#### 6.1.1.18 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 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.

#### 6.1.1.19 EXTERNXML int rtAsn1XmlpDecUnivStr (OSCTXT \* *pctxt*, const OS32BITCHAR \*\* *ppdata*, OSSIZE \* *pchars*)

This function decodes the contents of an ASN.1 UNIVERSAL string type.

Input is expected to be a string of OSUTF8CHAR 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.

#### 6.1.1.20 EXTERNXML int rtAsn1XmlpDecUTCTime (OSCTXT \* *pctxt*, const char \*\* *outdata*)

This function decodes the contents of an ASN.1 UTCTime type.

Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser. The encoding is expected to be in xsd:dateTime format (not in the XER format).

##### 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.

#### 6.1.1.21 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.

## 6.2 XML decode functions.

### Functions

- EXTERNXML int [rtXmlDecBase64Binary](#) (OSRTMEMBUF \*pMemBuf, const OSUTF8CHAR \*inpdata, OSIZE length)  
*This function decodes the contents of a Base64-encoded binary data type into a memory buffer.*
- EXTERNXML int [rtXmlDecBase64Str](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, OSINT32 bufsize)  
*This function decodes a contents of a Base64-encode binary string into a static memory structure.*
- EXTERNXML int [rtXmlDecBase64StrValue](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, OSIZE bufSize, OSSIZE srcDataLen)  
*This function decodes a contents of a Base64-encode binary string into the specified octet array.*
- EXTERNXML int [rtXmlDecBigInt](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*ppvalue)  
*This function will decode a variable of the XSD integer type.*
- EXTERNXML int [rtXmlDecBool](#) (OSCTXT \*pctxt, OSBOOL \*pvalue)  
*This function decodes a variable of the boolean type.*
- EXTERNXML int [rtXmlDecDate](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'date' type.*
- EXTERNXML int [rtXmlDecTime](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'time' type.*
- EXTERNXML int [rtXmlDecDateTime](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'dateTime' type.*
- EXTERNXML int [rtXmlDecDecimal](#) (OSCTXT \*pctxt, OSREAL \*pvalue)  
*This function decodes the contents of a decimal data type.*
- EXTERNXML int [rtXmlDecDouble](#) (OSCTXT \*pctxt, OSREAL \*pvalue)  
*This function decodes the contents of a float or double data type.*
- EXTERNXML int [rtXmlDecDynBase64Str](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)  
*This function decodes a contents of a Base64-encode binary string.*
- EXTERNXML int [rtXmlDecDynHexStr](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)  
*This function decodes a contents of a hexBinary string.*
- EXTERNXML int [rtXmlDecEmptyElement](#) (OSCTXT \*pctxt)  
*This function is used to enforce a requirement that an element be empty.*
- EXTERNXML int [rtXmlDecUTF8Str](#) (OSCTXT \*pctxt, OSUTF8CHAR \*outdata, OSSIZE max\_len)  
*This function decodes the contents of a UTF-8 string data type.*
- EXTERNXML int [rtXmlDecDynUTF8Str](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*outdata)



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

- EXTERNXML int **rtXmlDecHexBinary** (OSRTMEMBUF \*pMemBuf, const OSUTF8CHAR \*inpdata, OS-SIZE length)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- EXTERNXML int [rtXmlDecNSAttr](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*attrName, const OSUTF8CHAR \*attrValue, OSRTDList \*pNSAttrs, const OSUTF8CHAR \*nsTable[ ], OSUINT32 nsTableRowCount)  
*This function decodes an XML namespace attribute (xmlns).*
- EXTERNXML const OSUTF8CHAR \* [rtXmlDecQName](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*qname, const OSUTF8CHAR \*\*prefix)  
*This function decodes an XML qualified name string (QName) type.*
- EXTERNXML int [rtXmlDecXSIAttr](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*attrName, const OSUTF8CHAR \*attrValue)  
*This function decodes XML schema instance (XSI) attribute.*
- EXTERNXML int [rtXmlDecXSIAttrs](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*const \*attrs, const char \*typeName)  
*This function decodes XML schema instance (XSI) attributes.*
- EXTERNXML int [rtXmlDecXmlStr](#) (OSCTXT \*pctxt, OSXMLSTRING \*outdata)  
*This function decodes the contents of an XML string data type.*
- EXTERNXML int [rtXmlParseElementName](#) (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppName)  
*This function parses the initial tag from an XML message.*
- EXTERNXML int [rtXmlParseElemQName](#) (OSCTXT \*pctxt, OSXMLQName \*pQName)  
*This function parses the initial tag from an XML message.*

## 6.2.1 Function Documentation

### 6.2.1.1 EXTERNXML int [rtXmlDecBase64Binary](#) (OSRTMEMBUF \*pMemBuf, const OSUTF8CHAR \*inpdata, OSSIZE length)

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

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

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

#### 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.

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

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

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

#### 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.

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

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

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

#### 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.

#### 6.2.1.4 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.

#### 6.2.1.5 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.

#### 6.2.1.6 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.

### 6.2.1.7 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.

### 6.2.1.8 EXTERNXML int rtXmlDecDecimal (OSCTXT \* *pctxt*, OSREAL \* *pvalue*)

This function decodes the contents of a decimal data type.

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

#### Parameters

*pctxt* Pointer to context block structure.

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

#### Returns

Completion status of operation:

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

### 6.2.1.9 EXTERNXML int rtXmlDecDouble (OSCTXT \* *pctxt*, OSREAL \* *pvalue*)

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

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

#### Parameters

*pctxt* Pointer to context block structure.

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

#### Returns

Completion status of operation:

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

#### 6.2.1.10 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.

#### 6.2.1.11 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.

#### 6.2.1.12 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.

### 6.2.1.13 EXTERNXML int rtXmlDecEmptyElement (OSCTXT \* *pctxt*)

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

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

#### Parameters

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

#### Returns

Completion status of operation:

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

### 6.2.1.14 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.

### 6.2.1.15 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.

#### 6.2.1.16 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.

#### 6.2.1.17 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.

#### 6.2.1.18 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.



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

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

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

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

##### 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.

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

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

##### 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.

#### 6.2.1.21 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.

#### 6.2.1.22 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.

#### 6.2.1.23 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.

#### 6.2.1.24 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.

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

This function decodes an XML namespace attribute (xmlns).

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

## Parameters

*pctxt* Pointer to context structure.

*attrName* Name of the XML namespace attribute. This is assumed to contain either 'xmlns' (a default namespace declaration) or 'xmlns:prefix' where prefix is a namespace prefix.

*attrValue* XML namespace attribute value (a URI).

*pNSAttrs* List to receive parsed namespace values.

*nsTable* Namespace URI's parsed from schema.

*nsTableRowCount* Number of rows (URI's) in namespace table.

## Returns

Zero if success or negative error code.

**6.2.1.26 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.

### 6.2.1.27 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.

### 6.2.1.28 EXTERNXML int rtXmlDecUInt (OSCTXT \* *pctxt*, OSUINT32 \* *pvalue*)

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

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

## 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.

### 6.2.1.29 EXTERNXML int rtXmlDecUInt16 (OSCTXT \* *pctxt*, OSUINT16 \* *pvalue*)

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

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

## 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.

#### 6.2.1.30 EXTERNXML int rtXmlDecUInt64 (OSCTXT \* *pctxt*, OSUINT64 \* *pvalue*)

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

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

#### 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.

#### 6.2.1.31 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.

#### 6.2.1.32 EXTERNXML int rtXmlDecUTF8Str (OSCTXT \* *pctxt*, OSUTF8CHAR \* *outdata*, OSSIZE *max\_len*)

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

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

## Parameters

*pctxt* Pointer to context block structure.

*outdata* Pointer to a block of memory to receive decoded UTF8 string.

*max\_len* Size of memory block.

## Returns

Completion status of operation:

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

### 6.2.1.33 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.

### 6.2.1.34 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.

### 6.2.1.35 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.

### 6.2.1.36 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.

### 6.2.1.37 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.

## 6.3 XML encode functions.

### Defines

- #define **rtxPrintNSAttrs**(name, data) rtxPrintNSAttrs(name,&data)
- #define **rtXmlFinalizeMemBuf**(pMemBuf)
- #define **rtXmlGetEncBufPtr**(pctx) (pctx)->buffer.data  
*This macro returns the start address of the encoded XML message.*
- #define **rtXmlGetEncBufLen**(pctx) (pctx)->buffer.byteIndex  
*This macro returns the length of the encoded XML message.*

### Functions

- EXTERNXML int **rtXmlEncAny** (OSCTXT \*pctx, OSXMLSTRING \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD any type.*
- EXTERNXML int **rtXmlEncAnyStr** (OSCTXT \*pctx, const OSUTF8CHAR \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)
- EXTERNXML int **rtXmlEncAnyTypeValue** (OSCTXT \*pctx, const OSUTF8CHAR \*pvalue)  
*This function encodes a variable of the XSD anyType type.*
- EXTERNXML int **rtXmlEncAnyAttr** (OSCTXT \*pctx, OSRTDList \*pAnyAttrList)  
*This function encodes a list of OSAnyAttr attributes in which the name and value are given as a UTF-8 string.*
- EXTERNXML int **rtXmlEncBase64Binary** (OSCTXT \*pctx, OSUINT32 noctx, const OSOCTET \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD base64Binary type.*
- EXTERNXML int **rtXmlEncBase64BinaryAttr** (OSCTXT \*pctx, OSUINT32 noctx, const OSOCTET \*value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)  
*This function encodes a variable of the XSD base64Binary type as an attribute.*
- EXTERNXML int **rtXmlEncBase64StrValue** (OSCTXT \*pctx, OSUINT32 noctx, const OSOCTET \*value)  
*This function encodes a variable of the XSD base64Binary type.*
- EXTERNXML int **rtXmlEncBigInt** (OSCTXT \*pctx, const OSUTF8CHAR \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD integer type.*
- EXTERNXML int **rtXmlEncBigIntAttr** (OSCTXT \*pctx, const OSUTF8CHAR \*value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)  
*This function encodes an XSD integer attribute value.*
- EXTERNXML int **rtXmlEncBigIntValue** (OSCTXT \*pctx, const OSUTF8CHAR \*value)  
*This function encodes an XSD integer attribute value.*



- EXTERNXML int [rtXmlEncBitString](#) (OSCTXT \*pctxt, OSUINT32 nbits, const OSOCTET \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the ASN.1 BIT STRING type.*
- 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.*
- EXTERNXML int [rtXmlEncBool](#) (OSCTXT \*pctxt, OSBOOL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD boolean type.*
- EXTERNXML int [rtXmlEncBoolValue](#) (OSCTXT \*pctxt, OSBOOL value)  
*This function encodes a variable of the XSD boolean type.*
- EXTERNXML int [rtXmlEncBoolAttr](#) (OSCTXT \*pctxt, OSBOOL value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)  
*This function encodes an XSD boolean attribute value.*
- EXTERNXML int [rtXmlEncComment](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*comment)  
*This function encodes an XML comment.*
- EXTERNXML int [rtXmlEncDate](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD 'date' type as a string.*
- EXTERNXML int [rtXmlEncDateValue](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)  
*This function encodes a variable of the XSD 'date' type as a string.*
- EXTERNXML int [rtXmlEncTime](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD 'time' type as a string.*
- EXTERNXML int [rtXmlEncTimeValue](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)  
*This function encodes a variable of the XSD 'time' type as a string.*
- EXTERNXML int [rtXmlEncDateTime](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a numeric date/time value into an XML string representation.*
- EXTERNXML int [rtXmlEncDateTimeValue](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric date/time value into an XML string representation.*
- EXTERNXML int [rtXmlEncDecimal](#) (OSCTXT \*pctxt, OSREAL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSDecimalFmt \*pFmtSpec)  
*This function encodes a variable of the XSD decimal type.*
- EXTERNXML int [rtXmlEncDecimalAttr](#) (OSCTXT \*pctxt, OSREAL value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen, const OSDecimalFmt \*pFmtSpec)  
*This function encodes a variable of the XSD decimal type as an attribute.*

- EXTERNXML int [rtXmlEncDecimalValue](#) (OSCTXT \*pctx, OSREAL value, const OSDecimalFmt \*pFmtSpec, char \*pDestBuf, OSSIZE destBufSize)  
*This function encodes a value of the XSD decimal type.*
- EXTERNXML int [rtXmlEncDouble](#) (OSCTXT \*pctx, OSREAL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSDoubleFmt \*pFmtSpec)  
*This function encodes a variable of the XSD double type.*
- EXTERNXML int [rtXmlEncDoubleAttr](#) (OSCTXT \*pctx, OSREAL value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen, const OSDoubleFmt \*pFmtSpec)  
*This function encodes a variable of the XSD double type as an attribute.*
- EXTERNXML int [rtXmlEncDoubleNormalValue](#) (OSCTXT \*pctx, OSREAL value, const OSDoubleFmt \*pFmtSpec, int defaultPrecision)  
*This function encodes a normal (not +/-INF or NaN) value of the XSD double or float type.*
- EXTERNXML int [rtXmlEncDoubleValue](#) (OSCTXT \*pctx, OSREAL value, const OSDoubleFmt \*pFmtSpec, int defaultPrecision)  
*This function encodes a value of the XSD double or float type.*
- EXTERNXML int [rtXmlEncEmptyElement](#) (OSCTXT \*pctx, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs, OSBOOL terminate)  
*This function encodes an empty element tag value (<elemName/>).*
- EXTERNXML int [rtXmlEncEndDocument](#) (OSCTXT \*pctx)  
*This function adds trailer information and a null terminator at the end of the XML document being encoded.*
- EXTERNXML int [rtXmlEncEndElement](#) (OSCTXT \*pctx, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes an end element tag value (</elemName>).*
- EXTERNXML int [rtXmlEncEndSoapEnv](#) (OSCTXT \*pctx)  
*This function encodes a SOAP envelope end element tag (<SOAP-ENV:Envelope/>).*
- EXTERNXML int [rtXmlEncEndSoapElements](#) (OSCTXT \*pctx, OSXMLSOAPMsgType msgtype)  
*This function encodes SOAP end element tags.*
- EXTERNXML int [rtXmlEncFloat](#) (OSCTXT \*pctx, OSREAL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSDoubleFmt \*pFmtSpec)  
*This function encodes a variable of the XSD float type.*
- EXTERNXML int [rtXmlEncFloatAttr](#) (OSCTXT \*pctx, OSREAL value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen, const OSDoubleFmt \*pFmtSpec)  
*This function encodes a variable of the XSD float type as an attribute.*
- EXTERNXML int [rtXmlEncGYear](#) (OSCTXT \*pctx, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a numeric gYear element into an XML string representation.*
- EXTERNXML int [rtXmlEncGYearMonth](#) (OSCTXT \*pctx, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

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

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

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

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

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

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

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

- EXTERNXML int [rtXmlEncGYearValue](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)

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

- EXTERNXML int [rtXmlEncGYearMonthValue](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)

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

- EXTERNXML int [rtXmlEncGMonthValue](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)

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

- EXTERNXML int [rtXmlEncGMonthDayValue](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)

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

- EXTERNXML int [rtXmlEncGDayValue](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)

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

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

*This function encodes a variable of the XSD hexBinary type.*

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

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

- EXTERNXML int [rtXmlEncHexStrValue](#) (OSCTXT \*pctxt, OSSIZE nocts, const OSOCTET \*data)

*This function encodes a variable of the XSD hexBinary type.*

- EXTERNXML int [rtXmlEncIndent](#) (OSCTXT \*pctxt)

*This function adds indentation whitespace to the output stream.*

- EXTERNXML int [rtXmlEncInt](#) (OSCTXT \*pctxt, OSINT32 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

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

- EXTERNXML int [rtXmlEncIntValue](#) (OSCTXT \*pctxt, OSINT32 value)

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

- EXTERNXML int [rtXmlEncIntAttr](#) (OSCTXT \*pctxt, OSINT32 value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)  
*This function encodes a variable of the XSD integer type as an attribute (name="value").*
- EXTERNXML int [rtXmlEncIntPattern](#) (OSCTXT \*pctxt, OSINT32 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSUTF8CHAR \*pattern)  
*This function encodes a variable of the XSD integer type using a pattern to specify the format of the integer value.*
- EXTERNXML int [rtXmlEncIntPatternValue](#) (OSCTXT \*pctxt, OSINT32 value, const OSUTF8CHAR \*pattern)
- EXTERNXML int [rtXmlEncUIntPattern](#) (OSCTXT \*pctxt, OSUINT32 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSUTF8CHAR \*pattern)
- EXTERNXML int [rtXmlEncUIntPatternValue](#) (OSCTXT \*pctxt, OSUINT32 value, const OSUTF8CHAR \*pattern)
- EXTERNXML int [rtXmlEncInt64](#) (OSCTXT \*pctxt, OSINT64 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD integer type.*
- EXTERNXML int [rtXmlEncInt64Pattern](#) (OSCTXT \*pctxt, OSINT64 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSUTF8CHAR \*pattern)
- EXTERNXML int [rtXmlEncInt64Value](#) (OSCTXT \*pctxt, OSINT64 value)  
*This function encodes a variable of the XSD integer type.*
- EXTERNXML int [rtXmlEncInt64PatternValue](#) (OSCTXT \*pctxt, OSINT64 value, const OSUTF8CHAR \*pattern)
- EXTERNXML int [rtXmlEncInt64Attr](#) (OSCTXT \*pctxt, OSINT64 value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)  
*This function encodes a variable of the XSD integer type as an attribute (name="value").*
- EXTERNXML int [rtXmlEncNamedBits](#) (OSCTXT \*pctxt, const OSBitMapItem \*pBitMap, OSUINT32 nbits, const OSOCTET \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the ASN.1 BIT STRING type.*
- EXTERNXML int [rtXmlEncNamedBitsValue](#) (OSCTXT \*pctxt, const OSBitMapItem \*pBitMap, OSUINT32 nbits, const OSOCTET \*pvalue)
- EXTERNXML int [rtXmlEncNSAttrs](#) (OSCTXT \*pctxt, OSRTDList \*pNSAttrs)  
*This function encodes namespace declaration attributes at the beginning of an XML document.*
- EXTERNXML int [rtXmlPrintNSAttrs](#) (const char \*name, const OSRTDList \*data)  
*This function prints a list of namespace attributes.*
- EXTERNXML int [rtXmlEncReal10](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the ASN.1 REAL base 10 type.*
- EXTERNXML int [rtXmlEncSoapArrayTypeAttr](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*name, const OSUTF8CHAR \*value, OSSIZE itemCount)  
*This function encodes the special SOAP encoding attrType attribute which specifies the number and type of elements in a SOAP array.*
- EXTERNXML int [rtXmlEncSoapArrayTypeAttr2](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*name, OSSIZE nameLen, const OSUTF8CHAR \*value, OSSIZE valueLen, OSSIZE itemCount)

- EXTERNXML int [rtXmlEncStartDocument](#) (OSCTXT \*pctxt)  
*This function encodes the XML header text at the beginning of an XML document.*
- EXTERNXML int [rtXmlEncBOM](#) (OSCTXT \*pctxt)  
*This function encodes the Unicode byte order mark header at the start of the document.*
- EXTERNXML int [rtXmlEncStartElement](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*elemName, OSXML-  
Namespace \*pNS, OSRTDList \*pNSAttrs, OSBOOL terminate)  
*This function encodes a start element tag value (<elemName>).*
- EXTERNXML int [rtXmlEncStartSoapEnv](#) (OSCTXT \*pctxt, OSRTDList \*pNSAttrs)  
*This function encodes a SOAP envelope start element tag.*
- EXTERNXML int [rtXmlEncStartSoapElems](#) (OSCTXT \*pctxt, OSXMLSOAPMsgType msgtype)  
*This function encodes a SOAP envelope start element tag and an optional SOAP body or fault tag.*
- EXTERNXML int [rtXmlEncString](#) (OSCTXT \*pctxt, OSXMLSTRING \*pxmlstr, const OSUTF8CHAR  
\*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD string type.*
- EXTERNXML int [rtXmlEncStringValue](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*value)  
*This function encodes a variable of the XSD string type.*
- EXTERNXML int [rtXmlEncStringValue2](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*value, OSSIZE value-  
Len)  
*This function encodes a variable of the XSD string type.*
- EXTERNXML int [rtXmlEncTermStartElement](#) (OSCTXT \*pctxt)  
*This function terminates a currently open XML start element by adding either a '>' or '/>' (if empty) terminator.*
- EXTERNXML int [rtXmlEncUnicodeStr](#) (OSCTXT \*pctxt, const OSUNICHAR \*value, OSSIZE nchars, const  
OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a Unicode string value.*
- EXTERNXML int [rtXmlEncUTF8Attr](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*name, const OSUTF8CHAR  
\*value)  
*This function encodes an attribute in which the name and value are given as a null-terminated UTF-8 strings.*
- EXTERNXML int [rtXmlEncUTF8Attr2](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*name, OSSIZE nameLen,  
const OSUTF8CHAR \*value, OSSIZE valueLen)  
*This function encodes an attribute in which the name and value are given as a UTF-8 strings with lengths.*
- EXTERNXML int [rtXmlEncUTF8Str](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*value, const OSUTF8CHAR  
\*elemName, OSXMLNamespace \*pNS)  
*This function encodes a UTF-8 string value.*
- EXTERNXML int [rtXmlEncUInt](#) (OSCTXT \*pctxt, OSUINT32 value, const OSUTF8CHAR \*elemName, OS-  
XMLNamespace \*pNS)  
*This function encodes a variable of the XSD unsigned integer type.*
- EXTERNXML int [rtXmlEncUIntValue](#) (OSCTXT \*pctxt, OSUINT32 value)

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

- EXTERNXML int [rtXmlEncUIntAttr](#) (OSCTXT \*pctx, OSUINT32 value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)

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

- EXTERNXML int [rtXmlEncUInt64](#) (OSCTXT \*pctx, OSUINT64 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

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

- EXTERNXML int [rtXmlEncUInt64Pattern](#) (OSCTXT \*pctx, OSUINT64 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSUTF8CHAR \*pattern)
- EXTERNXML int [rtXmlEncUInt64Value](#) (OSCTXT \*pctx, OSUINT64 value)

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

- EXTERNXML int [rtXmlEncUInt64PatternValue](#) (OSCTXT \*pctx, OSUINT64 value, const OSUTF8CHAR \*pattern)
- EXTERNXML int [rtXmlEncUInt64Attr](#) (OSCTXT \*pctx, OSUINT64 value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)

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

- EXTERNXML int [rtXmlEncXSIAttrs](#) (OSCTXT \*pctx, OSBOOL needXSI)

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

- EXTERNXML int [rtXmlEncXSITypeAttr](#) (OSCTXT \*pctx, const OSUTF8CHAR \*value)

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

- EXTERNXML int [rtXmlEncXSITypeAttr2](#) (OSCTXT \*pctx, const OSUTF8CHAR \*typeNsUri, const OSUTF8CHAR \*typeName)

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

- EXTERNXML int [rtXmlEncXSINilAttr](#) (OSCTXT \*pctx)

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

- EXTERNXML int [rtXmlFreeInputSource](#) (OSCTXT \*pctx)

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

- EXTERNXML OSBOOL [rtXmlStrCmpAsc](#) (const OSUTF8CHAR \*text1, const char \*text2)
- EXTERNXML OSBOOL [rtXmlStrnCmpAsc](#) (const OSUTF8CHAR \*text1, const char \*text2, OSSIZE len)
- EXTERNXML int [rtXmlSetEncBufPtr](#) (OSCTXT \*pctx, OSOCTET \*bufaddr, OSSIZE bufsiz)

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

- EXTERNXML int [rtXmlGetIndent](#) (OSCTXT \*pctx)

*This function returns current XML output indent value.*

- EXTERNXML OSBOOL [rtXmlGetWriteBOM](#) (OSCTXT \*pctx)

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

- EXTERNXML int [rtXmlGetIndentChar](#) (OSCTXT \*pctx)

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

## 6.3.1 Define Documentation

### 6.3.1.1 #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)
```

Definition at line 2443 of file osrtxml.h.

### 6.3.1.2 #define rtXmlGetEncBufLen(pctxt) (pctxt)->buffer.byteIndex

This macro returns the length of the encoded XML message.

#### Parameters

*pctxt* Pointer to a context structure.

Definition at line 2495 of file osrtxml.h.

### 6.3.1.3 #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.

Definition at line 2488 of file osrtxml.h.

## 6.3.2 Function Documentation

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

This function encodes a variable of the XSD any type.

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

#### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Value to be encoded. This is a string containing the fully-encoded XML text to be copied to the output stream.

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

*pNS* Pointer to namespace structure.

### Returns

Completion status of operation:

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

#### 6.3.2.2 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.

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

This function encodes a variable of the XSD anyType type.

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

### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Value to be encoded.

### Returns

Completion status of operation:

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

#### 6.3.2.4 EXTERNXML int rtXmlEncBase64Binary (OSCTXT \* *pctxt*, OSUINT32 *nocts*, const OSOCTET \* *value*, const OSUTF8CHAR \* *elemName*, OSXMLNamespace \* *pNS*)

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.

*pNS* Pointer to namespace structure.

### Returns

Completion status of operation:

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

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

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

### 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.

#### 6.3.2.6 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.

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

This function encodes a variable of the XSD integer type.

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

Items of this type are stored in character string constant variables. They can be represented as decimal strings (with no prefixes), as hexadecimal strings starting with a "0x" prefix, as octal strings starting with a "0o" prefix or as binary strings starting with a "0b" prefix. Other 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.

*pNS* Pointer to namespace structure.

#### Returns

Completion status of operation:

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

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

This function encodes an XSD integer attribute value.

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

#### 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.

### 6.3.2.9 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.

### 6.3.2.10 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.

### 6.3.2.11 EXTERNXML int rtXmlEncBitString (OSCTXT \* *pctxt*, OSUINT32 *nbits*, const OSOCTET \* *value*, const OSUTF8CHAR \* *elemName*, OSXMLNamespace \* *pNS*)

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

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

## See also

[rtXmlEncNamedBits](#)).

## Parameters

*pctxt* Pointer to context block structure.

*nbits* Number of bits in the bit string.

*value* Value to be encoded.

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

*pNS* Pointer to namespace structure.

## Returns

Completion status of operation:

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

### 6.3.2.12 EXTERNXML int rtXmlEncBOM (OSCTXT \* *pctxt*)

This function encodes the Unicode byte order mark header at the start of the document. It is called by `rtXmlEncStartDocument` and does not need to be called manually.

#### Parameters

*pctxt* Pointer to context block structure.

#### Returns

Completion status of operation:

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

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

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.

*pNS* Pointer to namespace structure.

#### Returns

Completion status of operation:

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

### 6.3.2.14 EXTERNXML int rtXmlEncBoolAttr (OSCTXT \* *pctxt*, OSBOOL *value*, const OSUTF8CHAR \* *attrName*, OSSIZE *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.

### 6.3.2.15 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.

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

This function encodes an XML comment.

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

#### Parameters

*pctxt* Pointer to context block structure.

*comment* The comment text.

#### Returns

Completion status of operation:

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

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

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

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.

*pNS* Pointer to namespace structure.

#### Returns

Completion status of operation:

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

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

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

#### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

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

*pNS* Pointer to namespace structure.

#### Returns

Completion status of operation:

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

### 6.3.2.19 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.

### 6.3.2.20 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.

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

This function encodes a variable of the XSD decimal type.

**Parameters**

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

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

*pNS* Pointer to namespace structure.

*pFmtSpec* Pointer to format specification structure.

**Returns**

Completion status of operation:

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

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

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

**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.

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

This function encodes a value of the XSD decimal type.

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

**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 `pctx->buffer` or in stream associated with the `pctx`.

*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.

#### 6.3.2.24 EXTERNXML int rtXmlEncDouble (OSCTXT \* *pctx*, OSREAL *value*, const OSUTF8CHAR \* *elemName*, OSXMLNamespace \* *pNS*, const OSDoubleFmt \* *pFmtSpec*)

This function encodes a variable of the XSD double type.

### Parameters

*pctx* Pointer to context block structure.

*value* Value to be encoded.

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

*pNS* Pointer to namespace structure.

*pFmtSpec* Pointer to format specification structure.

### Returns

Completion status of operation:

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

#### 6.3.2.25 EXTERNXML int rtXmlEncDoubleAttr (OSCTXT \* *pctx*, OSREAL *value*, const OSUTF8CHAR \* *attrName*, OSSIZE *attrNameLen*, const OSDoubleFmt \* *pFmtSpec*)

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

### Parameters

*pctx* 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.



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

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

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

**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.

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

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

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

**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.

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*elemName* XML element name.

*pNS* XML namespace information (prefix and URI).

*pNSAttrs* List of namespace attributes to be added to element.

*terminate* Add closing '>' character.

### Returns

Completion status of operation:

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

### 6.3.2.29 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.

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

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

### Parameters

*pctxt* Pointer to context block structure.

*elemName* XML element name.

*pNS* XML namespace information (prefix and URI).

### Returns

Completion status of operation:

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

### 6.3.2.31 EXTERNXML int rtXmlEncEndSoapElems (OSCTXT \* *pctxt*, OSXMLSOAPMsgType *msgtype*)

This function encodes SOAP end element tags.

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

## Parameters

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

## Returns

Completion status of operation:

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

### 6.3.2.32 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.

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

This function encodes a variable of the XSD float type.

## Parameters

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

## Returns

Completion status of operation:

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

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

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

**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.

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

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

**Parameters**

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

**Returns**

Completion status of operation:

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

**6.3.2.36 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.

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

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

*pNS* XML namespace information (prefix and URI).

**Returns**

Completion status of operation:

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

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

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

*pNS* XML namespace information (prefix and URI).

**Returns**

Completion status of operation:

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

**6.3.2.39 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.

#### 6.3.2.40 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.

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

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

*pNS* XML namespace information (prefix and URI).

##### Returns

Completion status of operation:

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

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

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

*pNS* XML namespace information (prefix and URI).

##### Returns

Completion status of operation:

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

#### 6.3.2.43 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.

#### 6.3.2.44 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.

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

This function encodes a variable of the XSD hexBinary type.

##### Parameters

*pctxt* Pointer to context block structure.

*nocts* Number of octets in the value string.

*value* Value to be encoded.

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

*pNS* XML namespace information (prefix and URI).

##### Returns

Completion status of operation:

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

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

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

**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.

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

This function encodes a variable of the XSD hexBinary type.

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

**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.

**6.3.2.48 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 OSXML-LINDENT constant value.

**Parameters**

- pctxt* Pointer to context block structure.

**Returns**

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



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

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.

*pNS* XML namespace information (prefix and URI).

**Returns**

Completion status of operation:

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

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

This function encodes a variable of the XSD integer type.

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

**Parameters**

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

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

*pNS* XML namespace information (prefix and URI).

**Returns**

Completion status of operation:

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

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

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

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

**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.

### 6.3.2.52 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.

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

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

### 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.

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

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

*pNS* XML namespace information (prefix and URI).

*pattern* Pattern of the encoded value.

**Returns**

Completion status of operation:

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

**6.3.2.55 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.

**6.3.2.56 EXTERNXML int rtXmlEncNamedBits (OSCTXT \* *pctxt*, const OSBitMapItem \* *pBitMap*, OSUINT32 *nbits*, const OSOCTET \* *pvalue*, const OSUTF8CHAR \* *elemName*, OSXMLNamespace \* *pNS*)**

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

In this case, a set of named bits was provided in the schema for the bit values. The encoding is a list of the named bit 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.

*pNS* Pointer to namespace structure.

### Returns

Completion status of operation:

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

### 6.3.2.57 EXTERNXML int rtXmlEncNSAttrs (OSCTXT \* *pctxt*, OSRTDList \* *pNSAttrs*)

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

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

### Parameters

*pctxt* Pointer to context block structure.

*pNSAttrs* Pointer to list of namespace attributes.

### Returns

Completion status of operation:

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

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

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

### Parameters

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

*pvalue* A pointer to an REAL base 10 value.

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

*pNS* XML namespace information (prefix and URI).

### Returns

Completion status of operation:

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

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

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

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

**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.

**6.3.2.60 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.

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

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

**Parameters**

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

*terminate* Add closing '>' character.

### Returns

Completion status of operation:

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

### 6.3.2.62 EXTERNXML int rtXmlEncStartSoapElems (OSCTXT \* *pctxt*, OSXMLSOAPMsgType *msgtype*)

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

This includes all of the standard SOAP namespace attributes.

### Parameters

*pctxt* Pointer to context block structure.

*msgtype* SOAP message type (body, fault, or none)

### Returns

Completion status of operation:

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

### 6.3.2.63 EXTERNXML int rtXmlEncStartSoapEnv (OSCTXT \* *pctxt*, OSRTDList \* *pNSAttrs*)

This function encodes a SOAP envelope start element tag.

This includes all of the standard SOAP namespace attributes.

### Parameters

*pctxt* Pointer to context block structure.

*pNSAttrs* List of namespace attributes to be added to SOAP envelope.

### Returns

Completion status of operation:

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

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

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. If either null or empty string is passed, no element tag is added to the encoded value.

*pNS* XML namespace information (prefix and URI).

### Returns

Completion status of operation:

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

### 6.3.2.65 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.

### 6.3.2.66 EXTERNXML int rtXmlEncStringValue2 (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *value*, OSSIZE *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.

### 6.3.2.67 EXTERNXML int rtXmlEncTermStartElement (OSCTXT \* *pctxt*)

This function terminates a currently open XML start element by adding either a '>' or '/>' (if empty) terminator. It will also add XSI attributes to the element.

#### Parameters

*pctxt* Pointer to context block structure.

#### Returns

Completion status of operation:

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

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

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.

*pNS* Pointer to namespace structure.

#### Returns

Completion status of operation:

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

### 6.3.2.69 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.



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

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.

*pNS* XML namespace information (prefix and URI).

**Returns**

Completion status of operation:

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

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

This function encodes a variable of the XSD integer type.

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

**Parameters**

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

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

*pNS* XML namespace information (prefix and URI).

**Returns**

Completion status of operation:

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

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

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

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

**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.

### 6.3.2.73 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.

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

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

### 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.

### 6.3.2.75 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.

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

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.

*pNS* XML namespace information (prefix and URI).

#### Returns

Completion status of operation:

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

### 6.3.2.77 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.

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

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

**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.

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

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.
- pNS* XML namespace information (prefix and URI).

**Returns**

Completion status of operation:

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

**6.3.2.80 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`.

### Returns

Completion status of operation:

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

#### 6.3.2.81 EXTERNXML int rtXmlEncXSINilAttr (OSCTXT \* *pctxt*)

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

### Parameters

*pctxt* Pointer to context block structure.

### Returns

Completion status of operation:

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

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

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

### Parameters

*pctxt* Pointer to context block structure.

*value* XSI type attribute value.

### Returns

Completion status of operation:

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

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

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

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

### Parameters

*pctxt* Pointer to context block structure.

*typeNsUri* The type's namespace URI. Either null or empty if none.

*typeName* The type's name.

## Returns

Completion status of operation:

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

### 6.3.2.84 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.

### 6.3.2.85 EXTERNXML int rtXmlGetIndent (OSCTXT \* *pctxt*)

This function returns current XML output indent value.

## Parameters

*pctxt* Pointer to OSCTXT structure

## Returns

Current indent value ( $\geq 0$ ) if OK, negative status code if error.

### 6.3.2.86 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.

### 6.3.2.87 EXTERNXML OSBOOL rtXmlGetWriteBOM (OSCTXT \* *pctxt*)

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

## Parameters

*pctxt* Pointer to OSCTXT structure.

## Returns

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

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

This function prints a list of namespace attributes.

## Parameters

*name* Name to print.

*data* List of namespace attributes.

## Returns

Completion status of operation:

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

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

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

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

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

## 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).

## 6.4 XML utility functions.

### Functions

- EXTERNXML int **rtXmlWriteToFile** (OSCTXT \*pctxt, const char \*filename)  
*This function writes the encoded XML message stored in the context message buffer out to a file.*
- EXTERNXML int **rtXmlWriteUTF16ToFile** (OSCTXT \*pctxt, const char \*filename)
- EXTERNXML void **rtXmlTreatWhitespaces** (OSCTXT \*pctxt, int whiteSpaceType)
- EXTERNXML int **rtXmlCheckBuffer** (OSCTXT \*pctxt, OSSIZE byte\_count)

### 6.4.1 Function Documentation

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

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

#### Parameters

*pctxt* Pointer to OSCTXT structure.

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

#### Returns

0 - if success, negative value if error.



## 6.5 XML pull-parser decode functions.

### Defines

- #define **OSXMLREALENC\_OBJSYS** 0x1F
- #define **OSXMLREALENC\_BXER** 0x10
- #define **OSXMLREALENC\_EXERMODS** 0x1B
- #define **OSXMLREALENC\_EXERDECIMAL** 0x03

### Functions

- EXTERNXML int **rtXmlpDecAny** (OSCTXT \*pctxt, const OSUTF8CHAR \*\*pvalue)  
*This function decodes an arbitrary XML section of code as defined by the XSD any type (xsd:any).*
- EXTERNXML int **rtXmlpDecAny2** (OSCTXT \*pctxt, OSUTF8CHAR \*\*pvalue)  
*This version of the rtXmlpDecAny function returns the string in a mutable buffer.*
- EXTERNXML int **rtXmlpDecAnyAttrStr** (OSCTXT \*pctxt, const OSUTF8CHAR \*\*ppAttrStr, OSSIZE attrIndex)  
*This function decodes an any attribute string.*
- EXTERNXML int **rtXmlpDecAnyElem** (OSCTXT \*pctxt, const OSUTF8CHAR \*\*pvalue)  
*This function decodes an arbitrary XML section of code as defined by the XSD any type (xsd:any).*
- EXTERNXML int **rtXmlpDecBase64Str** (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, OSSIZE bufsize)  
*This function decodes a contents of a Base64-encode binary string into a static memory structure.*
- EXTERNXML int **rtXmlpDecBigInt** (OSCTXT \*pctxt, const OSUTF8CHAR \*\*pvalue)  
*This function will decode a variable of the XSD integer type.*
- EXTERNXML int **rtXmlpDecBitString** (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnbits, OSUINT32 bufsize)  
*This function decodes a bit string value.*
- EXTERNXML int **rtXmlpDecBool** (OSCTXT \*pctxt, OSBOOL \*pvalue)  
*This function decodes a variable of the boolean type.*
- EXTERNXML int **rtXmlpDecDate** (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'date' type.*
- EXTERNXML int **rtXmlpDecDateTime** (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'dateTime' type.*
- EXTERNXML int **rtXmlpDecDecimal** (OSCTXT \*pctxt, OSREAL \*pvalue, int totalDigits, int fractionDigits)  
*This function decodes the contents of a decimal data type.*
- EXTERNXML int **rtXmlpDecDouble** (OSCTXT \*pctxt, OSREAL \*pvalue)  
*This function decodes the contents of a float or double data type.*

- EXTERNXML int [rtXmlpDecDoubleExt](#) (OSCTXT \*pctx, OSUINT8 flags, OSREAL \*pvalue)  
*This function decodes the contents of a float or double data type.*
- EXTERNXML int [rtXmlpDecDynBase64Str](#) (OSCTXT \*pctx, OSDynOctStr \*pvalue)  
*This function decodes a contents of a Base64-encode binary string.*
- EXTERNXML int [rtXmlpDecDynBitString](#) (OSCTXT \*pctx, OSDynOctStr \*pvalue)  
*This function decodes a bit string value.*
- EXTERNXML int [rtXmlpDecDynHexStr](#) (OSCTXT \*pctx, OSDynOctStr \*pvalue)  
*This function decodes a contents of a hexBinary string.*
- EXTERNXML int [rtXmlpDecDynUnicodeStr](#) (OSCTXT \*pctx, const OSUNICHAR \*\*ppdata, OSSIZE \*pnchars)  
*This function decodes a Unicode string data type.*
- EXTERNXML int [rtXmlpDecDynUTF8Str](#) (OSCTXT \*pctx, const OSUTF8CHAR \*\*outdata)  
*This function decodes the contents of a UTF-8 string data type.*
- EXTERNXML int [rtXmlpDecUTF8Str](#) (OSCTXT \*pctx, OSUTF8CHAR \*out, OSSIZE max\_len)  
*This function decodes the contents of a UTF-8 string data type.*
- EXTERNXML int [rtXmlpDecGDay](#) (OSCTXT \*pctx, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gDay' type.*
- EXTERNXML int [rtXmlpDecGMonth](#) (OSCTXT \*pctx, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gMonth' type.*
- EXTERNXML int [rtXmlpDecGMonthDay](#) (OSCTXT \*pctx, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gMonthDay' type.*
- EXTERNXML int [rtXmlpDecGYear](#) (OSCTXT \*pctx, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gYear' type.*
- EXTERNXML int [rtXmlpDecGYearMonth](#) (OSCTXT \*pctx, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gYearMonth' type.*
- EXTERNXML int [rtXmlpDecHexStr](#) (OSCTXT \*pctx, OSOCTET \*pvalue, OSUINT32 \*pnocts, OSINT32 bufsize)  
*This function decodes the contents of a hexBinary string into a static memory structure.*
- EXTERNXML int [rtXmlpDecInt](#) (OSCTXT \*pctx, OSINT32 \*pvalue)  
*This function decodes the contents of a 32-bit integer data type.*
- EXTERNXML int [rtXmlpDecInt8](#) (OSCTXT \*pctx, OSINT8 \*pvalue)  
*This function decodes the contents of an 8-bit integer data type (i.e.*
- EXTERNXML int [rtXmlpDecInt16](#) (OSCTXT \*pctx, OSINT16 \*pvalue)  
*This function decodes the contents of a 16-bit integer data type.*

- EXTERNXML int [rtXmlpDecInt64](#) (OSCTXT \*pctxt, OSINT64 \*pvalue)  
*This function decodes the contents of a 64-bit integer data type.*
- EXTERNXML int [rtXmlpDecNamedBits](#) (OSCTXT \*pctxt, const OSBitMapItem \*pBitMap, OSOCTET \*pvalue, OSUINT32 \*pnbits, OSUINT32 bufsize)  
*This function decodes the contents of a named bit field.*
- EXTERNXML int [rtXmlpDecStrList](#) (OSCTXT \*pctxt, OSRTDList \*plist)  
*This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.*
- EXTERNXML int [rtXmlpDecTime](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'time' type.*
- EXTERNXML int [rtXmlpDecUInt](#) (OSCTXT \*pctxt, OSUINT32 \*pvalue)  
*This function decodes the contents of an unsigned 32-bit integer data type.*
- EXTERNXML int [rtXmlpDecUInt8](#) (OSCTXT \*pctxt, OSOCTET \*pvalue)  
*This function decodes the contents of an unsigned 8-bit integer data type (i.e.*
- EXTERNXML int [rtXmlpDecUInt16](#) (OSCTXT \*pctxt, OSUINT16 \*pvalue)  
*This function decodes the contents of an unsigned 16-bit integer data type.*
- EXTERNXML int [rtXmlpDecUInt64](#) (OSCTXT \*pctxt, OSUINT64 \*pvalue)  
*This function decodes the contents of an unsigned 64-bit integer data type.*
- EXTERNXML int [rtXmlpDecXmlStr](#) (OSCTXT \*pctxt, OSXMLSTRING \*outdata)  
*This function decodes the contents of an XML string data type.*
- EXTERNXML int [rtXmlpDecXmlStrList](#) (OSCTXT \*pctxt, OSRTDList \*plist)  
*This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.*
- EXTERNXML int [rtXmlpDecXSIAttr](#) (OSCTXT \*pctxt, const OSXMLNameFragments \*attrName)  
*This function decodes XSI (XML Schema Instance) attributes that may be present in any arbitrary XML element within a document.*
- EXTERNXML int [rtXmlpDecXSITypeAttr](#) (OSCTXT \*pctxt, const OSXMLNameFragments \*attrName, const OSUTF8CHAR \*\*ppAttrValue)  
*This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).*
- EXTERNXML int [rtXmlpGetAttributeID](#) (const OSXMLStrFragment \*attrName, OSINT16 nsidx, OSSIZE nAttr, const OSXMLAttrDescr attrNames[ ], OSUINT32 attrPresent[ ])  
*This function finds an attribute in the descriptor table.*
- EXTERNXML int [rtXmlpGetNextElem](#) (OSCTXT \*pctxt, OSXMLElemDescr \*pElem, OSINT32 level)  
*This function parse the next element start tag.*
- EXTERNXML int [rtXmlpGetNextElemID](#) (OSCTXT \*pctxt, const OSXMLElemIDRec \*tab, OSSIZE nrows, OSINT32 level, OSBOOL continueParse)  
*This function parses the next start tag and finds the index of the element name in the descriptor table.*

- EXTERNXML int [rtXmlpMarkLastEventActive](#) (OSCTXT \*pctxt)  
*This function marks current tag as unprocessed.*
- EXTERNXML int [rtXmlpMatchStartTag](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*elemLocalName, OSINT16 nsidx)  
*This function parses the next start tag that matches with given name.*
- EXTERNXML int [rtXmlpMatchEndTag](#) (OSCTXT \*pctxt, OSINT32 level)  
*This function parse next end tag that matches with given name.*
- EXTERNXML OSBOOL [rtXmlpHasAttributes](#) (OSCTXT \*pctxt)  
*This function checks accessibility of attributes.*
- EXTERNXML int [rtXmlpGetAttributeCount](#) (OSCTXT \*pctxt)  
*This function returns number of attributes in last processed start tag.*
- EXTERNXML int [rtXmlpSelectAttribute](#) (OSCTXT \*pctxt, [OSXMLNameFragments](#) \*pAttr, OSINT16 \*nsidx, OSSIZE attrIndex)  
*This function selects attribute to decode.*
- EXTERNXML OSINT32 [rtXmlpGetCurrentLevel](#) (OSCTXT \*pctxt)  
*This function returns current nesting level.*
- EXTERNXML void [rtXmlpSetWhiteSpaceMode](#) (OSCTXT \*pctxt, [OSXMLWhiteSpaceMode](#) whiteSpaceMode)  
*Sets the whitespace treatment mode.*
- EXTERNXML OSBOOL [rtXmlpSetMixedContentMode](#) (OSCTXT \*pctxt, OSBOOL mixedContentMode)  
*Sets mixed content mode.*
- EXTERNXML void [rtXmlpSetListMode](#) (OSCTXT \*pctxt)  
*Sets list mode.*
- EXTERNXML OSBOOL [rtXmlpListHasItem](#) (OSCTXT \*pctxt)  
*Check for end of decoded token list.*
- EXTERNXML void [rtXmlpCountListItems](#) (OSCTXT \*pctxt, OSSIZE \*itemCnt)  
*Count tokens in list.*
- EXTERNXML int [rtXmlpGetNextSeqElemID2](#) (OSCTXT \*pctxt, const [OSXMLElemIDRec](#) \*tab, const [OSXMLGroupDesc](#) \*pGroup, int groups, int curID, int lastMandatoryID, OSBOOL groupMode, OSBOOL checkRepeat)  
*This function parses the next start tag and finds index of element name in descriptor table.*
- EXTERNXML int [rtXmlpGetNextSeqElemID](#) (OSCTXT \*pctxt, const [OSXMLElemIDRec](#) \*tab, const [OSXMLGroupDesc](#) \*pGroup, int curID, int lastMandatoryID, OSBOOL groupMode)  
*This function parses the next start tag and finds index of element name in descriptor table.*
- EXTERNXML int [rtXmlpGetNextSeqElemIDExt](#) (OSCTXT \*pctxt, const [OSXMLElemIDRec](#) \*tab, const [OSXMLGroupDesc](#) \*ppGroup, const OSBOOL \*extRequired, int postExtRootID, int curID, int lastMandatoryID, OSBOOL groupMode)

*This is an ASN.1 extension-supporting version of rtXmlpGetNextSeqElemID.*

- EXTERNXML int [rtXmlpGetNextAllElemID](#) (OSCTXT \*pctx, const OSXMLElemIDRec \*tab, OSSIZE nRows, const OSUINT8 \*pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)  
*This function parses the next start tag and finds index of element name in descriptor table.*
- EXTERNXML int [rtXmlpGetNextAllElemID16](#) (OSCTXT \*pctx, const OSXMLElemIDRec \*tab, OSSIZE nRows, const OSUINT16 \*pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)  
*This function parses the next start tag and finds index of element name in descriptor table.*
- EXTERNXML int [rtXmlpGetNextAllElemID32](#) (OSCTXT \*pctx, const OSXMLElemIDRec \*tab, OSSIZE nRows, const OSUINT32 \*pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)  
*This function parses the next start tag and finds index of element name in descriptor table.*
- EXTERNXML void [rtXmlpSetNamespaceTable](#) (OSCTXT \*pctx, const OSUTF8CHAR \*namespaceTable[], OSSIZE nmNamespaces)  
*Sets user namespace table.*
- EXTERNXML int [rtXmlpCreateReader](#) (OSCTXT \*pctx)  
*Creates pull parser reader structure within the context.*
- EXTERNXML void [rtXmlpHideAttributes](#) (OSCTXT \*pctx)  
*Disable access to attributes.*
- EXTERNXML OSBOOL [rtXmlpNeedDecodeAttributes](#) (OSCTXT \*pctx)  
*This function checks if attributes were previously decoded.*
- EXTERNXML void [rtXmlpMarkPos](#) (OSCTXT \*pctx)  
*Save current decode position.*
- EXTERNXML void [rtXmlpRewindToMarkedPos](#) (OSCTXT \*pctx)  
*Rewind to saved decode position.*
- EXTERNXML void [rtXmlpResetMarkedPos](#) (OSCTXT \*pctx)  
*Reset saved decode position.*
- EXTERNXML int [rtXmlpGetXSITypeAttr](#) (OSCTXT \*pctx, const OSUTF8CHAR \*\*ppAttrValue, OSINT16 \*nsidx, OSSIZE \*pLocalOffs)  
*This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).*
- EXTERNXML int [rtXmlpGetXmlnsAttrs](#) (OSCTXT \*pctx, OSRTDList \*pNSAttrs)  
*This function decodes namespace attributes from start tag and adds them to the given list.*
- EXTERNXML int [rtXmlpDecXSIAttrs](#) (OSCTXT \*pctx)  
*This function decodes XSI (XML Schema Instance) that may be present in any arbitrary XML element within a document.*
- EXTERNXML OSBOOL [rtXmlpIsEmptyElement](#) (OSCTXT \*pctx)  
*Check element content: empty or not.*
- EXTERNXML int [rtXmlEncAttrC14N](#) (OSCTXT \*pctx)  
*This function used only in C14 mode.*

- EXTERNXML struct OSXMLReader \* [rtXmlpGetReader](#) (OSCTXT \*pctxt)  
*This function fetches the XML reader structure from the context for use in low-level pull parser calls.*
- EXTERNXML OSBOOL [rtXmlpIsLastEventDone](#) (OSCTXT \*pctxt)  
*Check processing status of current tag.*
- EXTERNXML int [rtXmlpGetXSITypeIndex](#) (OSCTXT \*pctxt, const OSXMLItemDescr typetab[], OSSIZE typetabsiz)  
*This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type) and find type index in descriptor table.*
- EXTERNXML int [rtXmlpLookupXSITypeIndex](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*pXsiType, OSINT16 xsiTypeIdx, const OSXMLItemDescr typetab[], OSSIZE typetabsiz)  
*This function find index of XSI (XML Schema Instance) type in descriptor table.*
- EXTERNXML void [rtXmlpForceDecodeAsGroup](#) (OSCTXT \*pctxt)  
*Disable skipping of unknown elements in optional sequence tail.*
- EXTERNXML OSBOOL [rtXmlpIsDecodeAsGroup](#) (OSCTXT \*pctxt)  
*This function checks if "decode as group" mode was forced.*
- EXTERNXML OSBOOL [rtXmlpIsUTF8Encoding](#) (OSCTXT \*pctxt)  
*This function checks if the encoding specified in XML header is UTF-8.*
- EXTERNXML int [rtXmlpReadBytes](#) (OSCTXT \*pctxt, OSOCTET \*pbuf, OSSIZE nbytes)  
*This function reads the specified number of bytes directly from the underlying XML parser stream.*

## 6.5.1 Function Documentation

### 6.5.1.1 EXTERNXML int rtXmlEncAttrC14N (OSCTXT \* pctxt)

This function used only in C14 mode.

It provide attributes sorting.

#### Parameters

*pctxt* Pointer to context block structure.

#### Returns

Completion status of operation:

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

### 6.5.1.2 EXTERNXML void rtXmlpCountListItems (OSCTXT \* *pctxt*, OSSIZE \* *itemCnt*)

Count tokens in list.

#### Parameters

*pctxt* Pointer to context block structure.

*itemCnt* Pointer to number of elements in list.

#### Returns

Token number. For element content function check accessed part of content only. Returned value may be below then real token number.

### 6.5.1.3 EXTERNXML int rtXmlpCreateReader (OSCTXT \* *pctxt*)

Creates pull parser reader structure within the context.

#### Parameters

*pctxt* Pointer to context block structure.

#### Returns

Completion status of operation:

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

### 6.5.1.4 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.

#### 6.5.1.5 EXTERNXML int rtXmlpDecAny2 (OSCTXT \* *pctxt*, OSUTF8CHAR \*\* *pvalue*)

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

##### 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.

#### 6.5.1.6 EXTERNXML int rtXmlpDecAnyAttrStr (OSCTXT \* *pctxt*, const OSUTF8CHAR \*\* *ppAttrStr*, OSSIZE *attrIndex*)

This function decodes an any attribute string.

The full attribute string (name="value") is decoded and returned on the string output argument. Memory is allocated for the string using the rtxMemAlloc function.

##### Parameters

*pctxt* Pointer to context block structure.

*ppAttrStr* Pointer to UTF8 character string pointer to receive decoded attribute string. Memory is allocated for the string using the run-time memory manager.

*attrIndex* Index of attribute.

##### Returns

Completion status of operation:

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

#### 6.5.1.7 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.

### 6.5.1.8 EXTERNXML int rtXmlpDecBase64Str (OSCTXT \* *pctxt*, OSOCTET \* *pvalue*, OSUINT32 \* *pnocts*, OSSIZE *bufsize*)

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

The octet string must be Base64 encoded. This function call is used to decode a sized base64Binary string production. Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

## 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.

### 6.5.1.9 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.

*pvalue* Pointer to a pointer to receive decoded UTF-8 string. Dynamic memory is allocated for the variable using the `rtMemAlloc` function. The decoded variable is represented as a string starting with appropriate prefix.

## Returns

Completion status of operation:

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

#### 6.5.1.10 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.

#### 6.5.1.11 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.

#### 6.5.1.12 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.

#### 6.5.1.13 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.

#### 6.5.1.14 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. 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.

#### 6.5.1.15 EXTERNXML int rtXmlpDecDouble (OSCTXT \* *pctxt*, OSREAL \* *pvalue*)

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

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

##### Parameters

*pctxt* Pointer to context block structure.

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

##### Returns

Completion status of operation:

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

#### 6.5.1.16 EXTERNXML int rtXmlpDecDoubleExt (OSCTXT \* *pctxt*, OSUINT8 *flags*, OSREAL \* *pvalue*)

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*flags* Specifies alternatives allowed in the lexical value. See OSXMLREALENC\* constants. bit 0 (rightmost) set: recognize INF, -INF, NaN text values bit 1 set: recognize leading '+' on normal reals bit 2 set: recognize leading '+' on INF bit 3 set: recognize leading zeros in exponent bit 4 set: recognize exponents

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

##### Returns

Completion status of operation:

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

#### 6.5.1.17 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.

#### 6.5.1.18 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.

### 6.5.1.19 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 OS-UTF8CHAR 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.

### 6.5.1.20 EXTERNXML int rtXmlpDecDynUnicodeStr (OSCTXT \* *pctxt*, const OSUNICHAR \*\* *ppdata*, OSSIZE \* *pnchars*)

This function decodes a Unicode string data type.

The input is assumed to be in UTF-8 format. This function reads each character and converts it into its Unicode equivalent.

## 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.

#### 6.5.1.21 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.

#### 6.5.1.22 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.

#### 6.5.1.23 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.

#### 6.5.1.24 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.

#### 6.5.1.25 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.

#### 6.5.1.26 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.

**6.5.1.27 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.

**6.5.1.28 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.

**6.5.1.29 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.

### 6.5.1.30 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.

### 6.5.1.31 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.

### 6.5.1.32 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.

### 6.5.1.33 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.

### 6.5.1.34 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.

#### 6.5.1.35 EXTERNXML int rtXmlpDecUInt (OSCTXT \* *pctxt*, OSUINT32 \* *pvalue*)

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

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

##### 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.

#### 6.5.1.36 EXTERNXML int rtXmlpDecUInt16 (OSCTXT \* *pctxt*, OSUINT16 \* *pvalue*)

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

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

##### 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.

#### 6.5.1.37 EXTERNXML int rtXmlpDecUInt64 (OSCTXT \* *pctxt*, OSUINT64 \* *pvalue*)

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

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

##### 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.

#### 6.5.1.38 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.

#### 6.5.1.39 EXTERNXML int rtXmlpDecUTF8Str (OSCTXT \* *pctxt*, OSUTF8CHAR \* *out*, OSSIZE *max\_len*)

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*out* Pointer to an array of OSUTF8CHAR to receive decoded UTF-8 string.

*max\_len* Length of out array.

##### Returns

Completion status of operation:

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

#### 6.5.1.40 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.

#### 6.5.1.41 EXTERNXML int rtXmlpDecXmlStrList (OSCTXT \* *pctxt*, OSRTDList \* *plist*)

This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.

Memory is allocated for the list nodes and token values using the rtx memory management functions. List contains OSXMLSTRING structures.

##### 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.

#### 6.5.1.42 EXTERNXML int rtXmlpDecXSIAttr (OSCTXT \* *pctxt*, const OSXMLNameFragments \* *attrName*)

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

##### 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.

#### 6.5.1.43 EXTERNXML int rtXmlpDecXSIAttrs (OSCTXT \* *pctxt*)

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

##### Parameters

*pctxt* Pointer to context block structure.

##### Returns

Completion status of operation:

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

**6.5.1.44 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.

**6.5.1.45 EXTERNXML void rtXmlpForceDecodeAsGroup (OSCTXT \* *pctxt*)**

Disable skipping of unknown elements in optional sequence tail.

Function used in outer types to break decode on first unknown element after decoding mandatory sequence part.

**Parameters**

*pctxt* Pointer to context block structure.

**6.5.1.46 EXTERNXML int rtXmlpGetAttributeCount (OSCTXT \* *pctxt*)**

This function returns number of attributes in last processed start tag.

**Parameters**

*pctxt* Pointer to context block structure.

**Returns**

Completion status of operation:

- zero or positive value is attributes number,
- negative return value is error.

**6.5.1.47 EXTERNXML int rtXmlpGetAttributeID (const OSXMLStrFragment \* *attrName*, OSINT16 *nsidx*, OSSIZE *nAttr*, const OSXMLAttrDescr *attrNames*[], OSUINT32 *attrPresent*[])**

This function finds an attribute in the descriptor table.

## Parameters

***attrName*** A pointer to a structure holding various parts of an attribute name. The parts are in the form of string fragments meaning they are not null terminated. The user must be careful to use the value and length when working with them.

***nsidx*** Namespace index:

- 0 = attribute is unqualified,
- positive value is user namespace from generated namespace table,
- negative value is predefined namespace like XSD instance and ect.

***nAttr*** Number of descriptors in table.

***attrNames*** Attributes descriptor table.

***attrPresent*** Bit array to mark already decoded attributes. It is used to identify duplicate attributes.

## Returns

Completion status of operation:

- positive or zero return value is attribute index in descriptor table,
- negative return value is error.

### 6.5.1.48 EXTERNXML OSINT32 rtXmlpGetCurrentLevel (OSCTXT \* *pctxt*)

This function returns current nesting level.

## Parameters

***pctxt*** Pointer to context block structure.

## Returns

Current nesting level.

### 6.5.1.49 EXTERNXML int rtXmlpGetNextAllElemID (OSCTXT \* *pctxt*, const OSXMLElemIDRec \* *tab*, OSSIZE *nrows*, const OSUINT8 \* *pOrder*, OSSIZE *nOrder*, OSSIZE *maxOrder*, int *anyID*)

This function parses the next start tag and finds index of element name in descriptor table.

It used for decode "all" content model in strict mode.

## Parameters

***pctxt*** Pointer to context block structure.

***tab*** Elements descriptor table.

***nrows*** Number of descriptors in table.

***pOrder*** Pointer to array to receive elements order.

***nOrder*** Last element's index in order array.

***maxOrder*** Size of order array.

***anyID*** Identifier of xsd:any element.

## Returns

Completion status of operation:

- positive or zero value is element identifier,
- negative return value is error.

**6.5.1.50 EXTERNXML int rtXmlpGetNextAllElemID16 (OSCTXT \* *pctxt*, const OSXMLElemIDRec \* *tab*, OSSIZE *nrows*, const OSUINT16 \* *pOrder*, OSSIZE *nOrder*, OSSIZE *maxOrder*, int *anyID*)**

This function parses the next start tag and finds index of element name in descriptor table.

It used for decode "all" content model in strict mode. This variant used when xsd:all has above 256 elements.

**Parameters**

- pctxt* Pointer to context block structure.
- tab* Elements descriptor table.
- nrows* Number of descriptors in table.
- pOrder* Pointer to array to receive elements order.
- nOrder* Last element's index in order array.
- maxOrder* Size of order array.
- anyID* Identifier of xsd:any element.

**Returns**

Completion status of operation:

- positive or zero value is element identifier,
- negative return value is error.

**6.5.1.51 EXTERNXML int rtXmlpGetNextAllElemID32 (OSCTXT \* *pctxt*, const OSXMLElemIDRec \* *tab*, OSSIZE *nrows*, const OSUINT32 \* *pOrder*, OSSIZE *nOrder*, OSSIZE *maxOrder*, int *anyID*)**

This function parses the next start tag and finds index of element name in descriptor table.

It used for decode "all" content model in strict mode.

**Parameters**

- pctxt* Pointer to context block structure.
- tab* Elements descriptor table.
- nrows* Number of descriptors in table.
- pOrder* Pointer to array to receive elements order.
- nOrder* Last element's index in order array.
- maxOrder* Size of order array.
- anyID* Identifier of xsd:any element.

**Returns**

Completion status of operation:

- positive or zero value is element identifier,
- negative return value is error.



**6.5.1.52 EXTERNXML int rtXmlpGetNextElem (OSCTXT \* *pctxt*, OSXMLElemDesc \* *pElem*, OSINT32 *level*)**

This function parse the next element start tag.

**Parameters**

*pctxt* Pointer to context block structure.

*pElem* Pointer to a structure to receive the decoded element descriptor.

*level* Nesting level of parsed start tag. When value equal -1 parsed next start tag.

**Returns**

Completion status of operation:

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

**6.5.1.53 EXTERNXML int rtXmlpGetNextElemID (OSCTXT \* *pctxt*, const OSXMLElemIDRec \* *tab*, OSSIZE *nrows*, OSINT32 *level*, OSBOOL *continueParse*)**

This function parses the next start tag and finds the index of the element name in the descriptor table.

If this reaches the closing tag for the current level, it returns XML\_OK\_EOB. If this encounters an unexpected element and !continueParse, it returns RTERR\_UNEXPELEM (without logging it).

**Parameters**

*pctxt* Pointer to context block structure.

*tab* Elements descriptor table.

*nrows* Number of descriptors in table.

*level* Nesting level of parsed start tag. When value equal -1 function parse next start tag.

*continueParse* When value equals TRUE function skips unrecognized elements; skipped elements are logged, but an error is not returned.

**Returns**

Completion status of operation:

- positive or zero value is element identifier,
- negative return value is error.

**6.5.1.54 EXTERNXML int rtXmlpGetNextSeqElemID (OSCTXT \* *pctxt*, const OSXMLElemIDRec \* *tab*, const OSXMLGroupDesc \* *pGroup*, int *curID*, int *lastMandatoryID*, OSBOOL *groupMode*)**

This function parses the next start tag and finds index of element name in descriptor table.

It is used to decode sequences in strict mode.

Handling of unexpected elements:

- If the current decode group includes an any case, unexpected elements will be matched against it (the any case id will be returned).

- Otherwise, if `groupMode` is true, and the element identified by `lastMandatoryID` has been reached, `XML_OK_EOB` is returned. The unexpected element may belong to something following the elements in `tab`.
- If neither of the above applies, unknown elements will be logged and skipped over, with this method continuing as if the unexpected element were not present. Handling of the case of reaching closing tag for current level:
- If the last mandatory element is reached, return `XML_OK_EOB`.
- Otherwise, log and return `XML_E_ELEMSMISRQ`.

This function is equivalent to: `rtXmlpGetNextSeqElemID2(pctxt, tab, pGroup, curID, lastMandatoryID, groupMode, FALSE);`

### Parameters

*pctxt* Pointer to context block structure.

*tab* Elements descriptor table.

*pGroup* Decode groups table.

*curID* Current decode group.

*lastMandatoryID* Identifier of last mandatory element.

*groupMode* This parameter must be set to TRUE when decoding groups or base types.

### Returns

Completion status of operation:

- positive or zero value is element identifier,
- negative return value is error.

**6.5.1.55 EXTERNXML int rtXmlpGetNextSeqElemID2 (OSCTXT \* *pctxt*, const OSXMLElemIDRec \* *tab*, const OSXMLGroupDesc \* *pGroup*, int *groups*, int *curID*, int *lastMandatoryID*, OSBOOL *groupMode*, OSBOOL *checkRepeat*)**

This function parses the next start tag and finds index of element name in descriptor table.

It is used to decode sequences in strict mode.

### Parameters

*pctxt* Pointer to context block structure.

*tab* Elements descriptor table.

*pGroup* Decode groups table.

*groups* Number of groups in groups table. If `checkRepeat` is FALSE, you can pass 0 for this if the value is unknown.

*curID* Current decode group.

*lastMandatoryID* Identifier of last mandatory element.

*groupMode* This parameter must be set to TRUE when decode groups or base types.

*checkRepeat* If true, this method is being called to check for a repeat of `curID`. In this case, we do not treat `curID` as being required. Otherwise, `curID` is required if `curID <= lastMandatoryID`.

### Returns

Completion status of operation:

- positive or zero value is element identifier,
- negative return value is error.

**6.5.1.56 EXTERNXML int rtXmlpGetNextSeqElemIDExt (OSCTXT \* *pctxt*, const OSXMLElemIDRec \* *tab*, const OSXMLGroupDesc \* *ppGroup*, const OSBOOL \* *extRequired*, int *postExtRootID*, int *curID*, int *lastMandatoryID*, OSBOOL *groupMode*)**

This is an ASN.1 extension-supporting version of `rtXmlpGetNextSeqElemID`.

It allows required extension elements to be absent, except that when an extension group is present, all required extension elements in that group must be present. It otherwise behaves the same as `rtXmlpGetNextSeqElemID`.

#### Parameters

*pctxt* Pointer to context block structure.

*tab* Elements descriptor table.

*pGroup* Decode groups table.

*extRequired* `extRequired[i]` corresponds to `pGroup[i]`. This is TRUE if and only if the decode group ends with a non-optional, non-default extension element that must be present in the encoding (because it is inside an extension group for which some element has already been decoded).

*postExtRootID* This is the group id corresponding to the decode group that begins with the first root element that follows the extension additions. If there is no such element, this is -1.

*curID* Current decode group.

*lastMandatoryID* Identifier of last mandatory element.

*groupMode* This parameter must be set to TRUE when decoding groups or base types.

#### Returns

Completion status of operation:

- positive or zero value is element identifier,
- negative return value is error.

**6.5.1.57 EXTERNXML struct OSXMLReader\* rtXmlpGetReader (OSCTXT \* *pctxt*) [read]**

This function fetches the XML reader structure from the context for use in low-level pull parser calls.

If the reader structure does not exist, it is created and initialized.

#### Parameters

*pctxt* Pointer to context block structure.

#### Returns

Pointer to XML reader structure or NULL if an error occurred.

**6.5.1.58 EXTERNXML int rtXmlpGetXmlnsAttrs (OSCTXT \* *pctxt*, OSRTDList \* *pNSAttrs*)**

This function decodes namespace attributes from start tag and adds them to the given list.

#### Parameters

*pctxt* Pointer to context block structure.

*pNSAttrs* A pointer to a linked list of OSXMLNamespace structures.

## Returns

Completion status of operation:

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

### 6.5.1.59 EXTERNXML int rtXmlpGetXSITypeAttr (OSCTXT \* *pctxt*, const OSUTF8CHAR \*\* *ppAttrValue*, OSINT16 \* *nsidx*, OSSIZE \* *pLocalOffs*)

This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).

## Parameters

*pctxt* Pointer to context block structure.

*ppAttrValue* A pointer to a pointer to a UTF8 character string to received the decoded XSI type QName.

*nsidx* A pointer to OSINT16 value to received the decoded XSI type namespace index.

*pLocalOffs* A pointer to size\_t value to received the local name offset in *ppAttrValue*. When *pLocalOffs* value equal NULL function return in *ppAttrValue* local name only.

## Returns

Completion status of operation:

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

### 6.5.1.60 EXTERNXML int rtXmlpGetXSITypeIndex (OSCTXT \* *pctxt*, const OSXMLItemDescr *typetab*[], OSSIZE *typetabsiz*)

This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type) and find type index in descriptor table.

## Parameters

*pctxt* Pointer to context block structure.

*typetab* XSI types descriptor table.

*typetabsiz* Number of descriptors in table.

## Returns

Completion status of operation:

- positive or zero value is type index,
- negative return value is error.

### 6.5.1.61 EXTERNXML OSBOOL rtXmlpHasAttributes (OSCTXT \* *pctxt*)

This function checks accessibility of attributes.

## Parameters

*pctxt* Pointer to context block structure.

## Returns

Completion status of operation:

- TRUE = attributes are present and access is enabled,
- FALSE = attributes are absent or access is disabled.

### 6.5.1.62 EXTERNXML void rtXmlpHideAttributes (OSCTXT \* *pctxt*)

Disable access to attributes.

Function used in derived types to disable repeated decode in base type.

## Parameters

*pctxt* Pointer to context block structure.

### 6.5.1.63 EXTERNXML OSBOOL rtXmlpIsDecodeAsGroup (OSCTXT \* *pctxt*)

This function checks if "decode as group" mode was forced.

## Parameters

*pctxt* Pointer to context block structure.

## Returns

State of mode:

- TRUE = need decode as group,
- FALSE = normal sequence decoding.

### 6.5.1.64 EXTERNXML OSBOOL rtXmlpIsEmptyElement (OSCTXT \* *pctxt*)

Check element content: empty or not.

## Parameters

*pctxt* Pointer to context block structure.

## Returns

Status of element content:

- TRUE = element is empty,
- FALSE = element has content.

### 6.5.1.65 EXTERNXML OSBOOL rtXmlpIsLastEventDone (OSCTXT \* *pctxt*)

Check processing status of current tag.

## Parameters

*pctxt* Pointer to context block structure.

## Returns

Status of element content:

- TRUE = tag marked as processed,
- FALSE = tag will be processed again.

### 6.5.1.66 EXTERNXML OSBOOL rtXmlpIsUTF8Encoding (OSCTXT \* *pctxt*)

This function checks if the encoding specified in XML header is UTF-8.

## Parameters

*pctxt* Pointer to context block structure.

## Returns

State of mode:

- TRUE = is in UTF-8 encoding,
- FALSE = is not in UTF-8 encoding.

### 6.5.1.67 EXTERNXML OSBOOL rtXmlpListHasItem (OSCTXT \* *pctxt*)

Check for end of decoded token list.

## Parameters

*pctxt* Pointer to context block structure.

## Returns

State of decoded list:

- TRUE = list is not finished,
- FALSE = all tokens was decoded.

### 6.5.1.68 EXTERNXML int rtXmlpLookupXSITypeIndex (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *pXsiType*, OSINT16 *xsiTypeIdx*, const OSXMLItemDescr *typetab*[], OSSIZE *typetabsiz*)

This function find index of XSI (XML Schema Instance) type in descriptor table.

## Parameters

*pctxt* Pointer to context block structure.

*pXsiType* A pointer to XSI type name.

*xsiTypeIdx* A XSI type namespace index.

*typetab* XSI types descriptor table.

*typetabsiz* Number of descriptors in table.

## Returns

Completion status of operation:

- positive or zero value is type index,
- negative return value is error.

#### 6.5.1.69 EXTERNXML int rtXmlpMarkLastEventActive (OSCTXT \* *pctxt*)

This function marks current tag as unprocessed.

This will cause the element to be processed again in the next pull-parser function.

##### Parameters

*pctxt* Pointer to context block structure.

##### Returns

Completion status of operation:

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

#### 6.5.1.70 EXTERNXML void rtXmlpMarkPos (OSCTXT \* *pctxt*)

Save current decode position.

##### Parameters

*pctxt* Pointer to context block structure.

#### 6.5.1.71 EXTERNXML int rtXmlpMatchEndTag (OSCTXT \* *pctxt*, OSINT32 *level*)

This function parse next end tag that matches with given name.

##### Parameters

*pctxt* Pointer to context block structure.

*level* Nesting level of parsed start tag. When value equal -1 function parse next end tag with current level.

##### Returns

Completion status of operation:

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

#### 6.5.1.72 EXTERNXML int rtXmlpMatchStartTag (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *elemLocalName*, OSINT16 *nsidx*)

This function parses the next start tag that matches with given name.

##### Parameters

*pctxt* Pointer to context block structure.

*elemLocalName* Name of parsed element.

*nsidx* Namespace index:

- 0 = attribute is unqualified,

- positive value is user namespace from generated namespace table,
- negative value is predefined namespace like XSD instance and ect.

### Returns

Completion status of operation:

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

#### 6.5.1.73 EXTERNXML OSBOOL rtXmlpNeedDecodeAttributes (OSCTXT \* *pctxt*)

This function checks if attributes were previously decoded.

### Parameters

*pctxt* Pointer to context block structure.

### Returns

State of attributes:

- TRUE = attributes was not decoded,
- FALSE = attributes had been decoded.

#### 6.5.1.74 EXTERNXML int rtXmlpReadBytes (OSCTXT \* *pctxt*, OSOCTET \* *pbuf*, OSSIZE *nbytes*)

This function reads the specified number of bytes directly from the underlying XML parser stream.

It has no effect on any of the parser state variables.

### Parameters

*pctxt* Pointer to context block structure.

*pbuf* Pointer to static buffer (byte array) to receive data. The buffer must be at least large enough to hold the requested number of bytes.

*nbytes* Number of bytes to read.

### Returns

State of mode:

- TRUE = is in UTF-8 encoding,
- FALSE = is not in UTF-8 encoding.

#### 6.5.1.75 EXTERNXML void rtXmlpResetMarkedPos (OSCTXT \* *pctxt*)

Reset saved decode position.

### Parameters

*pctxt* Pointer to context block structure.



#### 6.5.1.76 EXTERNXML void rtXmlpRewindToMarkedPos (OSCTXT \* *pctxt*)

Rewind to saved decode position.

##### Parameters

*pctxt* Pointer to context block structure.

#### 6.5.1.77 EXTERNXML int rtXmlpSelectAttribute (OSCTXT \* *pctxt*, OSXMLNameFragments \* *pAttr*, OSINT16 \* *nsidx*, OSSIZE *attrIndex*)

This function selects attribute to decode.

##### Parameters

*pctxt* Pointer to context block structure.

*pAttr* Pointer to structure to receive the various parts of an attribute name.

*nsidx* Pointer to value to receive namespace index:

- 0 = attribute is unqualified,
- positive value is user namespace from generated namespace table,
- negative value is predefined namespace like XSD instance and ect (see enum OSXMLNsIndex)

*attrIndex* Index of selected attribute.

##### Returns

Completion status of operation:

- positive or zero return value is attribute index in descriptor table,
- negative return value is error.

#### 6.5.1.78 EXTERNXML void rtXmlpSetListMode (OSCTXT \* *pctxt*)

Sets list mode.

Attribute value or element content is decoded by tokens.

##### Parameters

*pctxt* Pointer to context block structure.

#### 6.5.1.79 EXTERNXML OSBOOL rtXmlpSetMixedContentMode (OSCTXT \* *pctxt*, OSBOOL *mixedContentMode*)

Sets mixed content mode.

##### Parameters

*pctxt* Pointer to context block structure.

*mixedContentMode* Enable/disable mixed mode.

##### Returns

Previously state of mixed mode.

**6.5.1.80 EXTERNXML void rtXmlpSetNamespaceTable (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *namespaceTable*[], OSSIZE *nmNamespaces*)**

Sets user namespace table.

**Parameters**

- pctxt* Pointer to context block structure.
- namespaceTable* Array of namespace URI strings.
- nmNamespaces* Number of namespaces in table.

**6.5.1.81 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.

**Parameters**

- pctxt* Pointer to context block structure.
- whiteSpaceMode* White space mode.

**Returns**

Previously set whitespace mode.

## 6.6 XML run-time error status codes.

This is a list of status codes that can be returned by XML run-time functions and generated code.

### Defines

- #define `XML_OK_EOB` 0x7fffffff  
*End of block marker.*
- #define `XML_OK_FRAG` `XML_OK_EOB`  
*Maintained for backward compatibility.*
- #define `XML_E_BASE` -200  
*Error base.*
- #define `XML_E_GENERR` (`XML_E_BASE`)  
*General error.*
- #define `XML_E_INVSYMBOL` (`XML_E_BASE-1`)  
*An invalid XML symbol (character) was detected at the given point in the parse stream.*
- #define `XML_E_TAGMISMATCH` (`XML_E_BASE-2`)  
*Start/end tag mismatch.*
- #define `XML_E_DUPLATTR` (`XML_E_BASE-3`)  
*Duplicate attribute found.*
- #define `XML_E_BADCHARREF` (`XML_E_BASE-4`)  
*Bad character reference found.*
- #define `XML_E_INVMODE` (`XML_E_BASE-5`)  
*Invalid mode.*
- #define `XML_E_UNEXPEOF` (`XML_E_BASE-6`)  
*Unexpected end of file (document).*
- #define `XML_E_NOMATCH` (`XML_E_BASE-7`)  
*Current tag is not matched to specified one.*
- #define `XML_E_ELEMMISRQ` (`XML_E_BASE-8`)  
*Missing required element.*
- #define `XML_E_ELEMSISRQ` (`XML_E_BASE-9`)  
*Missing required elements.*
- #define `XML_E_TOOFWELEMS` (`XML_E_BASE-10`)  
*The number of elements in a repeating collection was less than the number of elements specified in the XSD minOccurs facet for this type or element.*
- #define `XML_E_UNEXPSTARTTAG` (`XML_E_BASE-11`)

*Unexpected start tag.*

- #define `XML_E_UNEXPENDTAG` (XML\_E\_BASE-12)  
*Unexpected end tag.*
- #define `XML_E_IDNOTFOU` (XML\_E\_BASE-13)  
*Expected identifier not found.*
- #define `XML_E_INVTYPEINFO` (XML\_E\_BASE-14)  
*Unknown xsi:type.*
- #define `XML_E_NSURINOTFOU` (XML\_E\_BASE-15)  
*Namespace URI not defined for given prefix.*
- #define `XML_E_KEYNOTFOU` (XML\_E\_BASE-16)  
*Keyref constraint has some key that not present in refered constraint.*
- #define `XML_E_DUPLKEY` (XML\_E\_BASE-17)  
*Key or unique constraint has duplicated key.*
- #define `XML_E_FLDABSENT` (XML\_E\_BASE-18)  
*Some key has no full set of fields.*
- #define `XML_E_DUPLFLD` (XML\_E\_BASE-19)  
*Some key has more than one value for field.*
- #define `XML_E_NOTEMPTY` (XML\_E\_BASE-20)  
*An element was not empty when expected.*

## 6.6.1 Detailed Description

This is a list of status codes that can be returned by XML 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.

## 6.6.2 Define Documentation

### 6.6.2.1 #define XML\_E\_BASE -200

Error base.

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

Definition at line 60 of file `rtXmlErrCodes.h`.

#### **6.6.2.2 #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.

Definition at line 121 of file rtXmlErrCodes.h.

#### **6.6.2.3 #define XML\_E\_ELEMSISRQ (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.

Definition at line 128 of file rtXmlErrCodes.h.

#### **6.6.2.4 #define XML\_E\_FLDABSENT (XML\_E\_BASE-18)**

Some key has no full set of fields.

It is not valid for key constraint.

Definition at line 199 of file rtXmlErrCodes.h.

#### **6.6.2.5 #define XML\_E\_NOMATCH (XML\_E\_BASE-7)**

Current tag is not matched to specified one.

Informational code.

Definition at line 115 of file rtXmlErrCodes.h.

#### **6.6.2.6 #define XML\_E\_NSURINOTFOU (XML\_E\_BASE-15)**

Namespace URI not defined for given prefix.

A namespace URI was not defined using an xmlns attribute for the given prefix.

Definition at line 166 of file rtXmlErrCodes.h.

#### **6.6.2.7 #define XML\_E\_TAGMISMATCH (XML\_E\_BASE-2)**

Start/end tag mismatch.

The parsed end tag does not match the start tag that was parsed earlier at this level. Indicates document is not well-formed.

Definition at line 90 of file rtXmlErrCodes.h.

#### **6.6.2.8 #define XML\_OK\_EOB 0x7fffffff**

End of block marker.

Definition at line 51 of file rtXmlErrCodes.h.

### **6.6.2.9 #define XML\_OK\_FRAG XML\_OK\_EOB**

Maintained for backward compatibility.

Definition at line 54 of file rtXmlErrCodes.h.

# Chapter 7

## Data Structure Documentation

### 7.1 OSIntegerFmt Struct Reference

#### Data Fields

- OSINT8 `integerMaxDigits`
- OSBOOL `signPresent`

#### 7.1.1 Detailed Description

Definition at line 274 of file `osrxml.h`.

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

- [osrxml.h](#)

## 7.2 OSXMLCtxtInfo Struct Reference

### Data Fields

- OSFreeCtxtAppInfoPtr **pFreeFunc**
- OSResetCtxtAppInfoPtr **pResetFunc**
- OSUTF8CHAR \* **schemaLocation**
- OSUTF8CHAR \* **noNSSchemaLoc**
- OSUTF8CHAR \* **xsiTypeAttr**
- OSXMLEncoding **encoding**
- OSRTDList **namespaceList**
- OSRTDList **encodedNSList**
- OSRTDList **sortedAttrList**
- OSXMLNSPfxLinkStack **nsPfxLinkStack**
- OSXMLNSURITable **nsURITable**
- OSRTMEMBUF **memBuf**
- OSINT32 **mSaxLevel**
- OSINT32 **mSkipLevel**
- OSUINT32 **maxSaxErrors**
- OSUINT32 **errorsCnt**
- OSUINT8 **indent**
- OSBOOL **mbCdataProcessed**
- char **indentChar**
- OSUINT8 **soapVersion**
- [OSXMLFacets](#) **facets**
- const OSUTF8CHAR \* **encodingStr**
- OSXMLBOM **byteOrderMark**
- struct OSXMLReader \* **pXmlPPReader**
- OSRTBuffer **savedBuffer**
- OSRTFLAGS **savedFlags**
- OSOCTET \* **attrsBuff**
- OSSIZE **attrsBuffSize**
- OSSIZE **attrStartPos**

### 7.2.1 Detailed Description

Definition at line 208 of file `osrxml.h`.

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

- [osrxml.h](#)

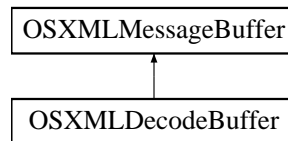


## 7.3 OSXMLDecodeBuffer Class Reference

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

```
#include <OSXMLDecodeBuffer.h>
```

Inheritance diagram for OSXMLDecodeBuffer:



### Public Member Functions

- [OSXMLDecodeBuffer](#) (const char \*xmlFile)  
*This version of the [OSXMLDecodeBuffer](#) constructor takes a name of a file that contains XML data to be decoded and constructs a buffer.*
- [OSXMLDecodeBuffer](#) (const OSOCTET \*msgbuf, size\_t bufsiz)  
*This version of the [OSXMLDecodeBuffer](#) constructor takes parameters describing a message in memory to be decoded and constructs a buffer.*
- [OSXMLDecodeBuffer](#) (OSRTInputStream &inputStream)  
*This version of the [OSXMLDecodeBuffer](#) constructor takes a reference to the OSInputStream object.*
- EXTXMLMETHOD int [decodeXML](#) (OSXMLReaderClass \*pReader)  
*This method decodes an XML message associated with this buffer.*
- virtual EXTXMLMETHOD int [init](#) ()  
*This method initializes the decode message buffer.*
- EXTXMLMETHOD OSBOOL [isWellFormed](#) ()  
*This method determines if an XML fragment is well-formed.*
- EXTXMLMETHOD int [parseElementName](#) (OSUTF8CHAR \*\*ppName)  
*This method parses the initial tag from an XML message.*
- EXTXMLMETHOD int [parseElemQName](#) (OSXMLQName \*pQName)  
*This method parses the initial tag from an XML message.*
- EXTXMLMETHOD OSUINT32 [setMaxErrors](#) (OSUINT32 maxErrors)  
*This method sets the maximum number of errors returned by the SAX parser.*
- virtual OSBOOL [isA](#) (Type bufferType)  
*This is a virtual method that must be overridden by derived classes to allow identification of the class.*

## Protected Attributes

- OSRTInputStream \* [mpInputStream](#)  
*Input source for message to be decoded.*
- OSBOOL [mbOwnStream](#)  
*This is set to true if this object creates the underlying stream object.*

### 7.3.1 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 an input buffer or stream containing a message to be decoded.

Note that the XML decode buffer object does not take a message buffer argument because buffer management is handled by the XML parser.

Definition at line 45 of file OSXMLDecodeBuffer.h.

### 7.3.2 Constructor & Destructor Documentation

#### 7.3.2.1 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.

#### 7.3.2.2 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.

#### 7.3.2.3 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

## 7.3.3 Member Function Documentation

### 7.3.3.1 EXTXMLMETHOD int OSXMLDecodeBuffer::decodeXML (OSXMLReaderClass \* *pReader*)

This method decodes an XML message associated with this buffer.

#### Returns

stat Status of the operation. Possible values are 0 if successful or one of the negative error status codes defined in Appendix A of the C/C++ runtime Common Functions Reference Manual.

#### Parameters

*pReader* Pointer to OSXMLReaderClass object.

#### Returns

Completion status.

### 7.3.3.2 virtual EXTXMLMETHOD int OSXMLDecodeBuffer::init () [virtual]

This method initializes the decode message buffer.

#### Returns

Completion status of operation:

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

### 7.3.3.3 virtual OSBOOL OSXMLDecodeBuffer::isA (Type *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.

Definition at line 169 of file OSXMLDecodeBuffer.h.

### 7.3.3.4 EXTXMLMETHOD OSBOOL OSXMLDecodeBuffer::isWellFormed ()

This method determines if an XML fragment is well-formed.

The stream is reset to the start position following the test.

#### Returns

Boolean result true if fragment well-formed; false otherwise.

### 7.3.3.5 EXTXMLMETHOD int OSXMLDecodeBuffer::parseElementName (OSUTF8CHAR \*\* ppName)

This method parses the initial tag from an XML message.

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

#### 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.

### 7.3.3.6 EXTXMLMETHOD 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.

### 7.3.3.7 EXTXMLMETHOD 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.

## 7.3.4 Field Documentation

### 7.3.4.1 OSBOOL OSXMLDecodeBuffer::mbOwnStream [protected]

This is set to true if this object creates the underlying stream object.

In this case, the stream will be deleted in the object's destructor.

Definition at line 57 of file `OSXMLDecodeBuffer.h`.

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

- [OSXMLDecodeBuffer.h](#)

## 7.4 OSXMLElemIDRec Struct Reference

### Data Fields

- [OSXMLElemDescr](#) descr
- OSUINT16 id

### 7.4.1 Detailed Description

Definition at line 127 of file osrtxml.h.

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

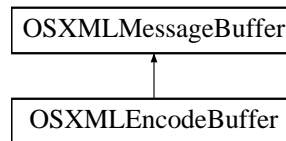
- [osrtxml.h](#)

## 7.5 OSXMLEncodeBuffer Class Reference

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

```
#include <OSXMLEncodeBuffer.h>
```

Inheritance diagram for OSXMLEncodeBuffer:



### Public Member Functions

- EXTXMLMETHOD [OSXMLEncodeBuffer](#) ()  
*Default constructor.*
- EXTXMLMETHOD [OSXMLEncodeBuffer](#) (OSOCKET \*pMsgBuf, size\_t msgBufLen)  
*This constructor allows a static message buffer to be specified to receive the encoded message.*
- int [addXMLHeader](#) (const OSUTF8CHAR \*version=OSUTF8("1.0"), const OSUTF8CHAR \*encoding=OSUTF8(OSXMLHDRUTF8), OSBOOL newLine=TRUE)  
*This method adds XML header text to the output buffer with the given version number and encoding attributes.*
- EXTXMLMETHOD int [addXMLText](#) (const OSUTF8CHAR \*text)  
*This method adds encoded XML text to the encode buffer.*
- virtual size\_t [getMsgLen](#) ()  
*This method returns the length of a previously encoded XML message.*
- virtual EXTXMLMETHOD int [init](#) ()  
*This method reinitializes the encode buffer to allow a new message to be encoded.*
- virtual OSBOOL [isA](#) (Type bufferType)  
*This is a virtual method that must be overridden by derived classes to allow identification of the class.*
- void [nullTerminate](#) ()  
*This method adds a null-terminator character (") at the current buffer position.*
- EXTXMLMETHOD void [setFragment](#) (OSBOOL value=TRUE)  
*This method sets a flag indicating that the data is to be encoded as an XML fragment instead of as a complete XML document (i.e.*
- virtual EXTXMLMETHOD long [write](#) (const char \*filename)  
*This method writes the encoded message to the given file.*
- virtual EXTXMLMETHOD long [write](#) (FILE \*fp)  
*This version of the write method writes to a file that is specified by a FILE pointer.*

## Protected Member Functions

- **OSXMLEncodeBuffer** (OSRTContext \*pContext)

### 7.5.1 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.

Definition at line 38 of file OSXMLEncodeBuffer.h.

### 7.5.2 Constructor & Destructor Documentation

#### 7.5.2.1 EXTMLMETHOD 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.

### 7.5.3 Member Function Documentation

#### 7.5.3.1 int OSXMLEncodeBuffer::addXMLHeader (const OSUTF8CHAR \* version = OSUTF8 ("1.0"), const OSUTF8CHAR \* encoding = OSUTF8 (OSXMLHDRUTF8), OSBOOL newLine = TRUE)

This method adds XML header text to the output buffer with the given version number and encoding attributes.

#### Parameters

- version* XML version (default is 1.0)  
*encoding* Character encoding (default is UTF-8)  
*newLine* Add newline char at end of header

#### Returns

Zero if success or negative error code.

#### 7.5.3.2 EXTMLMETHOD int OSXMLEncodeBuffer::addXMLText (const OSUTF8CHAR \* text)

This method adds encoded XML text to the encode buffer.

It is assumed that the user has already processed the text to do character escaping, etc.. The text is copied directly to the buffer as-is.

#### Parameters

- text* Encoded XML text to be added to the buffer.

#### Returns

Zero if success or negative error code.

### 7.5.3.3 `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.

Definition at line 90 of file OSXMLEncodeBuffer.h.

### 7.5.3.4 `virtual EXTXMLMETHOD 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.

### 7.5.3.5 `virtual OSBOOL OSXMLEncodeBuffer::isA (Type 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.

Definition at line 118 of file OSXMLEncodeBuffer.h.

### 7.5.3.6 `EXTXMLMETHOD void OSXMLEncodeBuffer::setFragment (OSBOOL value = TRUE)`

This method sets a flag indicating that the data is to be encoded as an XML fragment instead of as a complete XML document (i.e.

an XML header will not be added).

### 7.5.3.7 `virtual EXTXMLMETHOD 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.



### 7.5.3.8 virtual EXXMLMETHOD 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:

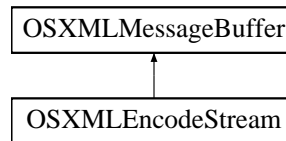
- [OSXMLEncodeBuffer.h](#)

## 7.6 OSXMLEncodeStream Class Reference

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

```
#include <OSXMLEncodeStream.h>
```

Inheritance diagram for OSXMLEncodeStream:



### Public Member Functions

- EXTXMLMETHOD [OSXMLEncodeStream](#) (OSRTOutputStream &outputStream)  
*This version of the [OSXMLEncodeStream](#) constructor takes a reference to the OSOutputStream object.*
- [OSXMLEncodeStream](#) (OSRTOutputStream \*pOutputStream, OSBOOL ownStream=TRUE)  
*This version of the [OSXMLEncodeStream](#) constructor takes a pointer to the OSRTOutputStream object.*
- EXTXMLMETHOD int [encodeAttr](#) (const OSUTF8CHAR \*name, const OSUTF8CHAR \*value)  
*This function encodes an attribute in which the name and value are given as null-terminated UTF-8 strings.*
- EXTXMLMETHOD int [encodeText](#) (const OSUTF8CHAR \*value)  
*This method encodes XML textual content.*
- EXTXMLMETHOD int [endDocument](#) ()  
*This method ends an XML document by flushing any remaining data to the stream.*
- EXTXMLMETHOD int [endElement](#) (const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS=0)  
*This method encodes an end element tag value (</elemName>).*
- virtual EXTXMLMETHOD int [init](#) ()  
*This method reinitializes the encode stream to allow a new message to be encoded.*
- virtual OSBOOL [isA](#) (Type bufferType)  
*This is a virtual method that must be overridden by derived classes to allow identification of the class.*
- virtual const OSOCTET \* [getMsgPtr](#) ()  
*This is a virtual method that must be overridden by derived classes to allow access to the stored message.*
- OSRTOutputStream \* [getStream](#) () const  
*This method returns the output stream associated with the object.*
- EXTXMLMETHOD int [startDocument](#) ()  
*This method writes information to start an XML document to the encode stream.*
- EXTXMLMETHOD int [startElement](#) (const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS=0, OSRTDList \*pNSAttrs=0, OSBOOL terminate=FALSE)

*This method writes information to start an XML element to the encode stream.*

- EXTXMLMETHOD int [termStartElement](#) ()

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

## Protected Attributes

- OSRTOutputStream \* [mpStream](#)

*A pointer to an OSRTOutputStream object.*

- OSBOOL [mbOwnStream](#)

*TRUE if the OSXMLEncodeStream object will close and free the stream in the destructor.*

- OSCTXT \* [mpCtxt](#)

*Internal pointer to the context structure associated with the stream for making C function calls.*

## 7.6.1 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.

Definition at line 40 of file OSXMLEncodeStream.h.

## 7.6.2 Constructor & Destructor Documentation

### 7.6.2.1 EXTXMLMETHOD OSXMLEncodeStream::OSXMLEncodeStream (OSRTOutputStream & *outputStream*)

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

The stream is assumed to have been previously initialized.

#### Parameters

*outputStream* reference to the OSOutputStream object

### 7.6.2.2 OSXMLEncodeStream::OSXMLEncodeStream (OSRTOutputStream \* *pOutputStream*, OSBOOL *ownStream* = TRUE)

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

The stream is assumed to have been previously initialized. If *ownStream* is set to TRUE, then stream will be closed and freed in the destructor.

#### Parameters

*pOutputStream* reference to the OSOutputStream object

*ownStream* set ownership for the passed stream object.

## 7.6.3 Member Function Documentation

### 7.6.3.1 **EXTXMLMETHOD** `int OSXMLEncodeStream::encodeAttr (const OSUTF8CHAR * name, const OSUTF8CHAR * value)`

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

#### Parameters

*name* Attribute name.

*value* UTF-8 string value to be encoded.

#### Returns

Completion status of operation:

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

### 7.6.3.2 **EXTXMLMETHOD** `int OSXMLEncodeStream::encodeText (const OSUTF8CHAR * value)`

This method encodes XML textual content.

XML metadata characters such as '<' are escaped. The input value is specified in UTF-8 character format but may be transformed if a different character encoding is enabled.

#### Parameters

*value* UTF-8 string value to be encoded.

#### Returns

Completion status of operation:

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

### 7.6.3.3 **EXTXMLMETHOD** `int OSXMLEncodeStream::endElement (const OSUTF8CHAR * elemName, OSXMLNamespace * pNS = 0)`

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

#### Parameters

*elemName* XML element name.

*pNS* XML namespace information (prefix and URI).

#### Returns

Completion status of operation:

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

#### 7.6.3.4 **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.

Definition at line 154 of file OSXMLEncodeStream.h.

#### 7.6.3.5 **OSRTOutputStream\* OSXMLEncodeStream::getStream () const [inline]**

This method returns the output stream associated with the object.

##### **Returns**

A pointer to the output stream.

Definition at line 161 of file OSXMLEncodeStream.h.

#### 7.6.3.6 **virtual EXTXMLMETHOD int OSXMLEncodeStream::init () [virtual]**

This method reinitializes the encode stream to allow a new message to be encoded.

This makes it possible to reuse one stream object in a loop to encode multiple messages.

#### 7.6.3.7 **virtual OSBOOL OSXMLEncodeStream::isA (Type *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*.

Definition at line 143 of file OSXMLEncodeStream.h.

#### 7.6.3.8 **EXTXMLMETHOD int OSXMLEncodeStream::startDocument ()**

This method writes information to start an XML document to the encode stream.

This includes the XML header declaration.

### 7.6.3.9 EXTXMLMETHOD int OSXMLEncodeStream::startElement (const OSUTF8CHAR \* *elemName*, OSXMLNamespace \* *pNS* = 0, OSRTDList \* *pNSAttrs* = 0, OSBOOL *terminate* = FALSE)

This method writes information to start an XML element to the encode stream.

It can leave the element open so that attributes can be added.

#### Parameters

*elemName* XML element name.

*pNS* XML namespace information (prefix and URI). If the prefix is NULL, this method will search the context's namespace stack for a prefix to use.

*pNSAttrs* List of namespace attributes to be added to element.

*terminate* Add closing '>' character.

#### Returns

Completion status of operation:

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

### 7.6.3.10 EXTXMLMETHOD int OSXMLEncodeStream::termStartElement ()

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

It will also add XSI attributes to the element. Note that it is important to use this method to terminate the element rather than writing a closing angle bracket text to the stream directly due to the way state is maintained in the context.

#### Returns

Completion status of operation:

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

## 7.6.4 Field Documentation

### 7.6.4.1 OSBOOL OSXMLEncodeStream::mbOwnStream [protected]

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

Definition at line 47 of file OSXMLEncodeStream.h.

### 7.6.4.2 OSCTXT\* OSXMLEncodeStream::mpCtxt [protected]

Internal pointer to the context structure associated with the stream for making C function calls.

Definition at line 51 of file OSXMLEncodeStream.h.

### 7.6.4.3 OSRTOutputStream\* OSXMLEncodeStream::mpStream [protected]

A pointer to an OSRTOutputStream object.

Definition at line 43 of file OSXMLEncodeStream.h.

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

- [OSXMLEncodeStream.h](#)

## 7.7 OSXMLFacets Struct Reference

### Data Fields

- int **totalDigits**
- int **fractionDigits**

### 7.7.1 Detailed Description

Definition at line 103 of file osrxml.h.

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

- [osrxml.h](#)



## 7.8 OSXMLGroupDesc Struct Reference

[OSXMLGroupDesc](#) describes how entries in an [OSXMLElemIDRec](#) array make up a group.

```
#include <osrtxml.h>
```

### Data Fields

- int **row**
- int **num**
- int **anyCase**

### 7.8.1 Detailed Description

[OSXMLGroupDesc](#) describes how entries in an [OSXMLElemIDRec](#) array make up a group. Here, "group" means a set of elements, any of which may be matched next. This does not correspond directly to an XSD group.

For example, if elementA is optional and followed by non-optional elementB, then there will be a group that contains both elements. There will also be a group that contains only elementB; this will be the group of interest after elementA is matched.

Definition at line 142 of file osrtxml.h.

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

- [osrtxml.h](#)

## 7.9 OSXMLItemDescr Struct Reference

### Data Fields

- [OSXMLStrFragment](#) `localName`
- OSINT16 `nsidx`

### 7.9.1 Detailed Description

Definition at line 119 of file `osrxml.h`.

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

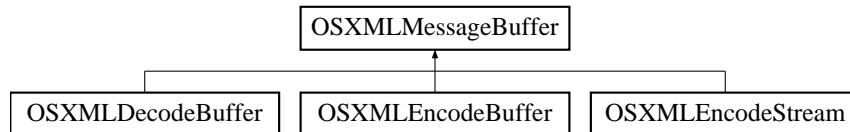
- [osrxml.h](#)

## 7.10 OSXMLMessageBuffer Class Reference

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

```
#include <OSXMLMessageBuffer.h>
```

Inheritance diagram for OSXMLMessageBuffer:



### Public Member Functions

- virtual EXTXMLMETHOD void \* [getAppInfo](#) ()  
*The getAppInfo method returns the pointer to application context information.*
- EXTXMLMETHOD int [getIndent](#) ()  
*This method returns current XML output indent value.*
- EXTXMLMETHOD int [getIndentChar](#) ()  
*This method returns current XML output indent character value (default is space).*
- EXTXMLMETHOD OSBOOL [getWriteBOM](#) ()  
*This function returns whether writing the Unicode BOM is currently enabled or disabled.*
- virtual EXTXMLMETHOD void [setNamespace](#) (const OSUTF8CHAR \*prefix, const OSUTF8CHAR \*uri, OSRTDList \*pNSAttrs=0)  
*This method sets a namespace in the context namespace list.*
- virtual EXTXMLMETHOD void [setAppInfo](#) (void \*pXMLInfo)  
*This method sets application specific context information within the common context structure.*
- EXTXMLMETHOD void [setFormatting](#) (OSBOOL doFormatting)  
*This method sets XML output formatting to the given value.*
- EXTXMLMETHOD void [setIndent](#) (OSUINT8 indent)  
*This method sets XML output indent to the given value.*
- EXTXMLMETHOD void [setIndentChar](#) (char indentChar)  
*This method sets XML output indent character to the given value.*
- EXTXMLMETHOD void [setWriteBOM](#) (OSBOOL write)  
*This method sets whether to write the Unicode byte order mark before the XML header.*

## Protected Member Functions

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

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

### 7.10.1 Detailed Description

The XML message buffer class is derived from the OSMessageBuffer base class. It is the base class for the [OSXML-LEncodeBuffer](#) and [OSXMLDecodeBuffer](#) classes. It contains variables and methods specific to encoding or decoding XML messages. It is used to manage the buffer into which a message is to be encoded or decoded.

Definition at line 42 of file OSXMLMessageBuffer.h.

### 7.10.2 Constructor & Destructor Documentation

- 7.10.2.1 EXTXMLMETHOD [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.

### 7.10.3 Member Function Documentation

- 7.10.3.1 EXTXMLMETHOD [int OSXMLMessageBuffer::getIndent](#) ()

This method returns current XML output indent value.

#### Returns

Current indent value ( $\geq 0$ ) if OK, negative status code if error.

- 7.10.3.2 EXTXMLMETHOD [int OSXMLMessageBuffer::getIndentChar](#) ()

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

#### Returns

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

- 7.10.3.3 EXTXMLMETHOD [OSBOOL OSXMLMessageBuffer::getWriteBOM](#) ()

This function returns whether writing the Unicode BOM is currently enabled or disabled.

#### Returns

TRUE if writing BOM is enabled, FALSE otherwise.

**7.10.3.4 virtual EXTXMLMETHOD 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.

**7.10.3.5 EXTXMLMETHOD void OSXMLMessageBuffer::setFormatting (OSBOOL *doFormatting*)**

This method sets XML output formatting to the given value.

If TRUE (the default), the XML document is formatted with indentation and newlines. If FALSE, all whitespace between elements is suppressed. Turning formatting off can provide more compressed documents and also a more canonical representation which is important for security applications.

**Parameters**

*doFormatting* Boolean value indicating if formatting is to be done

**Returns**

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

**7.10.3.6 EXTXMLMETHOD 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.

**7.10.3.7 EXTXMLMETHOD void OSXMLMessageBuffer::setIndentChar (char *indentChar*)**

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

**Parameters**

*indentChar* Indent character. Default is space.

**7.10.3.8 virtual EXTXMLMETHOD void OSXMLMessageBuffer::setNamespace (const OSUTF8CHAR \*  
*prefix*, const OSUTF8CHAR \* *uri*, OSRTDList \* *pNSAttrs* = 0) [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

*pNSAttrs* Namespace list to which namespace is to be added

### 7.10.3.9 EXTMLMETHOD void OSXMLMessageBuffer::setWriteBOM (OSBOOL *write*)

This method sets whether to write the Unicode byte order mark before the XML header.

## Parameters

*write* TRUE if BOM should be written, FALSE otherwise.

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

- [OSXMLMessageBuffer.h](#)

## 7.11 OSXMLNameFragments Struct Reference

### Data Fields

- [OSXMLStrFragment](#) **mQName**
- [OSXMLStrFragment](#) **mLocalName**
- [OSXMLStrFragment](#) **mPrefix**

### 7.11.1 Detailed Description

Definition at line 113 of file osrxml.h.

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

- [osrxml.h](#)

## 7.12 OSXMLQName Struct Reference

### Data Fields

- const OSUTF8CHAR \* **nsPrefix**
- const OSUTF8CHAR \* **ncName**

### 7.12.1 Detailed Description

Definition at line 267 of file osrxml.h.

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

- [osrxml.h](#)



## 7.13 OSXMLSortedAttrOffset Struct Reference

### Data Fields

- OSIZE **offset**
- OSIZE **length**
- OSIZE **prefixLength**
- OSIZE **nameLength**

### 7.13.1 Detailed Description

Definition at line 282 of file osrxml.h.

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

- [osrxml.h](#)

## 7.14 OSXMLStrFragment Struct Reference

### Data Fields

- const OSUTF8CHAR \* **value**
- OSSIZE **length**

### 7.14.1 Detailed Description

Definition at line 108 of file osrxml.h.

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

- [osrxml.h](#)

## 7.15 OSXSDAnyType Struct Reference

### Data Fields

- OSXMLSTRING **value**
- OSRTDList **attrs**

### 7.15.1 Detailed Description

Definition at line 160 of file osrxml.h.

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

- [osrxml.h](#)

# Chapter 8

## File Documentation

### 8.1 osrtxml.h File Reference

XML low-level C encode/decode functions.

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

#### Data Structures

- struct [OSXMLFacets](#)
- struct [OSXMLStrFragment](#)
- struct [OSXMLNameFragments](#)
- struct [OSXMLItemDescr](#)
- struct [OSXMLElemIDRec](#)
- struct [OSXMLGroupDesc](#)

*OSXMLGroupDesc* describes how entries in an *OSXMLElemIDRec* array make up a group.

- struct [OSXSDAnyType](#)
- struct [OSXMLCtxtInfo](#)
- struct [OSXMLQName](#)
- struct [OSIntegerFmt](#)
- struct [OSXMLSortedAttrOffset](#)

#### Defines

- #define [OSXMLNS12](#)
- #define [OSUPCASE](#) 0x00008000

- #define **OSTERMSTART** 0x00004000
- #define **OEMPTYELEM** 0x00002000
- #define **OSQUALATTR** 0x00001000
- #define **OSXMLFRAG** 0x00000800
- #define **OSXMLNSSET** 0x00000400
- #define **OSXMLC14N** 0x00000200
- #define **OSXSIATTR** 0x00000100
- #define **OSXMLNOCMPNS** 0x00000080
- #define **OSXSINIL** 0x00000040
- #define **OSXMLNOBLANKS** 0x00000020
- #define **OSHASDEFAULT** 0x00000010
- #define **OSASN1XER** 0x00000008
- #define **OSXMLFRAGSEQUAL**(frag1, frag2) (frag1.length==frag2.length && !memcmp(frag1.value,frag2.value,frag1.length))
- #define **OSXMLQNAMEEQUALS**(xnamefrag, qnametext)
- #define **OSXMLSETUTF8DECPTR**(pctxt, str)
- #define **IS\_XMLNSATTR**(name)
- #define **IS\_XSIATTR**(name)
- #define **OSXMLINDENT** 3
- #define **rtXmlErrAddStrParm** rtxErrAddStrParm
- #define **rtXPrintNSAttr**(name, data) rtXPrintNSAttr(name,&data)
- #define **rtXmlFinalizeMemBuf**(pMemBuf)
- #define **rtXmlGetEncBufPtr**(pctxt) (pctxt)->buffer.data  
*This macro returns the start address of the encoded XML message.*
- #define **rtXmlGetEncBufLen**(pctxt) (pctxt)->buffer.byteIndex  
*This macro returns the length of the encoded XML message.*
- #define **OSXMLREALENC\_OBJSYS** 0x1F
- #define **OSXMLREALENC\_BXER** 0x10
- #define **OSXMLREALENC\_EXERMODS** 0x1B
- #define **OSXMLREALENC\_EXERDECIMAL** 0x03

## Typedefs

- typedef struct **OSXMLFacets** **OSXMLFacets**
- typedef struct **OSXMLItemDescr** **OSXMLItemDescr**
- typedef **OSXMLItemDescr** **OSXMLAttrDescr**
- typedef **OSXMLItemDescr** **OSXMLElemDescr**
- typedef struct **OSXMLElemIDRec** **OSXMLElemIDRec**
- typedef struct **OSXMLGroupDesc** **OSXMLGroupDesc**  
*OSXMLGroupDesc describes how entries in an OSXMLElemIDRec array make up a group.*
- typedef struct **OSXSDAnyType** **OSXSDAnyType**
- typedef struct **OSXMLQName** **OSXMLQName**
- typedef struct **OSIntegerFmt** **OSIntegerFmt**

## Enumerations

- enum **OSXMLEncoding** {  
    **OSXMLUTF8**, **OSXMLUTF16**, **OSXMLUTF16BE**, **OSXMLUTF16LE**,  
    **OSXMLLATIN1** }
- enum **OSXMLSOAPMsgType** { **OSSOAPNONE**, **OSSOAPHEADER**, **OSSOAPBODY**, **OSSOAPFAULT** }
- enum **OSXMLBOM** {  
    **OSXMLBOM\_NO\_BOM**, **OSXMLBOM\_UTF32\_BE**, **OSXMLBOM\_UTF32\_LE**, **OSXMLBOM\_UTF16\_BE**,  
    **OSXMLBOM\_UTF16\_LE**, **OSXMLBOM\_UTF8**, **OSXMLBOM\_CHECK** }
- enum **OSXMLNsIndex** {  
    **OSXMLNSI\_UNQUALIFIED** = 0, **OSXMLNSI\_UNKNOWN** = -1, **OSXMLNSI\_UNCHECKED** = -2,  
    **OSXMLNSI\_XSI** = -3,  
    **OSXMLNSI\_XMLNS** = -4, **OSXMLNSI\_XML** = -5, **OSXMLNSI\_SOAP\_ENVELOPE** = -6,  
    **OSXMLNSI\_XSD** = -7 }
- enum **OSXMLREALEncoding** { **OSXMLREALOBJSYS**, **OSXMLREALBXR**, **OSXMLREALEXERMODS**, **OSXMLREALEXERDEC** }
- enum **OSXMLState** {  
    **OSXMLINIT**, **OSXMLHEADER**, **OSXMLSTART**, **OSXMLATTR**,  
    **OSXMLDATA**, **OSXMLEND**, **OSXMLCOMMENT** }
- enum **OSXMLWhiteSpaceMode** { **OSXMLWSM\_PRESERVE** = 0, **OSXMLWSM\_REPLACE**, **OSXMLWSM\_COLLAPSE** }

*Whitespace treatment options.*

## Functions

- EXTERNXML int **rtXmlInitContext** (OSCTXT \*pctxt)  
*This function initializes a context variable for XML encoding or decoding.*
- EXTERNXML int **rtXmlInitContextUsingKey** (OSCTXT \*pctxt, const OSOCTET \*key, OSSIZE keylen)  
*This function initializes a context using a run-time key.*
- EXTERNXML int **rtXmlInitCtxtAppInfo** (OSCTXT \*pctxt)  
*This function initializes the XML application info section of the given context.*
- EXTERNXML int **rtXmlCreateFileInputSource** (OSCTXT \*pctxt, const char \*filepath)  
*This function creates an XML document file input source.*
- EXTERNXML OSBOOL **rtXmlCmpQName** (const OSUTF8CHAR \*qname1, const OSUTF8CHAR \*name2, const OSUTF8CHAR \*nsPrefix2)
- EXTERNXML int **rtXmlGetBase64StrDecodedLen** (const OSUTF8CHAR \*inpdata, OSSIZE srcDataSize, OSSIZE \*pNumOcts, OSSIZE \*pSrcDataLen)
- EXTERNXML void **rtXmlMemFreeAnyAttrs** (OSCTXT \*pctxt, OSRTDList \*pAnyAttrList)  
*This function frees a list of anyAttribute that is a member of **OSXSDAnyType** structure.*
- EXTERNXML int **rtXmlDecBase64Binary** (OSRTMEMBUF \*pMemBuf, const OSUTF8CHAR \*inpdata, OSSIZE length)

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

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

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

- EXTERNXML int [rtXmlDecBase64StrValue](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, OS-SIZE bufSize, OSSIZE srcDataLen)

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

- EXTERNXML int [rtXmlDecBigInt](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*ppvalue)

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

- EXTERNXML int [rtXmlDecBool](#) (OSCTXT \*pctxt, OSBOOL \*pvalue)

*This function decodes a variable of the boolean type.*

- EXTERNXML int [rtXmlDecDate](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)

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

- EXTERNXML int [rtXmlDecTime](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)

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

- EXTERNXML int [rtXmlDecDateTime](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)

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

- EXTERNXML int [rtXmlDecDecimal](#) (OSCTXT \*pctxt, OSREAL \*pvalue)

*This function decodes the contents of a decimal data type.*

- EXTERNXML int [rtXmlDecDouble](#) (OSCTXT \*pctxt, OSREAL \*pvalue)

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

- EXTERNXML int [rtXmlDecDynBase64Str](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)

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

- EXTERNXML int [rtXmlDecDynHexStr](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)

*This function decodes a contents of a hexBinary string.*

- EXTERNXML int [rtXmlDecEmptyElement](#) (OSCTXT \*pctxt)

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

- EXTERNXML int [rtXmlDecUTF8Str](#) (OSCTXT \*pctxt, OSUTF8CHAR \*outdata, OSSIZE max\_len)

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

- EXTERNXML int [rtXmlDecDynUTF8Str](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*outdata)

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

- EXTERNXML int [rtXmlDecHexBinary](#) (OSRTMEMBUF \*pMemBuf, const OSUTF8CHAR \*inpdata, OS-SIZE length)

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

- EXTERNXML int [rtXmlDecHexStr](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, OSINT32 bufsize)  
*This function decodes the contents of a hexBinary string into a static memory structure.*
- EXTERNXML int [rtXmlDecHexStrValue](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*const inpdata, OSSIZE nbytes, OSOCTET \*pvalue, OSUINT32 \*pnbits, OSINT32 bufsize)
- EXTERNXML int [rtXmlDecGYear](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gYear' type.*
- EXTERNXML int [rtXmlDecGYearMonth](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gYearMonth' type.*
- EXTERNXML int [rtXmlDecGMonth](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gMonth' type.*
- EXTERNXML int [rtXmlDecGMonthDay](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gMonthDay' type.*
- EXTERNXML int [rtXmlDecGDay](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gDay' type.*
- EXTERNXML int [rtXmlDecInt](#) (OSCTXT \*pctxt, OSINT32 \*pvalue)  
*This function decodes the contents of a 32-bit integer data type.*
- EXTERNXML int [rtXmlDecInt8](#) (OSCTXT \*pctxt, OSINT8 \*pvalue)  
*This function decodes the contents of an 8-bit integer data type (i.e.*
- EXTERNXML int [rtXmlDecInt16](#) (OSCTXT \*pctxt, OSINT16 \*pvalue)  
*This function decodes the contents of a 16-bit integer data type.*
- EXTERNXML int [rtXmlDecInt64](#) (OSCTXT \*pctxt, OSINT64 \*pvalue)  
*This function decodes the contents of a 64-bit integer data type.*
- EXTERNXML int [rtXmlDecUInt](#) (OSCTXT \*pctxt, OSUINT32 \*pvalue)  
*This function decodes the contents of an unsigned 32-bit integer data type.*
- EXTERNXML int [rtXmlDecUInt8](#) (OSCTXT \*pctxt, OSUINT8 \*pvalue)  
*This function decodes the contents of an unsigned 8-bit integer data type (i.e.*
- EXTERNXML int [rtXmlDecUInt16](#) (OSCTXT \*pctxt, OSUINT16 \*pvalue)  
*This function decodes the contents of an unsigned 16-bit integer data type.*
- EXTERNXML int [rtXmlDecUInt64](#) (OSCTXT \*pctxt, OSUINT64 \*pvalue)  
*This function decodes the contents of an unsigned 64-bit integer data type.*
- EXTERNXML int [rtXmlDecNSAttr](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*attrName, const OSUTF8CHAR \*attrValue, OSRTDList \*pNSAttrs, const OSUTF8CHAR \*nsTable[ ], OSUINT32 nsTableRowCount)  
*This function decodes an XML namespace attribute (xmlns).*
- EXTERNXML const OSUTF8CHAR \* [rtXmlDecQName](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*qname, const OSUTF8CHAR \*\*prefix)



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

- EXTERNXML int [rtXmlDecXSIAttr](#) (OSCTXT \*pctx, const OSUTF8CHAR \*attrName, const OSUTF8CHAR \*attrValue)

*This function decodes XML schema instance (XSI) attribute.*

- EXTERNXML int [rtXmlDecXSIAttrs](#) (OSCTXT \*pctx, const OSUTF8CHAR \*const \*attrs, const char \*typeName)

*This function decodes XML schema instance (XSI) attributes.*

- EXTERNXML int [rtXmlDecXmlStr](#) (OSCTXT \*pctx, OSXMLSTRING \*outdata)

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

- EXTERNXML int [rtXmlParseElementName](#) (OSCTXT \*pctx, OSUTF8CHAR \*\*ppName)

*This function parses the initial tag from an XML message.*

- EXTERNXML int [rtXmlParseElemQName](#) (OSCTXT \*pctx, OSXMLQName \*pQName)

*This function parses the initial tag from an XML message.*

- EXTERNXML int [rtXmlEncAny](#) (OSCTXT \*pctx, OSXMLSTRING \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

*This function encodes a variable of the XSD any type.*

- EXTERNXML int [rtXmlEncAnyStr](#) (OSCTXT \*pctx, const OSUTF8CHAR \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

- EXTERNXML int [rtXmlEncAnyTypeValue](#) (OSCTXT \*pctx, const OSUTF8CHAR \*pvalue)

*This function encodes a variable of the XSD anyType type.*

- EXTERNXML int [rtXmlEncAnyAttr](#) (OSCTXT \*pctx, OSRTDList \*pAnyAttrList)

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

- EXTERNXML int [rtXmlEncBase64Binary](#) (OSCTXT \*pctx, OSUINT32 noctx, const OSOCTET \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

*This function encodes a variable of the XSD base64Binary type.*

- EXTERNXML int [rtXmlEncBase64BinaryAttr](#) (OSCTXT \*pctx, OSUINT32 noctx, const OSOCTET \*value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)

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

- EXTERNXML int [rtXmlEncBase64StrValue](#) (OSCTXT \*pctx, OSUINT32 noctx, const OSOCTET \*value)

*This function encodes a variable of the XSD base64Binary type.*

- EXTERNXML int [rtXmlEncBigInt](#) (OSCTXT \*pctx, const OSUTF8CHAR \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

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

- EXTERNXML int [rtXmlEncBigIntAttr](#) (OSCTXT \*pctx, const OSUTF8CHAR \*value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)

*This function encodes an XSD integer attribute value.*

- EXTERNXML int [rtXmlEncBigIntValue](#) (OSCTXT \*pctx, const OSUTF8CHAR \*value)

*This function encodes an XSD integer attribute value.*

- EXTERNXML int [rtXmlEncBitString](#) (OSCTXT \*pctx, OSUINT32 nbits, const OSOCTET \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

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

- EXTERNXML int [rtXmlEncBinStrValue](#) (OSCTXT \*pctx, OSUINT32 nbits, const OSOCTET \*data)

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

- EXTERNXML int [rtXmlEncBool](#) (OSCTXT \*pctx, OSBOOL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

*This function encodes a variable of the XSD boolean type.*

- EXTERNXML int [rtXmlEncBoolValue](#) (OSCTXT \*pctx, OSBOOL value)

*This function encodes a variable of the XSD boolean type.*

- EXTERNXML int [rtXmlEncBoolAttr](#) (OSCTXT \*pctx, OSBOOL value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)

*This function encodes an XSD boolean attribute value.*

- EXTERNXML int [rtXmlEncComment](#) (OSCTXT \*pctx, const OSUTF8CHAR \*comment)

*This function encodes an XML comment.*

- EXTERNXML int [rtXmlEncDate](#) (OSCTXT \*pctx, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

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

- EXTERNXML int [rtXmlEncDateValue](#) (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)

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

- EXTERNXML int [rtXmlEncTime](#) (OSCTXT \*pctx, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

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

- EXTERNXML int [rtXmlEncTimeValue](#) (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)

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

- EXTERNXML int [rtXmlEncDateTime](#) (OSCTXT \*pctx, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

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

- EXTERNXML int [rtXmlEncDateTimeValue](#) (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)

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

- EXTERNXML int [rtXmlEncDecimal](#) (OSCTXT \*pctx, OSREAL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSDecimalFmt \*pFmtSpec)

*This function encodes a variable of the XSD decimal type.*

- EXTERNXML int [rtXmlEncDecimalAttr](#) (OSCTXT \*pctx, OSREAL value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen, const OSDecimalFmt \*pFmtSpec)

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

- EXTERNXML int [rtXmlEncDecimalValue](#) (OSCTXT \*pctxt, OSREAL value, const OSDecimalFmt \*pFmtSpec, char \*pDestBuf, OSSIZE destBufSize)  
*This function encodes a value of the XSD decimal type.*
- EXTERNXML int [rtXmlEncDouble](#) (OSCTXT \*pctxt, OSREAL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSDoubleFmt \*pFmtSpec)  
*This function encodes a variable of the XSD double type.*
- EXTERNXML int [rtXmlEncDoubleAttr](#) (OSCTXT \*pctxt, OSREAL value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen, const OSDoubleFmt \*pFmtSpec)  
*This function encodes a variable of the XSD double type as an attribute.*
- EXTERNXML int [rtXmlEncDoubleNormalValue](#) (OSCTXT \*pctxt, OSREAL value, const OSDoubleFmt \*pFmtSpec, int defaultPrecision)  
*This function encodes a normal (not +/-INF or NaN) value of the XSD double or float type.*
- EXTERNXML int [rtXmlEncDoubleValue](#) (OSCTXT \*pctxt, OSREAL value, const OSDoubleFmt \*pFmtSpec, int defaultPrecision)  
*This function encodes a value of the XSD double or float type.*
- EXTERNXML int [rtXmlEncEmptyElement](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs, OSBOOL terminate)  
*This function encodes an empty element tag value (<elemName/>).*
- EXTERNXML int [rtXmlEncEndDocument](#) (OSCTXT \*pctxt)  
*This function adds trailer information and a null terminator at the end of the XML document being encoded.*
- EXTERNXML int [rtXmlEncEndElement](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes an end element tag value (</elemName>).*
- EXTERNXML int [rtXmlEncEndSoapEnv](#) (OSCTXT \*pctxt)  
*This function encodes a SOAP envelope end element tag (<SOAP-ENV:Envelope/>).*
- EXTERNXML int [rtXmlEncEndSoapElems](#) (OSCTXT \*pctxt, OSXMLSOAPMsgType msgtype)  
*This function encodes SOAP end element tags.*
- EXTERNXML int [rtXmlEncFloat](#) (OSCTXT \*pctxt, OSREAL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSDoubleFmt \*pFmtSpec)  
*This function encodes a variable of the XSD float type.*
- EXTERNXML int [rtXmlEncFloatAttr](#) (OSCTXT \*pctxt, OSREAL value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen, const OSDoubleFmt \*pFmtSpec)  
*This function encodes a variable of the XSD float type as an attribute.*
- EXTERNXML int [rtXmlEncGYear](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a numeric gYear element into an XML string representation.*

- EXTERNXML int [rtXmlEncGYearMonth](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a numeric gYearMonth element into an XML string representation.*
- EXTERNXML int [rtXmlEncGMonth](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a numeric gMonth element into an XML string representation.*
- EXTERNXML int [rtXmlEncGMonthDay](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a numeric gMonthDay element into an XML string representation.*
- EXTERNXML int [rtXmlEncGDay](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a numeric gDay element into an XML string representation.*
- EXTERNXML int [rtXmlEncGYearValue](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gYear value into an XML string representation.*
- EXTERNXML int [rtXmlEncGYearMonthValue](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gYearMonth value into an XML string representation.*
- EXTERNXML int [rtXmlEncGMonthValue](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gMonth value into an XML string representation.*
- EXTERNXML int [rtXmlEncGMonthDayValue](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gMonthDay value into an XML string representation.*
- EXTERNXML int [rtXmlEncGDayValue](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gDay value into an XML string representation.*
- EXTERNXML int [rtXmlEncHexBinary](#) (OSCTXT \*pctxt, OSSIZE nocts, const OSOCTET \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD hexBinary type.*
- EXTERNXML int [rtXmlEncHexBinaryAttr](#) (OSCTXT \*pctxt, OSUINT32 nocts, const OSOCTET \*value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)  
*This function encodes a variable of the XSD hexBinary type as an attribute.*
- EXTERNXML int [rtXmlEncHexStrValue](#) (OSCTXT \*pctxt, OSSIZE nocts, const OSOCTET \*data)  
*This function encodes a variable of the XSD hexBinary type.*
- EXTERNXML int [rtXmlEncIndent](#) (OSCTXT \*pctxt)  
*This function adds indentation whitespace to the output stream.*
- EXTERNXML int [rtXmlEncInt](#) (OSCTXT \*pctxt, OSINT32 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD integer type.*
- EXTERNXML int [rtXmlEncIntValue](#) (OSCTXT \*pctxt, OSINT32 value)  
*This function encodes a variable of the XSD integer type.*

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

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

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

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

- EXTERNXML int [rtXmlEncIntPatternValue](#) (OSCTXT \*pctxt, OSINT32 value, const OSUTF8CHAR \*pattern)

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

- EXTERNXML int [rtXmlEncUIntPatternValue](#) (OSCTXT \*pctxt, OSUINT32 value, const OSUTF8CHAR \*pattern)

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

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

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

- EXTERNXML int [rtXmlEncInt64Value](#) (OSCTXT \*pctxt, OSINT64 value)

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

- EXTERNXML int [rtXmlEncInt64PatternValue](#) (OSCTXT \*pctxt, OSINT64 value, const OSUTF8CHAR \*pattern)

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

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

- EXTERNXML int [rtXmlEncNamedBits](#) (OSCTXT \*pctxt, const OSBitMapItem \*pBitMap, OSUINT32 nbits, const OSOCTET \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

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

- EXTERNXML int [rtXmlEncNamedBitsValue](#) (OSCTXT \*pctxt, const OSBitMapItem \*pBitMap, OSUINT32 nbits, const OSOCTET \*pvalue)

- EXTERNXML int [rtXmlEncNSAttrs](#) (OSCTXT \*pctxt, OSRTDList \*pNSAttrs)

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

- EXTERNXML int [rtXmlPrintNSAttrs](#) (const char \*name, const OSRTDList \*data)

*This function prints a list of namespace attributes.*

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

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

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

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

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

*This function encodes the XML header text at the beginning of an XML document.*
- EXTERNXML int **rtXmlEncBOM** (OSCTXT \*pctxt)
 

*This function encodes the Unicode byte order mark header at the start of the document.*
- EXTERNXML int **rtXmlEncStartElement** (OSCTXT \*pctxt, const OSUTF8CHAR \*elemName, OSXML-  
Namespace \*pNS, OSRTDList \*pNSAttrs, OSBOOL terminate)
 

*This function encodes a start element tag value (<elemName>).*
- EXTERNXML int **rtXmlEncStartSoapEnv** (OSCTXT \*pctxt, OSRTDList \*pNSAttrs)
 

*This function encodes a SOAP envelope start element tag.*
- EXTERNXML int **rtXmlEncStartSoapElems** (OSCTXT \*pctxt, OSXMLSOAPMsgType msgtype)
 

*This function encodes a SOAP envelope start element tag and an optional SOAP body or fault tag.*
- EXTERNXML int **rtXmlEncString** (OSCTXT \*pctxt, OSXMLSTRING \*pxmlstr, const OSUTF8CHAR  
\*elemName, OSXMLNamespace \*pNS)
 

*This function encodes a variable of the XSD string type.*
- EXTERNXML int **rtXmlEncStringValue** (OSCTXT \*pctxt, const OSUTF8CHAR \*value)
 

*This function encodes a variable of the XSD string type.*
- EXTERNXML int **rtXmlEncStringValue2** (OSCTXT \*pctxt, const OSUTF8CHAR \*value, OSSIZE value-  
Len)
 

*This function encodes a variable of the XSD string type.*
- EXTERNXML int **rtXmlEncTermStartElement** (OSCTXT \*pctxt)
 

*This function terminates a currently open XML start element by adding either a '>' or '/>' (if empty) terminator.*
- EXTERNXML int **rtXmlEncUnicodeStr** (OSCTXT \*pctxt, const OSUNICHAR \*value, OSSIZE nchars, const  
OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)
 

*This function encodes a Unicode string value.*
- EXTERNXML int **rtXmlEncUTF8Attr** (OSCTXT \*pctxt, const OSUTF8CHAR \*name, const OSUTF8CHAR  
\*value)
 

*This function encodes an attribute in which the name and value are given as a null-terminated UTF-8 strings.*
- EXTERNXML int **rtXmlEncUTF8Attr2** (OSCTXT \*pctxt, const OSUTF8CHAR \*name, OSSIZE nameLen,  
const OSUTF8CHAR \*value, OSSIZE valueLen)
 

*This function encodes an attribute in which the name and value are given as a UTF-8 strings with lengths.*
- EXTERNXML int **rtXmlEncUTF8Str** (OSCTXT \*pctxt, const OSUTF8CHAR \*value, const OSUTF8CHAR  
\*elemName, OSXMLNamespace \*pNS)
 

*This function encodes a UTF-8 string value.*
- EXTERNXML int **rtXmlEncUInt** (OSCTXT \*pctxt, OSUINT32 value, const OSUTF8CHAR \*elemName, OS-  
XMLNamespace \*pNS)
 

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

- EXTERNXML int [rtXmlEncUIntValue](#) (OSCTXT \*pctxt, OSUINT32 value)  
*This function encodes a variable of the XSD unsigned integer type.*
- EXTERNXML int [rtXmlEncUIntAttr](#) (OSCTXT \*pctxt, OSUINT32 value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)  
*This function encodes a variable of the XSD unsigned integer type as an attribute (name="value").*
- EXTERNXML int [rtXmlEncUInt64](#) (OSCTXT \*pctxt, OSUINT64 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD integer type.*
- EXTERNXML int [rtXmlEncUInt64Pattern](#) (OSCTXT \*pctxt, OSUINT64 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSUTF8CHAR \*pattern)
- EXTERNXML int [rtXmlEncUInt64Value](#) (OSCTXT \*pctxt, OSUINT64 value)  
*This function encodes a variable of the XSD integer type.*
- EXTERNXML int [rtXmlEncUInt64PatternValue](#) (OSCTXT \*pctxt, OSUINT64 value, const OSUTF8CHAR \*pattern)
- EXTERNXML int [rtXmlEncUInt64Attr](#) (OSCTXT \*pctxt, OSUINT64 value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)  
*This function encodes a variable of the XSD integer type as an attribute (name="value").*
- EXTERNXML int [rtXmlEncXSIAttrs](#) (OSCTXT \*pctxt, OSBOOL needXSI)  
*This function encodes XML schema instance (XSI) attributes at the beginning of an XML document.*
- EXTERNXML int [rtXmlEncXSITypeAttr](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*value)  
*This function encodes an XML schema instance (XSI) type attribute value (xsi:type="value").*
- EXTERNXML int [rtXmlEncXSITypeAttr2](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*typeNsUri, const OSUTF8CHAR \*typeName)  
*This function encodes an XML schema instance (XSI) type attribute value (xsi:type="pfx:value").*
- EXTERNXML int [rtXmlEncXSINilAttr](#) (OSCTXT \*pctxt)  
*This function encodes an XML nil attribute (xsi:nil="true").*
- EXTERNXML int [rtXmlFreeInputSource](#) (OSCTXT \*pctxt)  
*This function closes an input source that was previously created with one of the create input source functions such as 'rtXmlCreateFileInputSource'.*
- EXTERNXML OSBOOL [rtXmlStrCmpAsc](#) (const OSUTF8CHAR \*text1, const char \*text2)
- EXTERNXML OSBOOL [rtXmlStrnCmpAsc](#) (const OSUTF8CHAR \*text1, const char \*text2, OSSIZE len)
- EXTERNXML int [rtXmlSetEncBufPtr](#) (OSCTXT \*pctxt, OSOCTET \*bufaddr, OSSIZE bufsiz)  
*This function is used to set the internal buffer within the run-time library encoding context.*
- EXTERNXML int [rtXmlGetIndent](#) (OSCTXT \*pctxt)  
*This function returns current XML output indent value.*
- EXTERNXML OSBOOL [rtXmlGetWriteBOM](#) (OSCTXT \*pctxt)  
*This function returns whether the Unicode byte order mark will be encoded.*



- EXTERNXML int [rtXmlGetIndentChar](#) (OSCTXT \*pctxt)
 

*This function returns current XML output indent character value (default is space).*
- EXTERNXML int [rtXmlPrepareContext](#) (OSCTXT \*pctxt)
 

*This function prepares the context for another encode by setting the state back to OSXMLINIT and moving the buffer's cursor back to the beginning of the buffer.*
- EXTERNXML int [rtXmlSetEncC14N](#) (OSCTXT \*pctxt, OSBOOL value)
 

*This function sets the option to encode in C14N mode.*
- EXTERNXML int [rtXmlSetEncXSINamespace](#) (OSCTXT \*pctxt, OSBOOL value)
 

*This function sets a flag in the context that indicates the XSI namespace declaration (xmlns:xsi) should be added to the encoded XML instance.*
- EXTERNXML int [rtXmlSetEncXSINilAttr](#) (OSCTXT \*pctxt, OSBOOL value)
 

*This function sets a flag in the context that indicates the XSI attribute declaration (xmlns:xsi) should be added to the encoded XML instance.*
- EXTERNXML int [rtXmlSetDigitsFacets](#) (OSCTXT \*pctxt, int totalDigits, int fractionDigits)
- EXTERNXML int [rtXmlSetEncDocHdr](#) (OSCTXT \*pctxt, OSBOOL value)
 

*This function sets the option to add the XML document header (i.e.*
- EXTERNXML int [rtXmlSetEncodingStr](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*encodingStr)
 

*This function sets the XML output encoding to the given value.*
- EXTERNXML int [rtXmlSetFormatting](#) (OSCTXT \*pctxt, OSBOOL doFormatting)
 

*This function sets XML output formatting to the given value.*
- EXTERNXML int [rtXmlSetIndent](#) (OSCTXT \*pctxt, OSUINT8 indent)
 

*This function sets XML output indent to the given value.*
- EXTERNXML int [rtXmlSetIndentChar](#) (OSCTXT \*pctxt, char indentChar)
 

*This function sets XML output indent character to the given value.*
- EXTERNXML void [rtXmlSetNamespacesSet](#) (OSCTXT \*pctxt, OSBOOL value)
 

*This function sets the context 'namespaces are set' flag.*
- EXTERNXML int [rtXmlSetNSPrefixLinks](#) (OSCTXT \*pctxt, OSRTDList \*pNSAttrs)
 

*This function sets namespace prefix/URI links in the namespace prefix stack in the context structure.*
- EXTERNXML int [rtXmlSetSchemaLocation](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*schemaLocation)
 

*This function sets the XML Schema Instance (xsi) schema location attribute to be added to an encoded document.*
- EXTERNXML int [rtXmlSetNoNSSchemaLocation](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*schemaLocation)
 

*This function sets the XML Schema Instance (xsi) no namespace schema location attribute to be added to an encoded document.*
- EXTERNXML void [rtXmlSetSoapVersion](#) (OSCTXT \*pctxt, OSUINT8 version)
 

*This function sets the SOAP version number.*



- EXTERNXML int [rtXmlSetXSITypeAttr](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*xsiType)  
*This function sets the XML Schema Instance (xsi) type attribute value.*
- EXTERNXML int [rtXmlSetWriteBOM](#) (OSCTXT \*pctxt, OSBOOL write)  
*This function sets whether the Unicode byte order mark is encoded.*
- EXTERNXML int [rtXmlMatchHexStr](#) (OSCTXT \*pctxt, OSSIZE minLength, OSSIZE maxLength)  
*This function tests the context buffer for containing a correct hexadecimal string.*
- EXTERNXML int [rtXmlMatchBase64Str](#) (OSCTXT \*pctxt, OSSIZE minLength, OSSIZE maxLength)  
*This function tests the context buffer for containing a correct base64 string.*
- EXTERNXML int [rtXmlMatchDate](#) (OSCTXT \*pctxt)  
*This function tests the context buffer for containing a correct date string.*
- EXTERNXML int [rtXmlMatchTime](#) (OSCTXT \*pctxt)  
*This function tests the context buffer for containing a correct time string.*
- EXTERNXML int [rtXmlMatchDateTime](#) (OSCTXT \*pctxt)  
*This function tests the context buffer for containing a correct dateTime string.*
- EXTERNXML int [rtXmlMatchGYear](#) (OSCTXT \*pctxt)  
*This function tests the context buffer for containing a correct gYear string.*
- EXTERNXML int [rtXmlMatchGYearMonth](#) (OSCTXT \*pctxt)  
*This function tests the context buffer for containing a correct gYearMonth string.*
- EXTERNXML int [rtXmlMatchGMonth](#) (OSCTXT \*pctxt)  
*This function tests the context buffer for containing a correct gMonth string.*
- EXTERNXML int [rtXmlMatchGMonthDay](#) (OSCTXT \*pctxt)  
*This function tests the context buffer for containing a correct gMonthDay string.*
- EXTERNXML int [rtXmlMatchGDay](#) (OSCTXT \*pctxt)  
*This function tests the context buffer for containing a correct gDay string.*
- EXTERNXML OSUTF8CHAR \* [rtXmlNewQName](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*localName, const OSUTF8CHAR \*prefix)  
*This function creates a new QName given the localName and prefix parts.*
- EXTERNXML OSBOOL [rtXmlCmpBase64Str](#) (OSUINT32 nocts1, const OSOCTET \*data1, const OSUTF8CHAR \*data2)  
*This function compares an array of octets to a base64 string.*
- EXTERNXML OSBOOL [rtXmlCmpHexStr](#) (OSUINT32 nocts1, const OSOCTET \*data1, const OSUTF8CHAR \*data2)  
*This function compares an array of octets to a hex string.*
- EXTERNXML OSBOOL [rtXmlCmpHexChar](#) (OSUTF8CHAR ch, OSOCTET hexval)
- EXTERNXML int [rtSaxGetAttributeID](#) (const OSUTF8CHAR \*attrName, OSSIZE nAttr, const OSUTF8CHAR \*attrNames[], OSUINT32 attrPresent[])

- EXTERNXML const OSUTF8CHAR \* **rtSaxGetAttrValue** (const OSUTF8CHAR \*attrName, const OSUTF8CHAR \*const \*attrs)  
*This function looks up an attribute in the attribute array returned by SAX to the startElement function.*
- EXTERNXML OSINT16 **rtSaxGetElemID** (OSINT16 \*pState, OSINT16 prevElemIdx, const OSUTF8CHAR \*localName, OSINT32 nsidx, const OSSAXElemTableRec idtab[], const OSINT16 \*fstab, OSINT16 fstabRows, OSINT16 fstabCols)  
*This function looks up a sequence element name in the given element info array.*
- EXTERNXML OSINT16 **rtSaxGetElemID8** (OSINT16 \*pState, OSINT16 prevElemIdx, const OSUTF8CHAR \*localName, OSINT32 nsidx, const OSSAXElemTableRec idtab[], const OSINT8 \*fstab, OSINT16 fstabRows, OSINT16 fstabCols)  
*This function is a space optimized version of rtSaxGetElemID.*
- EXTERNXML OSINT16 **rtSaxFindElemID** (OSINT16 \*pState, OSINT16 prevElemIdx, const OSUTF8CHAR \*localName, OSINT32 nsidx, const OSSAXElemTableRec idtab[], const OSINT16 \*fstab, OSINT16 fstabRows, OSINT16 fstabCols)
- EXTERNXML OSINT16 **rtSaxFindElemID8** (OSINT16 \*pState, OSINT16 prevElemIdx, const OSUTF8CHAR \*localName, OSINT32 nsidx, const OSSAXElemTableRec idtab[], const OSINT8 \*fstab, OSINT16 fstabRows, OSINT16 fstabCols)
- EXTERNXML OSBOOL **rtSaxHasXMLNSAttrs** (const OSUTF8CHAR \*const \*attrs)  
*This function checks if the given attribute list contains one or more XML namespace attributes (xmlns).*
- EXTERNXML OSBOOL **rtSaxIsEmptyBuffer** (OSCTXT \*pctxt)  
*This function checks if the buffer in the context is empty or not.*
- EXTERNXML OSINT16 **rtSaxLookupElemID** (OSCTXT \*pctxt, OSINT16 \*pState, OSINT16 prevElemIdx, const OSUTF8CHAR \*localName, const OSUTF8CHAR \*qName, OSINT32 nsidx, const OSSAXElemTableRec idtab[], const OSINT16 \*fstab, OSINT16 fstabRows, OSINT16 fstabCols)
- EXTERNXML OSINT16 **rtSaxLookupElemID8** (OSCTXT \*pctxt, OSINT16 \*pState, OSINT16 prevElemIdx, const OSUTF8CHAR \*localName, const OSUTF8CHAR \*qName, OSINT32 nsidx, const OSSAXElemTableRec idtab[], const OSINT8 \*fstab, OSINT16 fstabRows, OSINT16 fstabCols)
- EXTERNXML int **rtSaxStrListParse** (OSCTXT \*pctxt, OSRTMEMBUF \*pMemBuf, OSRTDList \*pvalue)  
*This function parses the list of strings.*
- EXTERNXML int **rtSaxSortAttrs** (OSCTXT \*pctxt, const OSUTF8CHAR \*const \*attrs, OSUINT16 \*\*order)  
*This function sorts a SAX attribute list in ascending order based on attribute name.*
- EXTERNXML int **rtSaxStrListMatch** (OSCTXT \*pctxt)  
*This function matches the list of strings.*
- EXTERNXML OSBOOL **rtSaxTestFinal** (OSINT16 state, OSINT16 currElemIdx, const int \*fstab, int fstabRows, int fstabCols)
- EXTERNXML OSBOOL **rtSaxTestFinal8** (OSINT16 state, OSINT16 currElemIdx, const OSINT8 \*fstab, int fstabRows, int fstabCols)
- EXTERNXML int **rtSaxSetSkipLevelToCurrent** (OSCTXT \*pctxt, int stat)
- EXTERNXML OSUINT32 **rtSaxSetMaxErrors** (OSCTXT \*pctxt, OSUINT32 maxErrors)
- EXTERNXML OSUINT32 **rtSaxGetMaxErrors** (OSCTXT \*pctxt)
- EXTERNXML int **rtSaxTestAttributesPresent** (OSCTXT \*pctxt, const OSUINT32 \*attrPresent, const OSUINT32 \*reqAttrMask, const OSUTF8CHAR \*const \*attrNames, OSSIZE numOfAttrs, const char \*parentTypeName)

- EXTERNXML OSBOOL **rtSaxIncErrors** (OSCTXT \*pctx)
- EXTERNXML int **rtSaxReportUnexpAttrs** (OSCTXT \*pctx, const OSUTF8CHAR \*const \*attrs, const char \*typeName)
- 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.*
- EXTERNXML int **rtXmlWriteUTF16ToFile** (OSCTXT \*pctx, const char \*filename)
- EXTERNXML void **rtXmlTreatWhitespaces** (OSCTXT \*pctx, int whiteSpaceType)
- EXTERNXML int **rtXmlCheckBuffer** (OSCTXT \*pctx, OSSIZE byte\_count)
- EXTERNXML void **rtErrXmlInit** (OSVOIDARG)
- EXTERNXML int **rtXmlPutChar** (OSCTXT \*pctx, const OSUTF8CHAR value)
- EXTERNXML int **rtXmlWriteChars** (OSCTXT \*pctx, const OSUTF8CHAR \*value, OSSIZE len)
- EXTERNXML int **rtXmlpDecAny** (OSCTXT \*pctx, const OSUTF8CHAR \*\*pvalue)
 

*This function decodes an arbitrary XML section of code as defined by the XSD any type (xsd:any).*
- EXTERNXML int **rtXmlpDecAny2** (OSCTXT \*pctx, OSUTF8CHAR \*\*pvalue)
 

*This version of the rtXmlpDecAny function returns the string in a mutable buffer.*
- EXTERNXML int **rtXmlpDecAnyAttrStr** (OSCTXT \*pctx, const OSUTF8CHAR \*\*ppAttrStr, OSSIZE attrIndex)
 

*This function decodes an any attribute string.*
- EXTERNXML int **rtXmlpDecAnyElem** (OSCTXT \*pctx, const OSUTF8CHAR \*\*pvalue)
 

*This function decodes an arbitrary XML section of code as defined by the XSD any type (xsd:any).*
- EXTERNXML int **rtXmlpDecBase64Str** (OSCTXT \*pctx, OSOCTET \*pvalue, OSUINT32 \*pnocets, OSSIZE bufsize)
 

*This function decodes a contents of a Base64-encode binary string into a static memory structure.*
- EXTERNXML int **rtXmlpDecBigInt** (OSCTXT \*pctx, const OSUTF8CHAR \*\*pvalue)
 

*This function will decode a variable of the XSD integer type.*
- EXTERNXML int **rtXmlpDecBitString** (OSCTXT \*pctx, OSOCTET \*pvalue, OSUINT32 \*pnbits, OSUINT32 bufsize)
 

*This function decodes a bit string value.*
- EXTERNXML int **rtXmlpDecBool** (OSCTXT \*pctx, OSBOOL \*pvalue)
 

*This function decodes a variable of the boolean type.*
- EXTERNXML int **rtXmlpDecDate** (OSCTXT \*pctx, OSXSDDateTime \*pvalue)
 

*This function decodes a variable of the XSD 'date' type.*
- EXTERNXML int **rtXmlpDecDateTime** (OSCTXT \*pctx, OSXSDDateTime \*pvalue)
 

*This function decodes a variable of the XSD 'dateTime' type.*
- EXTERNXML int **rtXmlpDecDecimal** (OSCTXT \*pctx, OSREAL \*pvalue, int totalDigits, int fractionDigits)
 

*This function decodes the contents of a decimal data type.*
- EXTERNXML int **rtXmlpDecDouble** (OSCTXT \*pctx, OSREAL \*pvalue)

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

- EXTERNXML int [rtXmlpDecDoubleExt](#) (OSCTXT \*pctxt, OSUINT8 flags, OSREAL \*pvalue)  
*This function decodes the contents of a float or double data type.*
- EXTERNXML int [rtXmlpDecDynBase64Str](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)  
*This function decodes a contents of a Base64-encode binary string.*
- EXTERNXML int [rtXmlpDecDynBitString](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)  
*This function decodes a bit string value.*
- EXTERNXML int [rtXmlpDecDynHexStr](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)  
*This function decodes a contents of a hexBinary string.*
- EXTERNXML int [rtXmlpDecDynUnicodeStr](#) (OSCTXT \*pctxt, const OSUNICHAR \*\*ppdata, OSSIZE \*pnchars)  
*This function decodes a Unicode string data type.*
- EXTERNXML int [rtXmlpDecDynUTF8Str](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*outdata)  
*This function decodes the contents of a UTF-8 string data type.*
- EXTERNXML int [rtXmlpDecUTF8Str](#) (OSCTXT \*pctxt, OSUTF8CHAR \*out, OSSIZE max\_len)  
*This function decodes the contents of a UTF-8 string data type.*
- EXTERNXML int [rtXmlpDecGDay](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gDay' type.*
- EXTERNXML int [rtXmlpDecGMonth](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gMonth' type.*
- EXTERNXML int [rtXmlpDecGMonthDay](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gMonthDay' type.*
- EXTERNXML int [rtXmlpDecGYear](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gYear' type.*
- EXTERNXML int [rtXmlpDecGYearMonth](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gYearMonth' type.*
- EXTERNXML int [rtXmlpDecHexStr](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, OSINT32 bufsize)  
*This function decodes the contents of a hexBinary string into a static memory structure.*
- EXTERNXML int [rtXmlpDecInt](#) (OSCTXT \*pctxt, OSINT32 \*pvalue)  
*This function decodes the contents of a 32-bit integer data type.*
- EXTERNXML int [rtXmlpDecInt8](#) (OSCTXT \*pctxt, OSINT8 \*pvalue)  
*This function decodes the contents of an 8-bit integer data type (i.e.*
- EXTERNXML int [rtXmlpDecInt16](#) (OSCTXT \*pctxt, OSINT16 \*pvalue)

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

- EXTERNXML int [rtXmlpDecInt64](#) (OSCTXT \*pctx, OSINT64 \*pvalue)

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

- EXTERNXML int [rtXmlpDecNamedBits](#) (OSCTXT \*pctx, const OSBitMapItem \*pBitMap, OSOCTET \*pvalue, OSUINT32 \*pnbits, OSUINT32 bufsize)

*This function decodes the contents of a named bit field.*

- EXTERNXML int [rtXmlpDecStrList](#) (OSCTXT \*pctx, OSRTDList \*plist)

*This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.*

- EXTERNXML int [rtXmlpDecTime](#) (OSCTXT \*pctx, OSXSDDateTime \*pvalue)

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

- EXTERNXML int [rtXmlpDecUInt](#) (OSCTXT \*pctx, OSUINT32 \*pvalue)

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

- EXTERNXML int [rtXmlpDecUInt8](#) (OSCTXT \*pctx, OSOCTET \*pvalue)

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

- EXTERNXML int [rtXmlpDecUInt16](#) (OSCTXT \*pctx, OSUINT16 \*pvalue)

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

- EXTERNXML int [rtXmlpDecUInt64](#) (OSCTXT \*pctx, OSUINT64 \*pvalue)

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

- EXTERNXML int [rtXmlpDecXmlStr](#) (OSCTXT \*pctx, OSXMLSTRING \*outdata)

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

- EXTERNXML int [rtXmlpDecXmlStrList](#) (OSCTXT \*pctx, OSRTDList \*plist)

*This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.*

- EXTERNXML int [rtXmlpDecXSIAAttr](#) (OSCTXT \*pctx, const OSXMLNameFragments \*attrName)

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

- EXTERNXML int [rtXmlpDecXSITypeAttr](#) (OSCTXT \*pctx, const OSXMLNameFragments \*attrName, const OSUTF8CHAR \*\*ppAttrValue)

*This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).*

- EXTERNXML int [rtXmlpGetAttributeID](#) (const OSXMLStrFragment \*attrName, OSINT16 nsidx, OSSIZE nAttr, const OSXMLAttrDescr attrNames[ ], OSUINT32 attrPresent[ ])

*This function finds an attribute in the descriptor table.*

- EXTERNXML int [rtXmlpGetNextElem](#) (OSCTXT \*pctx, OSXMLElemDescr \*pElem, OSINT32 level)

*This function parse the next element start tag.*

- EXTERNXML int [rtXmlpGetNextElemID](#) (OSCTXT \*pctx, const OSXMLElemIDRec \*tab, OSSIZE nrows, OSINT32 level, OSBOOL continueParse)

*This function parses the next start tag and finds the index of the element name in the descriptor table.*

- EXTERNXML int [rtXmlpMarkLastEventActive](#) (OSCTXT \*pctxt)
 

*This function marks current tag as unprocessed.*
- EXTERNXML int [rtXmlpMatchStartTag](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*elemLocalName, OSINT16 nsidx)
 

*This function parses the next start tag that matches with given name.*
- EXTERNXML int [rtXmlpMatchEndTag](#) (OSCTXT \*pctxt, OSINT32 level)
 

*This function parse next end tag that matches with given name.*
- EXTERNXML OSBOOL [rtXmlpHasAttributes](#) (OSCTXT \*pctxt)
 

*This function checks accessibility of attributes.*
- EXTERNXML int [rtXmlpGetAttributeCount](#) (OSCTXT \*pctxt)
 

*This function returns number of attributes in last processed start tag.*
- EXTERNXML int [rtXmlpSelectAttribute](#) (OSCTXT \*pctxt, OSXMLNameFragments \*pAttr, OSINT16 \*nsidx, OSSIZE attrIndex)
 

*This function selects attribute to decode.*
- EXTERNXML OSINT32 [rtXmlpGetCurrentLevel](#) (OSCTXT \*pctxt)
 

*This function returns current nesting level.*
- EXTERNXML void [rtXmlpSetWhiteSpaceMode](#) (OSCTXT \*pctxt, OSXMLWhiteSpaceMode whiteSpaceMode)
 

*Sets the whitespace treatment mode.*
- EXTERNXML OSBOOL [rtXmlpSetMixedContentMode](#) (OSCTXT \*pctxt, OSBOOL mixedContentMode)
 

*Sets mixed content mode.*
- EXTERNXML void [rtXmlpSetListMode](#) (OSCTXT \*pctxt)
 

*Sets list mode.*
- EXTERNXML OSBOOL [rtXmlpListHasItem](#) (OSCTXT \*pctxt)
 

*Check for end of decoded token list.*
- EXTERNXML void [rtXmlpCountListItems](#) (OSCTXT \*pctxt, OSSIZE \*itemCnt)
 

*Count tokens in list.*
- EXTERNXML int [rtXmlpGetNextSeqElemID2](#) (OSCTXT \*pctxt, const OSXMLElemIDRec \*tab, const OSXMLGroupDesc \*pGroup, int groups, int curID, int lastMandatoryID, OSBOOL groupMode, OSBOOL checkRepeat)
 

*This function parses the next start tag and finds index of element name in descriptor table.*
- EXTERNXML int [rtXmlpGetNextSeqElemID](#) (OSCTXT \*pctxt, const OSXMLElemIDRec \*tab, const OSXMLGroupDesc \*pGroup, int curID, int lastMandatoryID, OSBOOL groupMode)
 

*This function parses the next start tag and finds index of element name in descriptor table.*

- EXTERNXML int [rtXmlpGetNextSeqElemIDExt](#) (OSCTXT \*pctx, const [OSXMLElemIDRec](#) \*tab, const [OSXMLGroupDesc](#) \*ppGroup, const OSBOOL \*extRequired, int postExtRootID, int curID, int lastMandatoryID, OSBOOL groupMode)  
*This is an ASN.1 extension-supporting version of [rtXmlpGetNextSeqElemID](#).*
- EXTERNXML int [rtXmlpGetNextAllElemID](#) (OSCTXT \*pctx, const [OSXMLElemIDRec](#) \*tab, OSSIZE nRows, const OSUINT8 \*pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)  
*This function parses the next start tag and finds index of element name in descriptor table.*
- EXTERNXML int [rtXmlpGetNextAllElemID16](#) (OSCTXT \*pctx, const [OSXMLElemIDRec](#) \*tab, OSSIZE nRows, const OSUINT16 \*pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)  
*This function parses the next start tag and finds index of element name in descriptor table.*
- EXTERNXML int [rtXmlpGetNextAllElemID32](#) (OSCTXT \*pctx, const [OSXMLElemIDRec](#) \*tab, OSSIZE nRows, const OSUINT32 \*pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)  
*This function parses the next start tag and finds index of element name in descriptor table.*
- EXTERNXML void [rtXmlpSetNamespaceTable](#) (OSCTXT \*pctx, const OSUTF8CHAR \*namespaceTable[], OSSIZE nmNamespaces)  
*Sets user namespace table.*
- EXTERNXML int [rtXmlpCreateReader](#) (OSCTXT \*pctx)  
*Creates pull parser reader structure within the context.*
- EXTERNXML void [rtXmlpHideAttributes](#) (OSCTXT \*pctx)  
*Disable access to attributes.*
- EXTERNXML OSBOOL [rtXmlpNeedDecodeAttributes](#) (OSCTXT \*pctx)  
*This function checks if attributes were previously decoded.*
- EXTERNXML void [rtXmlpMarkPos](#) (OSCTXT \*pctx)  
*Save current decode position.*
- EXTERNXML void [rtXmlpRewindToMarkedPos](#) (OSCTXT \*pctx)  
*Rewind to saved decode position.*
- EXTERNXML void [rtXmlpResetMarkedPos](#) (OSCTXT \*pctx)  
*Reset saved decode position.*
- EXTERNXML int [rtXmlpGetXSITypeAttr](#) (OSCTXT \*pctx, const OSUTF8CHAR \*\*ppAttrValue, OSINT16 \*nsidx, OSSIZE \*pLocalOffs)  
*This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).*
- EXTERNXML int [rtXmlpGetXmlnsAttrs](#) (OSCTXT \*pctx, OSRTDList \*pNSAttrs)  
*This function decodes namespace attributes from start tag and adds them to the given list.*
- EXTERNXML int [rtXmlpDecXSIAttrs](#) (OSCTXT \*pctx)  
*This function decodes XSI (XML Schema Instance) that may be present in any arbitrary XML element within a document.*
- EXTERNXML OSBOOL [rtXmlpIsEmptyElement](#) (OSCTXT \*pctx)  
*Check element content: empty or not.*



- EXTERNXML int [rtXmlEncAttrC14N](#) (OSCTXT \*pctxt)  
*This function used only in C14 mode.*
- EXTERNXML struct OSXMLReader \* [rtXmlpGetReader](#) (OSCTXT \*pctxt)  
*This function fetches the XML reader structure from the context for use in low-level pull parser calls.*
- EXTERNXML OSBOOL [rtXmlpIsLastEventDone](#) (OSCTXT \*pctxt)  
*Check processing status of current tag.*
- EXTERNXML int [rtXmlpGetXSITypeIndex](#) (OSCTXT \*pctxt, const OSXMLItemDescr typetab[ ], OSSIZE typetabsiz)  
*This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type) and find type index in descriptor table.*
- EXTERNXML int [rtXmlpLookupXSITypeIndex](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*pXsiType, OSINT16 xsiTypeId, const OSXMLItemDescr typetab[ ], OSSIZE typetabsiz)  
*This function find index of XSI (XML Schema Instance) type in descriptor table.*
- EXTERNXML void [rtXmlpForceDecodeAsGroup](#) (OSCTXT \*pctxt)  
*Disable skipping of unknown elements in optional sequence tail.*
- EXTERNXML OSBOOL [rtXmlpIsDecodeAsGroup](#) (OSCTXT \*pctxt)  
*This function checks if "decode as group" mode was forced.*
- EXTERNXML OSBOOL [rtXmlpIsUTF8Encoding](#) (OSCTXT \*pctxt)  
*This function checks if the encoding specified in XML header is UTF-8.*
- EXTERNXML int [rtXmlpReadBytes](#) (OSCTXT \*pctxt, OSOCTET \*pbuf, OSSIZE nbytes)  
*This function reads the specified number of bytes directly from the underlying XML parser stream.*

### 8.1.1 Detailed Description

XML low-level C encode/decode functions.

Definition in file [osrxml.h](#).

### 8.1.2 Define Documentation

#### 8.1.2.1 #define IS\_XMLNSATTR(name)

**Value:**

```
((OSUTF8LEN(name) >= 5) && name[0] == 'x' && name[1] == 'm' && \
name[2] == 'l' && name[3] == 'n' && name[4] == 's')
```

Definition at line 188 of file [osrxml.h](#).



### 8.1.2.2 #define IS\_XSIATTR(name)

#### Value:

```
((OSUTF8LEN(name) >= 4) && name[0] == 'x' && name[1] == 's' && \
 name[2] == 'i' && name[3] == ':')
```

Definition at line 192 of file osrxml.h.

### 8.1.2.3 #define OSXMLQNAMEEQUALS(xnamefrag, qnametext)

#### Value:

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

Definition at line 181 of file osrxml.h.

### 8.1.2.4 #define OSXMLSETUTF8DECPTR(pctxt, str)

#### Value:

```
rtxInitContextBuffer (pctxt, OSRTSAFECONSTCAST (OSOCKET*, str), \
OSUTF8LEN (str))
```

Definition at line 184 of file osrxml.h.

## 8.1.3 Typedef Documentation

### 8.1.3.1 typedef struct OSXMLGroupDesc OSXMLGroupDesc

[OSXMLGroupDesc](#) describes how entries in an [OSXMLElemIDRec](#) array make up a group.

Here, "group" means a set of elements, any of which may be matched next. This does not correspond directly to an XSD group.

For example, if elementA is optional and followed by non-optional elementB, then there will be a group that contains both elements. There will also be a group that contains only elementB; this will be the group of interest after elementA is matched.

## 8.1.4 Function Documentation

### 8.1.4.1 EXTERNXML const OSUTF8CHAR\* rtSaxGetAttrValue (const OSUTF8CHAR \* attrName, const OSUTF8CHAR \*const \* attrs)

This function looks up an attribute in the attribute array returned by SAX to the startElement function.

#### Parameters

*attrName* Name of the attribute to find.

*attrs* Attribute array returned in SAX startElement function. This is an array of character strings containing name1, value1, name2, value2, ... List is terminated by a null name.

#### Returns

Pointer to character string containing attribute value or NULL if attrName not found.

**8.1.4.2 EXTERNXML OSINT16 rtSaxGetElemID (OSINT16 \* *pState*, OSINT16 *prevElemIdx*, const OSUTF8CHAR \* *localName*, OSINT32 *nsidx*, const OSSAXElemTableRec *idtab*[], const OSINT16 \* *fstab*, OSINT16 *fstabRows*, OSINT16 *fstabCols*)**

This function looks up a sequence element name in the given element info array.

If 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

*nsidx* Namespace index

*idtab* Element ID table

*fstab* Finite state table

*fstabRows* Number of rows in *fstab*.

*fstabCols* Number of columns in *fstab*.

**8.1.4.3 EXTERNXML OSINT16 rtSaxGetElemID8 (OSINT16 \* *pState*, OSINT16 *prevElemIdx*, const OSUTF8CHAR \* *localName*, OSINT32 *nsidx*, const OSSAXElemTableRec *idtab*[], const OSINT8 \* *fstab*, OSINT16 *fstabRows*, OSINT16 *fstabCols*)**

This function is a space optimized version of `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

*nsidx* Namespace index

*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*.

**8.1.4.4 EXTERNXML OSBOOL rtSaxHasXMLNSAttrs (const OSUTF8CHAR \**const* \* *attrs*)**

This function checks if the given attribute list contains one or more XML namespace attributes (xmlns).

**Parameters**

*attrs* Attribute list in form passed by parser into SAX `startElement` function.

**Returns**

TRUE, if xmlns attribute found in list.

#### 8.1.4.5 EXTERNXML OSBOOL rtSaxIsEmptyBuffer (OSCTXT \* *pctxt*)

This function checks if the buffer in the context is empty or not.

##### Parameters

*pctxt* Pointer to OSCTXT structure

##### Returns

TRUE, if the buffer contains empty string.

#### 8.1.4.6 EXTERNXML int rtSaxSortAttrs (OSCTXT \* *pctxt*, const OSUTF8CHAR \*const \* *attrs*, OSUINT16 \*\* *order*)

This function sorts a SAX attribute list in ascending order based on attribute name.

It currently only supports unqualified attributes.

##### Parameters

*pctxt* Pointer to OSCTXT structure.

*attrs* Standard SAX attribute list. Entry *i* is attribute name and *i*+1 is value. List is terminated by a null name.

*order* Order array containing the order of sorted attributes. This array is allocated using `rtxMemAlloc`, it can be freed using `rtxMemFreePtr` or will be freed when the context is freed. The list holds indices to name items in the attribute list that is passed in.

##### Returns

If success, positive value contains number of attributes in *attrs*; if failure, negative status code.

#### 8.1.4.7 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.

#### 8.1.4.8 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.

### 8.1.4.9 EXTERNXML OSBOOL rtXmlCmpBase64Str (OSUINT32 *noct1*, const OSOCTET \* *data1*, const OSUTF8CHAR \* *data2*)

This function compares an array of octets to a base64 string.

## Parameters

*noct1* Number of octets in data1.

*data1* Pointer to array of OSOCTET.

*data2* Pointer to null-terminated array of OSUTF8CHAR.

## Returns

TRUE if data2 is a base64 string representation of data1, false otherwise.

### 8.1.4.10 EXTERNXML OSBOOL rtXmlCmpHexStr (OSUINT32 *noct1*, const OSOCTET \* *data1*, const OSUTF8CHAR \* *data2*)

This function compares an array of octets to a hex string.

## Parameters

*noct1* Number of octets in data1.

*data1* Pointer to array of OSOCTET.

*data2* Pointer to null-terminated array of OSUTF8CHAR.

## Returns

TRUE if data2 is a hex string representation of data1, false otherwise.

### 8.1.4.11 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.

### 8.1.4.12 EXTERNXML int rtXmlInitContext (OSCTXT \* *pctxt*)

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

#### Parameters

*pctxt* Pointer to OSCTXT structure

### 8.1.4.13 EXTERNXML int rtXmlInitContextUsingKey (OSCTXT \* *pctxt*, const OSOCTET \* *key*, OSSIZE *keylen*)

This function initializes a context using a run-time key.

This form is required for evaluation and limited distribution software. The compiler will generate a macro for rtXmlInitContext in the rtkey.h file that will invoke this function with the generated run-time key.

#### Parameters

*pctxt* The pointer to the context structure variable to be initialized.

*key* Key data generated by ASN1C compiler.

*keylen* Key data field length.

## Returns

Completion status of operation:

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

### 8.1.4.14 EXTERNXML int rtXmlInitCtxtAppInfo (OSCTXT \* *pctxt*)

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

#### Parameters

*pctxt* Pointer to OSCTXT structure

### 8.1.4.15 EXTERNXML int rtXmlMatchBase64Str (OSCTXT \* *pctxt*, OSSIZE *minLength*, OSSIZE *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.

### 8.1.4.16 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.

### 8.1.4.17 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.

### 8.1.4.18 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.

#### 8.1.4.19 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.

#### 8.1.4.20 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.

#### 8.1.4.21 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.

#### 8.1.4.22 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.

#### 8.1.4.23 EXTERNXML int rtXmlMatchHexStr (OSCTXT \* *pctxt*, OSSIZE *minLength*, OSSIZE *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.

#### 8.1.4.24 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.

#### 8.1.4.25 EXTERNXML void rtXmlMemFreeAnyAttrs (OSCTXT \* *pctxt*, OSRTDList \* *pAnyAttrList*)

This function frees a list of anyAttribute that is a member of [OSXSDAnyType](#) structure.

##### Parameters

*pctxt* Pointer to context block structure.

*pAnyAttrList* Pointer to list of anyAttribute that is to be freed.

#### 8.1.4.26 EXTERNXML OSUTF8CHAR\* rtXmlNewQName (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *localName*, const OSUTF8CHAR \* *prefix*)

This function creates a new QName given the localName and prefix parts.

##### Parameters

*pctxt* Pointer to a context structure.

*localName* Element local name.

*prefix* Namespace prefix.



## Returns

QName value. Memory for the value will have been allocated by `rtxMemAlloc` and thus must be freed using one of the `rtxMemFree` functions. The value will be NULL if no dynamic memory was available.

### 8.1.4.27 EXTERNXML int rtXmlPrepareContext (OSCTXT \* *pctxt*)

This function prepares the context for another encode by setting the state back to OSXMLINIT and moving the buffer's cursor back to the beginning of the buffer.

## Parameters

*pctxt* Pointer to OSCTXT structure

## Returns

0 if OK, negative status code if error.

### 8.1.4.28 EXTERNXML int rtXmlSetEncC14N (OSCTXT \* *pctxt*, OSBOOL *value*)

This function sets the option to encode in C14N mode.

## Parameters

*pctxt* Pointer to OSCTXT structure

*value* Boolean value: true = C14N mode enabled.

## Returns

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

### 8.1.4.29 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.

### 8.1.4.30 EXTERNXML int rtXmlSetEncodingStr (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *encodingStr*)

This function sets the XML output encoding to the given value. Currently, UTF-8/UTF-16/ISO-8859-1 encodings are supported.

## Parameters

*pctxt* Pointer to OSCTXT structure  
*encodingStr* XML output encoding format

## Returns

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

### 8.1.4.31 EXTERNXML int rtXmlSetEncXSINamespace (OSCTXT \* *pctxt*, OSBOOL *value*)

This function sets a flag in the context that indicates the XSI namespace declaration (xmlns:xsi) should be added to the encoded XML instance.

## Parameters

*pctxt* Pointer to OSCTXT structure  
*value* Boolean value: true = encode XSI namespace attribute.

## Returns

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

### 8.1.4.32 EXTERNXML int rtXmlSetEncXSINilAttr (OSCTXT \* *pctxt*, OSBOOL *value*)

This function sets a flag in the context that indicates the XSI attribute declaration (xmlns:xsi) should be added to the encoded XML instance.

## Parameters

*pctxt* Pointer to OSCTXT structure  
*value* Boolean value: true = encode xsi:nil attribute.

## Returns

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

### 8.1.4.33 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.

#### 8.1.4.34 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.

#### 8.1.4.35 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.

#### 8.1.4.36 EXTERNXML void rtXmlSetNamespacesSet (OSCTXT \* *pctxt*, OSBOOL *value*)

This function sets the context 'namespaces are set' flag.

This indicates that namespace declarations have been set either by the decoder or externally by the end user. It is used by the encoder to know not to set the default namespaces specified in the schema before starting encoding.

##### Parameters

*pctxt* Pointer to OSCTXT structure.  
*value* Boolean value to which flag is to be set.

#### 8.1.4.37 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.

#### 8.1.4.38 EXTERNXML int rtXmlSetNSPrefixLinks (OSCTXT \* *pctxt*, OSRTDList \* *pNSAttrs*)

This function sets namespace prefix/URI links in the namespace prefix stack in the context structure.

##### Parameters

*pctxt* Pointer to OSCTXT structure.

*pNSAttrs* List of namespace attributes.

##### Returns

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

#### 8.1.4.39 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.

#### 8.1.4.40 EXTERNXML void rtXmlSetSoapVersion (OSCTXT \* *pctxt*, OSUINT8 *version*)

This function sets the SOAP version number.

##### Parameters

*pctxt* Pointer to OSCTXT structure

*version* SOAP version number as 2 digit integer (for example, 11 is SOAP version 1.1, 12 is version 1.2, etc.)

#### 8.1.4.41 EXTERNXML int rtXmlSetWriteBOM (OSCTXT \* *pctxt*, OSBOOL *write*)

This function sets whether the Unicode byte order mark is encoded.

##### Parameters

*pctxt* Pointer to OSCTXT structure

*write* TRUE to encode BOM, FALSE to not encode BOM.

##### Returns

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

#### **8.1.4.42 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.

## 8.2 OSXMLDecodeBuffer.h File Reference

XML decode buffer or stream class definition.

```
#include "rtxsrc/OSRTInputStream.h"  
#include "rtxmlsrc/OSXMLMessageBuffer.h"  
#include "rtxmlsrc/rtSaxCppParserIF.h"
```

### Data Structures

- class [OSXMLDecodeBuffer](#)

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

### 8.2.1 Detailed Description

XML decode buffer or stream class definition.

Definition in file [OSXMLDecodeBuffer.h](#).

## 8.3 OSXMLEncodeBuffer.h File Reference

XML encode message buffer class definition.

```
#include "rtxmlsrc/OSXMLMessageBuffer.h"
```

### Data Structures

- class [OSXMLEncodeBuffer](#)

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

### 8.3.1 Detailed Description

XML encode message buffer class definition.

Definition in file [OSXMLEncodeBuffer.h](#).

## 8.4 OSXMLEncodeStream.h File Reference

XML encode stream class definition.

```
#include "rtxsrc/OSRTOutputStream.h"  
#include "rtxmlsrc/OSXMLMessageBuffer.h"
```

### Data Structures

- class [OSXMLEncodeStream](#)

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

### 8.4.1 Detailed Description

XML encode stream class definition.

Definition in file [OSXMLEncodeStream.h](#).



## 8.5 OSXMLMessageBuffer.h File Reference

XML encode/decode buffer and stream base class.

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

```
#include "rtxmlsrc/osrtxml.h"
```

### Data Structures

- class [OSXMLMessageBuffer](#)

*The XML message buffer class is derived from the OSMMessageBuffer base class.*

### 8.5.1 Detailed Description

XML encode/decode buffer and stream base class.

Definition in file [OSXMLMessageBuffer.h](#).

## 8.6 rtXmlErrCodes.h File Reference

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

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

### Defines

- #define `XML_OK_EOB` 0x7fffffff  
*End of block marker.*
- #define `XML_OK_FRAG` `XML_OK_EOB`  
*Maintained for backward compatibility.*
- #define `XML_E_BASE` -200  
*Error base.*
- #define `XML_E_GENERR` (`XML_E_BASE`)  
*General error.*
- #define `XML_E_INVSYMBOL` (`XML_E_BASE-1`)  
*An invalid XML symbol (character) was detected at the given point in the parse stream.*
- #define `XML_E_TAGMISMATCH` (`XML_E_BASE-2`)  
*Start/end tag mismatch.*
- #define `XML_E_DUPLATTR` (`XML_E_BASE-3`)  
*Duplicate attribute found.*
- #define `XML_E_BADCHARREF` (`XML_E_BASE-4`)  
*Bad character reference found.*
- #define `XML_E_INVMODE` (`XML_E_BASE-5`)  
*Invalid mode.*
- #define `XML_E_UNEXPEOF` (`XML_E_BASE-6`)  
*Unexpected end of file (document).*
- #define `XML_E_NOMATCH` (`XML_E_BASE-7`)  
*Current tag is not matched to specified one.*
- #define `XML_E_ELEMMISRQ` (`XML_E_BASE-8`)  
*Missing required element.*
- #define `XML_E_ELEMSISRQ` (`XML_E_BASE-9`)  
*Missing required elements.*
- #define `XML_E_TOOFWELEMS` (`XML_E_BASE-10`)  
*The number of elements in a repeating collection was less than the number of elements specified in the XSD minOccurs facet for this type or element.*

- #define [XML\\_E\\_UNEXPSTARTTAG](#) (XML\_E\_BASE-11)  
*Unexpected start tag.*
- #define [XML\\_E\\_UNEXPENDTAG](#) (XML\_E\_BASE-12)  
*Unexpected end tag.*
- #define [XML\\_E\\_IDNOTFOU](#) (XML\_E\_BASE-13)  
*Expected identifier not found.*
- #define [XML\\_E\\_INVTYPEINFO](#) (XML\_E\_BASE-14)  
*Unknown xsi:type.*
- #define [XML\\_E\\_NSURINOTFOU](#) (XML\_E\_BASE-15)  
*Namespace URI not defined for given prefix.*
- #define [XML\\_E\\_KEYNOTFOU](#) (XML\_E\_BASE-16)  
*Keyref constraint has some key that not present in refered constraint.*
- #define [XML\\_E\\_DUPLKEY](#) (XML\_E\_BASE-17)  
*Key or unique constraint has duplicated key.*
- #define [XML\\_E\\_FLDABSENT](#) (XML\_E\_BASE-18)  
*Some key has no full set of fields.*
- #define [XML\\_E\\_DUPLFLD](#) (XML\_E\_BASE-19)  
*Some key has more than one value for field.*
- #define [XML\\_E\\_NOTEMPTY](#) (XML\_E\_BASE-20)  
*An element was not empty when expected.*

### 8.6.1 Detailed Description

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

Definition in file [rtXmlErrCodes.h](#).

# Index

- addXMLHeader
  - OSXMLEncodeBuffer, 119
- addXMLText
  - OSXMLEncodeBuffer, 119
- ASN.1-XML encode/decode functions., 6
- asn1xml
  - rtAsn1XmlAddAnyAttr, 7
  - rtAsn1XmlEncGenTime, 8
  - rtAsn1XmlEncObjId, 8
  - rtAsn1XmlEncOpenType, 8
  - rtAsn1XmlEncOpenTypeExt, 9
  - rtAsn1XmlEncReal, 9
  - rtAsn1XmlEncRelOID, 10
  - rtAsn1XmlEncUnivStr, 10
  - rtAsn1XmlEncUTCTime, 10
  - rtAsn1XmlFmtAttrStr, 11
  - rtAsn1XmlParseAttrStr, 11
  - rtAsn1XmlpDecDynBitStr, 12
  - rtAsn1XmlpDecDynBitStr64, 12
  - rtAsn1XmlpDecGenTime, 12
  - rtAsn1XmlpDecObjId, 13
  - rtAsn1XmlpDecOpenType, 13
  - rtAsn1XmlpDecReal, 13
  - rtAsn1XmlpDecRelOID, 14
  - rtAsn1XmlpDecUnivStr, 14
  - rtAsn1XmlpDecUTCTime, 14
  - rtXmlpDecListOfASN1DynBitStr, 15
- decodeXML
  - OSXMLDecodeBuffer, 115
- encodeAttr
  - OSXMLEncodeStream, 124
- encodeText
  - OSXMLEncodeStream, 124
- endElement
  - OSXMLEncodeStream, 124
- getIndent
  - OSXMLMessageBuffer, 132
- getIndentChar
  - OSXMLMessageBuffer, 132
- getMsgLen
  - OSXMLEncodeBuffer, 119
- getMsgPtr
  - OSXMLEncodeStream, 124
- getStream
  - OSXMLEncodeStream, 125
- getWriteBOM
  - OSXMLMessageBuffer, 132
- init
  - OSXMLDecodeBuffer, 115
  - OSXMLEncodeBuffer, 120
  - OSXMLEncodeStream, 125
- IS\_XMLNSATTR
  - osrtxml.h, 160
- IS\_XSIATTR
  - osrtxml.h, 160
- isA
  - OSXMLDecodeBuffer, 115
  - OSXMLEncodeBuffer, 120
  - OSXMLEncodeStream, 125
- isWellFormed
  - OSXMLDecodeBuffer, 115
- mbOwnStream
  - OSXMLDecodeBuffer, 116
  - OSXMLEncodeStream, 126
- mpCtxt
  - OSXMLEncodeStream, 126
- mpStream
  - OSXMLEncodeStream, 126
- OSIntegerFmt, 111
- osrtxml.h, 140
  - IS\_XMLNSATTR, 160
  - IS\_XSIATTR, 160
  - OSXMLGroupDesc, 161
  - OSXMLQNAMEEQUALS, 161
  - OSXMLSETUTF8DECPTR, 161
  - rtSaxGetAttrValue, 161
  - rtSaxGetElemID, 161
  - rtSaxGetElemID8, 162
  - rtSaxHasXMLNSAttrs, 162
  - rtSaxIsEmptyBuffer, 162
  - rtSaxSortAttrs, 163
  - rtSaxStrListMatch, 163
  - rtSaxStrListParse, 163
  - rtXmlCmpBase64Str, 164
  - rtXmlCmpHexStr, 164
  - rtXmlCreateFileInputSource, 164

- rtXmlInitContext, 165
- rtXmlInitContextUsingKey, 165
- rtXmlInitCtxAppInfo, 165
- rtXmlMatchBase64Str, 165
- rtXmlMatchDate, 166
- rtXmlMatchDateTime, 166
- rtXmlMatchGDay, 166
- rtXmlMatchGMonth, 166
- rtXmlMatchGMonthDay, 167
- rtXmlMatchGYear, 167
- rtXmlMatchGYearMonth, 167
- rtXmlMatchHexStr, 167
- rtXmlMatchTime, 168
- rtXmlMemFreeAnyAttrs, 168
- rtXmlNewQName, 168
- rtXmlPrepareContext, 169
- rtXmlSetEncC14N, 169
- rtXmlSetEncDocHdr, 169
- rtXmlSetEncodingStr, 169
- rtXmlSetEncXSINamespace, 170
- rtXmlSetEncXSINilAttr, 170
- rtXmlSetFormatting, 170
- rtXmlSetIndent, 170
- rtXmlSetIndentChar, 171
- rtXmlSetNamespacesSet, 171
- rtXmlSetNoNSSchemaLocation, 171
- rtXmlSetNSPrefixLinks, 171
- rtXmlSetSchemaLocation, 172
- rtXmlSetSoapVersion, 172
- rtXmlSetWriteBOM, 172
- rtXmlSetXSITypeAttr, 172
- OSXMLCtxtInfo, 112
- OSXMLDecodeBuffer, 113
  - decodeXML, 115
  - init, 115
  - isA, 115
  - isWellFormed, 115
  - mbOwnStream, 116
  - OSXMLDecodeBuffer, 114
  - parseElementName, 115
  - parseElemQName, 116
  - setMaxErrors, 116
- OSXMLDecodeBuffer.h, 174
- OSXMLElemIDRec, 117
- OSXMLEncodeBuffer, 118
  - addXMLHeader, 119
  - addXMLText, 119
  - getMsgLen, 119
  - init, 120
  - isA, 120
  - OSXMLEncodeBuffer, 119
  - setFragment, 120
  - write, 120
- OSXMLEncodeBuffer.h, 175
- OSXMLEncodeStream, 122
  - encodeAttr, 124
  - encodeText, 124
  - endElement, 124
  - getMsgPtr, 124
  - getStream, 125
  - init, 125
  - isA, 125
  - mbOwnStream, 126
  - mpCtxt, 126
  - mpStream, 126
  - OSXMLEncodeStream, 123
  - startDocument, 125
  - startElement, 125
  - termStartElement, 126
- OSXMLEncodeStream.h, 176
- OSXMLFacets, 128
- OSXMLGroupDesc, 129
  - osrtxml.h, 161
- OSXMLItemDescr, 130
- OSXMLMessageBuffer, 131
  - getIndent, 132
  - getIndentChar, 132
  - getWriteBOM, 132
  - OSXMLMessageBuffer, 132
  - setAppInfo, 132
  - setFormatting, 133
  - setIndent, 133
  - setIndentChar, 133
  - setNamespace, 133
  - setWriteBOM, 134
- OSXMLMessageBuffer.h, 177
- OSXMLNameFragments, 135
- OSXMLQName, 136
- OSXMLQNAMEEQUALS
  - osrtxml.h, 161
- OSXMLSETUTF8DECPTR
  - osrtxml.h, 161
- OSXMLSortedAttrOffset, 137
- OSXMLStrFragment, 138
- OSXSDAnyType, 139
- parseElementName
  - OSXMLDecodeBuffer, 115
- parseElemQName
  - OSXMLDecodeBuffer, 116
- rtAsn1XmlAddAnyAttr
  - asn1xml, 7
- rtAsn1XmlEncGenTime
  - asn1xml, 8
- rtAsn1XmlEncObjId
  - asn1xml, 8
- rtAsn1XmlEncOpenType

- asn1xml, 8
- rtAsn1XmlEncOpenTypeExt
  - asn1xml, 9
- rtAsn1XmlEncReal
  - asn1xml, 9
- rtAsn1XmlEncRelOID
  - asn1xml, 10
- rtAsn1XmlEncUnivStr
  - asn1xml, 10
- rtAsn1XmlEncUTCtime
  - asn1xml, 10
- rtAsn1XmlFmtAttrStr
  - asn1xml, 11
- rtAsn1XmlParseAttrStr
  - asn1xml, 11
- rtAsn1XmlpDecDynBitStr
  - asn1xml, 12
- rtAsn1XmlpDecDynBitStr64
  - asn1xml, 12
- rtAsn1XmlpDecGenTime
  - asn1xml, 12
- rtAsn1XmlpDecObjId
  - asn1xml, 13
- rtAsn1XmlpDecOpenType
  - asn1xml, 13
- rtAsn1XmlpDecReal
  - asn1xml, 13
- rtAsn1XmlpDecRelOID
  - asn1xml, 14
- rtAsn1XmlpDecUnivStr
  - asn1xml, 14
- rtAsn1XmlpDecUTCtime
  - asn1xml, 14
- rtSaxGetAttrValue
  - osrtxml.h, 161
- rtSaxGetElemID
  - osrtxml.h, 161
- rtSaxGetElemID8
  - osrtxml.h, 162
- rtSaxHasXMLNSAttrs
  - osrtxml.h, 162
- rtSaxIsEmptyBuffer
  - osrtxml.h, 162
- rtSaxSortAttrs
  - osrtxml.h, 163
- rtSaxStrListMatch
  - osrtxml.h, 163
- rtSaxStrListParse
  - osrtxml.h, 163
- rtXmlCmpBase64Str
  - osrtxml.h, 164
- rtXmlCmpHexStr
  - osrtxml.h, 164
- rtXmlCreateFileInputSource
  - osrtxml.h, 164
- rtXmlDec
  - rtXmlDecBase64Binary, 18
  - rtXmlDecBase64Str, 18
  - rtXmlDecBase64StrValue, 19
  - rtXmlDecBigInt, 19
  - rtXmlDecBool, 20
  - rtXmlDecDate, 20
  - rtXmlDecDateTime, 20
  - rtXmlDecDecimal, 21
  - rtXmlDecDouble, 21
  - rtXmlDecDynBase64Str, 21
  - rtXmlDecDynHexStr, 22
  - rtXmlDecDynUTF8Str, 22
  - rtXmlDecEmptyElement, 22
  - rtXmlDecGDay, 23
  - rtXmlDecGMonth, 23
  - rtXmlDecGMonthDay, 23
  - rtXmlDecGYear, 24
  - rtXmlDecGYearMonth, 24
  - rtXmlDecHexBinary, 24
  - rtXmlDecHexStr, 25
  - rtXmlDecInt, 25
  - rtXmlDecInt16, 26
  - rtXmlDecInt64, 26
  - rtXmlDecInt8, 26
  - rtXmlDecNSAttr, 27
  - rtXmlDecQName, 27
  - rtXmlDecTime, 28
  - rtXmlDecUInt, 28
  - rtXmlDecUInt16, 28
  - rtXmlDecUInt64, 29
  - rtXmlDecUInt8, 29
  - rtXmlDecUTF8Str, 29
  - rtXmlDecXmlStr, 30
  - rtXmlDecXSIAAttr, 30
  - rtXmlDecXSIAAttrs, 30
  - rtXmlParseElementName, 31
  - rtXmlParseElemQName, 31
- rtXmlDecBase64Binary
  - rtXmlDec, 18
- rtXmlDecBase64Str
  - rtXmlDec, 18
- rtXmlDecBase64StrValue
  - rtXmlDec, 19
- rtXmlDecBigInt
  - rtXmlDec, 19
- rtXmlDecBool
  - rtXmlDec, 20
- rtXmlDecDate
  - rtXmlDec, 20
- rtXmlDecDateTime
  - rtXmlDec, 20
- rtXmlDecDecimal

- rtXmlDec, 21
- rtXmlDecDouble
  - rtXmlDec, 21
- rtXmlDecDynBase64Str
  - rtXmlDec, 21
- rtXmlDecDynHexStr
  - rtXmlDec, 22
- rtXmlDecDynUTF8Str
  - rtXmlDec, 22
- rtXmlDecEmptyElement
  - rtXmlDec, 22
- rtXmlDecGDay
  - rtXmlDec, 23
- rtXmlDecGMonth
  - rtXmlDec, 23
- rtXmlDecGMonthDay
  - rtXmlDec, 23
- rtXmlDecGYear
  - rtXmlDec, 24
- rtXmlDecGYearMonth
  - rtXmlDec, 24
- rtXmlDecHexBinary
  - rtXmlDec, 24
- rtXmlDecHexStr
  - rtXmlDec, 25
- rtXmlDecInt
  - rtXmlDec, 25
- rtXmlDecInt16
  - rtXmlDec, 26
- rtXmlDecInt64
  - rtXmlDec, 26
- rtXmlDecInt8
  - rtXmlDec, 26
- rtXmlDecNSAttr
  - rtXmlDec, 27
- rtXmlDecQName
  - rtXmlDec, 27
- rtXmlDecTime
  - rtXmlDec, 28
- rtXmlDecUInt
  - rtXmlDec, 28
- rtXmlDecUInt16
  - rtXmlDec, 28
- rtXmlDecUInt64
  - rtXmlDec, 29
- rtXmlDecUInt8
  - rtXmlDec, 29
- rtXmlDecUTF8Str
  - rtXmlDec, 29
- rtXmlDecXmlStr
  - rtXmlDec, 30
- rtXmlDecXSIAttr
  - rtXmlDec, 30
- rtXmlDecXSIAttrs

- rtXmlDec, 30
- rtXmlEnc
  - rtXmlEncAny, 39
  - rtXmlEncAnyAttr, 40
  - rtXmlEncAnyTypeValue, 40
  - rtXmlEncBase64Binary, 40
  - rtXmlEncBase64BinaryAttr, 41
  - rtXmlEncBase64StrValue, 41
  - rtXmlEncBigInt, 41
  - rtXmlEncBigIntAttr, 42
  - rtXmlEncBigIntValue, 42
  - rtXmlEncBinStrValue, 43
  - rtXmlEncBitString, 43
  - rtXmlEncBOM, 43
  - rtXmlEncBool, 44
  - rtXmlEncBoolAttr, 44
  - rtXmlEncBoolValue, 44
  - rtXmlEncComment, 45
  - rtXmlEncDate, 45
  - rtXmlEncDateTime, 45
  - rtXmlEncDateTimeValue, 46
  - rtXmlEncDateValue, 46
  - rtXmlEncDecimal, 46
  - rtXmlEncDecimalAttr, 47
  - rtXmlEncDecimalValue, 47
  - rtXmlEncDouble, 48
  - rtXmlEncDoubleAttr, 48
  - rtXmlEncDoubleNormalValue, 48
  - rtXmlEncDoubleValue, 49
  - rtXmlEncEmptyElement, 49
  - rtXmlEncEndDocument, 50
  - rtXmlEncEndElement, 50
  - rtXmlEncEndSoapElems, 50
  - rtXmlEncEndSoapEnv, 51
  - rtXmlEncFloat, 51
  - rtXmlEncFloatAttr, 51
  - rtXmlEncGDay, 52
  - rtXmlEncGDayValue, 52
  - rtXmlEncGMonth, 52
  - rtXmlEncGMonthDay, 53
  - rtXmlEncGMonthDayValue, 53
  - rtXmlEncGMonthValue, 53
  - rtXmlEncGYear, 54
  - rtXmlEncGYearMonth, 54
  - rtXmlEncGYearMonthValue, 54
  - rtXmlEncGYearValue, 55
  - rtXmlEncHexBinary, 55
  - rtXmlEncHexBinaryAttr, 55
  - rtXmlEncHexStrValue, 56
  - rtXmlEncIndent, 56
  - rtXmlEncInt, 56
  - rtXmlEncInt64, 57
  - rtXmlEncInt64Attr, 57
  - rtXmlEncInt64Value, 58

- rtXmlEncIntAttr, [58](#)
- rtXmlEncIntPattern, [58](#)
- rtXmlEncIntValue, [59](#)
- rtXmlEncNamedBits, [59](#)
- rtXmlEncNSAttrs, [60](#)
- rtXmlEncReal10, [60](#)
- rtXmlEncSoapArrayTypeAttr, [60](#)
- rtXmlEncStartDocument, [61](#)
- rtXmlEncStartElement, [61](#)
- rtXmlEncStartSoapElems, [62](#)
- rtXmlEncStartSoapEnv, [62](#)
- rtXmlEncString, [62](#)
- rtXmlEncStringValue, [63](#)
- rtXmlEncStringValue2, [63](#)
- rtXmlEncTermStartElement, [63](#)
- rtXmlEncTime, [64](#)
- rtXmlEncTimeValue, [64](#)
- rtXmlEncUInt, [64](#)
- rtXmlEncUInt64, [65](#)
- rtXmlEncUInt64Attr, [65](#)
- rtXmlEncUInt64Value, [66](#)
- rtXmlEncUIntAttr, [66](#)
- rtXmlEncUIntValue, [66](#)
- rtXmlEncUnicodeStr, [67](#)
- rtXmlEncUTF8Attr, [67](#)
- rtXmlEncUTF8Attr2, [67](#)
- rtXmlEncUTF8Str, [68](#)
- rtXmlEncXSIAAttrs, [68](#)
- rtXmlEncXSINilAttr, [69](#)
- rtXmlEncXSITypeAttr, [69](#)
- rtXmlEncXSITypeAttr2, [69](#)
- rtXmlFinalizeMemBuf, [39](#)
- rtXmlFreeInputSource, [70](#)
- rtXmlGetEncBufLen, [39](#)
- rtXmlGetEncBufPtr, [39](#)
- rtXmlGetIndent, [70](#)
- rtXmlGetIndentChar, [70](#)
- rtXmlGetWriteBOM, [70](#)
- rtXmlPrintNSAttrs, [71](#)
- rtXmlSetEncBufPtr, [71](#)
- rtXmlEncAny
  - rtXmlEnc, [39](#)
- rtXmlEncAnyAttr
  - rtXmlEnc, [40](#)
- rtXmlEncAnyTypeValue
  - rtXmlEnc, [40](#)
- rtXmlEncAttrC14N
  - rtXmlpDec, [78](#)
- rtXmlEncBase64Binary
  - rtXmlEnc, [40](#)
- rtXmlEncBase64BinaryAttr
  - rtXmlEnc, [41](#)
- rtXmlEncBase64StrValue
  - rtXmlEnc, [41](#)
- rtXmlEncBigInt
  - rtXmlEnc, [41](#)
- rtXmlEncBigIntAttr
  - rtXmlEnc, [42](#)
- rtXmlEncBigIntValue
  - rtXmlEnc, [42](#)
- rtXmlEncBinStrValue
  - rtXmlEnc, [43](#)
- rtXmlEncBitString
  - rtXmlEnc, [43](#)
- rtXmlEncBOM
  - rtXmlEnc, [43](#)
- rtXmlEncBool
  - rtXmlEnc, [44](#)
- rtXmlEncBoolAttr
  - rtXmlEnc, [44](#)
- rtXmlEncBoolValue
  - rtXmlEnc, [44](#)
- rtXmlEncComment
  - rtXmlEnc, [45](#)
- rtXmlEncDate
  - rtXmlEnc, [45](#)
- rtXmlEncDateTime
  - rtXmlEnc, [45](#)
- rtXmlEncDateTimeValue
  - rtXmlEnc, [46](#)
- rtXmlEncDateValue
  - rtXmlEnc, [46](#)
- rtXmlEncDecimal
  - rtXmlEnc, [46](#)
- rtXmlEncDecimalAttr
  - rtXmlEnc, [47](#)
- rtXmlEncDecimalValue
  - rtXmlEnc, [47](#)
- rtXmlEncDouble
  - rtXmlEnc, [48](#)
- rtXmlEncDoubleAttr
  - rtXmlEnc, [48](#)
- rtXmlEncDoubleNormalValue
  - rtXmlEnc, [48](#)
- rtXmlEncDoubleValue
  - rtXmlEnc, [49](#)
- rtXmlEncEmptyElement
  - rtXmlEnc, [49](#)
- rtXmlEncEndDocument
  - rtXmlEnc, [50](#)
- rtXmlEncEndElement
  - rtXmlEnc, [50](#)
- rtXmlEncEndSoapElems
  - rtXmlEnc, [50](#)
- rtXmlEncEndSoapEnv
  - rtXmlEnc, [51](#)
- rtXmlEncFloat
  - rtXmlEnc, [51](#)



rtXmlEncFloatAttr	rtXmlEncStartElement
rtXmlEnc, <a href="#">51</a>	rtXmlEnc, <a href="#">61</a>
rtXmlEncGDay	rtXmlEncStartSoapElems
rtXmlEnc, <a href="#">52</a>	rtXmlEnc, <a href="#">62</a>
rtXmlEncGDayValue	rtXmlEncStartSoapEnv
rtXmlEnc, <a href="#">52</a>	rtXmlEnc, <a href="#">62</a>
rtXmlEncGMonth	rtXmlEncString
rtXmlEnc, <a href="#">52</a>	rtXmlEnc, <a href="#">62</a>
rtXmlEncGMonthDay	rtXmlEncStringValue
rtXmlEnc, <a href="#">53</a>	rtXmlEnc, <a href="#">63</a>
rtXmlEncGMonthDayValue	rtXmlEncStringValue2
rtXmlEnc, <a href="#">53</a>	rtXmlEnc, <a href="#">63</a>
rtXmlEncGMonthValue	rtXmlEncTermStartElement
rtXmlEnc, <a href="#">53</a>	rtXmlEnc, <a href="#">63</a>
rtXmlEncGYear	rtXmlEncTime
rtXmlEnc, <a href="#">54</a>	rtXmlEnc, <a href="#">64</a>
rtXmlEncGYearMonth	rtXmlEncTimeValue
rtXmlEnc, <a href="#">54</a>	rtXmlEnc, <a href="#">64</a>
rtXmlEncGYearMonthValue	rtXmlEncUInt
rtXmlEnc, <a href="#">54</a>	rtXmlEnc, <a href="#">64</a>
rtXmlEncGYearValue	rtXmlEncUInt64
rtXmlEnc, <a href="#">55</a>	rtXmlEnc, <a href="#">65</a>
rtXmlEncHexBinary	rtXmlEncUInt64Attr
rtXmlEnc, <a href="#">55</a>	rtXmlEnc, <a href="#">65</a>
rtXmlEncHexBinaryAttr	rtXmlEncUInt64Value
rtXmlEnc, <a href="#">55</a>	rtXmlEnc, <a href="#">66</a>
rtXmlEncHexStringValue	rtXmlEncUIntAttr
rtXmlEnc, <a href="#">56</a>	rtXmlEnc, <a href="#">66</a>
rtXmlEncIndent	rtXmlEncUIntValue
rtXmlEnc, <a href="#">56</a>	rtXmlEnc, <a href="#">66</a>
rtXmlEncInt	rtXmlEncUnicodeStr
rtXmlEnc, <a href="#">56</a>	rtXmlEnc, <a href="#">67</a>
rtXmlEncInt64	rtXmlEncUTF8Attr
rtXmlEnc, <a href="#">57</a>	rtXmlEnc, <a href="#">67</a>
rtXmlEncInt64Attr	rtXmlEncUTF8Attr2
rtXmlEnc, <a href="#">57</a>	rtXmlEnc, <a href="#">67</a>
rtXmlEncInt64Value	rtXmlEncUTF8Str
rtXmlEnc, <a href="#">58</a>	rtXmlEnc, <a href="#">68</a>
rtXmlEncIntAttr	rtXmlEncXSIAttrs
rtXmlEnc, <a href="#">58</a>	rtXmlEnc, <a href="#">68</a>
rtXmlEncIntPattern	rtXmlEncXSINilAttr
rtXmlEnc, <a href="#">58</a>	rtXmlEnc, <a href="#">69</a>
rtXmlEncIntValue	rtXmlEncXSITypeAttr
rtXmlEnc, <a href="#">59</a>	rtXmlEnc, <a href="#">69</a>
rtXmlEncNamedBits	rtXmlEncXSITypeAttr2
rtXmlEnc, <a href="#">59</a>	rtXmlEnc, <a href="#">69</a>
rtXmlEncNSAttrs	rtXmlEncErrCodes.h, <a href="#">178</a>
rtXmlEnc, <a href="#">60</a>	rtXmlFinalizeMemBuf
rtXmlEncReal10	rtXmlEnc, <a href="#">39</a>
rtXmlEnc, <a href="#">60</a>	rtXmlFreeInputSource
rtXmlEncSoapArrayTypeAttr	rtXmlEnc, <a href="#">70</a>
rtXmlEnc, <a href="#">60</a>	rtXmlGetEncBufLen
rtXmlEncStartDocument	rtXmlEnc, <a href="#">39</a>
rtXmlEnc, <a href="#">61</a>	rtXmlGetEncBufPtr

- rtXmlEnc, 39
- rtXmlGetIndent
  - rtXmlEnc, 70
- rtXmlGetIndentChar
  - rtXmlEnc, 70
- rtXmlGetWriteBOM
  - rtXmlEnc, 70
- rtXmlInitContext
  - osrtxml.h, 165
- rtXmlInitContextUsingKey
  - osrtxml.h, 165
- rtXmlInitCtxtAppInfo
  - osrtxml.h, 165
- rtXmlMatchBase64Str
  - osrtxml.h, 165
- rtXmlMatchDate
  - osrtxml.h, 166
- rtXmlMatchDateTime
  - osrtxml.h, 166
- rtXmlMatchGDay
  - osrtxml.h, 166
- rtXmlMatchGMonth
  - osrtxml.h, 166
- rtXmlMatchGMonthDay
  - osrtxml.h, 167
- rtXmlMatchGYear
  - osrtxml.h, 167
- rtXmlMatchGYearMonth
  - osrtxml.h, 167
- rtXmlMatchHexStr
  - osrtxml.h, 167
- rtXmlMatchTime
  - osrtxml.h, 168
- rtXmlMemFreeAnyAttrs
  - osrtxml.h, 168
- rtXmlNewQName
  - osrtxml.h, 168
- rtXmlParseElementName
  - rtXmlDec, 31
- rtXmlParseElemQName
  - rtXmlDec, 31
- rtXmlpCountListItems
  - rtXmlpDec, 78
- rtXmlpCreateReader
  - rtXmlpDec, 79
- rtXmlpDec
  - rtXmlEncAttrC14N, 78
  - rtXmlpCountListItems, 78
  - rtXmlpCreateReader, 79
  - rtXmlpDecAny, 79
  - rtXmlpDecAny2, 79
  - rtXmlpDecAnyAttrStr, 80
  - rtXmlpDecAnyElem, 80
  - rtXmlpDecBase64Str, 81

- rtXmlpDecBigInt, 81
- rtXmlpDecBitString, 81
- rtXmlpDecBool, 82
- rtXmlpDecDate, 82
- rtXmlpDecDateTime, 82
- rtXmlpDecDecimal, 83
- rtXmlpDecDouble, 83
- rtXmlpDecDoubleExt, 83
- rtXmlpDecDynBase64Str, 84
- rtXmlpDecDynBitString, 84
- rtXmlpDecDynHexStr, 85
- rtXmlpDecDynUnicodeStr, 85
- rtXmlpDecDynUTF8Str, 85
- rtXmlpDecGDay, 86
- rtXmlpDecGMonth, 86
- rtXmlpDecGMonthDay, 86
- rtXmlpDecGYear, 87
- rtXmlpDecGYearMonth, 87
- rtXmlpDecHexStr, 87
- rtXmlpDecInt, 88
- rtXmlpDecInt16, 88
- rtXmlpDecInt64, 89
- rtXmlpDecInt8, 89
- rtXmlpDecNamedBits, 89
- rtXmlpDecStrList, 90
- rtXmlpDecTime, 90
- rtXmlpDecUInt, 90
- rtXmlpDecUInt16, 91
- rtXmlpDecUInt64, 91
- rtXmlpDecUInt8, 91
- rtXmlpDecUTF8Str, 92
- rtXmlpDecXmlStr, 92
- rtXmlpDecXmlStrList, 92
- rtXmlpDecXSIAAttr, 93
- rtXmlpDecXSIAAttrs, 93
- rtXmlpDecXSISTypeAttr, 93
- rtXmlpForceDecodeAsGroup, 94
- rtXmlpGetAttributeCount, 94
- rtXmlpGetAttributeID, 94
- rtXmlpGetCurrentLevel, 95
- rtXmlpGetNextAllElemID, 95
- rtXmlpGetNextAllElemID16, 95
- rtXmlpGetNextAllElemID32, 96
- rtXmlpGetNextElem, 96
- rtXmlpGetNextElemID, 97
- rtXmlpGetNextSeqElemID, 97
- rtXmlpGetNextSeqElemID2, 98
- rtXmlpGetNextSeqElemIDExt, 98
- rtXmlpGetReader, 99
- rtXmlpGetXmInsAttrs, 99
- rtXmlpGetXSITypeAttr, 100
- rtXmlpGetXSITypeIndex, 100
- rtXmlpHasAttributes, 100
- rtXmlpHideAttributes, 101

- rtXmlpIsDecodeAsGroup, 101
- rtXmlpIsEmptyElement, 101
- rtXmlpIsLastEventDone, 101
- rtXmlpIsUTF8Encoding, 102
- rtXmlpListHasItem, 102
- rtXmlpLookupXSITypeIndex, 102
- rtXmlpMarkLastEventActive, 102
- rtXmlpMarkPos, 103
- rtXmlpMatchEndTag, 103
- rtXmlpMatchStartTag, 103
- rtXmlpNeedDecodeAttributes, 104
- rtXmlpReadBytes, 104
- rtXmlpResetMarkedPos, 104
- rtXmlpRewindToMarkedPos, 104
- rtXmlpSelectAttribute, 105
- rtXmlpSetListMode, 105
- rtXmlpSetMixedContentMode, 105
- rtXmlpSetNamespaceTable, 105
- rtXmlpSetWhiteSpaceMode, 106
- rtXmlpDecAny
  - rtXmlpDec, 79
- rtXmlpDecAny2
  - rtXmlpDec, 79
- rtXmlpDecAnyAttrStr
  - rtXmlpDec, 80
- rtXmlpDecAnyElem
  - rtXmlpDec, 80
- rtXmlpDecBase64Str
  - rtXmlpDec, 81
- rtXmlpDecBigInt
  - rtXmlpDec, 81
- rtXmlpDecBitString
  - rtXmlpDec, 81
- rtXmlpDecBool
  - rtXmlpDec, 82
- rtXmlpDecDate
  - rtXmlpDec, 82
- rtXmlpDecDateTime
  - rtXmlpDec, 82
- rtXmlpDecDecimal
  - rtXmlpDec, 83
- rtXmlpDecDouble
  - rtXmlpDec, 83
- rtXmlpDecDoubleExt
  - rtXmlpDec, 83
- rtXmlpDecDynBase64Str
  - rtXmlpDec, 84
- rtXmlpDecDynBitString
  - rtXmlpDec, 84
- rtXmlpDecDynHexStr
  - rtXmlpDec, 85
- rtXmlpDecDynUnicodeStr
  - rtXmlpDec, 85
- rtXmlpDecDynUTF8Str
  - rtXmlpDec, 85
- rtXmlpDecGDay
  - rtXmlpDec, 86
- rtXmlpDecGMonth
  - rtXmlpDec, 86
- rtXmlpDecGMonthDay
  - rtXmlpDec, 86
- rtXmlpDecGYear
  - rtXmlpDec, 87
- rtXmlpDecGYearMonth
  - rtXmlpDec, 87
- rtXmlpDecHexStr
  - rtXmlpDec, 87
- rtXmlpDecInt
  - rtXmlpDec, 88
- rtXmlpDecInt16
  - rtXmlpDec, 88
- rtXmlpDecInt64
  - rtXmlpDec, 89
- rtXmlpDecInt8
  - rtXmlpDec, 89
- rtXmlpDecListOfASN1DynBitStr
  - asn1xml, 15
- rtXmlpDecNamedBits
  - rtXmlpDec, 89
- rtXmlpDecStrList
  - rtXmlpDec, 90
- rtXmlpDecTime
  - rtXmlpDec, 90
- rtXmlpDecUInt
  - rtXmlpDec, 90
- rtXmlpDecUInt16
  - rtXmlpDec, 91
- rtXmlpDecUInt64
  - rtXmlpDec, 91
- rtXmlpDecUInt8
  - rtXmlpDec, 91
- rtXmlpDecUTF8Str
  - rtXmlpDec, 92
- rtXmlpDecXmlStr
  - rtXmlpDec, 92
- rtXmlpDecXmlStrList
  - rtXmlpDec, 92
- rtXmlpDecXSIAAttr
  - rtXmlpDec, 93
- rtXmlpDecXSIAAttrs
  - rtXmlpDec, 93
- rtXmlpDecXSITypeAttr
  - rtXmlpDec, 93
- rtXmlpForceDecodeAsGroup
  - rtXmlpDec, 94
- rtXmlpGetAttributeCount
  - rtXmlpDec, 94
- rtXmlpGetAttributeID
  - rtXmlpDec, 94

[rtXmlpDec](#), [94](#)  
[rtXmlpGetCurrentLevel](#)  
[rtXmlpDec](#), [95](#)  
[rtXmlpGetNextAllElemID](#)  
[rtXmlpDec](#), [95](#)  
[rtXmlpGetNextAllElemID16](#)  
[rtXmlpDec](#), [95](#)  
[rtXmlpGetNextAllElemID32](#)  
[rtXmlpDec](#), [96](#)  
[rtXmlpGetNextElem](#)  
[rtXmlpDec](#), [96](#)  
[rtXmlpGetNextElemID](#)  
[rtXmlpDec](#), [97](#)  
[rtXmlpGetNextSeqElemID](#)  
[rtXmlpDec](#), [97](#)  
[rtXmlpGetNextSeqElemID2](#)  
[rtXmlpDec](#), [98](#)  
[rtXmlpGetNextSeqElemIDExt](#)  
[rtXmlpDec](#), [98](#)  
[rtXmlpGetReader](#)  
[rtXmlpDec](#), [99](#)  
[rtXmlpGetXmInAttrs](#)  
[rtXmlpDec](#), [99](#)  
[rtXmlpGetXSITypeAttr](#)  
[rtXmlpDec](#), [100](#)  
[rtXmlpGetXSITypeIndex](#)  
[rtXmlpDec](#), [100](#)  
[rtXmlpHasAttributes](#)  
[rtXmlpDec](#), [100](#)  
[rtXmlpHideAttributes](#)  
[rtXmlpDec](#), [101](#)  
[rtXmlpIsDecodeAsGroup](#)  
[rtXmlpDec](#), [101](#)  
[rtXmlpIsEmptyElement](#)  
[rtXmlpDec](#), [101](#)  
[rtXmlpIsLastEventDone](#)  
[rtXmlpDec](#), [101](#)  
[rtXmlpIsUTF8Encoding](#)  
[rtXmlpDec](#), [102](#)  
[rtXmlpListHasItem](#)  
[rtXmlpDec](#), [102](#)  
[rtXmlpLookupXSITypeIndex](#)  
[rtXmlpDec](#), [102](#)  
[rtXmlpMarkLastEventActive](#)  
[rtXmlpDec](#), [102](#)  
[rtXmlpMarkPos](#)  
[rtXmlpDec](#), [103](#)  
[rtXmlpMatchEndTag](#)  
[rtXmlpDec](#), [103](#)  
[rtXmlpMatchStartTag](#)  
[rtXmlpDec](#), [103](#)  
[rtXmlpNeedDecodeAttributes](#)  
[rtXmlpDec](#), [104](#)  
[rtXmlpReadBytes](#)  
[rtXmlpDec](#), [104](#)  
[rtXmlPrepareContext](#)  
[osrtxml.h](#), [169](#)  
[rtXmlpResetMarkedPos](#)  
[rtXmlpDec](#), [104](#)  
[rtXmlpRewindToMarkedPos](#)  
[rtXmlpDec](#), [104](#)  
[rtXmlPrintNSAttrs](#)  
[rtXmlEnc](#), [71](#)  
[rtXmlpSelectAttribute](#)  
[rtXmlpDec](#), [105](#)  
[rtXmlpSetListMode](#)  
[rtXmlpDec](#), [105](#)  
[rtXmlpSetMixedContentMode](#)  
[rtXmlpDec](#), [105](#)  
[rtXmlpSetNamespaceTable](#)  
[rtXmlpDec](#), [105](#)  
[rtXmlpSetWhiteSpaceMode](#)  
[rtXmlpDec](#), [106](#)  
[rtXmlSetEncBufPtr](#)  
[rtXmlEnc](#), [71](#)  
[rtXmlSetEncC14N](#)  
[osrtxml.h](#), [169](#)  
[rtXmlSetEncDocHdr](#)  
[osrtxml.h](#), [169](#)  
[rtXmlSetEncodingStr](#)  
[osrtxml.h](#), [169](#)  
[rtXmlSetEncXSINamespace](#)  
[osrtxml.h](#), [170](#)  
[rtXmlSetEncXSINilAttr](#)  
[osrtxml.h](#), [170](#)  
[rtXmlSetFormatting](#)  
[osrtxml.h](#), [170](#)  
[rtXmlSetIndent](#)  
[osrtxml.h](#), [170](#)  
[rtXmlSetIndentChar](#)  
[osrtxml.h](#), [171](#)  
[rtXmlSetNamespacesSet](#)  
[osrtxml.h](#), [171](#)  
[rtXmlSetNoNSSchemaLocation](#)  
[osrtxml.h](#), [171](#)  
[rtXmlSetNSPrefixLinks](#)  
[osrtxml.h](#), [171](#)  
[rtXmlSetSchemaLocation](#)  
[osrtxml.h](#), [172](#)  
[rtXmlSetSoapVersion](#)  
[osrtxml.h](#), [172](#)  
[rtXmlSetWriteBOM](#)  
[osrtxml.h](#), [172](#)  
[rtXmlSetXSITypeAttr](#)  
[osrtxml.h](#), [172](#)  
[rtXmlUtil](#)  
[rtXmlWriteToFile](#), [72](#)  
[rtXmlWriteToFile](#)

- rtXmlUtil, [72](#)
- setAppInfo
  - OSXMLMessageBuffer, [132](#)
- setFormatting
  - OSXMLMessageBuffer, [133](#)
- setFragment
  - OSXMLEncodeBuffer, [120](#)
- setIndent
  - OSXMLMessageBuffer, [133](#)
- setIndentChar
  - OSXMLMessageBuffer, [133](#)
- setMaxErrors
  - OSXMLDecodeBuffer, [116](#)
- setNamespace
  - OSXMLMessageBuffer, [133](#)
- setWriteBOM
  - OSXMLMessageBuffer, [134](#)
- startDocument
  - OSXMLEncodeStream, [125](#)
- startElement
  - OSXMLEncodeStream, [125](#)
- termStartElement
  - OSXMLEncodeStream, [126](#)
- write
  - OSXMLEncodeBuffer, [120](#)
- XML decode functions., [16](#)
- XML encode functions., [32](#)
- XML pull-parser decode functions., [73](#)
- XML run-time error status codes., [107](#)
- XML utility functions., [72](#)
- XML\_E\_BASE
  - xmlErrCodes, [108](#)
- XML\_E\_ELEMMISRQ
  - xmlErrCodes, [108](#)
- XML\_E\_ELEMSMISRQ
  - xmlErrCodes, [109](#)
- XML\_E\_FLDABSENT
  - xmlErrCodes, [109](#)
- XML\_E\_NOMATCH
  - xmlErrCodes, [109](#)
- XML\_E\_NSURINOTFOU
  - xmlErrCodes, [109](#)
- XML\_E\_TAGMISMATCH
  - xmlErrCodes, [109](#)
- XML\_OK\_EOB
  - xmlErrCodes, [109](#)
- XML\_OK\_FRAG
  - xmlErrCodes, [109](#)
- xmlErrCodes
  - XML\_E\_BASE, [108](#)
  - XML\_E\_ELEMMISRQ, [108](#)
  - XML\_E\_ELEMSMISRQ, [109](#)
  - XML\_E\_FLDABSENT, [109](#)
  - XML\_E\_NOMATCH, [109](#)
  - XML\_E\_NSURINOTFOU, [109](#)
  - XML\_E\_TAGMISMATCH, [109](#)
  - XML\_OK\_EOB, [109](#)
  - XML\_OK\_FRAG, [109](#)