

# **XBinder**

---

XML Schema Compiler  
Version 2.9  
C 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 ©1997–2023 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 XBinder may be submitted via electronic mail to [info@obj-sys.com](mailto:info@obj-sys.com).



# Contents

<b>1</b>	<b>C XML Runtime Library Functions</b>	<b>1</b>
<b>2</b>	<b>C DOM Runtime Library Functions</b>	<b>3</b>
<b>3</b>	<b>Module Index</b>	<b>5</b>
3.1	Modules . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List . . . . .	7
<b>5</b>	<b>File Index</b>	<b>9</b>
5.1	File List . . . . .	9
<b>6</b>	<b>Module Documentation</b>	<b>11</b>
6.1	XML decode functions. . . . .	11
6.1.1	Detailed Description . . . . .	13
6.1.2	Function Documentation . . . . .	13
6.1.2.1	rtXmlDecBase64Binary() . . . . .	13
6.1.2.2	rtXmlDecBase64Str() . . . . .	14
6.1.2.3	rtXmlDecBase64Str64() . . . . .	14
6.1.2.4	rtXmlDecBase64StrValue() . . . . .	15
6.1.2.5	rtXmlDecBase64StrValue64() . . . . .	16
6.1.2.6	rtXmlDecBigInt() . . . . .	16
6.1.2.7	rtXmlDecBool() . . . . .	17

6.1.2.8	<code>rtXmlDecDate()</code>	17
6.1.2.9	<code>rtXmlDecDateTime()</code>	18
6.1.2.10	<code>rtXmlDecDecimal()</code>	18
6.1.2.11	<code>rtXmlDecDouble()</code>	19
6.1.2.12	<code>rtXmlDecDynBase64Str()</code>	19
6.1.2.13	<code>rtXmlDecDynBase64Str64()</code>	20
6.1.2.14	<code>rtXmlDecDynHexStr()</code>	20
6.1.2.15	<code>rtXmlDecDynHexStr64()</code>	21
6.1.2.16	<code>rtXmlDecDynUTF8Str()</code>	21
6.1.2.17	<code>rtXmlDecEmptyElement()</code>	22
6.1.2.18	<code>rtXmlDecGDay()</code>	22
6.1.2.19	<code>rtXmlDecGMonth()</code>	23
6.1.2.20	<code>rtXmlDecGMonthDay()</code>	23
6.1.2.21	<code>rtXmlDecGYear()</code>	24
6.1.2.22	<code>rtXmlDecGYearMonth()</code>	24
6.1.2.23	<code>rtXmlDecHexBinary()</code>	25
6.1.2.24	<code>rtXmlDecHexStr()</code>	25
6.1.2.25	<code>rtXmlDecHexStr64()</code>	26
6.1.2.26	<code>rtXmlDecInt()</code>	27
6.1.2.27	<code>rtXmlDecInt16()</code>	27
6.1.2.28	<code>rtXmlDecInt64()</code>	28
6.1.2.29	<code>rtXmlDecInt8()</code>	28
6.1.2.30	<code>rtXmlDecNSAttr()</code>	29
6.1.2.31	<code>rtXmlDecQName()</code>	29
6.1.2.32	<code>rtXmlDecTime()</code>	30
6.1.2.33	<code>rtXmlDecUInt()</code>	31
6.1.2.34	<code>rtXmlDecUInt16()</code>	31
6.1.2.35	<code>rtXmlDecUInt64()</code>	32

6.1.2.36	rtXmlDecUInt8()	32
6.1.2.37	rtXmlDecUTF8Str()	33
6.1.2.38	rtXmlDecXmlStr()	33
6.1.2.39	rtXmlDecXSIAAttr()	34
6.1.2.40	rtXmlDecXSIAAttrs()	34
6.1.2.41	rtXmlParseElementName()	35
6.1.2.42	rtXmlParseElemQName()	36
6.2	XML encode functions.	37
6.2.1	Detailed Description	41
6.2.2	Macro Definition Documentation	41
6.2.2.1	rtXmlGetEncBufLen	42
6.2.2.2	rtXmlGetEncBufPtr	42
6.2.3	Function Documentation	42
6.2.3.1	rtXmlEncAny()	42
6.2.3.2	rtXmlEncAnyAttr()	43
6.2.3.3	rtXmlEncAnyTypeValue()	43
6.2.3.4	rtXmlEncBase64Binary()	44
6.2.3.5	rtXmlEncBase64BinaryAttr()	44
6.2.3.6	rtXmlEncBase64StrValue()	45
6.2.3.7	rtXmlEncBigInt()	46
6.2.3.8	rtXmlEncBigIntAttr()	46
6.2.3.9	rtXmlEncBigIntValue()	47
6.2.3.10	rtXmlEncBinStrValue()	47
6.2.3.11	rtXmlEncBitString()	48
6.2.3.12	rtXmlEncBitStringExt()	48
6.2.3.13	rtXmlEncBOM()	49
6.2.3.14	rtXmlEncBool()	50
6.2.3.15	rtXmlEncBoolAttr()	50

6.2.3.16	rtXmlEncBoolValue()	51
6.2.3.17	rtXmlEncCanonicalSort()	51
6.2.3.18	rtXmlEncComment()	52
6.2.3.19	rtXmlEncDate()	52
6.2.3.20	rtXmlEncDateTime()	53
6.2.3.21	rtXmlEncDateTimeValue()	53
6.2.3.22	rtXmlEncDateValue()	54
6.2.3.23	rtXmlEncDecimal()	54
6.2.3.24	rtXmlEncDecimalAttr()	55
6.2.3.25	rtXmlEncDecimalValue()	55
6.2.3.26	rtXmlEncDouble()	56
6.2.3.27	rtXmlEncDoubleAttr()	57
6.2.3.28	rtXmlEncDoubleNormalValue()	57
6.2.3.29	rtXmlEncDoubleValue()	58
6.2.3.30	rtXmlEncEmptyElement()	58
6.2.3.31	rtXmlEncEndDocument()	59
6.2.3.32	rtXmlEncEndElement()	59
6.2.3.33	rtXmlEncEndSoapElems()	60
6.2.3.34	rtXmlEncEndSoapEnv()	60
6.2.3.35	rtXmlEncFloat()	61
6.2.3.36	rtXmlEncFloatAttr()	61
6.2.3.37	rtXmlEncGDay()	62
6.2.3.38	rtXmlEncGDayValue()	63
6.2.3.39	rtXmlEncGMonth()	63
6.2.3.40	rtXmlEncGMonthDay()	64
6.2.3.41	rtXmlEncGMonthDayValue()	64
6.2.3.42	rtXmlEncGMonthValue()	65
6.2.3.43	rtXmlEncGYear()	65



6.2.3.44	<code>rtXmlEncGYearMonth()</code>	66
6.2.3.45	<code>rtXmlEncGYearMonthValue()</code>	66
6.2.3.46	<code>rtXmlEncGYearValue()</code>	67
6.2.3.47	<code>rtXmlEncHexBinary()</code>	67
6.2.3.48	<code>rtXmlEncHexBinaryAttr()</code>	68
6.2.3.49	<code>rtXmlEncHexStrValue()</code>	68
6.2.3.50	<code>rtXmlEncIndent()</code>	69
6.2.3.51	<code>rtXmlEncInt()</code>	69
6.2.3.52	<code>rtXmlEncInt64()</code>	70
6.2.3.53	<code>rtXmlEncInt64Attr()</code>	71
6.2.3.54	<code>rtXmlEncInt64Value()</code>	71
6.2.3.55	<code>rtXmlEncIntAttr()</code>	72
6.2.3.56	<code>rtXmlEncIntPattern()</code>	72
6.2.3.57	<code>rtXmlEncIntValue()</code>	73
6.2.3.58	<code>rtXmlEncNamedBits()</code>	73
6.2.3.59	<code>rtXmlEncNSAttrs()</code>	74
6.2.3.60	<code>rtXmlEncReal10()</code>	74
6.2.3.61	<code>rtXmlEncSoapArrayTypeAttr()</code>	75
6.2.3.62	<code>rtXmlEncStartDocument()</code>	76
6.2.3.63	<code>rtXmlEncStartElement()</code>	76
6.2.3.64	<code>rtXmlEncStartSoapElems()</code>	77
6.2.3.65	<code>rtXmlEncStartSoapEnv()</code>	77
6.2.3.66	<code>rtXmlEncString()</code>	78
6.2.3.67	<code>rtXmlEncStringValue()</code>	78
6.2.3.68	<code>rtXmlEncStringValue2()</code>	79
6.2.3.69	<code>rtXmlEncTermStartElement()</code>	79
6.2.3.70	<code>rtXmlEncTime()</code>	80
6.2.3.71	<code>rtXmlEncTimeValue()</code>	80

6.2.3.72	rtXmlEncUInt()	81
6.2.3.73	rtXmlEncUInt64()	81
6.2.3.74	rtXmlEncUInt64Attr()	82
6.2.3.75	rtXmlEncUInt64Value()	83
6.2.3.76	rtXmlEncUIntAttr()	83
6.2.3.77	rtXmlEncUIntValue()	84
6.2.3.78	rtXmlEncUnicodeStr()	84
6.2.3.79	rtXmlEncUTF8Attr()	85
6.2.3.80	rtXmlEncUTF8Attr2()	85
6.2.3.81	rtXmlEncUTF8Str()	86
6.2.3.82	rtXmlEncXSAttrs()	86
6.2.3.83	rtXmlEncXSNilAttr()	87
6.2.3.84	rtXmlEncXSTypeAttr()	87
6.2.3.85	rtXmlEncXSTypeAttr2()	88
6.2.3.86	rtXmlFreeInputSource()	88
6.2.3.87	rtXmlGetIndent()	89
6.2.3.88	rtXmlGetIndentChar()	89
6.2.3.89	rtXmlGetWriteBOM()	89
6.2.3.90	rtXmlPrintNSAttrs()	90
6.2.3.91	rtXmlSetEncBufPtr()	90
6.3	XML utility functions.	92
6.3.1	Detailed Description	92
6.3.2	Function Documentation	92
6.3.2.1	rtXmlWriteToFile()	92
6.4	XML pull-parser decode functions.	93
6.4.1	Detailed Description	97
6.4.2	Function Documentation	97
6.4.2.1	rtXmlEncAttrC14N()	97

6.4.2.2	<code>rtXmlpCountListItems()</code>	97
6.4.2.3	<code>rtXmlpCreateReader()</code>	98
6.4.2.4	<code>rtXmlpDecAny()</code>	98
6.4.2.5	<code>rtXmlpDecAny2()</code>	99
6.4.2.6	<code>rtXmlpDecAnyAttrStr()</code>	99
6.4.2.7	<code>rtXmlpDecAnyElem()</code>	100
6.4.2.8	<code>rtXmlpDecBase64Str()</code>	100
6.4.2.9	<code>rtXmlpDecBase64Str64()</code>	101
6.4.2.10	<code>rtXmlpDecBigInt()</code>	102
6.4.2.11	<code>rtXmlpDecBitString()</code>	102
6.4.2.12	<code>rtXmlpDecBitString64()</code>	103
6.4.2.13	<code>rtXmlpDecBitStringExt()</code>	104
6.4.2.14	<code>rtXmlpDecBitStringExt64()</code>	104
6.4.2.15	<code>rtXmlpDecBool()</code>	105
6.4.2.16	<code>rtXmlpDecDate()</code>	106
6.4.2.17	<code>rtXmlpDecDateTime()</code>	106
6.4.2.18	<code>rtXmlpDecDecimal()</code>	107
6.4.2.19	<code>rtXmlpDecDouble()</code>	107
6.4.2.20	<code>rtXmlpDecDoubleExt()</code>	108
6.4.2.21	<code>rtXmlpDecDynBase64Str()</code>	108
6.4.2.22	<code>rtXmlpDecDynBase64Str64()</code>	109
6.4.2.23	<code>rtXmlpDecDynBitString()</code>	109
6.4.2.24	<code>rtXmlpDecDynHexStr()</code>	110
6.4.2.25	<code>rtXmlpDecDynHexStr64()</code>	110
6.4.2.26	<code>rtXmlpDecDynUnicodeStr()</code>	111
6.4.2.27	<code>rtXmlpDecDynUTF8Str()</code>	111
6.4.2.28	<code>rtXmlpDecGDay()</code>	112
6.4.2.29	<code>rtXmlpDecGMonth()</code>	112

6.4.2.30	<code>rtXmlpDecGMonthDay()</code>	113
6.4.2.31	<code>rtXmlpDecGYear()</code>	113
6.4.2.32	<code>rtXmlpDecGYearMonth()</code>	114
6.4.2.33	<code>rtXmlpDecHexStr()</code>	114
6.4.2.34	<code>rtXmlpDecHexStr64()</code>	115
6.4.2.35	<code>rtXmlpDecInt()</code>	116
6.4.2.36	<code>rtXmlpDecInt16()</code>	116
6.4.2.37	<code>rtXmlpDecInt64()</code>	117
6.4.2.38	<code>rtXmlpDecInt8()</code>	117
6.4.2.39	<code>rtXmlpDecNamedBits()</code>	118
6.4.2.40	<code>rtXmlpDecNamedBits64()</code>	118
6.4.2.41	<code>rtXmlpDecStrList()</code>	119
6.4.2.42	<code>rtXmlpDecTime()</code>	120
6.4.2.43	<code>rtXmlpDecUInt()</code>	120
6.4.2.44	<code>rtXmlpDecUInt16()</code>	121
6.4.2.45	<code>rtXmlpDecUInt64()</code>	121
6.4.2.46	<code>rtXmlpDecUInt8()</code>	122
6.4.2.47	<code>rtXmlpDecUTF8Str()</code>	122
6.4.2.48	<code>rtXmlpDecXmlStr()</code>	123
6.4.2.49	<code>rtXmlpDecXmlStrList()</code>	123
6.4.2.50	<code>rtXmlpDecXSIAAttr()</code>	124
6.4.2.51	<code>rtXmlpDecXSIAAttrs()</code>	124
6.4.2.52	<code>rtXmlpDecXSISTypeAttr()</code>	125
6.4.2.53	<code>rtXmlpForceDecodeAsGroup()</code>	125
6.4.2.54	<code>rtXmlpGetAttributeCount()</code>	126
6.4.2.55	<code>rtXmlpGetAttributeID()</code>	126
6.4.2.56	<code>rtXmlpGetCurrentLevel()</code>	127
6.4.2.57	<code>rtXmlpGetNextAllElemID()</code>	127

6.4.2.58	<code>rtXmIpGetNextAllElemID16()</code>	128
6.4.2.59	<code>rtXmIpGetNextAllElemID32()</code>	129
6.4.2.60	<code>rtXmIpGetNextElem()</code>	129
6.4.2.61	<code>rtXmIpGetNextElemID()</code>	130
6.4.2.62	<code>rtXmIpGetNextSeqElemID()</code>	131
6.4.2.63	<code>rtXmIpGetNextSeqElemID2()</code>	132
6.4.2.64	<code>rtXmIpGetNextSeqElemIDExt()</code>	132
6.4.2.65	<code>rtXmIpGetReader()</code>	134
6.4.2.66	<code>rtXmIpGetXmInsAttrs()</code>	134
6.4.2.67	<code>rtXmIpGetXSITypeAttr()</code>	136
6.4.2.68	<code>rtXmIpGetXSITypeIndex()</code>	136
6.4.2.69	<code>rtXmIpHasAttributes()</code>	138
6.4.2.70	<code>rtXmIpHideAttributes()</code>	138
6.4.2.71	<code>rtXmIplsDecodeAsGroup()</code>	139
6.4.2.72	<code>rtXmIplsEmptyElement()</code>	139
6.4.2.73	<code>rtXmIplsLastEventDone()</code>	140
6.4.2.74	<code>rtXmIplsUTF8Encoding()</code>	140
6.4.2.75	<code>rtXmIpListHasItem()</code>	140
6.4.2.76	<code>rtXmIpLookupXSITypeIndex()</code>	141
6.4.2.77	<code>rtXmIpMarkLastEventActive()</code>	141
6.4.2.78	<code>rtXmIpMarkPos()</code>	142
6.4.2.79	<code>rtXmIpMatchEndTag()</code>	142
6.4.2.80	<code>rtXmIpMatchStartTag()</code>	143
6.4.2.81	<code>rtXmIpNeedDecodeAttributes()</code>	143
6.4.2.82	<code>rtXmIpReadBytes()</code>	144
6.4.2.83	<code>rtXmIpResetMarkedPos()</code>	144
6.4.2.84	<code>rtXmIpRewindToMarkedPos()</code>	144
6.4.2.85	<code>rtXmIpSelectAttribute()</code>	145

6.4.2.86	rtXmlpSetListMode()	145
6.4.2.87	rtXmlpSetMixedContentMode()	146
6.4.2.88	rtXmlpSetNamespaceTable()	146
6.4.2.89	rtXmlpSetWhiteSpaceMode()	146
6.5	XML run-time error status codes.	148
6.5.1	Detailed Description	149
6.5.2	Macro Definition Documentation	149
6.5.2.1	XML_E_BASE	149
6.5.2.2	XML_E_ELEMMISRQ	149
6.5.2.3	XML_E_ELEMSISRQ	150
6.5.2.4	XML_E_FLDABSENT	150
6.5.2.5	XML_E_NOMATCH	150
6.5.2.6	XML_E_NSURINOTFOU	150
6.5.2.7	XML_E_TAGMISMATCH	151
6.5.2.8	XML_OK_EOB	151
6.5.2.9	XML_OK_FRAG	151
6.6	DOM API functions.	152
6.6.1	Detailed Description	153
6.6.2	Function Documentation	153
6.6.2.1	domAddAttribute()	153
6.6.2.2	domAddCdata()	154
6.6.2.3	domAddContent()	154
6.6.2.4	domCreateChild()	155
6.6.2.5	domCreateDocument()	155
6.6.2.6	domFreeDoc()	156
6.6.2.7	domGetAttrData()	156
6.6.2.8	domGetChild()	157
6.6.2.9	domGetDoc()	157

6.6.2.10	domGetElementName()	157
6.6.2.11	domGetNext()	158
6.6.2.12	domGetNextAttr()	158
6.6.2.13	domGetNodeAttributesNum()	159
6.6.2.14	domGetNodeContent()	159
6.6.2.15	domGetNodeFirstAttribute()	160
6.6.2.16	domGetRootElement()	160
6.6.2.17	domParseFile()	161
6.6.2.18	domSaveDoc()	161
6.7	DOM runtime encode/decode functions.	162
6.7.1	Detailed Description	162
6.7.2	Function Documentation	162
6.7.2.1	rtDomAddAttr()	163
6.7.2.2	rtDomAddNode()	163
6.7.2.3	rtDomAddNSAttrs()	164
6.7.2.4	rtDomAddSubTree()	165
6.7.2.5	rtDomDecodeDoc()	165
6.7.2.6	rtDomEncAny()	166
6.7.2.7	rtDomEncAnyAttr()	166
6.7.2.8	rtDomEncString()	167
6.7.2.9	rtDomEncStringValue()	167
6.7.2.10	rtDomEncXSIAttrs()	168
6.7.2.11	rtDomSetNode()	168

<b>7</b>	<b>Class Documentation</b>	<b>171</b>
7.1	OSXMLGroupDesc Struct Reference . . . . .	171
7.1.1	Detailed Description . . . . .	171
7.2	OSXMLStringListParser Class Reference . . . . .	171
7.2.1	Detailed Description . . . . .	172
7.2.2	Constructor & Destructor Documentation . . . . .	172
7.2.2.1	OSXMLStringListParser() . . . . .	172
7.2.3	Member Function Documentation . . . . .	172
7.2.3.1	next() . . . . .	172
<b>8</b>	<b>File Documentation</b>	<b>175</b>
8.1	osrtdom.h File Reference . . . . .	175
8.1.1	Detailed Description . . . . .	176
8.2	osrtxml.h File Reference . . . . .	176
8.2.1	Detailed Description . . . . .	189
8.2.2	Typedef Documentation . . . . .	190
8.2.2.1	OSXMLGroupDesc . . . . .	190
8.2.3	Function Documentation . . . . .	190
8.2.3.1	rtSaxGetAttrValue() . . . . .	190
8.2.3.2	rtSaxGetElemID() . . . . .	190
8.2.3.3	rtSaxGetElemID8() . . . . .	191
8.2.3.4	rtSaxHasXMLNSAttrs() . . . . .	192
8.2.3.5	rtSaxIsEmptyBuffer() . . . . .	192
8.2.3.6	rtSaxSortAttrs() . . . . .	192
8.2.3.7	rtSaxStrListMatch() . . . . .	193
8.2.3.8	rtSaxStrListParse() . . . . .	193
8.2.3.9	rtXmlCmpBase64Str() . . . . .	194
8.2.3.10	rtXmlCmpHexStr() . . . . .	194



8.2.3.11	<code>rtXmlCreateFileInputSource()</code>	195
8.2.3.12	<code>rtXmlInitContext()</code>	195
8.2.3.13	<code>rtXmlInitContextUsingKey()</code>	195
8.2.3.14	<code>rtXmlInitCtxtAppInfo()</code>	196
8.2.3.15	<code>rtXmlMatchBase64Str()</code>	196
8.2.3.16	<code>rtXmlMatchDate()</code>	197
8.2.3.17	<code>rtXmlMatchDateTime()</code>	197
8.2.3.18	<code>rtXmlMatchGDay()</code>	198
8.2.3.19	<code>rtXmlMatchGMonth()</code>	198
8.2.3.20	<code>rtXmlMatchGMonthDay()</code>	199
8.2.3.21	<code>rtXmlMatchGYear()</code>	199
8.2.3.22	<code>rtXmlMatchGYearMonth()</code>	199
8.2.3.23	<code>rtXmlMatchHexStr()</code>	200
8.2.3.24	<code>rtXmlMatchTime()</code>	200
8.2.3.25	<code>rtXmlMemFreeAnyAttrs()</code>	201
8.2.3.26	<code>rtXmlNewQName()</code>	201
8.2.3.27	<code>rtXmlPrepareContext()</code>	202
8.2.3.28	<code>rtXmlSetEncC14N()</code>	202
8.2.3.29	<code>rtXmlSetEncDocHdr()</code>	202
8.2.3.30	<code>rtXmlSetEncodingStr()</code>	203
8.2.3.31	<code>rtXmlSetEncXSINamespace()</code>	203
8.2.3.32	<code>rtXmlSetEncXSINilAttr()</code>	204
8.2.3.33	<code>rtXmlSetFormatting()</code>	204
8.2.3.34	<code>rtXmlSetIndent()</code>	205
8.2.3.35	<code>rtXmlSetIndentChar()</code>	205
8.2.3.36	<code>rtXmlSetNamespacesSet()</code>	206
8.2.3.37	<code>rtXmlSetNoNSSchemaLocation()</code>	206
8.2.3.38	<code>rtXmlSetNSPrefixLinks()</code>	206

8.2.3.39	rtXmlSetSchemaLocation()	207
8.2.3.40	rtXmlSetSoapVersion()	207
8.2.3.41	rtXmlSetWriteBOM()	208
8.2.3.42	rtXmlSetXSITypeAttr()	208
8.3	rtXmlErrCodes.h File Reference	208
8.3.1	Detailed Description	210
8.4	rtXmlExternDefs.h File Reference	210
8.4.1	Detailed Description	210
8.5	rtXmlKeyArray.h File Reference	210
8.5.1	Detailed Description	211
8.5.2	Function Documentation	211
8.5.2.1	rtXmlKeyArrayAdd()	211
8.5.2.2	rtXmlKeyArrayContains()	211
8.5.2.3	rtXmlKeyArrayInit()	211
8.5.2.4	rtXmlKeyArraySetDecimal()	212
8.5.2.5	rtXmlKeyArraySetInt()	212
8.5.2.6	rtXmlKeyArraySetString()	212
8.5.2.7	rtXmlKeyArraySetUInt()	213
8.6	rtXmlNamespace.h File Reference	213
8.6.1	Detailed Description	215
8.6.2	Macro Definition Documentation	215
8.6.2.1	RTXMLNSSETQNAME	215
8.6.3	Function Documentation	215
8.6.3.1	rtXmlNSAddNamespace()	215
8.6.3.2	rtXmlNSEqual()	216
8.6.3.3	rtXmlNSFreeAttrList()	216
8.6.3.4	rtXmlNSGetAttrPrefix()	217
8.6.3.5	rtXmlNSGetAttrQName()	217

8.6.3.6	<code>rtXmlNSGetPrefix()</code>	218
8.6.3.7	<code>rtXmlNSGetPrefixCount()</code>	218
8.6.3.8	<code>rtXmlNSGetPrefixIndex()</code>	219
8.6.3.9	<code>rtXmlNSGetPrefixUsingIndex()</code>	219
8.6.3.10	<code>rtXmlNSGetQName()</code>	220
8.6.3.11	<code>rtXmlNSLookupPrefix()</code>	220
8.6.3.12	<code>rtXmlNSLookupPrefixForURI()</code>	221
8.6.3.13	<code>rtXmlNSLookupPrefixFrag()</code>	221
8.6.3.14	<code>rtXmlNSLookupURI()</code>	222
8.6.3.15	<code>rtXmlNSLookupURIInList()</code>	222
8.6.3.16	<code>rtXmlNSNewPrefix()</code>	223
8.6.3.17	<code>rtXmlNSRemoveAll()</code>	223
8.6.3.18	<code>rtXmlNSSetNamespace()</code>	223
8.7	<code>rtXmlpCppDecFuncs.h</code> File Reference	224
8.7.1	Detailed Description	224
<b>Index</b>		<b>225</b>



## Chapter 1

# C XML Runtime Library Functions

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

The categories of functions provided are as follows:

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



## Chapter 2

# C DOM Runtime Library Functions

The **C run-time DOM library** contains functions used to encode/decode XML data in a DOM tree. These functions are identified by their *rtDom* prefixes.

The categories of functions provided are as follows:

- Functions to add nodes.
- Functions to add attributes to a node.
- Functions to add namespace and xsi:type information.
- Functions to encode data types to DOM.





# Chapter 3

## Module Index

### 3.1 Modules

Here is a list of all modules:

XML decode functions. . . . .	11
XML encode functions. . . . .	37
XML utility functions. . . . .	92
XML pull-parser decode functions. . . . .	93
XML run-time error status codes. . . . .	148
DOM API functions. . . . .	152
DOM runtime encode/decode functions. . . . .	162



# Chapter 4

## Class Index

### 4.1 Class List

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

<a href="#">OSXMLGroupDesc</a>	
<a href="#">OSXMLGroupDesc</a>	describes how entries in an OSXMLElemIDRec array make up a group . . . . . 171
<a href="#">OSXMLStringListParser</a>	
Class enabling parsing of an XML Schema list into strings . . . . .	171



# Chapter 5

## File Index

### 5.1 File List

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

<b>domAPI.h</b>	??
<a href="#">osrtdom.h</a>	
DOM low-level C encode/decode functions	175
<a href="#">osrtxml.h</a>	
XML low-level C encode/decode functions	176
<b>rtLx2Dom.h</b>	??
<b>rtxDomDefs.h</b>	??
<b>rtXmlCtxtApplInfo.h</b>	??
<b>rtXmlEncDateTpl.h</b>	??
<a href="#">rtXmlErrCodes.h</a>	
List of numeric status codes that can be returned by ASN1C run-time functions and generated code	208
<a href="#">rtXmlExternDefs.h</a>	
XML external definitions macro	210
<a href="#">rtXmlKeyArray.h</a>	
Implementation of a dynamic pointer sorted array	210
<a href="#">rtXmlNamespace.h</a>	
XML namespace handling structures and function definitions	213
<a href="#">rtXmlpCppDecFuncs.h</a>	
XML low-level C++ decode functions	224
<b>rtXmlPull.h</b>	??



## Chapter 6

# Module Documentation

### 6.1 XML decode functions.

#### Functions

- 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.*
- 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.*
- int [rtXmlDecBase64Str64](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocts, OSSIZE bufsize)  
*This function is identical to [rtXmlDecBase64Str](#) except that it supports a 64-bit integer length on 64-bit systems.*
- int [rtXmlDecBase64StrValue](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, OSSIZE bufSize, OS↔SIZE srcDataLen)  
*This function decodes a contents of a Base64-encode binary string into the specified octet array.*
- int [rtXmlDecBase64StrValue64](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocts, OSSIZE bufSize, OS↔SIZE srcDataLen)  
*This function decodes is identical to [rtXmlDecBase64StrValue](#) except that it supports a 64-bit integer length on 64-bit systems.*
- int [rtXmlDecBigInt](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*ppvalue)  
*This function will decode a variable of the XSD integer type.*
- int [rtXmlDecBool](#) (OSCTXT \*pctxt, OSBOOL \*pvalue)  
*This function decodes a variable of the boolean type.*
- int [rtXmlDecDate](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'date' type.*
- int [rtXmlDecTime](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'time' type.*
- int [rtXmlDecDateTime](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'dateTime' type.*
- int [rtXmlDecDecimal](#) (OSCTXT \*pctxt, OSREAL \*pvalue)  
*This function decodes the contents of a decimal data type.*
- int [rtXmlDecDouble](#) (OSCTXT \*pctxt, OSREAL \*pvalue)  
*This function decodes the contents of a float or double data type.*

- int [rtXmlDecDynBase64Str](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)  
*This function decodes a contents of a Base64-encode binary string.*
- int [rtXmlDecDynBase64Str64](#) (OSCTXT \*pctxt, OSDynOctStr64 \*pvalue)  
*This function is identical to [rtXmlDecDynBase64Str](#) except that it supports a 64-bit integer length on 64-bit systems.*
- int [rtXmlDecDynHexStr](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)  
*This function decodes a contents of a hexBinary string.*
- int [rtXmlDecDynHexStr64](#) (OSCTXT \*pctxt, OSDynOctStr64 \*pvalue)  
*This function is identical to [rtXmlDecDynHexStr](#) except that it supports a 64-bit integer length on 64-bit systems.*
- int [rtXmlDecEmptyElement](#) (OSCTXT \*pctxt)  
*This function is used to enforce a requirement that an element be empty.*
- int [rtXmlDecUTF8Str](#) (OSCTXT \*pctxt, OSUTF8CHAR \*outdata, OSSIZE max\_len)  
*This function decodes the contents of a UTF-8 string data type.*
- int [rtXmlDecDynUTF8Str](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*outdata)  
*This function decodes the contents of a UTF-8 string data type.*
- 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.*
- int [rtXmlDecHexStr](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocets, OSINT32 bufsize)  
*This function decodes the contents of a hexBinary string into a static memory structure.*
- int [rtXmlDecHexStr64](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocets, OSSIZE bufsize)  
*This function is identical to [rtXmlDecHexStr](#) except that it supports a 64-bit integer length on 64-bit systems.*
- int [rtXmlDecGYear](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gYear' type.*
- int [rtXmlDecGYearMonth](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gYearMonth' type.*
- int [rtXmlDecGMonth](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gMonth' type.*
- int [rtXmlDecGMonthDay](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gMonthDay' type.*
- int [rtXmlDecGDay](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gDay' type.*
- int [rtXmlDecInt](#) (OSCTXT \*pctxt, OSINT32 \*pvalue)  
*This function decodes the contents of a 32-bit integer data type.*
- int [rtXmlDecInt8](#) (OSCTXT \*pctxt, OSINT8 \*pvalue)  
*This function decodes the contents of an 8-bit integer data type (i.e.*
- int [rtXmlDecInt16](#) (OSCTXT \*pctxt, OSINT16 \*pvalue)  
*This function decodes the contents of a 16-bit integer data type.*
- int [rtXmlDecInt64](#) (OSCTXT \*pctxt, OSINT64 \*pvalue)  
*This function decodes the contents of a 64-bit integer data type.*
- int [rtXmlDecUInt](#) (OSCTXT \*pctxt, OSUINT32 \*pvalue)  
*This function decodes the contents of an unsigned 32-bit integer data type.*
- int [rtXmlDecUInt8](#) (OSCTXT \*pctxt, OSUINT8 \*pvalue)  
*This function decodes the contents of an unsigned 8-bit integer data type (i.e.*
- int [rtXmlDecUInt16](#) (OSCTXT \*pctxt, OSUINT16 \*pvalue)  
*This function decodes the contents of an unsigned 16-bit integer data type.*
- int [rtXmlDecUInt64](#) (OSCTXT \*pctxt, OSUINT64 \*pvalue)  
*This function decodes the contents of an unsigned 64-bit integer data type.*



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

## 6.1.1 Detailed Description

## 6.1.2 Function Documentation

### 6.1.2.1 `rtXmlDecBase64Binary()`

```
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

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

## Returns

Completion status of operation:

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

### 6.1.2.2 rtXmlDecBase64Str()

```
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

<i>pctxt</i>	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.
<i>pvalue</i>	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.
<i>pnocts</i>	A pointer to an integer value to receive the decoded number of octets.
<i>bufsize</i>	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.

### 6.1.2.3 rtXmlDecBase64Str64()

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

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

## Parameters

<i>pctxt</i>	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.
<i>pvalue</i>	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.
<i>pnocts</i>	A pointer to an integer value to receive the decoded number of octets.
<i>bufsize</i>	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.

### 6.1.2.4 rtXmlDecBase64StrValue()

```
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

<i>pctxt</i>	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.
<i>pvalue</i>	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.
<i>pnocts</i>	A pointer to an integer value to receive the decoded number of octets.
<i>bufSize</i>	A maximum size (in octets) of <i>pvalue</i> buffer. An error will occur if the number of octets in the decoded string is larger than this value.
<i>srcDataLen</i>	An actual source data length (in octets) without whitespaces.

## Returns

Completion status of operation:

- 0 = success,

- negative return value is error.

#### 6.1.2.5 rtXmlDecBase64StrValue64()

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

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

#### Parameters

<i>pctxt</i>	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.
<i>pvalue</i>	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 <code>bufSize</code> input parameter.
<i>pnocts</i>	A pointer to an integer value to receive the decoded number of octets.
<i>bufSize</i>	A maximum size (in octets) of <code>pvalue</code> buffer. An error will occur if the number of octets in the decoded string is larger than this value.
<i>srcDataLen</i>	An actual source data length (in octets) without whitespaces.

#### Returns

Completion status of operation:

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

#### 6.1.2.6 rtXmlDecBigInt()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>ppvalue</i>	Pointer to a pointer to receive decoded UTF-8 string. Dynamic memory is allocated for the variable using the <code>rtMemAlloc</code> 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.

### 6.1.2.7 `rtXmlDecBool()`

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

This function decodes a variable of the boolean type.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to a variable to receive the decoded boolean value.

## Returns

Completion status of operation:

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

### 6.1.2.8 `rtXmlDecDate()`

```
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

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

#### Returns

Completion status of operation:

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

#### 6.1.2.9 rtXmlDecDateTime()

```
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

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

#### Returns

Completion status of operation:

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

#### 6.1.2.10 rtXmlDecDecimal()

```
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

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

### Returns

Completion status of operation:

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

#### 6.1.2.11 rtXmlDecDouble()

```
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

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

### Returns

Completion status of operation:

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

#### 6.1.2.12 rtXmlDecDynBase64Str()

```
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

<i>pctxt</i>	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.
<i>pvalue</i>	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the <code>::rtxMemAlloc</code> function.

## Returns

Completion status of operation:

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

### 6.1.2.13 `rtXmlDecDynBase64Str64()`

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

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

## Parameters

<i>pctxt</i>	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.
<i>pvalue</i>	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the <code>::rtxMemAlloc</code> function.

## Returns

Completion status of operation:

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

### 6.1.2.14 `rtXmlDecDynHexStr()`

```
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

<i>pctxt</i>	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.
<i>pvalue</i>	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the <code>::rtxMemAlloc</code> function.

## Returns

Completion status of operation:

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

### 6.1.2.15 rtXmlDecDynHexStr64()

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

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

## Parameters

<i>pctxt</i>	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.
<i>pvalue</i>	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the <code>::rtxMemAlloc</code> function.

## Returns

Completion status of operation:

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

### 6.1.2.16 rtXmlDecDynUTF8Str()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>outdata</i>	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.

### 6.1.2.17 rtXmlDecEmptyElement()

```
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

<i>pctxt</i>	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.

### 6.1.2.18 rtXmlDecGDay()

```
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

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

### Returns

Completion status of operation:

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

#### 6.1.2.19 rtXmlDecGMonth()

```
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

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

### Returns

Completion status of operation:

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

#### 6.1.2.20 rtXmlDecGMonthDay()

```
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

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

### Returns

Completion status of operation:

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

#### 6.1.2.21 rtXmlDecGYear()

```
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

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

### Returns

Completion status of operation:

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

#### 6.1.2.22 rtXmlDecGYearMonth()

```
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

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

## Returns

Completion status of operation:

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

### 6.1.2.23 rtXmlDecHexBinary()

```
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

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

## Returns

Completion status of operation:

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

### 6.1.2.24 rtXmlDecHexStr()

```
int rtXmlDecHexStr (
    OSCTXT * pctxt,
```

```

OSOCKET * pvalue,
OSUINT32 * pnocts,
OSINT32 bufsize )

```

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

#### Parameters

<i>pctxt</i>	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.
<i>pvalue</i>	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.
<i>pnocts</i>	A pointer to an integer value to receive the decoded number of octets.
<i>bufsize</i>	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.

#### 6.1.2.25 rtXmlDecHexStr64()

```

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

```

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

#### Parameters

<i>pctxt</i>	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.
<i>pvalue</i>	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.
<i>pnocts</i>	A pointer to an integer value to receive the decoded number of octets.
<i>bufsize</i>	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.

#### 6.1.2.26 rtXmlDecInt()

```
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

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

##### Returns

Completion status of operation:

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

#### 6.1.2.27 rtXmlDecInt16()

```
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

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

## Returns

Completion status of operation:

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

### 6.1.2.28 rtXmlDecInt64()

```
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

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

## Returns

Completion status of operation:

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

### 6.1.2.29 rtXmlDecInt8()

```
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

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



## Returns

Completion status of operation:

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

### 6.1.2.30 rtXmlDecNSAttr()

```
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

<i>pctxt</i>	Pointer to context structure.
<i>attrName</i>	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.
<i>attrValue</i>	XML namespace attribute value (a URI).
<i>pNSAttrs</i>	List to receive parsed namespace values.
<i>nsTable</i>	Namespace URI's parsed from schema.
<i>nsTableRowCount</i>	Number of rows (URI's) in namespace table.

## Returns

Zero if success or negative error code.

### 6.1.2.31 rtXmlDecQName()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>qname</i>	String containing XML QName to be decoded.
<i>prefix</i>	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 <code>rtxMemAlloc</code> which must be freed using one of the <code>rtxMemFree</code> 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.

#### 6.1.2.32 rtXmlDecTime()

```
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

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

## Returns

Completion status of operation:

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

### 6.1.2.33 rtXmlDecUInt()

```
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

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

## Returns

Completion status of operation:

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

### 6.1.2.34 rtXmlDecUInt16()

```
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

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

## Returns

Completion status of operation:

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

### 6.1.2.35 rtXmlDecUInt64()

```
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

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

## Returns

Completion status of operation:

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

### 6.1.2.36 rtXmlDecUInt8()

```
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

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

## Returns

Completion status of operation:

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

### 6.1.2.37 rtXmlDecUTF8Str()

```
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

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

## Returns

Completion status of operation:

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

### 6.1.2.38 rtXmlDecXmlStr()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>outdata</i>	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.

### 6.1.2.39 rtXmlDecXSIAttr()

```
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

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

## Returns

Completion status of operation:

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

### 6.1.2.40 rtXmlDecXSIAttrs()

```
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**

<i>pctxt</i>	Pointer to context block structure.
<i>attrs</i>	Attributes-values array [attr, value]. Should be null-terminated.
<i>typeName</i>	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.

**6.1.2.41 rtXmlParseElementName()**

```

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**

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

**Returns**

Completion status of operation:

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

### 6.1.2.42 rtXmlParseElemQName()

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

This function parses the initial tag from an XML message.

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>pQName</i>	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.



## 6.2 XML encode functions.

### Macros

- #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

- int `rtXmlEncAny` (OSCTXT \*pctx, OSXMLSTRING \*pvalue, const OSUTF8CHAR \*elemName, OSXML↔ Namespace \*pNS)  
*This function encodes a variable of the XSD any type.*
- int `rtXmlEncAnyTypeValue` (OSCTXT \*pctx, const OSUTF8CHAR \*pvalue)  
*This function encodes a variable of the XSD anyType type.*
- 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.*
- int `rtXmlEncBase64Binary` (OSCTXT \*pctx, OSSIZE nocts, const OSOCTET \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD base64Binary type.*
- int `rtXmlEncBase64BinaryAttr` (OSCTXT \*pctx, OSUINT32 nocts, const OSOCTET \*value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)  
*This function encodes a variable of the XSD base64Binary type as an attribute.*
- int `rtXmlEncBase64StringValue` (OSCTXT \*pctx, OSSIZE nocts, const OSOCTET \*value)  
*This function encodes a variable of the XSD base64Binary type.*
- int `rtXmlEncBigInt` (OSCTXT \*pctx, const OSUTF8CHAR \*value, const OSUTF8CHAR \*elemName, OSXML↔ Namespace \*pNS)  
*This function encodes a variable of the XSD integer type.*
- int `rtXmlEncBigIntAttr` (OSCTXT \*pctx, const OSUTF8CHAR \*value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)  
*This function encodes an XSD integer attribute value.*
- int `rtXmlEncBigIntValue` (OSCTXT \*pctx, const OSUTF8CHAR \*value)  
*This function encodes an XSD integer attribute value.*
- int `rtXmlEncBitString` (OSCTXT \*pctx, OSSIZE nbits, const OSOCTET \*value, const OSUTF8CHAR \*elem↔ Name, OSXMLNamespace \*pNS)  
*This function encodes a variable of the ASN.1 BIT STRING type.*
- int `rtXmlEncBitStringExt` (OSCTXT \*pctx, OSSIZE nbits, const OSOCTET \*value, OSSIZE dataSize, const O↔ SOCTET \*extValue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the ASN.1 BIT STRING type.*
- int `rtXmlEncBinStringValue` (OSCTXT \*pctx, OSSIZE nbits, const OSOCTET \*data)  
*This function encodes a binary string value as a sequence of '1's and '0's.*
- int `rtXmlEncBool` (OSCTXT \*pctx, OSBOOL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*p↔ NS)  
*This function encodes a variable of the XSD boolean type.*
- int `rtXmlEncBoolValue` (OSCTXT \*pctx, OSBOOL value)  
*This function encodes a variable of the XSD boolean type.*

- int **rtXmlEncBoolAttr** (OSCTXT \*pctx, OSBOOL value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)  
*This function encodes an XSD boolean attribute value.*
- int **rtXmlEncCanonicalSort** (OSCTXT \*pctx, OSCTXT \*pBufCtx, OSRTSList \*pList)  
*Sort (as for CXER, Canonical-XER) and encode previously encoded SET OF components.*
- int **rtXmlEncComment** (OSCTXT \*pctx, const OSUTF8CHAR \*comment)  
*This function encodes an XML comment.*
- 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.*
- int **rtXmlEncDateValue** (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a variable of the XSD 'date' type as a string.*
- 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.*
- int **rtXmlEncTimeValue** (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a variable of the XSD 'time' type as a string.*
- 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.*
- int **rtXmlEncDateTimeValue** (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric date/time value into an XML string representation.*
- int **rtXmlEncDecimal** (OSCTXT \*pctx, OSREAL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSDecimalFmt \*pFmtSpec)  
*This function encodes a variable of the XSD decimal type.*
- 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.*
- int **rtXmlEncDecimalValue** (OSCTXT \*pctx, OSREAL value, const OSDecimalFmt \*pFmtSpec, char \*pDestBuf, OSSIZE destBufSize)  
*This function encodes a value of the XSD decimal type.*
- int **rtXmlEncDouble** (OSCTXT \*pctx, OSREAL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSDoubleFmt \*pFmtSpec)  
*This function encodes a variable of the XSD double type.*
- 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.*
- 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.*
- int **rtXmlEncDoubleValue** (OSCTXT \*pctx, OSREAL value, const OSDoubleFmt \*pFmtSpec, int defaultPrecision)  
*This function encodes a value of the XSD double or float type.*
- int **rtXmlEncEmptyElement** (OSCTXT \*pctx, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs, OSBOOL terminate)  
*This function encodes an empty element tag value (<elemName/>).*
- int **rtXmlEncEndDocument** (OSCTXT \*pctx)  
*This function adds trailer information and a null terminator at the end of the XML document being encoded.*
- int **rtXmlEncEndElement** (OSCTXT \*pctx, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes an end element tag value (</elemName/>).*

- `int rtXmlEncEndSoapEnv` (OSCTXT \*pctxt)
 

*This function encodes a SOAP envelope end element tag (<SOAP-ENV:Envelope/>).*
- `int rtXmlEncEndSoapElems` (OSCTXT \*pctxt, OSXMLSOAPMsgType msgtype)
 

*This function encodes SOAP end element tags.*
- `int rtXmlEncFloat` (OSCTXT \*pctxt, OSREAL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSDoubleFmt \*pFmtSpec)
 

*This function encodes a variable of the XSD float type.*
- `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.*
- `int rtXmlEncGYear` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)
 

*This function encodes a numeric gYear element into an XML string representation.*
- `int rtXmlEncGYearMonth` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)
 

*This function encodes a numeric gYearMonth element into an XML string representation.*
- `int rtXmlEncGMonth` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)
 

*This function encodes a numeric gMonth element into an XML string representation.*
- `int rtXmlEncGMonthDay` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)
 

*This function encodes a numeric gMonthDay element into an XML string representation.*
- `int rtXmlEncGDay` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)
 

*This function encodes a numeric gDay element into an XML string representation.*
- `int rtXmlEncGYearValue` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)
 

*This function encodes a numeric gYear value into an XML string representation.*
- `int rtXmlEncGYearMonthValue` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)
 

*This function encodes a numeric gYearMonth value into an XML string representation.*
- `int rtXmlEncGMonthValue` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)
 

*This function encodes a numeric gMonth value into an XML string representation.*
- `int rtXmlEncGMonthDayValue` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)
 

*This function encodes a numeric gMonthDay value into an XML string representation.*
- `int rtXmlEncGDayValue` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)
 

*This function encodes a numeric gDay value into an XML string representation.*
- `int rtXmlEncHexBinary` (OSCTXT \*pctxt, OSSIZE nocts, const OSOCTET \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)
 

*This function encodes a variable of the XSD hexBinary type.*
- `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.*
- `int rtXmlEncHexStrValue` (OSCTXT \*pctxt, OSSIZE nocts, const OSOCTET \*data)
 

*This function encodes a variable of the XSD hexBinary type.*
- `int rtXmlEncIndent` (OSCTXT \*pctxt)
 

*This function adds indentation whitespace to the output stream.*
- `int rtXmlEncInt` (OSCTXT \*pctxt, OSINT32 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)
 

*This function encodes a variable of the XSD integer type.*
- `int rtXmlEncIntValue` (OSCTXT \*pctxt, OSINT32 value)

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

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

*This function encodes a variable of the XSD integer type.*
- int **rtXmlEncInt64Value** (OSCTXT \*pctxt, OSINT64 value)

*This function encodes a variable of the XSD integer type.*
- 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").*
- int **rtXmlEncNamedBits** (OSCTXT \*pctxt, const OSBitMapItem \*pBitMap, OSSIZE nbits, const OSOCTET \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

*This function encodes a variable of the ASN.1 BIT STRING type.*
- int **rtXmlEncNSAttrs** (OSCTXT \*pctxt, OSRTDList \*pNSAttrs)

*This function encodes namespace declaration attributes at the beginning of an XML document.*
- int **rtXmlPrintNSAttrs** (const char \*name, const OSRTDList \*data)

*This function prints a list of namespace attributes.*
- 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.*
- 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.*
- int **rtXmlEncStartDocument** (OSCTXT \*pctxt)

*This function encodes the XML header text at the beginning of an XML document.*
- int **rtXmlEncBOM** (OSCTXT \*pctxt)

*This function encodes the Unicode byte order mark header at the start of the document.*
- int **rtXmlEncStartElement** (OSCTXT \*pctxt, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs, OSBOOL terminate)

*This function encodes a start element tag value (<elemName>).*
- int **rtXmlEncStartSoapEnv** (OSCTXT \*pctxt, OSRTDList \*pNSAttrs)

*This function encodes a SOAP envelope start element tag.*
- int **rtXmlEncStartSoapElems** (OSCTXT \*pctxt, OSXMLSOAPMsgType msgtype)

*This function encodes a SOAP envelope start element tag and an optional SOAP body or fault tag.*
- int **rtXmlEncString** (OSCTXT \*pctxt, OSXMLSTRING \*pxmlstr, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

*This function encodes a variable of the XSD string type.*
- int **rtXmlEncStringValue** (OSCTXT \*pctxt, const OSUTF8CHAR \*value)

*This function encodes a variable of the XSD string type.*
- int **rtXmlEncStringValue2** (OSCTXT \*pctxt, const OSUTF8CHAR \*value, OSSIZE valueLen)

*This function encodes a variable of the XSD string type.*
- int **rtXmlEncTermStartElement** (OSCTXT \*pctxt)

*This function terminates a currently open XML start element by adding either a '>' or '>' (if empty) terminator.*
- int **rtXmlEncUnicodeStr** (OSCTXT \*pctxt, const OSUNICHAR \*value, OSSIZE nchars, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

- This function encodes a Unicode string value.*

  - 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.*
- 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.*
- int `rtXmlEncUTF8Str` (OSCTXT \*pctxt, const OSUTF8CHAR \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

*This function encodes a UTF-8 string value.*
- int `rtXmlEncUInt` (OSCTXT \*pctxt, OSUINT32 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

*This function encodes a variable of the XSD unsigned integer type.*
- int `rtXmlEncUIntValue` (OSCTXT \*pctxt, OSUINT32 value)

*This function encodes a variable of the XSD unsigned integer type.*
- 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").*
- int `rtXmlEncUInt64` (OSCTXT \*pctxt, OSUINT64 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

*This function encodes a variable of the XSD integer type.*
- int `rtXmlEncUInt64Value` (OSCTXT \*pctxt, OSUINT64 value)

*This function encodes a variable of the XSD integer type.*
- 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").*
- int `rtXmlEncXSIAttrs` (OSCTXT \*pctxt, OSBOOL needXSI)

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

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

*This function encodes an XML nil attribute (xsi:nil="true").*
- 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'.*
- int `rtXmlSetEncBufPtr` (OSCTXT \*pctxt, OSOCTET \*bufaddr, OSSIZE bufsiz)

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

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

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

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

## 6.2.1 Detailed Description

## 6.2.2 Macro Definition Documentation

### 6.2.2.1 rtXmlGetEncBufLen

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

This macro returns the length of the encoded XML message.

#### Parameters

<i>pctxt</i>	Pointer to a context structure.
--------------	---------------------------------

Definition at line 2657 of file osrtxml.h.

### 6.2.2.2 rtXmlGetEncBufPtr

```
#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

<i>pctxt</i>	Pointer to a context structure.
--------------	---------------------------------

Definition at line 2650 of file osrtxml.h.

## 6.2.3 Function Documentation

### 6.2.3.1 rtXmlEncAny()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.2 rtXmlEncAnyAttr()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pAnyAttrList</i>	List of attributes.

## Returns

Completion status of operation:

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

### 6.2.3.3 rtXmlEncAnyTypeValue()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Value to be encoded.

#### Returns

Completion status of operation:

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

#### 6.2.3.4 rtXmlEncBase64Binary()

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

This function encodes a variable of the XSD base64Binary type.

#### Parameters

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

#### Returns

Completion status of operation:

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

#### 6.2.3.5 rtXmlEncBase64BinaryAttr()

```
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

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

#### Returns

Completion status of operation:

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

#### 6.2.3.6 rtXmlEncBase64StrValue()

```

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

```

This function encodes a variable of the XSD base64Binary type.

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

#### Parameters

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

#### Returns

Completion status of operation:

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

### 6.2.3.7 rtXmlEncBigInt()

```
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

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

#### Returns

Completion status of operation:

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

### 6.2.3.8 rtXmlEncBigIntAttr()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.9 rtXmlEncBigIntValue()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.10 rtXmlEncBinStrValue()

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

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

## Parameters

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

## Returns

Completion status of operation:

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

### 6.2.3.11 rtXmlEncBitString()

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

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

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

## See also

[rtXmlEncNamedBits](#)).

## Parameters

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

## Returns

Completion status of operation:

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

### 6.2.3.12 rtXmlEncBitStringExt()

```
int rtXmlEncBitStringExt (
    OSCTXT * pctxt,
```

```

OSSIZE nbits,
const OSOCTET * value,
OSSIZE dataSize,
const OSOCTET * extValue,
const OSUTF8CHAR * elemName,
OSXMLNamespace * pNS )

```

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

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

#### Parameters

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

#### Returns

Completion status of operation:

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

#### 6.2.3.13 rtXmlEncBOM()

```

int rtXmlEncBOM (
    OSOCTET * 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

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

#### Returns

Completion status of operation:

- 0 = success,

- negative return value is error.

#### 6.2.3.14 rtXmlEncBool()

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

This function encodes a variable of the XSD boolean type.

##### Parameters

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

##### Returns

Completion status of operation:

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

#### 6.2.3.15 rtXmlEncBoolAttr()

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

This function encodes an XSD boolean attribute value.

##### Parameters

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

## Returns

Completion status of operation:

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

### 6.2.3.16 rtXmlEncBoolValue()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Boolean value to be encoded.

## Returns

Completion status of operation:

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

### 6.2.3.17 rtXmlEncCanonicalSort()

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

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

#### Parameters

<i>pctxt</i>	Context to use to encode the sorted encoding
<i>pBufCtxt</i>	Context previously used to encode the components which are to be sorted.
<i>pList</i>	List of OSRTBufLocDescr identifying the location of each of the components' encodings within pBufCtxt.

### 6.2.3.18 rtXmlEncComment()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>comment</i>	The comment text.

#### Returns

Completion status of operation:

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

### 6.2.3.19 rtXmlEncDate()

```
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

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



## Returns

Completion status of operation:

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

### 6.2.3.20 rtXmlEncDateTime()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.21 rtXmlEncDateTimeValue()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to be encoded.

## Returns

Completion status of operation:

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

### 6.2.3.22 rtXmlEncDateValue()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.23 rtXmlEncDecimal()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.
<i>elemName</i>	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value. 54
<i>pNS</i>	Pointer to namespace structure.
<i>pFmtSpec</i>	Pointer to format specification structure.

## Returns

Completion status of operation:

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

### 6.2.3.24 rtXmlEncDecimalAttr()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.25 rtXmlEncDecimalValue()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.
<i>pFmtSpec</i>	Pointer to format specification structure.
<i>pDestBuf</i>	Pointer to a destination buffer. If NULL (destBufSize should be 0) the encoded value will be put in <code>pctxt-&gt;buffer</code> or in stream associated with the <code>pctxt</code> .
<i>destBufSize</i>	The size of the destination buffer. Must be 0, if <code>pDestBuf</code> is NULL.

### Returns

Completion status of operation:

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

### 6.2.3.26 rtXmlEncDouble()

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

This function encodes a variable of the XSD double type.

### Parameters

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

### Returns

Completion status of operation:

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

### 6.2.3.27 rtXmlEncDoubleAttr()

```
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.

#### Parameters

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

#### Returns

Completion status of operation:

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

### 6.2.3.28 rtXmlEncDoubleNormalValue()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.
<i>pFmtSpec</i>	Pointer to format specification structure.
<i>defaultPrecision</i>	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.

### 6.2.3.29 rtXmlEncDoubleValue()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.
<i>pFmtSpec</i>	Pointer to format specification structure.
<i>defaultPrecision</i>	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.

### 6.2.3.30 rtXmlEncEmptyElement()

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

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

### Parameters

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

### Returns

Completion status of operation:

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

#### 6.2.3.31 rtXmlEncEndDocument()

```
int rtXmlEncEndDocument (
    OSCTXT * pctxt )
```

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

### Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

### Returns

Completion status of operation:

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

#### 6.2.3.32 rtXmlEncEndElement()

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

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

### Parameters

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

### Returns

Completion status of operation:

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

#### 6.2.3.33 rtXmlEncEndSoapElems()

```
int rtXmlEncEndSoapElems (
    OSCTXT * pctxt,
    OSXMLSOAPMsgType msgtype )
```

This function encodes SOAP end element tags.

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

### Parameters

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

### Returns

Completion status of operation:

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

#### 6.2.3.34 rtXmlEncEndSoapEnv()

```
int rtXmlEncEndSoapEnv (
    OSCTXT * pctxt )
```

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



## Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

## Returns

Completion status of operation:

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

### 6.2.3.35 rtXmlEncFloat()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.36 rtXmlEncFloatAttr()

```
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

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

#### Returns

Completion status of operation:

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

#### 6.2.3.37 rtXmlEncGDay()

```

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

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

#### Returns

Completion status of operation:

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

### 6.2.3.38 rtXmlEncGDayValue()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to be encoded.

#### Returns

Completion status of operation:

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

### 6.2.3.39 rtXmlEncGMonth()

```
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

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

#### Returns

Completion status of operation:

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

#### 6.2.3.40 rtXmlEncGMonthDay()

```
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

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

##### Returns

Completion status of operation:

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

#### 6.2.3.41 rtXmlEncGMonthDayValue()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to be encoded.

## Returns

Completion status of operation:

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

### 6.2.3.42 rtXmlEncGMonthValue()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to be encoded.

## Returns

Completion status of operation:

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

### 6.2.3.43 rtXmlEncGYear()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.44 rtXmlEncGYearMonth()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.45 rtXmlEncGYearMonthValue()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to be encoded.

## Returns

Completion status of operation:

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

### 6.2.3.46 rtXmlEncGYearValue()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to be encoded.

## Returns

Completion status of operation:

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

### 6.2.3.47 rtXmlEncHexBinary()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.48 rtXmlEncHexBinaryAttr()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.49 rtXmlEncHexStrValue()

```
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

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

### Returns

Completion status of operation:

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

#### 6.2.3.50 rtXmlEncIndent()

```
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

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

### Returns

Completion status of operation:

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

#### 6.2.3.51 rtXmlEncInt()

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

This function encodes a variable of the XSD integer type.

## Parameters

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

## Returns

Completion status of operation:

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

### 6.2.3.52 rtXmlEncInt64()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.53 rtXmlEncInt64Attr()

```
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

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

#### Returns

Completion status of operation:

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

### 6.2.3.54 rtXmlEncInt64Value()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.

#### Returns

Completion status of operation:

- 0 = success,

- negative return value is error.

### 6.2.3.55 rtXmlEncIntAttr()

```
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

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

#### Returns

Completion status of operation:

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

### 6.2.3.56 rtXmlEncIntPattern()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.
<i>elemName</i>	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
<i>pNS</i>	XML namespace information (prefix and URI).
<i>pattern</i>	Pattern of the encoded value. 72

## Returns

Completion status of operation:

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

### 6.2.3.57 rtXmlEncIntValue()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.

## Returns

Completion status of operation:

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

### 6.2.3.58 rtXmlEncNamedBits()

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

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

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

## Parameters

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

## Returns

Completion status of operation:

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

### 6.2.3.59 rtXmlEncNSAttrs()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pNSAttrs</i>	Pointer to list of namespace attributes.

## Returns

Completion status of operation:

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

### 6.2.3.60 rtXmlEncReal10()

```
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

<i>pctxt</i>	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.
<i>pvalue</i>	A pointer to an REAL base 10 value.
<i>elemName</i>	XML element name. A name must be provided. If an empty string is passed (""), no element tag is added to the encoded value.
<i>pNS</i>	XML namespace information (prefix and URI).

#### Returns

Completion status of operation:

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

#### 6.2.3.61 rtXmlEncSoapArrayTypeAttr()

```

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

<i>pctxt</i>	Pointer to context block structure.
<i>name</i>	Attribute name (NS prefix + arrayType)
<i>value</i>	UTF-8 string value to be encoded.
<i>itemCount</i>	Count of the number of elements in the array.

#### Returns

Completion status of operation:

- 0 = success,

- negative return value is error.

### 6.2.3.62 rtXmlEncStartDocument()

```
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

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

#### Returns

Completion status of operation:

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

### 6.2.3.63 rtXmlEncStartElement()

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

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>elemName</i>	XML element name. Empty string and null are treated equivalently.
<i>pNS</i>	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.
<i>pNSAttrs</i>	List of namespace attributes to be added to element.
<i>terminate</i>	Add closing '>' character.



## Returns

Completion status of operation:

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

### 6.2.3.64 rtXmlEncStartSoapElems()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.65 rtXmlEncStartSoapEnv()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.66 rtXmlEncString()

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

This function encodes a variable of the XSD string type.

#### Parameters

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

## Returns

Completion status of operation:

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

### 6.2.3.67 rtXmlEncStringValue()

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

This function encodes a variable of the XSD string type.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	XML string value to be encoded.

## Returns

Completion status of operation:

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

### 6.2.3.68 rtXmlEncStringValue2()

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

This function encodes a variable of the XSD string type.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	XML string value to be encoded.
<i>valueLen</i>	UTF-8 string value length (in octets).

## Returns

Completion status of operation:

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

### 6.2.3.69 rtXmlEncTermStartElement()

```
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

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

## Returns

Completion status of operation:

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

### 6.2.3.70 rtXmlEncTime()

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

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

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

## Parameters

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

## Returns

Completion status of operation:

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

### 6.2.3.71 rtXmlEncTimeValue()

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

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

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

### Parameters

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

### Returns

Completion status of operation:

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

### 6.2.3.72 rtXmlEncUInt()

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

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

### Parameters

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

### Returns

Completion status of operation:

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

### 6.2.3.73 rtXmlEncUInt64()

```
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**

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

**Returns**

Completion status of operation:

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

**6.2.3.74 rtXmlEncUInt64Attr()**

```

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**

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

**Returns**

Completion status of operation:

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

### 6.2.3.75 rtXmlEncUInt64Value()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.

#### Returns

Completion status of operation:

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

### 6.2.3.76 rtXmlEncUIntAttr()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.77 rtXmlEncUIntValue()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.

## Returns

Completion status of operation:

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

### 6.2.3.78 rtXmlEncUnicodeStr()

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

This function encodes a Unicode string value.

#### Parameters

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



## Returns

Completion status of operation:

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

### 6.2.3.79 rtXmlEncUTF8Attr()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.80 rtXmlEncUTF8Attr2()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.81 rtXmlEncUTF8Str()

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

This function encodes a UTF-8 string value.

#### Parameters

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

## Returns

Completion status of operation:

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

### 6.2.3.82 rtXmlEncXSIAttrs()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>needXSI</i>	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.

### 6.2.3.83 rtXmlEncXSINilAttr()

```
int rtXmlEncXSINilAttr (
    OSCTXT * pctxt )
```

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

## Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

## Returns

Completion status of operation:

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

### 6.2.3.84 rtXmlEncXSITypeAttr()

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

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

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	XSI type attribute value.

## Returns

Completion status of operation:

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

### 6.2.3.85 rtXmlEncXSITypeAttr2()

```
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

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

## Returns

Completion status of operation:

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

### 6.2.3.86 rtXmlFreeInputSource()

```
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

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

## Returns

Completion status of operation:

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

### 6.2.3.87 rtXmlGetIndent()

```
int rtXmlGetIndent (
    OSCTXT * pctxt )
```

This function returns current XML output indent value.

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
--------------	-----------------------------

## Returns

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

### 6.2.3.88 rtXmlGetIndentChar()

```
int rtXmlGetIndentChar (
    OSCTXT * pctxt )
```

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

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
--------------	-----------------------------

## Returns

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

### 6.2.3.89 rtXmlGetWriteBOM()

```
OSBOOL rtXmlGetWriteBOM (
    OSCTXT * pctxt )
```

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

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure.
--------------	------------------------------

#### Returns

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

#### 6.2.3.90 rtXmlPrintNSAttrs()

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

This function prints a list of namespace attributes.

#### Parameters

<i>name</i>	Name to print.
<i>data</i>	List of namespace attributes.

#### Returns

Completion status of operation:

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

#### 6.2.3.91 rtXmlSetEncBufPtr()

```
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 `rtXmlInitContext` 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

<i>pctxt</i>	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.
<i>bufaddr</i>	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).
<i>bufsiz</i>	The length of the memory buffer in bytes. Should be set to zero if NULL was specified for <i>bufaddr</i> (i.e. dynamic encoding was selected).

## 6.3 XML utility functions.

### Functions

- int [rtXmlWriteToFile](#) (OSCTXT \*pctxt, const char \*filename)

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

### 6.3.1 Detailed Description

### 6.3.2 Function Documentation

#### 6.3.2.1 rtXmlWriteToFile()

```
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

<i>pctxt</i>	Pointer to OSCTXT structure.
<i>filename</i>	Full path to file to which XML is to be written.

#### Returns

0 - if success, negative value if error.



## 6.4 XML pull-parser decode functions.

### Functions

- `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).*
- `int rtXmlpDecAny2` (OSCTXT \*pctx, OSUTF8CHAR \*\*pvalue)  
*This version of the rtXmlpDecAny function returns the string in a mutable buffer.*
- `int rtXmlpDecAnyAttrStr` (OSCTXT \*pctx, const OSUTF8CHAR \*\*ppAttrStr, OSSIZE attrIndex)  
*This function decodes an any attribute string.*
- `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).*
- `int rtXmlpDecBase64Str` (OSCTXT \*pctx, OSOCTET \*pvalue, OSUINT32 \*pnocTs, OSSIZE bufsize)  
*This function decodes a contents of a Base64-encode binary string into a static memory structure.*
- `int rtXmlpDecBase64Str64` (OSCTXT \*pctx, OSOCTET \*pvalue, OSSIZE \*pnocTs, OSSIZE bufsize)  
*This function is identical to rtXmlpDecBase64Str except that it supports 64-bit integer lengths on 64-bit systems.*
- `int rtXmlpDecBigInt` (OSCTXT \*pctx, const OSUTF8CHAR \*\*pvalue)  
*This function will decode a variable of the XSD integer type.*
- `int rtXmlpDecBitString` (OSCTXT \*pctx, OSOCTET \*pvalue, OSUINT32 \*pnbits, OSUINT32 bufsize)  
*This function decodes a bit string value.*
- `int rtXmlpDecBitString64` (OSCTXT \*pctx, OSOCTET \*pvalue, OSSIZE \*pnbits, OSSIZE bufsize)  
*This function is identical to rtXmlpDecBitString except that it supports lengths up to 64-bits in size on 64-bit machines.*
- `int rtXmlpDecBitStringExt` (OSCTXT \*pctx, OSOCTET \*pvalue, OSUINT32 \*pnbits, OSOCTET \*\*ppextdata, OSUINT32 bufsize)  
*This function decodes a bit string value.*
- `int rtXmlpDecBitStringExt64` (OSCTXT \*pctx, OSOCTET \*pvalue, OSSIZE \*pnbits, OSOCTET \*\*ppextdata, OSSIZE bufsize)  
*This function is identical to rtXmlpDecBitStringExt except that it supports lengths up to 64-bits in size on 64-bit machines.*
- `int rtXmlpDecBool` (OSCTXT \*pctx, OSBOOL \*pvalue)  
*This function decodes a variable of the boolean type.*
- `int rtXmlpDecDate` (OSCTXT \*pctx, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'date' type.*
- `int rtXmlpDecDateTime` (OSCTXT \*pctx, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'dateTime' type.*
- `int rtXmlpDecDecimal` (OSCTXT \*pctx, OSREAL \*pvalue, int totalDigits, int fractionDigits)  
*This function decodes the contents of a decimal data type.*
- `int rtXmlpDecDouble` (OSCTXT \*pctx, OSREAL \*pvalue)  
*This function decodes the contents of a float or double data type.*
- `int rtXmlpDecDoubleExt` (OSCTXT \*pctx, OSUINT8 flags, OSREAL \*pvalue)  
*This function decodes the contents of a float or double data type.*
- `int rtXmlpDecDynBase64Str` (OSCTXT \*pctx, OSDynOctStr \*pvalue)  
*This function decodes a contents of a Base64-encode binary string.*
- `int rtXmlpDecDynBase64Str64` (OSCTXT \*pctx, OSDynOctStr64 \*pvalue)  
*This function is identical to rtXmlpDecDynBase64Str except that it supports 64-bit integer lengths on 64-bit systems.*
- `int rtXmlpDecDynBitString` (OSCTXT \*pctx, OSDynOctStr \*pvalue)  
*This function decodes a bit string value.*
- `int rtXmlpDecDynHexStr` (OSCTXT \*pctx, OSDynOctStr \*pvalue)

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

  - int `rtXmlpDecDynHexStr64` (OSCTXT \*pctxt, OSDynOctStr64 \*pvalue)

*This function is identical to the `rtXmlpDecDynHexStr` except that it supports lengths up to 64 bits in size on 64-bit systems.*
- int `rtXmlpDecDynUnicodeStr` (OSCTXT \*pctxt, const OSUNICHAR \*\*ppdata, OSSIZE \*pnchars)

*This function decodes a Unicode string data type.*
- int `rtXmlpDecDynUTF8Str` (OSCTXT \*pctxt, const OSUTF8CHAR \*\*outdata)

*This function decodes the contents of a UTF-8 string data type.*
- int `rtXmlpDecUTF8Str` (OSCTXT \*pctxt, OSUTF8CHAR \*out, OSSIZE max\_len)

*This function decodes the contents of a UTF-8 string data type.*
- int `rtXmlpDecGDay` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)

*This function decodes a variable of the XSD 'gDay' type.*
- int `rtXmlpDecGMonth` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)

*This function decodes a variable of the XSD 'gMonth' type.*
- int `rtXmlpDecGMonthDay` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)

*This function decodes a variable of the XSD 'gMonthDay' type.*
- int `rtXmlpDecGYear` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)

*This function decodes a variable of the XSD 'gYear' type.*
- int `rtXmlpDecGYearMonth` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)

*This function decodes a variable of the XSD 'gYearMonth' type.*
- int `rtXmlpDecHexStr` (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocets, OSSIZE bufsize)

*This function decodes the contents of a hexBinary string into a static memory structure.*
- int `rtXmlpDecHexStr64` (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocets, OSSIZE bufsize)

*This function is identical to `rtXmlpDecHexStr` except that it supports lengths up to 64-bits in size on 64-bit machines.*
- int `rtXmlpDecInt` (OSCTXT \*pctxt, OSINT32 \*pvalue)

*This function decodes the contents of a 32-bit integer data type.*
- int `rtXmlpDecInt8` (OSCTXT \*pctxt, OSINT8 \*pvalue)

*This function decodes the contents of an 8-bit integer data type (i.e.*
- int `rtXmlpDecInt16` (OSCTXT \*pctxt, OSINT16 \*pvalue)

*This function decodes the contents of a 16-bit integer data type.*
- int `rtXmlpDecInt64` (OSCTXT \*pctxt, OSINT64 \*pvalue)

*This function decodes the contents of a 64-bit integer data type.*
- int `rtXmlpDecNamedBits` (OSCTXT \*pctxt, const OSBitMapItem \*pBitMap, OSOCTET \*pvalue, OSUINT32 \*pnbits, OSUINT32 bufsize)

*This function decodes the contents of a named bit field.*
- int `rtXmlpDecNamedBits64` (OSCTXT \*pctxt, const OSBitMapItem \*pBitMap, OSOCTET \*pvalue, OSSIZE \*pnbits, OSSIZE bufsize)

*This function decodes the contents of a named bit field.*
- 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.*
- int `rtXmlpDecTime` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)

*This function decodes a variable of the XSD 'time' type.*
- int `rtXmlpDecUInt` (OSCTXT \*pctxt, OSUINT32 \*pvalue)

*This function decodes the contents of an unsigned 32-bit integer data type.*
- int `rtXmlpDecUInt8` (OSCTXT \*pctxt, OSOCTET \*pvalue)

*This function decodes the contents of an unsigned 8-bit integer data type (i.e.*
- int `rtXmlpDecUInt16` (OSCTXT \*pctxt, OSUINT16 \*pvalue)

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

- int [rtXmIplDecUInt64](#) (OSCTXT \*pctxt, OSUINT64 \*pvalue)  
*This function decodes the contents of an unsigned 64-bit integer data type.*
- int [rtXmIplDecXmIStr](#) (OSCTXT \*pctxt, OSXMLSTRING \*outdata)  
*This function decodes the contents of an XML string data type.*
- int [rtXmIplDecXmIStrList](#) (OSCTXT \*pctxt, OSRDLList \*plist)  
*This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.*
- int [rtXmIplDecXSIAttr](#) (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.*
- int [rtXmIplDecXSITypeAttr](#) (OSCTXT \*pctxt, const OSXMLNameFragments \*attrName, const OSUTF8CHAR \*\*ppAttrValue)  
*This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).*
- int [rtXmIplGetAttributeID](#) (const OSXMLStrFragment \*attrName, OSINT16 nsidx, OSSIZE nAttr, const OSXMLAttrDescr attrNames[], OSUINT32 attrPresent[])  
*This function finds an attribute in the descriptor table.*
- int [rtXmIplGetNextElem](#) (OSCTXT \*pctxt, OSXMLElemDescr \*pElem, OSINT32 level)  
*This function parse the next element start tag.*
- int [rtXmIplGetNextElemID](#) (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.*
- int [rtXmIplMarkLastEventActive](#) (OSCTXT \*pctxt)  
*This function marks current tag as unprocessed.*
- int [rtXmIplMatchStartTag](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*elemLocalName, OSINT16 nsidx)  
*This function parses the next start tag that matches with given name.*
- int [rtXmIplMatchEndTag](#) (OSCTXT \*pctxt, OSINT32 level)  
*This function parse next end tag that matches with given name.*
- OSBOOL [rtXmIplHasAttributes](#) (OSCTXT \*pctxt)  
*This function checks accessibility of attributes.*
- int [rtXmIplGetAttributeCount](#) (OSCTXT \*pctxt)  
*This function returns number of attributes in last processed start tag.*
- int [rtXmIplSelectAttribute](#) (OSCTXT \*pctxt, OSXMLNameFragments \*pAttr, OSINT16 \*nsidx, OSSIZE attrIndex)  
*This function selects attribute to decode.*
- OSINT32 [rtXmIplGetCurrentLevel](#) (OSCTXT \*pctxt)  
*This function returns current nesting level.*
- void [rtXmIplSetWhiteSpaceMode](#) (OSCTXT \*pctxt, OSXMLWhiteSpaceMode whiteSpaceMode)  
*Sets the whitespace treatment mode.*
- OSBOOL [rtXmIplSetMixedContentMode](#) (OSCTXT \*pctxt, OSBOOL mixedContentMode)  
*Sets mixed content mode.*
- void [rtXmIplSetListMode](#) (OSCTXT \*pctxt)  
*Sets list mode.*
- OSBOOL [rtXmIplListHasItem](#) (OSCTXT \*pctxt)  
*Check for end of decoded token list.*
- void [rtXmIplCountListItems](#) (OSCTXT \*pctxt, OSSIZE \*itemCnt)  
*Count tokens in list.*
- int [rtXmIplGetNextSeqElemID2](#) (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.*

- int [rtXmIplGetNextSeqElemID](#) (OSCTXT \*pctx, 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.*
- int [rtXmIplGetNextSeqElemIDExt](#) (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 rtXmIplGetNextSeqElemID.*
- int [rtXmIplGetNextAllElemID](#) (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.*
- int [rtXmIplGetNextAllElemID16](#) (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.*
- int [rtXmIplGetNextAllElemID32](#) (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.*
- void [rtXmIplSetNamespaceTable](#) (OSCTXT \*pctx, const OSUTF8CHAR \*namespaceTable[], OSSIZE nNamespaces)
 

*Sets user namespace table.*
- int [rtXmIplCreateReader](#) (OSCTXT \*pctx)
 

*Creates pull parser reader structure within the context.*
- void [rtXmIplHideAttributes](#) (OSCTXT \*pctx)
 

*Disable access to attributes.*
- OSBOOL [rtXmIplNeedDecodeAttributes](#) (OSCTXT \*pctx)
 

*This function checks if attributes were previously decoded.*
- void [rtXmIplMarkPos](#) (OSCTXT \*pctx)
 

*Save current decode position.*
- void [rtXmIplRewindToMarkedPos](#) (OSCTXT \*pctx)
 

*Rewind to saved decode position.*
- void [rtXmIplResetMarkedPos](#) (OSCTXT \*pctx)
 

*Reset saved decode position.*
- int [rtXmIplGetXSITypeAttr](#) (OSCTXT \*pctx, const OSUTF8CHAR \*\*ppAttrValue, OSINT16 \*nsidx, OSSIZE \*pLocalOffs)
 

*This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).*
- int [rtXmIplGetXmIplNsAttrs](#) (OSCTXT \*pctx, OSRTDList \*pNSAttrs)
 

*This function decodes namespace attributes from start tag and adds them to the given list.*
- int [rtXmIplDecXSIAttrs](#) (OSCTXT \*pctx)
 

*This function decodes XSI (XML Schema Instance) that may be present in any arbitrary XML element within a document.*
- OSBOOL [rtXmIplsEmptyElement](#) (OSCTXT \*pctx)
 

*Check element content: empty or not.*
- int [rtXmIplEncAttrC14N](#) (OSCTXT \*pctx)
 

*This function used only in C14 mode.*
- struct OSXMLReader \* [rtXmIplGetReader](#) (OSCTXT \*pctx)
 

*This function fetches the XML reader structure from the context for use in low-level pull parser calls.*
- OSBOOL [rtXmIplsLastEventDone](#) (OSCTXT \*pctx)
 

*Check processing status of current tag.*
- int [rtXmIplGetXSITypeIndex](#) (OSCTXT \*pctx, 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.*

- int `rtXmlpLookupXSITypeIndex` (OSCTXT \*pctx, const OSUTF8CHAR \*pXsiType, OSINT16 xsiTypeIdx, const OSXMLItemDescr typetab[], OSSIZE typetabsiz)

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

- void `rtXmlpForceDecodeAsGroup` (OSCTXT \*pctx)

*Disable skipping of unknown elements in optional sequence tail.*

- OSBOOL `rtXmlplsDecodeAsGroup` (OSCTXT \*pctx)

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

- OSBOOL `rtXmlplsUTF8Encoding` (OSCTXT \*pctx)

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

- int `rtXmlpReadBytes` (OSCTXT \*pctx, OSOCTET \*pbuf, OSSIZE nbytes)

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

## 6.4.1 Detailed Description

## 6.4.2 Function Documentation

### 6.4.2.1 rtXmlEncAttrC14N()

```
int rtXmlEncAttrC14N (  
    OSCTXT * pctx )
```

This function used only in C14 mode.

It provide attributes sorting.

#### Parameters

<i>pctx</i>	Pointer to context block structure.
-------------	-------------------------------------

#### Returns

Completion status of operation:

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

### 6.4.2.2 rtXmlpCountListItems()

```
void rtXmlpCountListItems (  
    OSCTXT * pctx,  
    OSSIZE * itemCnt )
```

Count tokens in list.

**Parameters**

<i>pctx</i>	Pointer to context block structure.
<i>itemCnt</i>	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.

**6.4.2.3 rtXmlpCreateReader()**

```
int rtXmlpCreateReader (
    OSCTXT * pctx )
```

Creates pull parser reader structure within the context.

**Parameters**

<i>pctx</i>	Pointer to context block structure.
-------------	-------------------------------------

**Returns**

Completion status of operation:

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

**6.4.2.4 rtXmlpDecAny()**

```
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).

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

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	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.

### 6.4.2.5 rtXmlpDecAny2()

```
int rtXmlpDecAny2 (
    OSCTXT * pctxt,
    OSUTF8CHAR ** pvalue )
```

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

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	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.

### 6.4.2.6 rtXmlpDecAnyAttrStr()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>ppAttrStr</i>	Pointer to UTF8 character string pointer to receive decoded attribute string. Memory is allocated for the string using the run-time memory manager.
<i>attrIndex</i>	Index of attribute.

## Returns

Completion status of operation:

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

### 6.4.2.7 rtXmlpDecAnyElem()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	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.

### 6.4.2.8 rtXmlpDecBase64Str()

```
int rtXmlpDecBase64Str (
    OSCTXT * pctxt,
```



```

OSOCKET * 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

<i>pctxt</i>	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.
<i>pvalue</i>	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.
<i>pnocts</i>	A pointer to an integer value to receive the decoded number of octets.
<i>bufsize</i>	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.

#### 6.4.2.9 rtXmlpDecBase64Str64()

```

int rtXmlpDecBase64Str64 (
    OSCTXT * pctxt,
    OSOCKET * pvalue,
    OSSIZE * pnocts,
    OSSIZE bufsize )

```

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

#### Parameters

<i>pctxt</i>	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.
<i>pvalue</i>	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.
<i>pnocts</i>	A pointer to an integer value to receive the decoded number of octets.
<i>bufsize</i>	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.

### 6.4.2.10 rtXmlpDecBigInt()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to a pointer to receive decoded UTF-8 string. Dynamic memory is allocated for the variable using the <code>rtMemAlloc</code> 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.

### 6.4.2.11 rtXmlpDecBitString()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to a variable to receive the decoded boolean value.
<i>pnbits</i>	Pointer to hold decoded number of bits.
<i>bufsize</i>	Size of buffer passed in <i>pvalue</i> argument.

## Returns

Completion status of operation:

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

### 6.4.2.12 rtXmlpDecBitString64()

```
int rtXmlpDecBitString64 (
    OSCTXT * pctxt,
    OSOCTET * pvalue,
    OSSIZE * pnbits,
    OSSIZE bufsize )
```

This function is identical to `rtXmlpDecBitString` except that it supports lengths up to 64-bits in size on 64-bit machines.

This function decodes a bit string value. The string consists of a series of '1' and '0' characters. This is the static version in which the user provides a pre-allocated memory buffer to receive the decoded data. One byte in a memory buffer can hold 8 characters of encoded data. Bits are stored from MSB to LSB order.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to a variable to receive the decoded boolean value.
<i>pnbits</i>	Pointer to hold decoded number of bits.
<i>bufsize</i>	Size of buffer passed in <i>pvalue</i> argument.

## Returns

Completion status of operation:

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

## See also

[rtXmlpDecBitString](#)

### 6.4.2.13 rtXmlpDecBitStringExt()

```
int rtXmlpDecBitStringExt (
    OSCTXT * pctxt,
    OSOCTET * pvalue,
    OSUINT32 * pnbits,
    OSOCTET ** ppextdata,
    OSUINT32 bufsize )
```

This function decodes a bit string value.

The string consists of a series of '1' and '0' characters. This is the static version in which the user provides a pre-allocated memory buffer to receive the decoded data. One byte in a memory buffer can hold 8 characters of encoded data. Bits are stored from MSB to LSB order. This version supports BIT STRINGS with the extdata member present.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to a variable to receive the decoded boolean value.
<i>pnbits</i>	Pointer to hold decoded number of bits.
<i>ppextdata</i>	Pointer to byte array containing extension data.
<i>bufsize</i>	Size of buffer passed in <i>pvalue</i> argument.

#### Returns

Completion status of operation:

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

### 6.4.2.14 rtXmlpDecBitStringExt64()

```
int rtXmlpDecBitStringExt64 (
    OSCTXT * pctxt,
    OSOCTET * pvalue,
    OSSIZE * pnbits,
    OSOCTET ** ppextdata,
    OSSIZE bufsize )
```

This function is identical to `rtXmlpDecBitStringExt` except that it supports lengths up to 64-bits in size on 64-bit machines.

This function decodes a bit string value. The string consists of a series of '1' and '0' characters. This is the static version in which the user provides a pre-allocated memory buffer to receive the decoded data. One byte in a memory buffer can hold 8 characters of encoded data. Bits are stored from MSB to LSB order. This version supports BIT STRINGS with the extdata member present.

### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to a variable to receive the decoded boolean value.
<i>pnbits</i>	Pointer to hold decoded number of bits.
<i>ppextdata</i>	Pointer to byte array containing extension data.
<i>bufsize</i>	Size of buffer passed in <i>pvalue</i> argument.

### Returns

Completion status of operation:

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

### See also

[rtXmlpDecBitStringExt](#)

### 6.4.2.15 rtXmlpDecBool()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to a variable to receive the decoded boolean value.

### Returns

Completion status of operation:

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

#### 6.4.2.16 rtXmlpDecDate()

```
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

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

##### Returns

Completion status of operation:

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

#### 6.4.2.17 rtXmlpDecDateTime()

```
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

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

##### Returns

Completion status of operation:

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

#### 6.4.2.18 rtXmlpDecDecimal()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 64-bit double value to receive decoded result.
<i>totalDigits</i>	Number of total digits in the decimal number from XSD totalDigits facet value. Argument should be set to -1 if facet not given.
<i>fractionDigits</i>	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.

#### 6.4.2.19 rtXmlpDecDouble()

```
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

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

## Returns

Completion status of operation:

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

### 6.4.2.20 rtXmlpDecDoubleExt()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>flags</i>	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
<i>pvalue</i>	Pointer to 64-bit double value to receive decoded result.

## Returns

Completion status of operation:

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

### 6.4.2.21 rtXmlpDecDynBase64Str()

```
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

<i>pctxt</i>	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.
<i>pvalue</i>	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated for the string using the <code>::rtxMemAlloc</code> function.

## Returns

Completion status of operation:

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

### 6.4.2.22 `rtXmlpDecDynBase64Str64()`

```
int rtXmlpDecDynBase64Str64 (  
    OSCTXT * pctxt,  
    OSDynOctStr64 * pvalue )
```

This function is identical to `rtXmlpDecDynBase64Str` except that it supports 64-bit integer lengths on 64-bit systems.

## Parameters

<i>pctxt</i>	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.
<i>pvalue</i>	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated for the string using the <code>::rtxMemAlloc</code> function.

## Returns

Completion status of operation:

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

### 6.4.2.23 `rtXmlpDecDynBitString()`

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to a variable to receive the decoded boolean value.

#### Returns

Completion status of operation:

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

#### 6.4.2.24 rtXmlpDecDynHexStr()

```
int rtXmlpDecDynHexStr (
    OSCTXT * pctxt,
    OSDynOctStr * pvalue )
```

This function decodes a contents of a hexBinary string.

This function will allocate dynamic memory to store the decoded result. Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

#### Parameters

<i>pctxt</i>	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.
<i>pvalue</i>	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.

#### 6.4.2.25 rtXmlpDecDynHexStr64()

```
int rtXmlpDecDynHexStr64 (
    OSCTXT * pctxt,
    OSDynOctStr64 * pvalue )
```

This function is identical to the rtXmlpDecDynHexStr except that it supports lengths up to 64 bits in size on 64-bit systems.

## Parameters

<i>pctxt</i>	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.
<i>pvalue</i>	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the <code>::rtxMemAlloc</code> function.

## Returns

Completion status of operation:

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

### 6.4.2.26 `rtXmlpDecDynUnicodeStr()`

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>ppdata</i>	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.
<i>pnchars</i>	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.

### 6.4.2.27 `rtXmlpDecDynUTF8Str()`

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>outdata</i>	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.

#### 6.4.2.28 rtXmlpDecGDay()

```
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

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

#### Returns

Completion status of operation:

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

#### 6.4.2.29 rtXmlpDecGMonth()

```
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

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

#### Returns

Completion status of operation:

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

#### 6.4.2.30 rtXmlpDecGMonthDay()

```
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

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

#### Returns

Completion status of operation:

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

#### 6.4.2.31 rtXmlpDecGYear()

```
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

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

#### Returns

Completion status of operation:

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

#### 6.4.2.32 rtXmlpDecGYearMonth()

```
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

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

#### Returns

Completion status of operation:

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

#### 6.4.2.33 rtXmlpDecHexStr()

```
int rtXmlpDecHexStr (
    OSCTXT * pctxt,
```

```

OSOCKET * pvalue,
OSUINT32 * pnocts,
OSSIZE bufsize )

```

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

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

#### Parameters

<i>pctxt</i>	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.
<i>pvalue</i>	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.
<i>pnocts</i>	A pointer to an integer value to receive the decoded number of octets.
<i>bufsize</i>	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.

#### 6.4.2.34 rtXmlpDecHexStr64()

```

int rtXmlpDecHexStr64 (
    OSCTXT * pctxt,
    OSOCKET * pvalue,
    OSSIZE * pnocts,
    OSSIZE bufsize )

```

This function is identical to rtXmlpDecHexStr except that it supports lengths up to 64-bits in size on 64-bit machines.

#### Parameters

<i>pctxt</i>	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.
<i>pvalue</i>	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.
<i>pnocts</i>	A pointer to an integer value to receive the decoded number of octets.
<i>bufsize</i>	The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

## Returns

Completion status of operation:

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

## See also

[rtXmlpDecHexStr](#)

### 6.4.2.35 rtXmlpDecInt()

```
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

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

## Returns

Completion status of operation:

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

### 6.4.2.36 rtXmlpDecInt16()

```
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

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

#### Returns

Completion status of operation:

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

#### 6.4.2.37 rtXmlpDecInt64()

```
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

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

#### Returns

Completion status of operation:

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

#### 6.4.2.38 rtXmlpDecInt8()

```
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

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

#### Returns

Completion status of operation:

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

#### 6.4.2.39 rtXmlpDecNamedBits()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pBitMap</i>	Pointer to bit map structure that defined token to bit mappings.
<i>pvalue</i>	Pointer to buffer to receive decoded bit map.
<i>pnbits</i>	Number of bits in decoded bit map.
<i>bufsize</i>	Size of buffer passed in to received decoded bit values.

#### Returns

Completion status of operation:

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

#### 6.4.2.40 rtXmlpDecNamedBits64()

```
int rtXmlpDecNamedBits64 (
    OSCTXT * pctxt,
```

```

const OSBitMapItem * pBitMap,
OSOCKET * pvalue,
OSSIZE * pnbits,
OSSIZE bufsize )

```

This function decodes the contents of a named bit field.

This is a space-separated list of token values in which each token corresponds to a bit field in a bit map.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pBitMap</i>	Pointer to bit map structure that defined token to bit mappings.
<i>pvalue</i>	Pointer to buffer to receive decoded bit map.
<i>pnbits</i>	Number of bits in decoded bit map.
<i>bufsize</i>	Size of buffer passed in to received decoded bit values.

#### Returns

Completion status of operation:

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

#### 6.4.2.41 rtXmlpDecStrList()

```

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

<i>pctxt</i>	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.
<i>plist</i>	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.

#### 6.4.2.42 rtXmlpDecTime()

```
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

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

#### Returns

Completion status of operation:

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

#### 6.4.2.43 rtXmlpDecUInt()

```
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

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

## Returns

Completion status of operation:

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

### 6.4.2.44 rtXmlpDecUInt16()

```
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

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

## Returns

Completion status of operation:

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

### 6.4.2.45 rtXmlpDecUInt64()

```
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

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

## Returns

Completion status of operation:

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

### 6.4.2.46 rtXmlpDecUInt8()

```
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

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

## Returns

Completion status of operation:

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

### 6.4.2.47 rtXmlpDecUTF8Str()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>out</i>	Pointer to an array of OSUTF8CHAR to receive decoded UTF-8 string.
<i>max_len</i>	Length of out array.

## Returns

Completion status of operation:

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

### 6.4.2.48 rtXmlpDecXmlStr()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>outdata</i>	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.

### 6.4.2.49 rtXmlpDecXmlStrList()

```
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

<i>pctxt</i>	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.
<i>plist</i>	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.

### 6.4.2.50 rtXmlpDecXSIAttr()

```
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

<i>pctxt</i>	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.
<i>attrName</i>	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.

### 6.4.2.51 rtXmlpDecXSIAttr()

```
int rtXmlpDecXSIAttr (
    OSCTXT * pctxt )
```

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



## Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

## Returns

Completion status of operation:

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

### 6.4.2.52 rtXmlpDecXSITypeAttr()

```
int rtXmlpDecXSITypeAttr (
    OSCTXT * pctxt,
    const OSXMLNameFragments * attrName,
    const OSUTF8CHAR ** ppAttrValue )
```

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

Prior to calling this function, use `rtXmlpSelectAttribute` to select the attribute. After calling this function, if you want to read the same attribute again, you will need to call `rtXmlpSelectAttribute` again.

## Parameters

<i>pctxt</i>	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.
<i>attrName</i>	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.
<i>ppAttrValue</i>	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.

### 6.4.2.53 rtXmlpForceDecodeAsGroup()

```
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**

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

**6.4.2.54 rtXmlpGetAttributeCount()**

```
int rtXmlpGetAttributeCount (
    OSCTXT * pctxt )
```

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

**Parameters**

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

**Returns**

Completion status of operation:

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

**6.4.2.55 rtXmlpGetAttributeID()**

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

This function finds an attribute in the descriptor table.

**Parameters**

<i>attrName</i>	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.
-----------------	---

## Parameters

<i>nsidx</i>	Namespace index: <ul style="list-style-type: none"><li>• 0 = attribute is unqualified,</li><li>• positive value is user namespace from generated namespace table,</li><li>• negative value is predefined namespace like XSD instance and ect.</li></ul>
<i>nAttr</i>	Number of descriptors in table.
<i>attrNames</i>	Attributes descriptor table.
<i>attrPresent</i>	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.

### 6.4.2.56 rtXmlpGetCurrentLevel()

```
OSINT32 rtXmlpGetCurrentLevel (
    OSCTXT * pctx )
```

This function returns current nesting level.

## Parameters

<i>pctx</i>	Pointer to context block structure.
-------------	-------------------------------------

## Returns

Current nesting level.

### 6.4.2.57 rtXmlpGetNextAllElemID()

```
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.

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

**Parameters**

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

**Returns**

Completion status of operation:

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

**6.4.2.58 rtXmlpGetNextAllElemID16()**

```

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**

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

## Returns

Completion status of operation:

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

### 6.4.2.59 rtXmlpGetNextAllElemID32()

```
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

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

## Returns

Completion status of operation:

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

### 6.4.2.60 rtXmlpGetNextElem()

```
int rtXmlpGetNextElem (
    OSCTXT * pctxt,
    OSXMLElemDescr * pElem,
    OSINT32 level )
```

This function parse the next element start tag.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pElem</i>	Pointer to a structure to receive the decoded element descriptor.
<i>level</i>	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.

### 6.4.2.61 rtXmlpGetNextElemID()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>tab</i>	Elements descriptor table.
<i>nrows</i>	Number of descriptors in table.
<i>level</i>	Nesting level of parsed start tag. When value equal -1 function parse next start tag.
<i>continueParse</i>	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.

#### 6.4.2.62 rtXmlpGetNextSeqElemID()

```
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\_ELEMSISRQ.

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tab</i>	Elements descriptor table.
<i>pGroup</i>	Decode groups table.
<i>curID</i>	Current decode group.
<i>lastMandatoryID</i>	Identifier of last mandatory element.
<i>groupMode</i>	This parameter must be setted to TRUE when decoding groups or base types.

#### Returns

Completion status of operation:

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

#### 6.4.2.63 rtXmlpGetNextSeqElemID2()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>tab</i>	Elements descriptor table.
<i>pGroup</i>	Decode groups table.
<i>groups</i>	Number of groups in groups table. If checkRepeat is FALSE, you can pass 0 for this if the value is unknown.
<i>curID</i>	Current decode group.
<i>lastMandatoryID</i>	Identifier of last mandatory element.
<i>groupMode</i>	This parameter must be setted to TRUE when decode groups or base types.
<i>checkRepeat</i>	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.

#### 6.4.2.64 rtXmlpGetNextSeqElemIDExt()

```
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 rtXmpGetNextSeqElemID.

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 rtXmpGetNextSeqElemID.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>tab</i>	Elements descriptor table.
<i>pGroup</i>	Decode groups table.
<i>extRequired</i>	<i>extRequired[i]</i> corresponds to <i>pGroup[i]</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).
<i>postExtRootID</i>	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.
<i>curID</i>	Current decode group.
<i>lastMandatoryID</i>	Identifier of last mandatory element.
<i>groupMode</i>	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.

### 6.4.2.65 rtXmlpGetReader()

```
struct OSXMLReader* rtXmlpGetReader (
    OSCTXT * pctxt )
```

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

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

## Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

## Returns

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

### 6.4.2.66 rtXmlpGetXmInAttrs()

```
int rtXmlpGetXmInAttrs (
    OSCTXT * pctxt,
    OSRTDList * pNSAttrs )
```

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pNSAttrs</i>	A pointer to a linked list of OSXMLNamespace structures.

#### Returns

Completion status of operation:

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

#### 6.4.2.67 rtXmlpGetXSITypeAttr()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>ppAttrValue</i>	A pointer to a pointer to a UTF8 character string to received the decoded XSI type QName.
<i>nsidx</i>	A pointer to OSINT16 value to received the decoded XSI type namespace index.
<i>pLocalOffs</i>	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.

#### 6.4.2.68 rtXmlpGetXSITypeIndex()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>typetab</i>	XSI types descriptor table.
<i>typetabsiz</i>	Number of descriptors in table.

## Returns

Completion status of operation:

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

### 6.4.2.69 rtXmlpHasAttributes()

```
OSBOOL rtXmlpHasAttributes (  
    OSCTXT * pctxt )
```

This function checks accessibility of attributes.

## Parameters

<i>pctxt</i>	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.

### 6.4.2.70 rtXmlpHideAttributes()

```
void rtXmlpHideAttributes (  
    OSCTXT * pctxt )
```

Disable access to attributes.

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

#### 6.4.2.71 rtXmIplsDecodeAsGroup()

```
OSBOOL rtXmIplsDecodeAsGroup (  
    OSCTXT * pctxt )
```

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

#### Returns

State of mode:

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

#### 6.4.2.72 rtXmIplsEmptyElement()

```
OSBOOL rtXmIplsEmptyElement (  
    OSCTXT * pctxt )
```

Check element content: empty or not.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

#### Returns

Status of element content:

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

#### 6.4.2.73 rtXmpltLastEventDone()

```
OSBOOL rtXmpltIsLastEventDone (
    OSCTXT * pctxt )
```

Check processing status of current tag.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

##### Returns

Status of element content:

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

#### 6.4.2.74 rtXmpltUTF8Encoding()

```
OSBOOL rtXmpltIsUTF8Encoding (
    OSCTXT * pctxt )
```

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

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

##### Returns

State of mode:

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

#### 6.4.2.75 rtXmpltListHasItem()

```
OSBOOL rtXmpltListHasItem (
    OSCTXT * pctxt )
```

Check for end of decoded token list.



## Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

## Returns

State of decoded list:

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

### 6.4.2.76 rtXmlpLookupXSITypeIndex()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pXsiType</i>	A pointer to XSI type name.
<i>xsiTypeIdx</i>	A XSI type namespace index.
<i>typetab</i>	XSI types descriptor table.
<i>typetabsiz</i>	Number of descriptors in table.

## Returns

Completion status of operation:

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

### 6.4.2.77 rtXmlpMarkLastEventActive()

```
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

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

#### Returns

Completion status of operation:

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

#### 6.4.2.78 rtXmIplMarkPos()

```
void rtXmIplMarkPos (  
    OSCTXT * pctxt )
```

Save current decode position.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

#### 6.4.2.79 rtXmIplMatchEndTag()

```
int rtXmIplMatchEndTag (  
    OSCTXT * pctxt,  
    OSINT32 level )
```

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>level</i>	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.

#### 6.4.2.80 rtXmlpMatchStartTag()

```
int rtXmlpMatchStartTag (
    OSCTXT * pctxt,
    const OSUTF8CHAR * elemLocalName,
    OSINT16 nsidx )
```

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

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>elemLocalName</i>	Name of parsed element.
<i>nsidx</i>	Namespace index: <ul style="list-style-type: none"><li>• 0 = attribute is unqualified,</li><li>• positive value is user namespace from generated namespace table,</li><li>• negative value is predefined namespace like XSD instance and ect.</li></ul>

##### Returns

Completion status of operation:

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

#### 6.4.2.81 rtXmlpNeedDecodeAttributes()

```
OSBOOL rtXmlpNeedDecodeAttributes (
    OSCTXT * pctxt )
```

This function checks if attributes were previously decoded.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

##### Returns

State of attributes:

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

#### 6.4.2.82 rtXmlpReadBytes()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>pbuf</i>	Pointer to static buffer (byte array) to receive data. The buffer must be at least large enough to hold the requested number of bytes.
<i>nbytes</i>	Number of bytes to read.

##### Returns

State of mode:

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

#### 6.4.2.83 rtXmlpResetMarkedPos()

```
void rtXmlpResetMarkedPos (
    OSCTXT * pctxt )
```

Reset saved decode position.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

#### 6.4.2.84 rtXmlpRewindToMarkedPos()

```
void rtXmlpRewindToMarkedPos (
    OSCTXT * pctxt )
```

Rewind to saved decode position.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

### 6.4.2.85 rtXmlpSelectAttribute()

```
int rtXmlpSelectAttribute (
    OSCTXT * pctxt,
    OSXMLNameFragments * pAttr,
    OSINT16 * nsidx,
    OSSIZE attrIndex )
```

This function selects attribute to decode.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pAttr</i>	Pointer to structure to receive the various parts of an attribute name.
<i>nsidx</i>	Pointer to value to receive namespace index: <ul style="list-style-type: none"><li>• 0 = attribute is unqualified,</li><li>• positive value is user namespace from generated namespace table,</li><li>• negative value is predefined namespace like XSD instance and ect (see enum OSXMLNsIndex)</li></ul>
<i>attrIndex</i>	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.

### 6.4.2.86 rtXmlpSetListMode()

```
void rtXmlpSetListMode (
    OSCTXT * pctxt )
```

Sets list mode.

Attribute value or element content is decoded by tokens.

### Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

### 6.4.2.87 rtXmIplSetMixedContentMode()

```
OSBOOL rtXmIplSetMixedContentMode (
    OSCTXT * pctxt,
    OSBOOL mixedContentMode )
```

Sets mixed content mode.

### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>mixedContentMode</i>	Enable/disable mixed mode.

### Returns

Previously state of mixed mode.

### 6.4.2.88 rtXmIplSetNamespaceTable()

```
void rtXmIplSetNamespaceTable (
    OSCTXT * pctxt,
    const OSUTF8CHAR * namespaceTable[],
    OSSIZE nmNamespaces )
```

Sets user namespace table.

### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>namespaceTable</i>	Array of namespace URI strings.
<i>nmNamespaces</i>	Number of namespaces in table.

### 6.4.2.89 rtXmIplSetWhiteSpaceMode()

```
void rtXmIplSetWhiteSpaceMode (
```

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>whiteSpaceMode</i>	White space mode.

#### Returns

Previously set whitespace mode.

## 6.5 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.

### Macros

- #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.*

## 6.5.1 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.

## 6.5.2 Macro Definition Documentation

### 6.5.2.1 XML\_E\_BASE

```
#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`.

### 6.5.2.2 XML\_E\_ELEMMISRQ

```
#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`.

### 6.5.2.3 XML\_E\_ELEMSMISRQ

```
#define XML_E_ELEMSMISRQ (XML_E_BASE-9)
```

Missing required elements.

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

Definition at line 128 of file rtXmlErrCodes.h.

### 6.5.2.4 XML\_E\_FLDABSENT

```
#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.

### 6.5.2.5 XML\_E\_NOMATCH

```
#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.

### 6.5.2.6 XML\_E\_NSURINOTFOU

```
#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.

### 6.5.2.7 XML\_E\_TAGMISMATCH

```
#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.

### 6.5.2.8 XML\_OK\_EOB

```
#define XML_OK_EOB 0x7fffffff
```

End of block marker.

Definition at line 51 of file rtXmlErrCodes.h.

### 6.5.2.9 XML\_OK\_FRAG

```
#define XML_OK_FRAG XML_OK_EOB
```

Maintained for backward compatibility.

Definition at line 54 of file rtXmlErrCodes.h.

## 6.6 DOM API functions.

### Typedefs

- typedef int **OSRTDOMError**  
*0 - OK < 0 - Error code (see rtxsrc/rtxErrCodes.h) > 0 - Supplementary code*

### Functions

- EXTERNDOM void **domParserInit** ()  
*Inits parser.*
- EXTERNDOM void **domParserShutdown** ()  
*Shut downs parser.*
- EXTERNDOM **OSRTDOMError** **domParseFile** (const char \*fileSpec, OSRTDOMDocPtr \*pXmlDoc)  
*Parse the file into a DOM document.*
- EXTERNDOM **OSRTDOMError** **domGetRootElement** (OSRTDOMDocPtr xmlDoc, OSRTDOMNodePtr \*pNode)  
*Returns root node for the document.*
- EXTERNDOM **OSRTDOMError** **domGetDoc** (OSRTDOMNodePtr node, OSRTDOMDocPtr \*pXmlDoc)  
*Returns the document for the given element.*
- EXTERNDOM void **domFreeDoc** (OSRTDOMDocPtr xmlDoc)  
*Frees document.*
- EXTERNDOM **OSRTDOMError** **domGetNext** (const OSRTDOMNodePtr curNode, OSRTDOMNodePtr \*pNext↔Node)  
*Returns next element node (down sibling).*
- EXTERNDOM **OSRTDOMError** **domGetChild** (const OSRTDOMNodePtr curNode, OSRTDOMNodePtr \*p↔ChildNode)  
*Returns first (top) child node.*
- EXTERNDOM **OSRTDOMError** **domGetElementName** (const OSRTDOMNodePtr curNode, const OSUTF8C↔HAR \*\*ppName, const OSUTF8CHAR \*\*ppNsPrefix, const OSUTF8CHAR \*\*ppNsUri)  
*Returns node's name and prefix (if any).*
- EXTERNDOM int **domGetNodeContent** (OSRTDOMContCtxtPtr \*ppCtxt, const OSRTDOMNodePtr curNode, const OSUTF8CHAR \*\*ppValue, size\_t \*pValueLen, OSBOOL \*pCdataProcessed)  
*Gets the node's content.*
- EXTERNDOM **OSRTDOMError** **domGetNodeFirstAttribute** (OSRTDOMAttrCtxtPtr \*ppCtxt, const OSRTDOM↔NodePtr curNode, OSRTDOMAttrPtr \*pTopAttrNode)  
*Returns the pointer to the first (top) attribute of the node.*
- EXTERNDOM int **domGetNodeAttributesNum** (const OSRTDOMNodePtr curNode)  
*Returns number of attributes in the node.*
- EXTERNDOM **OSRTDOMError** **domGetNextAttr** (OSRTDOMAttrCtxtPtr \*ppCtxt, const OSRTDOMAttrPtr cur↔AttrNode, OSRTDOMAttrPtr \*pAttrNode)  
*Returns the next attribute.*
- EXTERNDOM **OSRTDOMError** **domGetAttrData** (const OSRTDOMAttrPtr curAttrNode, const OSUTF8CHAR \*\*ppAttrName, const OSUTF8CHAR \*\*ppAttrValue, const OSUTF8CHAR \*\*ppAttrPrefix, const OSUTF8CHAR \*\*ppAttrUri)  
*Returns the attribute's name and value.*
- EXTERNDOM **OSRTDOMError** **domSaveDoc** (OSRTDOMDocPtr xmlDoc, const char \*filename)  
*Saves DOM document to a file.*

- EXTERNDOM [OSRTError domCreateDocument](#) (OSCTXT \*pctxt, OSRTDOMDocPtr \*pXmiDoc, const OSUTF8CHAR \*pNodeName, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs)  
*Creates a new document and a root element.*
- EXTERNDOM [OSRTError domCreateChild](#) (OSCTXT \*pctxt, OSRTDOMNodePtr parentNode, const OSUTF8CHAR \*pNodeName, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs, OSRTDOMNodePtr \*pNewNode)  
*Creates new child node.*
- EXTERNDOM [OSRTError domAddAttribute](#) (OSCTXT \*pctxt, OSRTDOMNodePtr node, const OSUTF8CHAR \*pAttrName, const OSUTF8CHAR \*pAttrValue, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs)  
*Creates new attribute and adds it to the node.*
- EXTERNDOM [OSRTError domAddContent](#) (OSRTDOMNodePtr node, const OSUTF8CHAR \*pContent, size\_t contentLen)  
*Adds content (text) to the node.*
- EXTERNDOM [OSRTError domAddCdata](#) (OSRTDOMNodePtr node, const OSUTF8CHAR \*pContent, size\_t contentLen)  
*Adds CDATA section to the node.*

## 6.6.1 Detailed Description

## 6.6.2 Function Documentation

### 6.6.2.1 domAddAttribute()

```
EXTERNDOM OSRTError domAddAttribute (
    OSCTXT * pctxt,
    OSRTDOMNodePtr node,
    const OSUTF8CHAR * pAttrName,
    const OSUTF8CHAR * pAttrValue,
    OSXMLNamespace * pNS,
    OSRTDList * pNSAttrs )
```

Creates new attribute and adds it to the node.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>node</i>	Pointer to the node where attribute to be added.
<i>pAttrName</i>	Pointer to the null-terminated string containing name of the attribute.
<i>pAttrValue</i>	Pointer to the null-terminated string containing name of the value.
<i>pNS</i>	XML namespace information (prefix and URI).
<i>pNSAttrs</i>	List of namespace attributes to be added to element.

#### Returns

0 - success, otherwise error code.

### 6.6.2.2 domAddCdata()

```
EXTERNDOM OSRTDOMEError domAddCdata (  
    OSRTDOMNodePtr node,  
    const OSUTF8CHAR * pContent,  
    size_t contentLen )
```

Adds CDATA section to the node.

#### Parameters

<i>node</i>	Pointer to the node where content to be added.
<i>pContent</i>	Pointer to UTF-8 encoded content to be wrapped by CDATA section.
<i>contentLen</i>	Length of the content

#### Returns

0 - success, otherwise error code.

### 6.6.2.3 domAddContent()

```
EXTERNDOM OSRTDOMEError domAddContent (  
    OSRTDOMNodePtr node,  
    const OSUTF8CHAR * pContent,  
    size_t contentLen )
```

Adds content (text) to the node.

#### Parameters

<i>node</i>	Pointer to the node where content to be added.
<i>pContent</i>	Pointer to UTF-8 encoded content.
<i>contentLen</i>	Length of the content

#### Returns

0 - success, otherwise error code.

#### 6.6.2.4 domCreateChild()

```
EXTERNDOM OSRTDOMError domCreateChild (
    OSCTXT * pctxt,
    OSRTDOMNodePtr parentNode,
    const OSUTF8CHAR * pNodeName,
    OSXMLNamespace * pNS,
    OSRTDList * pNSAttrs,
    OSRTDOMNodePtr * pNewNode )
```

Creates new child node.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>parentNode</i>	Pointer to the parent node
<i>pNodeName</i>	Name of the node
<i>pNS</i>	XML namespace information (prefix and URI).
<i>pNSAttrs</i>	List of namespace attributes to be added to element.
<i>pNewNode</i>	Pointer to pointer to newly created node.

##### Returns

0 - success, otherwise error code.

#### 6.6.2.5 domCreateDocument()

```
EXTERNDOM OSRTDOMError domCreateDocument (
    OSCTXT * pctxt,
    OSRTDOMDocPtr * pXmlDoc,
    const OSUTF8CHAR * pNodeName,
    OSXMLNamespace * pNS,
    OSRTDList * pNSAttrs )
```

Creates a new document and a root element.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pXmlDoc</i>	Pointer to pointer to the newly created empty DOM document.
<i>pNodeName</i>	Name of the node
<i>pNS</i>	XML namespace information (prefix and URI).
<i>pNSAttrs</i>	List of namespace attributes to be added to element.

## Returns

0 - success, otherwise error code.

### 6.6.2.6 domFreeDoc()

```
EXTERNDOM void domFreeDoc (
    OSRTDOMDocPtr xmlDoc )
```

Frees document.

#### Parameters

<i>xmlDoc</i>	Pointer to the DOM document context
---------------	-------------------------------------

### 6.6.2.7 domGetAttrData()

```
EXTERNDOM OSRTDOMError domGetAttrData (
    const OSRTDOMAttrPtr curAttrNode,
    const OSUTF8CHAR ** ppAttrName,
    const OSUTF8CHAR ** ppAttrValue,
    const OSUTF8CHAR ** ppAttrPrefix,
    const OSUTF8CHAR ** ppAttrUri )
```

Returns the attribute's name and value.

#### Parameters

<i>curAttrNode</i>	Pointer to current attribute node,
<i>ppAttrName</i>	Pointer to pointer to receive attribute's name.
<i>ppAttrValue</i>	Pointer to pointer to receive attribute's value.
<i>ppAttrPrefix</i>	Pointer to pointer to receive namespace's prefix.
<i>ppAttrUri</i>	Pointer to pointer to receive namespace's uri.

## Returns

0 - success, otherwise error code.



### 6.6.2.8 domGetChild()

```
EXTERNDOM OSRTDOMError domGetChild (
    const OSRTDOMNodePtr curNode,
    OSRTDOMNodePtr * pChildNode )
```

Returns first (top) child node.

#### Parameters

<i>curNode</i>	Pointer to current node,
<i>pChildNode</i>	Pointer to pointer to receive the child node.

#### Returns

0 - success, otherwise error code.

### 6.6.2.9 domGetDoc()

```
EXTERNDOM OSRTDOMError domGetDoc (
    OSRTDOMNodePtr node,
    OSRTDOMDocPtr * pXmlDoc )
```

Returns the document for the given element.

#### Parameters

<i>node</i>	Pointer to a node.
<i>pXmlDoc</i>	Pointer to a pointer to the DOM document context

#### Returns

0 - success, otherwise error code.

### 6.6.2.10 domGetElementName()

```
EXTERNDOM OSRTDOMError domGetElementName (
    const OSRTDOMNodePtr curNode,
    const OSUTF8CHAR ** ppName,
    const OSUTF8CHAR ** ppNsPrefix,
    const OSUTF8CHAR ** ppNsUri )
```

Returns node's name and prefix (if any).

### Parameters

<i>curNode</i>	Pointer to current node,
<i>ppName</i>	Pointer to pointer to receive the node's local name.
<i>ppNsPrefix</i>	Pointer to pointer to receive the node's prefix. May be NULL, if prefix is not important.
<i>ppNsUri</i>	Pointer to pointer to receive the namespace's Uri.

### Returns

0 - success, otherwise error code.

### 6.6.2.11 domGetNext()

```
EXTERNDOM OSRDOMError domGetNext (
    const OSRDOMNodePtr curNode,
    OSRDOMNodePtr * pNextNode )
```

Returns next element node (down sibling).

### Parameters

<i>curNode</i>	Pointer to current node,
<i>pNextNode</i>	Pointer to pointer to receive the next node.

### Returns

0 - success, otherwise error code.

### 6.6.2.12 domGetNextAttr()

```
EXTERNDOM OSRDOMError domGetNextAttr (
    OSRDOMAttrCtxtPtr * ppCtxt,
    const OSRDOMAttrPtr curAttrNode,
    OSRDOMAttrPtr * pAttrNode )
```

Returns the next attribute.

### Parameters

<i>ppCtxt</i>	Pointer to an implementation-specific context pointer,
<i>curAttrNode</i>	Pointer to current attribute node,
<i>pAttrNode</i>	Pointer to pointer to receive the next attribute node.

**Returns**

0 - success, otherwise error code.

**6.6.2.13 domGetNodeAttributesNum()**

```
EXTERNDOM int domGetNodeAttributesNum (
    const OSRTDOMNodePtr curNode )
```

Returns number of attributes in the node.

**Parameters**

<i>curNode</i>	Pointer to current node,
----------------	--------------------------

**Returns**

Number of attributes in the node.

**6.6.2.14 domGetNodeContent()**

```
EXTERNDOM int domGetNodeContent (
    OSRTDOMContCtxtPtr * ppCtxt,
    const OSRTDOMNodePtr curNode,
    const OSUTF8CHAR ** ppValue,
    size_t * pValueLen,
    OSBOOL * pCdataProcessed )
```

Gets the node's content.

**Parameters**

<i>ppCtxt</i>	Pointer to an implementation-specific content context pointer;
<i>curNode</i>	Pointer to the current node;
<i>ppValue</i>	Pointer to pointer to receive value;
<i>pValueLen</i>	Pointer to value length; may be NULL.
<i>pCdataProcessed</i>	Pointer to boolean value; this value will be set to TRUE if CDATA section was proceeded; otherwise FALSE.

**Returns**

0 - if content retrieved successfully <0 - the error code;

### 6.6.2.15 domGetNodeFirstAttribute()

```
EXTERNDOM OSRTDOMError domGetNodeFirstAttribute (
    OSRTDOMAttrCtxtPtr * ppCtxt,
    const OSRTDOMNodePtr curNode,
    OSRTDOMAttrPtr * pTopAttrNode )
```

Returns the pointer to the first (top) attribute of the node.

The context pointer will be passed to subsequent calls to domGetNextAttr.

#### Parameters

<i>ppCtxt</i>	Pointer to an implementation-specific context pointer,
<i>curNode</i>	Pointer to current node,
<i>pTopAttrNode</i>	Pointer to pointer to receive the first attribute node. domGetNextAttr/domGetPrevAttr functions may be used to get next/previous attributes.

#### Returns

0 - success, otherwise error code.

### 6.6.2.16 domGetRootElement()

```
EXTERNDOM OSRTDOMError domGetRootElement (
    OSRTDOMDocPtr xmlDoc,
    OSRTDOMNodePtr * pNode )
```

Returns root node for the document.

#### Parameters

<i>xmlDoc</i>	Pointer to the DOM document context
<i>pNode</i>	Pointer to pointer to receive the root node.

#### Returns

0 - success, otherwise error code.

### 6.6.2.17 domParseFile()

```
EXTERNDOM OSRTDOMError domParseFile (  
    const char * fileSpec,  
    OSRTDOMDocPtr * pXmlDoc )
```

Parse the file into a DOM document.

#### Parameters

<i>fileSpec</i>	File name
<i>pXmlDoc</i>	Pointer to a pointer to newly created DOM document.

#### Returns

0 - success, otherwise error code.

### 6.6.2.18 domSaveDoc()

```
EXTERNDOM OSRTDOMError domSaveDoc (  
    OSRTDOMDocPtr xmlDoc,  
    const char * filename )
```

Saves DOM document to a file.

This function is supplementary and it is not obligatory to implement it.

#### Parameters

<i>xmlDoc</i>	Pointer to the DOM document context.
<i>filename</i>	Pointer to file name.

#### Returns

0 - success, otherwise error code.

## 6.7 DOM runtime encode/decode functions.

### Functions

- EXTERNDOM int [rtDomAddAttr](#) (OSCTXT \*pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr node, const OSUTF8CHAR \*attrName, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs)  
*Adds attribute to the node.*
- EXTERNDOM int [rtDomAddNode](#) (OSCTXT \*pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr parent←Node, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs)  
*Adds child node to the parent's node.*
- EXTERNDOM int [rtDomAddNSAttrs](#) (OSCTXT \*pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr root←Node, OSRTDList \*pNSAttrs)  
*Adds namespace attributes to the root node.*
- EXTERNDOM int [rtDomEncXSIAttrs](#) (OSCTXT \*pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr node, OSBOOL needXSI)  
*Adds XSI attributes to the node.*
- EXTERNDOM int [rtDomDecodeDoc](#) (OSRTDOMDocPtr domDoc, struct OSSAXHandlerBase \*pSaxBase)  
*This function starts decoding of the DOM document.*
- EXTERNDOM int [rtDomEncStringValue](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*pvalue)  
*This function encodes a variable of the XSD string type.*
- EXTERNDOM int [rtDomEncString](#) (OSCTXT \*pctxt, OSXMLSTRING \*pvalue)  
*This function encodes a variable of the XSD string type.*
- EXTERNDOM int [rtDomEncAny](#) (OSCTXT \*pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr parentNode, const OSXMLSTRING \*pXmlData, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs)  
*This function encodes a variable of the XSD any type.*
- EXTERNDOM int [rtDomEncAnyAttr](#) (OSCTXT \*pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr node, OSRTDList \*pAnyAttrList)  
*This function encodes a list of OSAnyAttr attributes in which the name and value are given as a UTF-8 string.*
- EXTERNDOM int [rtDomSetNode](#) (OSCTXT \*pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr curNode)  
*Adds content or CDATA section to the node.*
- EXTERNDOM int [rtDomAddSubTree](#) (OSCTXT \*pctxt, OSRTDOMDocPtr xmlDoc, OSRTDOMNodePtr node, const OSUTF8CHAR \*pXmlData, size\_t dataLen)  
*This function adds the subtree to the specified node.*

### 6.7.1 Detailed Description

### 6.7.2 Function Documentation

### 6.7.2.1 rtDomAddAttr()

```
EXTERNDOM int rtDomAddAttr (
    OSCTXT * pctxt,
    OSRTDOMDocPtr domDoc,
    OSRTDOMNodePtr node,
    const OSUTF8CHAR * attrName,
    OSXMLNamespace * pNS,
    OSRTDList * pNSAttrs )
```

Adds attribute to the node.

The value is taken from the context's buffer (*pctxt->buffer.data* with the length *pctxt->buffer.byteIndex*). This function resets *pctxt->buffer.byteIndex* to 0.

#### Parameters

<i>pctxt</i>	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.
<i>domDoc</i>	A pointer to a DOM-document.
<i>node</i>	A pointer to a node where attribute to be added.
<i>attrName</i>	A pointer to attribute's name.
<i>pNS</i>	XML namespace information (prefix and URI).
<i>pNSAttrs</i>	List of namespace attributes to be added to element.

#### Returns

Completion status of operation:

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

### 6.7.2.2 rtDomAddNode()

```
EXTERNDOM int rtDomAddNode (
    OSCTXT * pctxt,
    OSRTDOMDocPtr domDoc,
    OSRTDOMNodePtr parentNode,
    const OSUTF8CHAR * elemName,
    OSXMLNamespace * pNS,
    OSRTDList * pNSAttrs )
```

Adds child node to the parent's node.

The content (text) is taken from the context's buffer (*pctxt->buffer.data* with the length *pctxt->buffer.byteIndex*). If *pctxt->buffer.byteIndex* is 0 then empty element is added. This function resets *pctxt->buffer.byteIndex* to 0.

## Parameters

<i>pctxt</i>	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.
<i>domDoc</i>	A pointer to a DOM-document.
<i>parentNode</i>	A pointer to a parent node.
<i>elemName</i>	A pointer to element's name.
<i>pNS</i>	XML namespace information (prefix and URI).
<i>pNSAttrs</i>	List of namespace attributes to be added to element.

## Returns

Completion status of operation:

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

### 6.7.2.3 rtDomAddNSAttrs()

```
EXTERNDOM int rtDomAddNSAttrs (  
    OSCTXT * pctxt,  
    OSRTDOMDocPtr domDoc,  
    OSRTDOMNodePtr rootNode,  
    OSRTDList * pNSAttrs )
```

Adds namespace attributes to the root node.

## Parameters

<i>pctxt</i>	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.
<i>domDoc</i>	A pointer to a DOM-document.
<i>rootNode</i>	A pointer to a root node.
<i>pNSAttrs</i>	A pointer to a list of namespace attrs.

## Returns

Completion status of operation:

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



#### 6.7.2.4 rtDomAddSubTree()

```
EXTERNDOM int rtDomAddSubTree (
    OSCTXT * pctxt,
    OSRTDOMDocPtr xmlDoc,
    OSRTDOMNodePtr node,
    const OSUTF8CHAR * pXmlData,
    size_t dataLen )
```

This function adds the subtree to the specified node.

The *pXmlData* contains the fragment of XML data. This fragment is parsed to sub-tree and then inserted as children to the 'node'.

##### Parameters

<i>pctxt</i>	Pointer to the context structure. This context might be used for memory allocations, if necessary.
<i>xmlDoc</i>	Pointer to the document
<i>node</i>	Pointer to the node where sub-tree to be inserted.
<i>pXmlData</i>	Pointer to UTF-8 string representing fragment of XML data. For example, "<elem>data</elem>".
<i>dataLen</i>	Length of XML data.

##### Returns

Completion status of operation:

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

#### 6.7.2.5 rtDomDecodeDoc()

```
EXTERNDOM int rtDomDecodeDoc (
    OSRTDOMDocPtr domDoc,
    struct OSSAXHandlerBase * pSaxBase )
```

This function starts decoding of the DOM document.

##### Parameters

<i>domDoc</i>	A pointer to a DOM-document.
<i>pSaxBase</i>	A pointer to SAX handler's base.

##### Returns

Completion status of operation:

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

### 6.7.2.6 rtDomEncAny()

```

EXTERNDOM int rtDomEncAny (
    OSCTXT * pctxt,
    OSRTDOMDocPtr domDoc,
    OSRTDOMNodePtr parentNode,
    const OSXMLSTRING * pXmlData,
    const OSUTF8CHAR * elemName,
    OSXMLNamespace * pNS,
    OSRTDList * pNSAttrs )

```

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

<i>pctxt</i>	Pointer to context block structure.
<i>domDoc</i>	A pointer to a DOM-document.
<i>parentNode</i>	A pointer to a parent node.
<i>pXmlData</i>	Value to be encoded. This is a string containing the fully-encoded XML text to be copied to the output stream.
<i>elemName</i>	XML element name. A name must be provided. If NULL pointer is passed, no element tag is added to the encoded value.
<i>pNS</i>	XML namespace information (prefix and URI).
<i>pNSAttrs</i>	List of namespace attributes to be added to element.

#### Returns

Completion status of operation:

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

### 6.7.2.7 rtDomEncAnyAttr()

```

EXTERNDOM int rtDomEncAnyAttr (
    OSCTXT * pctxt,
    OSRTDOMDocPtr domDoc,
    OSRTDOMNodePtr node,
    OSRTDList * pAnyAttrList )

```

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

### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>domDoc</i>	A pointer to a DOM-document.
<i>node</i>	A pointer to a node where attributes to be added.
<i>pAnyAttrList</i>	List of attributes.

### Returns

Completion status of operation:

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

### 6.7.2.8 rtDomEncString()

```
EXTERNDOM int rtDomEncString (
    OSCTXT * pctxt,
    OSXMLSTRING * pvalue )
```

This function encodes a variable of the XSD string type.

### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	XML string value to be encoded.

### Returns

Completion status of operation:

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

### 6.7.2.9 rtDomEncStringValue()

```
EXTERNDOM int rtDomEncStringValue (
    OSCTXT * pctxt,
    const OSUTF8CHAR * pvalue )
```

This function encodes a variable of the XSD string type.

### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Null-terminated XML string value to be encoded.

### Returns

Completion status of operation:

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

### 6.7.2.10 rtDomEncXSIAttrs()

```
EXTERNDOM int rtDomEncXSIAttrs (  
    OSCTXT * pctxt,  
    OSRTDOMDocPtr domDoc,  
    OSRTDOMNodePtr node,  
    OSBOOL needXSI )
```

Adds XSI attributes to the node.

### Parameters

<i>pctxt</i>	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.
<i>domDoc</i>	A pointer to a DOM-document.
<i>node</i>	A pointer to a node.
<i>needXSI</i>	Determines whether XSI namespace declaration needed.

### Returns

Completion status of operation:

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

### 6.7.2.11 rtDomSetNode()

```
EXTERNDOM int rtDomSetNode (  
    OSCTXT * pctxt,
```

```
OSRDOMDocPtr domDoc,  
OSRDOMNodePtr curNode )
```

Adds content or CDATA section to the node.

The content (text) is taken from the context's buffer (pctx->buffer.data with the length pctx->buffer.byIndex). This function resets pctx->buffer.byIndex to 0.

#### Parameters

<i>pctx</i>	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.
<i>domDoc</i>	A pointer to a DOM-document.
<i>curNode</i>	A pointer to a current node.

#### Returns

Completion status of operation:

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



## Chapter 7

# Class Documentation

### 7.1 OSXMLGroupDesc Struct Reference

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

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

#### 7.1.1 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 140 of file osrtxml.h.

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

- [osrtxml.h](#)

### 7.2 OSXMLStringListParser Class Reference

Class enabling parsing of an XML Schema list into strings.

```
#include <rtXmlpCppDecFuncs.h>
```

## Public Member Functions

- [OSXMLStringListParser](#) (OSCTXT \*pctxt)  
*Create a parser on the given context.*
- [int next](#) (OSRTXMLString &value)  
*Assign the next string from the XML schema list to value.*

### 7.2.1 Detailed Description

Class enabling parsing of an XML Schema list into strings.

Definition at line 47 of file rtXmIpcppDecFuncs.h.

### 7.2.2 Constructor & Destructor Documentation

#### 7.2.2.1 OSXMLStringListParser()

```
OSXMLStringListParser::OSXMLStringListParser (  
    OSCTXT * pctxt )
```

Create a parser on the given context.

The OSXMLReader associated with the context is used to read the XML value.

#### Parameters

<i>pctxt</i>	Holds state information and provides access to the input that is to be parsed.
--------------	--

### 7.2.3 Member Function Documentation

#### 7.2.3.1 next()

```
int OSXMLStringListParser::next (  
    OSRTXMLString & value )
```

Assign the next string from the XML schema list to value.

To get all of the values from the list, call this function repeatedly until it returns zero or a negative value. After that, this object can no longer be used.



### Parameters

<i>value</i>	Receives the next string value or empty string if there isn't one.
--------------	--

### Returns

Return value indicates the result as follows: return < 0: error return == 0: no more strings; value is assigned empty return == 1: value is assigned the next string

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

- [rtXmlpCppDecFuncs.h](#)



## Chapter 8

# File Documentation

### 8.1 osrtdom.h File Reference

DOM low-level C encode/decode functions.

```
#include "rtxsrc/rtxCommon.h"
#include "rtdomsrc/rtxDomDefs.h"
#include "rtxmlsrc/osrtxml.h"
#include "rtdomsrc/domAPI.h"
#include "rtxmlsrc/rtXmlNamespace.h"
```

#### Functions

- EXTERNDOM int [rtDomAddAttr](#) (OSCTXT \*pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr node, const OSUTF8CHAR \*attrName, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs)  
*Adds attribute to the node.*
- EXTERNDOM int [rtDomAddNode](#) (OSCTXT \*pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr parent←Node, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs)  
*Adds child node to the parent's node.*
- EXTERNDOM int [rtDomAddNSAttrs](#) (OSCTXT \*pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr root←Node, OSRTDList \*pNSAttrs)  
*Adds namespace attributes to the root node.*
- EXTERNDOM int [rtDomEncXSIAttrs](#) (OSCTXT \*pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr node, OSBOOL needXSI)  
*Adds XSI attributes to the node.*
- EXTERNDOM int [rtDomDecodeDoc](#) (OSRTDOMDocPtr domDoc, struct OSSAXHandlerBase \*pSaxBase)  
*This function starts decoding of the DOM document.*
- EXTERNDOM int [rtDomEncStringValue](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*pvalue)  
*This function encodes a variable of the XSD string type.*
- EXTERNDOM int [rtDomEncString](#) (OSCTXT \*pctxt, OSXMLSTRING \*pvalue)  
*This function encodes a variable of the XSD string type.*

- EXTERNDOM int [rtDomEncAny](#) (OSCTXT \*pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr parentNode, const OSXMLSTRING \*pXmlData, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs)  
*This function encodes a variable of the XSD any type.*
- EXTERNDOM int [rtDomEncAnyAttr](#) (OSCTXT \*pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr node, OSRTDList \*pAnyAttrList)  
*This function encodes a list of OSAnyAttr attributes in which the name and value are given as a UTF-8 string.*
- EXTERNDOM int [rtDomSetNode](#) (OSCTXT \*pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr curNode)  
*Adds content or CDATA section to the node.*
- EXTERNDOM int [rtDomAddSubTree](#) (OSCTXT \*pctxt, OSRTDOMDocPtr xmlDoc, OSRTDOMNodePtr node, const OSUTF8CHAR \*pXmlData, size\_t dataLen)  
*This function adds the subtree to the specified node.*

### 8.1.1 Detailed Description

DOM low-level C encode/decode functions.

## 8.2 osrxml.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"
```

### Classes

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

### Macros

- #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.*

## Typedefs

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

## Enumerations

- enum [OSXMLWhiteSpaceMode](#)  
*Whitespace treatment options.*

## Functions

- int [rtXmlInitContext](#) (OSCTXT \*pctxt)  
*This function initializes a context variable for XML encoding or decoding.*
- int [rtXmlInitContextUsingKey](#) (OSCTXT \*pctxt, const OSOCTET \*key, OSSIZE keylen)  
*This function initializes a context using a run-time key.*
- int [rtXmlInitCtxtAppInfo](#) (OSCTXT \*pctxt)  
*This function initializes the XML application info section of the given context.*
- int [rtXmlCreateFileInputSource](#) (OSCTXT \*pctxt, const char \*filepath)  
*This function creates an XML document file input source.*
- void [rtXmlMemFreeAnyAttrs](#) (OSCTXT \*pctxt, OSRTDList \*pAnyAttrList)  
*This function frees a list of anyAttribute that is a member of OSXSDAnyType structure.*
- 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.*
- int [rtXmlDecBase64Str](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocets, OSINT32 bufsize)  
*This function decodes a contents of a Base64-encode binary string into a static memory structure.*
- int [rtXmlDecBase64Str64](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocets, OSSIZE bufsize)  
*This function is identical to [rtXmlDecBase64Str](#) except that is supports a 64-bit integer length on 64-bit systems.*
- int [rtXmlDecBase64StrValue](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocets, OSSIZE bufSize, OS←  
SIZE srcDataLen)  
*This function decodes a contents of a Base64-encode binary string into the specified octet array.*
- int [rtXmlDecBase64StrValue64](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocets, OSSIZE bufSize, OS←  
SIZE srcDataLen)  
*This function decodes is identical to [rtXmlDecBase64StrValue](#) except that it supports a 64-bit integer length on 64-bit systems.*
- int [rtXmlDecBigInt](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*ppvalue)  
*This function will decode a variable of the XSD integer type.*
- int [rtXmlDecBool](#) (OSCTXT \*pctxt, OSBOOL \*pvalue)  
*This function decodes a variable of the boolean type.*
- int [rtXmlDecDate](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'date' type.*
- int [rtXmlDecTime](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'time' type.*
- int [rtXmlDecDateTime](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'dateTime' type.*
- int [rtXmlDecDecimal](#) (OSCTXT \*pctxt, OSREAL \*pvalue)

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

  - int **rtXmlDecDouble** (OSCTXT \*pctx, OSREAL \*pvalue)
- This function decodes the contents of a float or double data type.*

  - int **rtXmlDecDynBase64Str** (OSCTXT \*pctx, OSDynOctStr \*pvalue)
- This function decodes a contents of a Base64-encode binary string.*

  - int **rtXmlDecDynBase64Str64** (OSCTXT \*pctx, OSDynOctStr64 \*pvalue)
- This function is identical to rtXmlDecDynBase64Str except that it supports a 64-bit integer length on 64-bit systems.*

  - int **rtXmlDecDynHexStr** (OSCTXT \*pctx, OSDynOctStr \*pvalue)
- This function decodes a contents of a hexBinary string.*

  - int **rtXmlDecDynHexStr64** (OSCTXT \*pctx, OSDynOctStr64 \*pvalue)
- This function is identical to rtXmlDecDynHexStr except that it supports a 64-bit integer length on 64-bit systems.*

  - int **rtXmlDecEmptyElement** (OSCTXT \*pctx)
- This function is used to enforce a requirement that an element be empty.*

  - int **rtXmlDecUTF8Str** (OSCTXT \*pctx, OSUTF8CHAR \*outdata, OSSIZE max\_len)
- This function decodes the contents of a UTF-8 string data type.*

  - int **rtXmlDecDynUTF8Str** (OSCTXT \*pctx, const OSUTF8CHAR \*\*outdata)
- This function decodes the contents of a UTF-8 string data type.*

  - 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.*

  - int **rtXmlDecHexStr** (OSCTXT \*pctx, OSOCTET \*pvalue, OSUINT32 \*pnocts, OSINT32 bufsize)
- This function decodes the contents of a hexBinary string into a static memory structure.*

  - int **rtXmlDecHexStr64** (OSCTXT \*pctx, OSOCTET \*pvalue, OSSIZE \*pnocts, OSSIZE bufsize)
- This function is identical to rtXmlDecHexStr except that it supports a 64-bit integer length on 64-bit systems.*

  - int **rtXmlDecGYear** (OSCTXT \*pctx, OSXSDDateTime \*pvalue)
- This function decodes a variable of the XSD 'gYear' type.*

  - int **rtXmlDecGYearMonth** (OSCTXT \*pctx, OSXSDDateTime \*pvalue)
- This function decodes a variable of the XSD 'gYearMonth' type.*

  - int **rtXmlDecGMonth** (OSCTXT \*pctx, OSXSDDateTime \*pvalue)
- This function decodes a variable of the XSD 'gMonth' type.*

  - int **rtXmlDecGMonthDay** (OSCTXT \*pctx, OSXSDDateTime \*pvalue)
- This function decodes a variable of the XSD 'gMonthDay' type.*

  - int **rtXmlDecGDay** (OSCTXT \*pctx, OSXSDDateTime \*pvalue)
- This function decodes a variable of the XSD 'gDay' type.*

  - int **rtXmlDeclnt** (OSCTXT \*pctx, OSINT32 \*pvalue)
- This function decodes the contents of a 32-bit integer data type.*

  - int **rtXmlDeclnt8** (OSCTXT \*pctx, OSINT8 \*pvalue)
- This function decodes the contents of an 8-bit integer data type (i.e.*

  - int **rtXmlDeclnt16** (OSCTXT \*pctx, OSINT16 \*pvalue)
- This function decodes the contents of a 16-bit integer data type.*

  - int **rtXmlDeclnt64** (OSCTXT \*pctx, OSINT64 \*pvalue)
- This function decodes the contents of a 64-bit integer data type.*

  - int **rtXmlDecUInt** (OSCTXT \*pctx, OSUINT32 \*pvalue)
- This function decodes the contents of an unsigned 32-bit integer data type.*

  - int **rtXmlDecUInt8** (OSCTXT \*pctx, OSUINT8 \*pvalue)
- This function decodes the contents of an unsigned 8-bit integer data type (i.e.*

  - int **rtXmlDecUInt16** (OSCTXT \*pctx, OSUINT16 \*pvalue)
- This function decodes the contents of an unsigned 16-bit integer data type.*

- int [rtXmlDecUInt64](#) (OSCTXT \*pctxt, OSUINT64 \*pvalue)  
*This function decodes the contents of an unsigned 64-bit integer data type.*
- 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).*
- const OSUTF8CHAR \* [rtXmlDecQName](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*qname, const OSUTF8CHAR \*\*prefix)  
*This function decodes an XML qualified name string (QName) type.*
- int [rtXmlDecXSIAttr](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*attrName, const OSUTF8CHAR \*attrValue)  
*This function decodes XML schema instance (XSI) attribute.*
- int [rtXmlDecXSIAttrs](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*const \*attrs, const char \*typeName)  
*This function decodes XML schema instance (XSI) attributes.*
- int [rtXmlDecXmlStr](#) (OSCTXT \*pctxt, OSXMLSTRING \*outdata)  
*This function decodes the contents of an XML string data type.*
- int [rtXmlParseElementName](#) (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppName)  
*This function parses the initial tag from an XML message.*
- int [rtXmlParseElemQName](#) (OSCTXT \*pctxt, OSXMLQName \*pQName)  
*This function parses the initial tag from an XML message.*
- int [rtXmlEncAny](#) (OSCTXT \*pctxt, OSXMLSTRING \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD any type.*
- int [rtXmlEncAnyTypeValue](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*pvalue)  
*This function encodes a variable of the XSD anyType type.*
- 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.*
- int [rtXmlEncBase64Binary](#) (OSCTXT \*pctxt, OSSIZE nocts, const OSOCTET \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD base64Binary type.*
- 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.*
- int [rtXmlEncBase64StrValue](#) (OSCTXT \*pctxt, OSSIZE nocts, const OSOCTET \*value)  
*This function encodes a variable of the XSD base64Binary type.*
- int [rtXmlEncBigInt](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD integer type.*
- int [rtXmlEncBigIntAttr](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)  
*This function encodes an XSD integer attribute value.*
- int [rtXmlEncBigIntValue](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*value)  
*This function encodes an XSD integer attribute value.*
- int [rtXmlEncBitString](#) (OSCTXT \*pctxt, OSSIZE nbits, const OSOCTET \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the ASN.1 BIT STRING type.*
- int [rtXmlEncBitStringExt](#) (OSCTXT \*pctxt, OSSIZE nbits, const OSOCTET \*value, OSSIZE dataSize, const OSOCTET \*extValue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the ASN.1 BIT STRING type.*
- int [rtXmlEncBinStrValue](#) (OSCTXT \*pctxt, OSSIZE nbits, const OSOCTET \*data)

- This function encodes a binary string value as a sequence of '1's and '0's.*
- int **rtXmlEncBool** (OSCTXT \*pctxt, OSBOOL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)
 

*This function encodes a variable of the XSD boolean type.*
  - int **rtXmlEncBoolValue** (OSCTXT \*pctxt, OSBOOL value)
 

*This function encodes a variable of the XSD boolean type.*
  - int **rtXmlEncBoolAttr** (OSCTXT \*pctxt, OSBOOL value, const OSUTF8CHAR \*attrName, OSSIZE attrNameLen)
 

*This function encodes an XSD boolean attribute value.*
  - int **rtXmlEncCanonicalSort** (OSCTXT \*pctxt, OSCTXT \*pBufCtxt, OSRTSList \*pList)
 

*Sort (as for CXER, Canonical-XER) and encode previously encoded SET OF components.*
  - int **rtXmlEncComment** (OSCTXT \*pctxt, const OSUTF8CHAR \*comment)
 

*This function encodes an XML comment.*
  - 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.*
  - int **rtXmlEncDateValue** (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)
 

*This function encodes a variable of the XSD 'date' type as a string.*
  - 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.*
  - int **rtXmlEncTimeValue** (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)
 

*This function encodes a variable of the XSD 'time' type as a string.*
  - 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.*
  - int **rtXmlEncDateTimeValue** (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)
 

*This function encodes a numeric date/time value into an XML string representation.*
  - int **rtXmlEncDecimal** (OSCTXT \*pctxt, OSREAL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSDecimalFmt \*pFmtSpec)
 

*This function encodes a variable of the XSD decimal type.*
  - 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.*
  - int **rtXmlEncDecimalValue** (OSCTXT \*pctxt, OSREAL value, const OSDecimalFmt \*pFmtSpec, char \*pDestBuf, OSSIZE destBufSize)
 

*This function encodes a value of the XSD decimal type.*
  - int **rtXmlEncDouble** (OSCTXT \*pctxt, OSREAL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSDoubleFmt \*pFmtSpec)
 

*This function encodes a variable of the XSD double type.*
  - 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.*
  - 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.*
  - int **rtXmlEncDoubleValue** (OSCTXT \*pctxt, OSREAL value, const OSDoubleFmt \*pFmtSpec, int defaultPrecision)
 

*This function encodes a value of the XSD double or float type.*
  - int **rtXmlEncEmptyElement** (OSCTXT \*pctxt, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs, OSBOOL terminate)



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

  - int **rtXmlEncEndDocument** (OSCTXT \*pctxt)

*This function adds trailer information and a null terminator at the end of the XML document being encoded.*
- int **rtXmlEncEndElement** (OSCTXT \*pctxt, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

*This function encodes an end element tag value (</elemName>).*
- int **rtXmlEncEndSoapEnv** (OSCTXT \*pctxt)

*This function encodes a SOAP envelope end element tag (<SOAP-ENV:Envelope/>).*
- int **rtXmlEncEndSoapElems** (OSCTXT \*pctxt, OSXMLSOAPMsgType msgtype)

*This function encodes SOAP end element tags.*
- int **rtXmlEncFloat** (OSCTXT \*pctxt, OSREAL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, const OSDoubleFmt \*pFmtSpec)

*This function encodes a variable of the XSD float type.*
- 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.*
- int **rtXmlEncGYear** (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

*This function encodes a numeric gYear element into an XML string representation.*
- int **rtXmlEncGYearMonth** (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

*This function encodes a numeric gYearMonth element into an XML string representation.*
- int **rtXmlEncGMonth** (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

*This function encodes a numeric gMonth element into an XML string representation.*
- int **rtXmlEncGMonthDay** (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

*This function encodes a numeric gMonthDay element into an XML string representation.*
- int **rtXmlEncGDay** (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

*This function encodes a numeric gDay element into an XML string representation.*
- int **rtXmlEncGYearValue** (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)

*This function encodes a numeric gYear value into an XML string representation.*
- int **rtXmlEncGYearMonthValue** (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)

*This function encodes a numeric gYearMonth value into an XML string representation.*
- int **rtXmlEncGMonthValue** (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)

*This function encodes a numeric gMonth value into an XML string representation.*
- int **rtXmlEncGMonthDayValue** (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)

*This function encodes a numeric gMonthDay value into an XML string representation.*
- int **rtXmlEncGDayValue** (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)

*This function encodes a numeric gDay value into an XML string representation.*
- int **rtXmlEncHexBinary** (OSCTXT \*pctxt, OSSIZE nocts, const OSOCTET \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

*This function encodes a variable of the XSD hexBinary type.*
- 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.*
- int **rtXmlEncHexStrValue** (OSCTXT \*pctxt, OSSIZE nocts, const OSOCTET \*data)

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

- int **rtXmlEncIndent** (OSCTXT \*pctxt)
 

*This function adds indentation whitespace to the output stream.*
- int **rtXmlEncInt** (OSCTXT \*pctxt, OSINT32 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)
 

*This function encodes a variable of the XSD integer type.*
- int **rtXmlEncIntValue** (OSCTXT \*pctxt, OSINT32 value)
 

*This function encodes a variable of the XSD integer type.*
- 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").*
- 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.*
- int **rtXmlEncInt64** (OSCTXT \*pctxt, OSINT64 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)
 

*This function encodes a variable of the XSD integer type.*
- int **rtXmlEncInt64Value** (OSCTXT \*pctxt, OSINT64 value)
 

*This function encodes a variable of the XSD integer type.*
- 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").*
- int **rtXmlEncNamedBits** (OSCTXT \*pctxt, const OSBitMapItem \*pBitMap, OSSIZE nbits, const OSOCTET \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)
 

*This function encodes a variable of the ASN.1 BIT STRING type.*
- int **rtXmlEncNSAttrs** (OSCTXT \*pctxt, OSRTDList \*pNSAttrs)
 

*This function encodes namespace declaration attributes at the beginning of an XML document.*
- int **rtXmlPrintNSAttrs** (const char \*name, const OSRTDList \*data)
 

*This function prints a list of namespace attributes.*
- 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.*
- 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.*
- int **rtXmlEncStartDocument** (OSCTXT \*pctxt)
 

*This function encodes the XML header text at the beginning of an XML document.*
- int **rtXmlEncBOM** (OSCTXT \*pctxt)
 

*This function encodes the Unicode byte order mark header at the start of the document.*
- int **rtXmlEncStartElement** (OSCTXT \*pctxt, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs, OSBOOL terminate)
 

*This function encodes a start element tag value (<elemName>).*
- int **rtXmlEncStartSoapEnv** (OSCTXT \*pctxt, OSRTDList \*pNSAttrs)
 

*This function encodes a SOAP envelope start element tag.*
- int **rtXmlEncStartSoapElems** (OSCTXT \*pctxt, OSXMLSOAPMsgType msgtype)
 

*This function encodes a SOAP envelope start element tag and an optional SOAP body or fault tag.*
- int **rtXmlEncString** (OSCTXT \*pctxt, OSXMLSTRING \*pxmlstr, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)
 

*This function encodes a variable of the XSD string type.*
- int **rtXmlEncStringValue** (OSCTXT \*pctxt, const OSUTF8CHAR \*value)
 

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

- int **rtXmlEncStringValue2** (OSCTXT \*pctx, const OSUTF8CHAR \*value, OSSIZE valueLen)  
*This function encodes a variable of the XSD string type.*
- int **rtXmlEncTermStartElement** (OSCTXT \*pctx)  
*This function terminates a currently open XML start element by adding either a '>' or '/>' (if empty) terminator.*
- int **rtXmlEncUnicodeStr** (OSCTXT \*pctx, const OSUNICHAR \*value, OSSIZE nchars, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a Unicode string value.*
- int **rtXmlEncUTF8Attr** (OSCTXT \*pctx, 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.*
- int **rtXmlEncUTF8Attr2** (OSCTXT \*pctx, 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.*
- int **rtXmlEncUTF8Str** (OSCTXT \*pctx, const OSUTF8CHAR \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a UTF-8 string value.*
- int **rtXmlEncUInt** (OSCTXT \*pctx, OSUINT32 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD unsigned integer type.*
- int **rtXmlEncUIntValue** (OSCTXT \*pctx, OSUINT32 value)  
*This function encodes a variable of the XSD unsigned integer type.*
- 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").*
- int **rtXmlEncUInt64** (OSCTXT \*pctx, OSUINT64 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD integer type.*
- int **rtXmlEncUInt64Value** (OSCTXT \*pctx, OSUINT64 value)  
*This function encodes a variable of the XSD integer type.*
- 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").*
- int **rtXmlEncXSIAttrs** (OSCTXT \*pctx, OSBOOL needXSI)  
*This function encodes XML schema instance (XSI) attributes at the beginning of an XML document.*
- int **rtXmlEncXSITypeAttr** (OSCTXT \*pctx, const OSUTF8CHAR \*value)  
*This function encodes an XML schema instance (XSI) type attribute value (xsi:type="value").*
- 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").*
- int **rtXmlEncXSINilAttr** (OSCTXT \*pctx)  
*This function encodes an XML nil attribute (xsi:nil="true").*
- 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'.*
- int **rtXmlSetEncBufPtr** (OSCTXT \*pctx, OSOCTET \*bufaddr, OSSIZE bufsiz)  
*This function is used to set the internal buffer within the run-time library encoding context.*
- int **rtXmlGetIndent** (OSCTXT \*pctx)  
*This function returns current XML output indent value.*
- OSBOOL **rtXmlGetWriteBOM** (OSCTXT \*pctx)  
*This function returns whether the Unicode byte order mark will be encoded.*
- int **rtXmlGetIndentChar** (OSCTXT \*pctx)

- This function returns current XML output indent character value (default is space).*
- 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.*
  - int **rtXmlSetEncC14N** (OSCTXT \*pctxt, OSBOOL value)
 

*This function sets the option to encode in C14N mode.*
  - int **rtXmlSetEncXSI\_NAMESPACE** (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.*
  - int **rtXmlSetEncXSIAttr** (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.*
  - int **rtXmlSetEncDocHdr** (OSCTXT \*pctxt, OSBOOL value)
 

*This function sets the option to add the XML document header (i.e.*
  - int **rtXmlSetEncodingStr** (OSCTXT \*pctxt, const OSUTF8CHAR \*encodingStr)
 

*This function sets the XML output encoding to the given value.*
  - int **rtXmlSetFormatting** (OSCTXT \*pctxt, OSBOOL doFormatting)
 

*This function sets XML output formatting to the given value.*
  - int **rtXmlSetIndent** (OSCTXT \*pctxt, OSUINT8 indent)
 

*This function sets XML output indent to the given value.*
  - int **rtXmlSetIndentChar** (OSCTXT \*pctxt, char indentChar)
 

*This function sets XML output indent character to the given value.*
  - void **rtXmlSetNamespacesSet** (OSCTXT \*pctxt, OSBOOL value)
 

*This function sets the context 'namespaces are set' flag.*
  - int **rtXmlSetNSPrefixLinks** (OSCTXT \*pctxt, OSRTDList \*pNSAttrs)
 

*This function sets namespace prefix/URI links in the namespace prefix stack in the context structure.*
  - 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.*
  - 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.*
  - void **rtXmlSetSoapVersion** (OSCTXT \*pctxt, OSUINT8 version)
 

*This function sets the SOAP version number.*
  - int **rtXmlSetXSITypeAttr** (OSCTXT \*pctxt, const OSUTF8CHAR \*xsiType)
 

*This function sets the XML Schema Instance (xsi) type attribute value.*
  - int **rtXmlSetWriteBOM** (OSCTXT \*pctxt, OSBOOL write)
 

*This function sets whether the Unicode byte order mark is encoded.*
  - int **rtXmlMatchHexStr** (OSCTXT \*pctxt, OSSIZE minLength, OSSIZE maxLength)
 

*This function tests the context buffer for containing a correct hexadecimal string.*
  - int **rtXmlMatchBase64Str** (OSCTXT \*pctxt, OSSIZE minLength, OSSIZE maxLength)
 

*This function tests the context buffer for containing a correct base64 string.*
  - int **rtXmlMatchDate** (OSCTXT \*pctxt)
 

*This function tests the context buffer for containing a correct date string.*
  - int **rtXmlMatchTime** (OSCTXT \*pctxt)
 

*This function tests the context buffer for containing a correct time string.*
  - int **rtXmlMatchDateTime** (OSCTXT \*pctxt)
 

*This function tests the context buffer for containing a correct dateTime string.*
  - int **rtXmlMatchGYear** (OSCTXT \*pctxt)

- This function tests the context buffer for containing a correct gYear string.*

  - int `rtXmlMatchGYearMonth` (OSCTXT \*pctxt)
- This function tests the context buffer for containing a correct gYearMonth string.*

  - int `rtXmlMatchGMonth` (OSCTXT \*pctxt)
- This function tests the context buffer for containing a correct gMonth string.*

  - int `rtXmlMatchGMonthDay` (OSCTXT \*pctxt)
- This function tests the context buffer for containing a correct gMonthDay string.*

  - int `rtXmlMatchGDay` (OSCTXT \*pctxt)
- This function tests the context buffer for containing a correct gDay string.*

  - OSUTF8CHAR \* `rtXmlNewQName` (OSCTXT \*pctxt, const OSUTF8CHAR \*localName, const OSUTF8CHAR \*prefix)
- This function creates a new QName given the localName and prefix parts.*

  - OSBOOL `rtXmlCmpBase64Str` (OSUINT32 nocts1, const OSOCTET \*data1, const OSUTF8CHAR \*data2)
- This function compares an array of octets to a base64 string.*

  - OSBOOL `rtXmlCmpHexStr` (OSUINT32 nocts1, const OSOCTET \*data1, const OSUTF8CHAR \*data2)
- This function compares an array of octets to a hex string.*

  - 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.*

  - 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.*

  - 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.*

  - OSBOOL `rtSaxHasXMLNSAttrs` (const OSUTF8CHAR \*const \*attrs)
- This function checks if the given attribute list contains one or more XML namespace attributes (xmlns).*

  - OSBOOL `rtSaxIsEmptyBuffer` (OSCTXT \*pctxt)
- This function checks if the buffer in the context is empty or not.*

  - int `rtSaxStrListParse` (OSCTXT \*pctxt, OSRTMEMBUF \*pMemBuf, OSRTDList \*pvalue)
- This function parses the list of strings.*

  - int `rtSaxSortAttrs` (OSCTXT \*pctxt, const OSUTF8CHAR \*const \*attrs, OSUINT16 \*\*order)
- This function sorts a SAX attribute list in ascending order based on attribute name.*

  - int `rtSaxStrListMatch` (OSCTXT \*pctxt)
- This function matches the list of strings.*

  - int `rtXmlWriteToFile` (OSCTXT \*pctxt, const char \*filename)
- This function writes the encoded XML message stored in the context message buffer out to a file.*

  - 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).*

  - int `rtXmlpDecAny2` (OSCTXT \*pctxt, OSUTF8CHAR \*\*pvalue)
- This version of the rtXmlpDecAny function returns the string in a mutable buffer.*

  - int `rtXmlpDecAnyAttrStr` (OSCTXT \*pctxt, const OSUTF8CHAR \*\*ppAttrStr, OSSIZE attrIndex)
- This function decodes an any attribute string.*

  - 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).*

  - 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.*

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

- This function is identical to `rtXmlpDecBase64Str` except that it supports 64-bit integer lengths on 64-bit systems.*
- int `rtXmlpDecBigInt` (OSCTXT \*pctxt, const OSUTF8CHAR \*\*pvalue)
 

*This function will decode a variable of the XSD integer type.*
  - int `rtXmlpDecBitString` (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnbits, OSUINT32 bufsize)
 

*This function decodes a bit string value.*
  - int `rtXmlpDecBitString64` (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnbits, OSSIZE bufsize)
 

*This function is identical to `rtXmlpDecBitString` except that it supports lengths up to 64-bits in size on 64-bit machines.*
  - int `rtXmlpDecBitStringExt` (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnbits, OSOCTET \*\*ppextdata, OSUINT32 bufsize)
 

*This function decodes a bit string value.*
  - int `rtXmlpDecBitStringExt64` (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnbits, OSOCTET \*\*ppextdata, OSSIZE bufsize)
 

*This function is identical to `rtXmlpDecBitStringExt` except that it supports lengths up to 64-bits in size on 64-bit machines.*
  - int `rtXmlpDecBool` (OSCTXT \*pctxt, OSBOOL \*pvalue)
 

*This function decodes a variable of the boolean type.*
  - int `rtXmlpDecDate` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
 

*This function decodes a variable of the XSD 'date' type.*
  - int `rtXmlpDecDateTime` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
 

*This function decodes a variable of the XSD 'dateTime' type.*
  - int `rtXmlpDecDecimal` (OSCTXT \*pctxt, OSREAL \*pvalue, int totalDigits, int fractionDigits)
 

*This function decodes the contents of a decimal data type.*
  - int `rtXmlpDecDouble` (OSCTXT \*pctxt, OSREAL \*pvalue)
 

*This function decodes the contents of a float or double data type.*
  - int `rtXmlpDecDoubleExt` (OSCTXT \*pctxt, OSUINT8 flags, OSREAL \*pvalue)
 

*This function decodes the contents of a float or double data type.*
  - int `rtXmlpDecDynBase64Str` (OSCTXT \*pctxt, OSDynOctStr \*pvalue)
 

*This function decodes a contents of a Base64-encode binary string.*
  - int `rtXmlpDecDynBase64Str64` (OSCTXT \*pctxt, OSDynOctStr64 \*pvalue)
 

*This function is identical to `rtXmlpDecDynBase64Str` except that it supports 64-bit integer lengths on 64-bit systems.*
  - int `rtXmlpDecDynBitString` (OSCTXT \*pctxt, OSDynOctStr \*pvalue)
 

*This function decodes a bit string value.*
  - int `rtXmlpDecDynHexStr` (OSCTXT \*pctxt, OSDynOctStr \*pvalue)
 

*This function decodes a contents of a hexBinary string.*
  - int `rtXmlpDecDynHexStr64` (OSCTXT \*pctxt, OSDynOctStr64 \*pvalue)
 

*This function is identical to the `rtXmlpDecDynHexStr` except that it supports lengths up to 64 bits in size on 64-bit systems.*
  - int `rtXmlpDecDynUnicodeStr` (OSCTXT \*pctxt, const OSUNICHAR \*\*ppdata, OSSIZE \*pnchars)
 

*This function decodes a Unicode string data type.*
  - int `rtXmlpDecDynUTF8Str` (OSCTXT \*pctxt, const OSUTF8CHAR \*\*outdata)
 

*This function decodes the contents of a UTF-8 string data type.*
  - int `rtXmlpDecUTF8Str` (OSCTXT \*pctxt, OSUTF8CHAR \*out, OSSIZE max\_len)
 

*This function decodes the contents of a UTF-8 string data type.*
  - int `rtXmlpDecGDay` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
 

*This function decodes a variable of the XSD 'gDay' type.*
  - int `rtXmlpDecGMonth` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
 

*This function decodes a variable of the XSD 'gMonth' type.*
  - int `rtXmlpDecGMonthDay` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
 

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



- int `rtXmlpDecGYear` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gYear' type.*
- int `rtXmlpDecGYearMonth` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gYearMonth' type.*
- int `rtXmlpDecHexStr` (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocets, OSSIZE bufsize)  
*This function decodes the contents of a hexBinary string into a static memory structure.*
- int `rtXmlpDecHexStr64` (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocets, OSSIZE bufsize)  
*This function is identical to `rtXmlpDecHexStr` except that it supports lengths up to 64-bits in size on 64-bit machines.*
- int `rtXmlpDecInt` (OSCTXT \*pctxt, OSINT32 \*pvalue)  
*This function decodes the contents of a 32-bit integer data type.*
- int `rtXmlpDecInt8` (OSCTXT \*pctxt, OSINT8 \*pvalue)  
*This function decodes the contents of an 8-bit integer data type (i.e.*
- int `rtXmlpDecInt16` (OSCTXT \*pctxt, OSINT16 \*pvalue)  
*This function decodes the contents of a 16-bit integer data type.*
- int `rtXmlpDecInt64` (OSCTXT \*pctxt, OSINT64 \*pvalue)  
*This function decodes the contents of a 64-bit integer data type.*
- int `rtXmlpDecNamedBits` (OSCTXT \*pctxt, const OSBitMapItem \*pBitMap, OSOCTET \*pvalue, OSUINT32 \*pnbits, OSUINT32 bufsize)  
*This function decodes the contents of a named bit field.*
- int `rtXmlpDecNamedBits64` (OSCTXT \*pctxt, const OSBitMapItem \*pBitMap, OSOCTET \*pvalue, OSSIZE \*pnbits, OSSIZE bufsize)  
*This function decodes the contents of a named bit field.*
- 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.*
- int `rtXmlpDecTime` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'time' type.*
- int `rtXmlpDecUInt` (OSCTXT \*pctxt, OSUINT32 \*pvalue)  
*This function decodes the contents of an unsigned 32-bit integer data type.*
- int `rtXmlpDecUInt8` (OSCTXT \*pctxt, OSOCTET \*pvalue)  
*This function decodes the contents of an unsigned 8-bit integer data type (i.e.*
- int `rtXmlpDecUInt16` (OSCTXT \*pctxt, OSUINT16 \*pvalue)  
*This function decodes the contents of an unsigned 16-bit integer data type.*
- int `rtXmlpDecUInt64` (OSCTXT \*pctxt, OSUINT64 \*pvalue)  
*This function decodes the contents of an unsigned 64-bit integer data type.*
- int `rtXmlpDecXmlStr` (OSCTXT \*pctxt, OSXMLSTRING \*outdata)  
*This function decodes the contents of an XML string data type.*
- 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.*
- 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.*
- 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).*
- int `rtXmlpGetAttributeID` (const OSXMLStrFragment \*attrName, OSINT16 nsidx, OSSIZE nAttr, const OSXMLAttrDescr attrNames[], OSUINT32 attrPresent[])  
*This function finds an attribute in the descriptor table.*
- int `rtXmlpGetNextElem` (OSCTXT \*pctxt, OSXMLElemDescr \*pElem, OSINT32 level)

- This function parse the next element start tag.*
- 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.*
  - int [rtXmlpMarkLastEventActive](#) (OSCTXT \*pctx)
 

*This function marks current tag as unprocessed.*
  - int [rtXmlpMatchStartTag](#) (OSCTXT \*pctx, const OSUTF8CHAR \*elemLocalName, OSINT16 nsidx)
 

*This function parses the next start tag that matches with given name.*
  - int [rtXmlpMatchEndTag](#) (OSCTXT \*pctx, OSINT32 level)
 

*This function parse next end tag that matches with given name.*
  - OSBOOL [rtXmlpHasAttributes](#) (OSCTXT \*pctx)
 

*This function checks accessibility of attributes.*
  - int [rtXmlpGetAttributeCount](#) (OSCTXT \*pctx)
 

*This function returns number of attributes in last processed start tag.*
  - int [rtXmlpSelectAttribute](#) (OSCTXT \*pctx, OSXMLNameFragments \*pAttr, OSINT16 \*nsidx, OSSIZE attrIndex)
 

*This function selects attribute to decode.*
  - OSINT32 [rtXmlpGetCurrentLevel](#) (OSCTXT \*pctx)
 

*This function returns current nesting level.*
  - void [rtXmlpSetWhiteSpaceMode](#) (OSCTXT \*pctx, OSXMLWhiteSpaceMode whiteSpaceMode)
 

*Sets the whitespace treatment mode.*
  - OSBOOL [rtXmlpSetMixedContentMode](#) (OSCTXT \*pctx, OSBOOL mixedContentMode)
 

*Sets mixed content mode.*
  - void [rtXmlpSetListMode](#) (OSCTXT \*pctx)
 

*Sets list mode.*
  - OSBOOL [rtXmlpListHasItem](#) (OSCTXT \*pctx)
 

*Check for end of decoded token list.*
  - void [rtXmlpCountListItems](#) (OSCTXT \*pctx, OSSIZE \*itemCnt)
 

*Count tokens in list.*
  - int [rtXmlpGetNextSeqElemID2](#) (OSCTXT \*pctx, const OSXMLElemIDRec \*tab, const OSXMLGroupDesc \*pGroup, int groups, int curlID, int lastMandatoryID, OSBOOL groupMode, OSBOOL checkRepeat)
 

*This function parses the next start tag and finds index of element name in descriptor table.*
  - int [rtXmlpGetNextSeqElemID](#) (OSCTXT \*pctx, const OSXMLElemIDRec \*tab, const OSXMLGroupDesc \*pGroup, int curlID, int lastMandatoryID, OSBOOL groupMode)
 

*This function parses the next start tag and finds index of element name in descriptor table.*
  - int [rtXmlpGetNextSeqElemIDExt](#) (OSCTXT \*pctx, const OSXMLElemIDRec \*tab, const OSXMLGroupDesc \*ppGroup, const OSBOOL \*extRequired, int postExtRootID, int curlID, int lastMandatoryID, OSBOOL groupMode)
 

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



- void [rtXmlpSetNamespaceTable](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*namespaceTable[], OSSIZE nm←  
Namespaces)  
*Sets user namespace table.*
- int [rtXmlpCreateReader](#) (OSCTXT \*pctxt)  
*Creates pull parser reader structure within the context.*
- void [rtXmlpHideAttributes](#) (OSCTXT \*pctxt)  
*Disable access to attributes.*
- OSBOOL [rtXmlpNeedDecodeAttributes](#) (OSCTXT \*pctxt)  
*This function checks if attributes were previously decoded.*
- void [rtXmlpMarkPos](#) (OSCTXT \*pctxt)  
*Save current decode position.*
- void [rtXmlpRewindToMarkedPos](#) (OSCTXT \*pctxt)  
*Rewind to saved decode position.*
- void [rtXmlpResetMarkedPos](#) (OSCTXT \*pctxt)  
*Reset saved decode position.*
- int [rtXmlpGetXSITypeAttr](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*ppAttrValue, OSINT16 \*nsidx, OSSIZE \*p←  
LocalOffs)  
*This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).*
- int [rtXmlpGetXmlnsAttrs](#) (OSCTXT \*pctxt, OSRTDList \*pNSAttrs)  
*This function decodes namespace attributes from start tag and adds them to the given list.*
- int [rtXmlpDecXSIAttrs](#) (OSCTXT \*pctxt)  
*This function decodes XSI (XML Schema Instance) that may be present in any arbitrary XML element within a document.*
- OSBOOL [rtXmlplsEmptyElement](#) (OSCTXT \*pctxt)  
*Check element content: empty or not.*
- int [rtXmlEncAttrC14N](#) (OSCTXT \*pctxt)  
*This function used only in C14 mode.*
- struct OSXMLReader \* [rtXmlpGetReader](#) (OSCTXT \*pctxt)  
*This function fetches the XML reader structure from the context for use in low-level pull parser calls.*
- OSBOOL [rtXmlplsLastEventDone](#) (OSCTXT \*pctxt)  
*Check processing status of current tag.*
- 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.*
- 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.*
- void [rtXmlpForceDecodeAsGroup](#) (OSCTXT \*pctxt)  
*Disable skipping of unknown elements in optional sequence tail.*
- OSBOOL [rtXmlplsDecodeAsGroup](#) (OSCTXT \*pctxt)  
*This function checks if "decode as group" mode was forced.*
- OSBOOL [rtXmlplsUTF8Encoding](#) (OSCTXT \*pctxt)  
*This function checks if the encoding specified in XML header is UTF-8.*
- int [rtXmlpReadBytes](#) (OSCTXT \*pctxt, OSOCTET \*pbuf, OSSIZE nbytes)  
*This function reads the specified number of bytes directly from the underlying XML parser stream.*

## 8.2.1 Detailed Description

XML low-level C encode/decode functions.

## 8.2.2 Typedef Documentation

### 8.2.2.1 OSXMLGroupDesc

```
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.

## 8.2.3 Function Documentation

### 8.2.3.1 rtSaxGetAttrValue()

```
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

<i>attrName</i>	Name of the attribute to find.
<i>attrs</i>	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.

### 8.2.3.2 rtSaxGetElemID()

```
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.

It ensures elements are received in the correct order and also sets the required element count variable.

#### Parameters

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

#### 8.2.3.3 rtSaxGetElemID8()

```

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

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

#### 8.2.3.4 rtSaxHasXMLNSAttrs()

```
OSBOOL rtSaxHasXMLNSAttrs (
    const OSUTF8CHAR *const * attrs )
```

This function checks if the given attribute list contains one or more XML namespace attributes (xmlns).

##### Parameters

<i>attrs</i>	Attribute list in form passed by parser into SAX startElement function.
--------------	---

##### Returns

TRUE, if xmlns attribute found in list.

#### 8.2.3.5 rtSaxIsEmptyBuffer()

```
OSBOOL rtSaxIsEmptyBuffer (
    OSCTXT * pctxt )
```

This function checks if the buffer in the context is empty or not.

##### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
--------------	-----------------------------

##### Returns

TRUE, if the buffer contains empty string.

#### 8.2.3.6 rtSaxSortAttrs()

```
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

<i>pctxt</i>	Pointer to OSCTXT structure.
<i>attrs</i>	Standard SAX attribute list. Entry <i>i</i> is attribute name and <i>i+1</i> is value. List is terminated by a null name.
<i>order</i>	Order array containing the order of sorted attributes. This array is allocated using <code>rtxMemAlloc</code> , it can be freed using <code>rtxMemFreePtr</code> 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.

#### 8.2.3.7 `rtSaxStrListMatch()`

```
int rtSaxStrListMatch (  
    OSCTXT * pctxt )
```

This function matches the list of strings.

It is used for matching NMTOKENS, IDREFS, NMENTITIES.

### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
--------------	-----------------------------

### Returns

0 - if success, negative value is error.

#### 8.2.3.8 `rtSaxStrListParse()`

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

This function parses the list of strings.

It is used for parsing NMTOKENS, IDREFS, NMENTITIES.

### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure. Can be NULL, if pMemBuf is not NULL.
<i>pMemBuf</i>	Pointer to memory buffer structure. Can be NULL, if pctxt is not NULL.
<i>pvalue</i>	Doubly-linked list for parsed strings.

### Returns

0 - if success, negative value is error.

### 8.2.3.9 rtXmlCmpBase64Str()

```
OSBOOL rtXmlCmpBase64Str (
    OSUINT32 nocts1,
    const OSOCTET * data1,
    const OSUTF8CHAR * data2 )
```

This function compares an array of octets to a base64 string.

### Parameters

<i>nocts1</i>	Number of octets in data1.
<i>data1</i>	Pointer to array of OSOCTET.
<i>data2</i>	Pointer to null-terminated array of OSUTF8CHAR.

### Returns

TRUE if data2 is a base64 string representation of data1, false otherwise.

### 8.2.3.10 rtXmlCmpHexStr()

```
OSBOOL rtXmlCmpHexStr (
    OSUINT32 nocts1,
    const OSOCTET * data1,
    const OSUTF8CHAR * data2 )
```

This function compares an array of octets to a hex string.

### Parameters

<i>nocts1</i>	Number of octets in data1.
<i>data1</i>	Pointer to array of OSOCTET.
<i>data2</i>	Pointer to null-terminated array of OSUTF8CHAR.

## Returns

TRUE if data2 is a hex string representation of data1 , false otherwise.

### 8.2.3.11 rtXmlCreateFileInputSource()

```
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

<i>pctxt</i>	Pointer to context block structure.
<i>filepath</i>	Full pathname of XML document file to open.

## Returns

Completion status of operation:

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

### 8.2.3.12 rtXmlInitContext()

```
int rtXmlInitContext (
    OSCTXT * pctxt )
```

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

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
--------------	-----------------------------

### 8.2.3.13 rtXmlInitContextUsingKey()

```
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

<i>pctxt</i>	The pointer to the context structure variable to be initialized.
<i>key</i>	Key data generated by ASN1C compiler.
<i>keylen</i>	Key data field length.

#### Returns

Completion status of operation:

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

#### 8.2.3.14 rtXmlInitCtxAppInfo()

```

int rtXmlInitCtxAppInfo (
    OSCTXT * pctxt )

```

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

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
--------------	-----------------------------

#### 8.2.3.15 rtXmlMatchBase64Str()

```

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

<i>pctxt</i>	Pointer to OSCTXT structure
<i>minLength</i>	A minimal length of expected string.
<i>maxLength</i>	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.

### 8.2.3.16 rtXmlMatchDate()

```
int rtXmlMatchDate (
    OSCTXT * pctxt )
```

This function tests the context buffer for containing a correct date string.

It does not decode the value.

### Parameters

<i>pctxt</i>	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.

### 8.2.3.17 rtXmlMatchDateTime()

```
int rtXmlMatchDateTime (
    OSCTXT * pctxt )
```

This function tests the context buffer for containing a correct dateTime string.

It does not decode the value.

### Parameters

<i>pctxt</i>	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.

### 8.2.3.18 rtXmlMatchGDay()

```
int rtXmlMatchGDay (
    OSCTXT * pctxt )
```

This function tests the context buffer for containing a correct gDay string.

It does not decode the value.

#### Parameters

<i>pctxt</i>	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.

### 8.2.3.19 rtXmlMatchGMonth()

```
int rtXmlMatchGMonth (
    OSCTXT * pctxt )
```

This function tests the context buffer for containing a correct gMonth string.

It does not decode the value.

#### Parameters

<i>pctxt</i>	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.

### 8.2.3.20 rtXmlMatchGMonthDay()

```
int rtXmlMatchGMonthDay (
    OSCTXT * pctx )
```

This function tests the context buffer for containing a correct gMonthDay string.

It does not decode the value.

#### Parameters

<i>pctx</i>	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.

### 8.2.3.21 rtXmlMatchGYear()

```
int rtXmlMatchGYear (
    OSCTXT * pctx )
```

This function tests the context buffer for containing a correct gYear string.

It does not decode the value.

#### Parameters

<i>pctx</i>	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.

### 8.2.3.22 rtXmlMatchGYearMonth()

```
int rtXmlMatchGYearMonth (
    OSCTXT * pctx )
```

This function tests the context buffer for containing a correct gYearMonth string.

It does not decode the value.

## Parameters

<i>pctxt</i>	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.

### 8.2.3.23 rtXmlMatchHexStr()

```
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

<i>pctxt</i>	Pointer to OSCTXT structure
<i>minLength</i>	A minimal length of expected string.
<i>maxLength</i>	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.

### 8.2.3.24 rtXmlMatchTime()

```
int rtXmlMatchTime (
    OSCTXT * pctxt )
```

This function tests the context buffer for containing a correct time string.

It does not decode the value.

## Parameters

<i>pctxt</i>	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.

### 8.2.3.25 rtXmlMemFreeAnyAttrs()

```
void rtXmlMemFreeAnyAttrs (
    OSCTXT * pctxt,
    OSRIDLlist * pAnyAttrList )
```

This function frees a list of anyAttribute that is a member of OSXSDAnyType structure.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pAnyAttrList</i>	Pointer to list of anyAttribute that is to be freed.

### 8.2.3.26 rtXmlNewQName()

```
OSUTF8CHAR* rtXmlNewQName (
    OSCTXT * pctxt,
    const OSUTF8CHAR * localName,
    const OSUTF8CHAR * prefix )
```

This function creates a new QName given the localName and prefix parts.

## Parameters

<i>pctxt</i>	Pointer to a context structure.
<i>localName</i>	Element local name.
<i>prefix</i>	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.

### 8.2.3.27 rtXmlPrepareContext()

```
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

<i>pctxt</i>	Pointer to OSCTXT structure
--------------	-----------------------------

#### Returns

0 if OK, negative status code if error.

### 8.2.3.28 rtXmlSetEncC14N()

```
int rtXmlSetEncC14N (
    OSCTXT * pctxt,
    OSBOOL value )
```

This function sets the option to encode in C14N mode.

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>value</i>	Boolean value: true = C14N mode enabled.

#### Returns

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

### 8.2.3.29 rtXmlSetEncDocHdr()

```
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

<i>pctxt</i>	Pointer to OSCTXT structure
<i>value</i>	Boolean value: true = add document header

#### Returns

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

#### 8.2.3.30 rtXmlSetEncodingStr()

```
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

<i>pctxt</i>	Pointer to OSCTXT structure
<i>encodingStr</i>	XML output encoding format

#### Returns

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

#### 8.2.3.31 rtXmlSetEncXSINamespace()

```
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

<i>pctxt</i>	Pointer to OSCTXT structure
<i>value</i>	Boolean value: true = encode XSI namespace attribute.

## Returns

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

### 8.2.3.32 rtXmlSetEncXSINilAttr()

```
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

<i>pctxt</i>	Pointer to OSCTXT structure
<i>value</i>	Boolean value: true = encode xsi:nil attribute.

## Returns

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

### 8.2.3.33 rtXmlSetFormatting()

```
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

<i>pctxt</i>	Pointer to OSCTXT structure
<i>doFormatting</i>	Boolean value indicating if formatting is to be done



## Returns

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

### 8.2.3.34 rtXmlSetIndent()

```
int rtXmlSetIndent (
    OSCTXT * pctxt,
    OSUINT8 indent )
```

This function sets XML output indent to the given value.

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>indent</i>	Number of spaces per indent. Default is 3.

## Returns

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

### 8.2.3.35 rtXmlSetIndentChar()

```
int rtXmlSetIndentChar (
    OSCTXT * pctxt,
    char indentChar )
```

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

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>indentChar</i>	Indent character. Default is space.

## Returns

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

### 8.2.3.36 rtXmlSetNamespacesSet()

```
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

<i>pctxt</i>	Pointer to OSCTXT structure.
<i>value</i>	Boolean value to which flag is to be set.

### 8.2.3.37 rtXmlSetNoNSSchemaLocation()

```
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

<i>pctxt</i>	Pointer to OSCTXT structure
<i>schemaLocation</i>	Schema location attribute value

#### Returns

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

### 8.2.3.38 rtXmlSetNSPrefixLinks()

```
int rtXmlSetNSPrefixLinks (
    OSCTXT * pctxt,
    OSRTDList * pNSAttrs )
```

This function sets namespace prefix/URI links in the namespace prefix stack in the context structure.

### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure.
<i>pNSAttrs</i>	List of namespace attributes.

### Returns

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

### 8.2.3.39 rtXmlSetSchemaLocation()

```
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

<i>pctxt</i>	Pointer to OSCTXT structure
<i>schemaLocation</i>	Schema location attribute value

### Returns

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

### 8.2.3.40 rtXmlSetSoapVersion()

```
void rtXmlSetSoapVersion (
    OSCTXT * pctxt,
    OSUINT8 version )
```

This function sets the SOAP version number.

### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>version</i>	SOAP version number as 2 digit integer (for example, 11 is SOAP version 1.1, 12 is version 1.2, etc.)

#### 8.2.3.41 rtXmlSetWriteBOM()

```
int rtXmlSetWriteBOM (
    OSCTXT * pctxt,
    OSBOOL write )
```

This function sets whether the Unicode byte order mark is encoded.

##### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>write</i>	TRUE to encode BOM, FALSE to not encode BOM.

##### Returns

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

#### 8.2.3.42 rtXmlSetXSITypeAttr()

```
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

<i>pctxt</i>	Pointer to OSCTXT structure
<i>xsiType</i>	xsi:type attribute value

##### Returns

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

## 8.3 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"
```

## Macros

- #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 referred 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.*

### 8.3.1 Detailed Description

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

## 8.4 rtXmlExternDefs.h File Reference

XML external definitions macro.

### 8.4.1 Detailed Description

XML external definitions macro.

This is used for Windows to properly declare function scope within DLL's.

## 8.5 rtXmlKeyArray.h File Reference

Implementation of a dynamic pointer sorted array.

```
#include "rtxsrc/rtxContext.h"  
#include "rtxmlsrc/rtXmlExternDefs.h"
```

### Functions

- int [rtXmlKeyArrayInit](#) (OSCTXT \*pctxt, OSXSDKeyArray \*pArray, OSUINT32 nmFields, OSBOOL key, const char \*name)  
*Initialize the array by allocating memory for it.*
- void [rtXmlKeyArraySetString](#) (OSXSDKeyArray \*pArray, const OSUTF8CHAR \*pValue, OSUINT32 fldNum)  
*Set the given field's value to the given string value.*
- void [rtXmlKeyArraySetInt](#) (OSXSDKeyArray \*pArray, OSINT32 value, OSUINT32 fldNum)  
*Set the given field's value to the given integer value.*
- void [rtXmlKeyArraySetUInt](#) (OSXSDKeyArray \*pArray, OSUINT32 value, OSUINT32 fldNum)  
*Set the given field's value to the given unsigned integer value.*
- void [rtXmlKeyArraySetDecimal](#) (OSXSDKeyArray \*pArray, const OSREAL \*value, OSUINT32 fldNum)  
*Set the given field's value to the given decimal value.*
- int [rtXmlKeyArrayAdd](#) (OSCTXT \*pctxt, OSXSDKeyArray \*pArray)  
*Once all the fields for a key are set, invoke this method to add the key.*
- int [rtXmlKeyArrayContains](#) (OSCTXT \*pctxt, OSXSDKeyArray \*pArray)  
*Once all the fields for a key are set, this method is used to check whether the key is already present in the array of keys.*

## 8.5.1 Detailed Description

Implementation of a dynamic pointer sorted array.

## 8.5.2 Function Documentation

### 8.5.2.1 rtXmlKeyArrayAdd()

```
int rtXmlKeyArrayAdd (
    OSCTXT * pctxt,
    OSXSDKeyArray * pArray )
```

Once all the fields for a key are set, invoke this method to add the key.

It will report an error if any of the following are true:

- a field was skipped and the constraint is a key constraint
- a field was set more than once This method adds the "new" key (pointed to by *pArray*) into the correct place within the array of keys (and their fields), also pointed to by *pArray*.

### 8.5.2.2 rtXmlKeyArrayContains()

```
int rtXmlKeyArrayContains (
    OSCTXT * pctxt,
    OSXSDKeyArray * pArray )
```

Once all the fields for a key are set, this method is used to check whether the key is already present in the array of keys.

Thus, to check a key ref, you set all the fields, then invoke this method. You do NOT invoke `rtXmlKeyArrayAdd`.

#### Returns

0 if key is found as expected

### 8.5.2.3 rtXmlKeyArrayInit()

```
int rtXmlKeyArrayInit (
    OSCTXT * pctxt,
    OSXSDKeyArray * pArray,
    OSUINT32 nmFields,
    OSBOOL key,
    const char * name )
```

Initialize the array by allocating memory for it.

## Parameters

<i>pctxt</i>	
<i>pArray</i>	The array whose data should be initialized.
<i>nmFields</i>	The number of fields in the key whose data will be held in <i>pArray-&gt;data</i> .
<i>key</i>	TRUE if the data is for a key; FALSE if for a unique identity constraint.
<i>name</i>	of the identity constraint

### 8.5.2.4 rtXmlKeyArraySetDecimal()

```
void rtXmlKeyArraySetDecimal (
    OSXSDKeyArray * pArray,
    const OSREAL * value,
    OSUINT32 fldNum )
```

Set the given field's value to the given decimal value.

see comment on set\* methods above.

### 8.5.2.5 rtXmlKeyArraySetInt()

```
void rtXmlKeyArraySetInt (
    OSXSDKeyArray * pArray,
    OSINT32 value,
    OSUINT32 fldNum )
```

Set the given field's value to the given integer value.

see comment on set\* methods above.

### 8.5.2.6 rtXmlKeyArraySetString()

```
void rtXmlKeyArraySetString (
    OSXSDKeyArray * pArray,
    const OSUTF8CHAR * pValue,
    OSUINT32 fldNum )
```

Set the given field's value to the given string value.

see comment on set\* methods above.



### 8.5.2.7 rtXmlKeyArraySetUInt()

```
void rtXmlKeyArraySetUInt (
    OSXSDKeyArray * pArray,
    OSUINT32 value,
    OSUINT32 fldNum )
```

Set the given field's value to the given unsigned integer value.

see comment on set\* methods above.

## 8.6 rtXmlNamespace.h File Reference

XML namespace handling structures and function definitions.

```
#include "rtxsrc/rtxContext.h"
#include "rtxsrc/rtxDynPtrArray.h"
#include "rtxsrc/rtxXmlQName.h"
#include "rtxmlsrc/rtXmlExternDefs.h"
```

### Macros

- #define [RTXMLNSSETQNAME](#)(qname, pNS)  
*This macro populates the given QName structure with information from the given namespace structure (namespace URI and prefix).*

### Functions

- OSXMLNamespace \* [rtXmlNSAddNamespace](#) (OSCTXT \*pctxt, OSRTDList \*pNSAttrs, const OSUTF8CHAR \*prefix, const OSUTF8CHAR \*uri)  
*This function adds a namespace to the context namespace list.*
- OSBOOL [rtXmlNSEqual](#) (OSXMLNamespace \*pNS1, OSXMLNamespace \*pNS2)  
*This function checks if two namespace records are equal.*
- void [rtXmlNSFreeAttrList](#) (OSCTXT \*pctxt, OSRTDList \*pNSAttrs)  
*This function frees dynamic memory used to hold namespace attribute values.*
- const OSUTF8CHAR \* [rtXmlNSGetPrefix](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*uri)  
*This function gets a namespace prefix assigned to the given URI.*
- const OSUTF8CHAR \* [rtXmlNSGetPrefixUsingIndex](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*uri, OSUINT32 idx)  
*This function gets a namespace prefix assigned to the given URI using the given index to select a specific prefix from the URI/prefix map.*
- OSUINT32 [rtXmlNSGetPrefixCount](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*uri)  
*This function returns the total number of prefixes currently assigned to the given URI.*
- int [rtXmlNSGetPrefixIndex](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*prefix, OSUINT32 \*pcount)

*This function gets the index of a given prefix in the internal list of prefixes maintained for a given URI in the namespace stack.*

- const OSUTF8CHAR \* [rtXmlINSGetQName](#) (OSCTXT \*pctxt, OSUTF8CHAR \*buf, size\_t bufsiz, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*localName)

*This function creates a QName in the given fixed-size buffer.*

- EXTXMLMETHOD const OSUTF8CHAR \* [rtXmlINSGetAttrPrefix](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*namespaceURI, OSRTDList \*pNSAttrs)

*This function returns a namespace prefix for use with attributes in the given namespace.*

- const OSUTF8CHAR \* [rtXmlINSGetAttrQName](#) (OSCTXT \*pctxt, OSUTF8CHAR \*buf, size\_t bufsiz, OSXMLNamespace \*pNS, const OSUTF8CHAR \*localName, OSRTDList \*pNSAttrs)

*This function creates a QName for a qualified attribute.*

- OSXMLNamespace \* [rtXmlINSLookupURI](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*uri)

*This function looks up a namespace in the context namespace stack using URI as the key value.*

- OSXMLNamespace \* [rtXmlINSLookupURIInList](#) (OSRTDList \*pNSAttrs, const OSUTF8CHAR \*uri)

*This function looks up a namespace in the given list using URI as the key value.*

- const OSUTF8CHAR \* [rtXmlINSLookupPrefixForURI](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*uri)

*This function looks up a namespace in the context namespace stack using URI as the key value and returns a non-empty prefix, if one has been defined.*

- const OSUTF8CHAR \* [rtXmlINSLookupPrefix](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*prefix)

*This function looks up a namespace in the context namespace list using the prefix as the key value.*

- const OSUTF8CHAR \* [rtXmlINSLookupPrefixFrag](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*prefix, size\_t prefixLen)

*This function looks up a namespace in the context namespace list using the prefix as the key value.*

- void [rtXmlINSRemoveAll](#) (OSCTXT \*pctxt)

*This function removes all namespaces from the context namespace list and frees the dynamic memory used to hold the names.*

- OSXMLNamespace \* [rtXmlINSSetNamespace](#) (OSCTXT \*pctxt, OSRTDList \*pNSAttrs, const OSUTF8CHAR \*prefix, const OSUTF8CHAR \*uri, OSBOOL override)

*This function sets a namespace in the context namespace list.*

- const OSUTF8CHAR \* [rtXmlINSNewPrefix](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*uri, OSRTDList \*pNSAttrs)

*This function returns the next unused prefix of the form "nsX" where X is a number.*

- int [rtXmlNSAddPrefixLink](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*prefix, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*nsTable[], OSUINT32 nsTableRowCount)

*Add a prefix link at the current stack level.*

- int [rtXmlNSFreeAllPrefixLinks](#) (OSCTXT \*pctxt, OSXMLNSPfxLinkStackNode \*pStackNode)

*Free all prefix links in the given namespace stack entry.*

- int [rtXmlNSFreePrefixLink](#) (OSCTXT \*pctxt, OSXMLNSPfxLink \*pLink)

*Free all data within a given namespace prefix link structure.*

- int [rtXmlNSGetIndex](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*prefix)

*Get namespace index for a given namespace prefix based on current namespace stack in context.*

- int [rtXmlNSPush](#) (OSCTXT \*pctxt)

*Push new namespace prefix mapping level onto stack.*

- int [rtXmlNSPop](#) (OSCTXT \*pctxt)

*Remove top namespace prefix mapping level from stack.*

- void [rtXmlNSSetURITable](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*data[], OSUINT32 nRows)

*Set namespace URI table in context.*

## 8.6.1 Detailed Description

XML namespace handling structures and function definitions.

## 8.6.2 Macro Definition Documentation

### 8.6.2.1 RTXMLNSSETQNAME

```
#define RTXMLNSSETQNAME(  
    qname,  
    pNS )
```

#### Value:

```
if (0 != pNS) { qname.nsPrefix = pNS->prefix; qname.nsURI = pNS->uri; } \  
else { qname.nsPrefix = qname.nsURI = 0; }
```

This macro populates the given QName structure with information from the given namespace structure (namespace URI and prefix).

#### Parameters

<i>qname</i>	Reference to QName structure to be populated. Pointers to items in pNS are assigned directly to fields in <i>qname</i> . No copies of data are made.
<i>pNS</i>	Pointer to namespace structure.

Definition at line 330 of file rtXmlNamespace.h.

## 8.6.3 Function Documentation

### 8.6.3.1 rtXmlNSAddNamespace()

```
OSXMLNamespace* rtXmlNSAddNamespace (  
    OSCTXT * pctxt,  
    OSRTDList * pNSAttrs,  
    const OSUTF8CHAR * prefix,  
    const OSUTF8CHAR * uri )
```

This function adds a namespace to the context namespace list.

### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>pNSAttrs</i>	Namespace attribute list to which namespace info is to be added.
<i>prefix</i>	Namespace prefix to be added
<i>uri</i>	Namespace URI to be added

### Returns

Pointer to namespace structure or NULL if not added.

### 8.6.3.2 rtXmlNSEqual()

```
OSBOOL rtXmlNSEqual (
    OSXMLNamespace * pNS1,
    OSXMLNamespace * pNS2 )
```

This function checks if two namespace records are equal.

This does a a deep compare in that it will first check if the pointers are equal and then it will check if the contents are equal (same prefix and URI).

### Parameters

<i>pNS1</i>	Pointer to first namespace records to check.
<i>pNS2</i>	Pointer to second record.

### Returns

True if records are equal, false otherwise.

### 8.6.3.3 rtXmlNSFreeAttrList()

```
void rtXmlNSFreeAttrList (
    OSCTXT * pctxt,
    OSRTDList * pNSAttrs )
```

This function frees dynamic memory used to hold namespace attribute values.

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>pNSAttrs</i>	Pointer to namespace attribute list to be freed.

#### 8.6.3.4 rtXmlNSGetAttrPrefix()

```
EXTXMLMETHOD const OSUTF8CHAR* rtXmlNSGetAttrPrefix (  
    OSCTXT * pctxt,  
    const OSUTF8CHAR * namespaceURI,  
    OSRTDList * pNSAttrs )
```

This function returns a namespace prefix for use with attributes in the given namespace.

If a prefix is not found for the namespace, a new namespace entry is created with a generated prefix.

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>namespaceURI</i>	Pointer to namespace URI
<i>pNSAttrs</i>	List of namespace records. If null, the namespace list in the context will be used.

#### Returns

The prefix.

#### 8.6.3.5 rtXmlNSGetAttrQName()

```
const OSUTF8CHAR* rtXmlNSGetAttrQName (  
    OSCTXT * pctxt,  
    OSUTF8CHAR * buf,  
    size_t bufsiz,  
    OSXMLNamespace * pNS,  
    const OSUTF8CHAR * localName,  
    OSRTDList * pNSAttrs )
```

This function creates a QName for a qualified attribute.

If a prefix is not found for the name, a new namespace entry is created with a generated prefix.

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>buf</i>	Buffer into which QName will be written.
<i>bufsiz</i>	Size of the buffer.
<i>pNS</i>	Pointer to namespace URI and prefix structure.
<i>localName</i>	Local name of the item.
<i>pNSAttrs</i>	List of namespace records. If null, the namespace list in the context will be used.

#### Returns

Pointer to QName buffer (*buf*).

#### 8.6.3.6 rtXmlNSGetPrefix()

```
const OSUTF8CHAR* rtXmlNSGetPrefix (
    OSCTXT * pctxt,
    const OSUTF8CHAR * uri )
```

This function gets a namespace prefix assigned to the given URI.

This gives preference to empty prefixes.

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>uri</i>	Namespace URI to be searched for

#### Returns

Pointer to namespace prefix string. If a NULL or empty prefix was assigned to the URI, an empty string is returned. Otherwise, any assigned prefix is returned. If no prefix was assigned, null is returned.

#### 8.6.3.7 rtXmlNSGetPrefixCount()

```
OSUINT32 rtXmlNSGetPrefixCount (
    OSCTXT * pctxt,
    const OSUTF8CHAR * uri )
```

This function returns the total number of prefixes currently assigned to the given URI.

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>uri</i>	Namespace URI to be searched for

#### Returns

Count of prefixes assigned to the URI.

#### 8.6.3.8 rtXmlNSGetPrefixIndex()

```
int rtXmlNSGetPrefixIndex (
    OSCTXT * pctxt,
    const OSUTF8CHAR * uri,
    const OSUTF8CHAR * prefix,
    OSUINT32 * pcount )
```

This function gets the index of a given prefix in the internal list of prefixes maintained for a given URI in the namespace stack.

It also may return the total number of prefixes currently assigned to the URI.

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>uri</i>	Namespace URI to be searched for
<i>prefix</i>	Namespace prefix for which to get index.
<i>pcount</i>	Optional pointer to an integer count variable. If provided, the total number of prefixes currently assigned to the URI will be returned.

#### Returns

Index to namespace prefix or -1 if the prefix is not assigned to the given URI.

#### 8.6.3.9 rtXmlNSGetPrefixUsingIndex()

```
const OSUTF8CHAR* rtXmlNSGetPrefixUsingIndex (
    OSCTXT * pctxt,
    const OSUTF8CHAR * uri,
    OSUINT32 idx )
```

This function gets a namespace prefix assigned to the given URI using the given index to select a specific prefix from the URI/prefix map.

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>uri</i>	Namespace URI to be searched for
<i>idx</i>	Index to prefix in map. This only has meaning when multiple prefixes have been assigned to the given URI.

#### Returns

Pointer to namespace prefix string

#### 8.6.3.10 rtXmlNSGetQName()

```
const OSUTF8CHAR* rtXmlNSGetQName (
    OSCTXT * pctxt,
    OSUTF8CHAR * buf,
    size_t bufsiz,
    const OSUTF8CHAR * uri,
    const OSUTF8CHAR * localName )
```

This function creates a QName in the given fixed-site buffer.

If the name will not fit in the buffer, it is truncated.

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>buf</i>	Buffer into which qname will be written.
<i>bufsiz</i>	Size of the buffer.
<i>uri</i>	Namespace URI.
<i>localName</i>	Local name of the item.

#### Returns

Pointer to QName buffer (buf).

#### 8.6.3.11 rtXmlNSLookupPrefix()

```
const OSUTF8CHAR* rtXmlNSLookupPrefix (
    OSCTXT * pctxt,
    const OSUTF8CHAR * prefix )
```

This function looks up a namespace in the context namespace list using the prefix as the key value.



### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>prefix</i>	Namespace Prefix to be found.

### Returns

Pointer to namespace URI or NULL if not found.

### 8.6.3.12 rtXmlNSLookupPrefixForURI()

```
const OSUTF8CHAR* rtXmlNSLookupPrefixForURI (
    OSCTXT * pctxt,
    const OSUTF8CHAR * uri )
```

This function looks up a namespace in the context namespace stack using URI as the key value and returns a non-empty prefix, if one has been defined.

### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>uri</i>	Namespace URI to be found.

### Returns

Pointer to non-empty prefix. NULL if URI is not found or URI has no associated non-empty prefix.

### 8.6.3.13 rtXmlNSLookupPrefixFrag()

```
const OSUTF8CHAR* rtXmlNSLookupPrefixFrag (
    OSCTXT * pctxt,
    const OSUTF8CHAR * prefix,
    size_t prefixLen )
```

This function looks up a namespace in the context namespace list using the prefix as the key value.

### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>prefix</i>	Namespace Prefix to be found.
<i>prefixLen</i>	Namespace Prefix length.

**Returns**

Pointer to namespace URI or NULL if not found.

**8.6.3.14 rtXmlNSLookupURI()**

```
OSXMLNamespace* rtXmlNSLookupURI (
    OSCTXT * pctxt,
    const OSUTF8CHAR * uri )
```

This function looks up a namespace in the context namespace stack using URI as the key value.

**Parameters**

<i>pctxt</i>	Pointer to OSCTXT structure
<i>uri</i>	Namespace URI to be found.

**Returns**

Pointer to namespace structure or NULL if not found.

**8.6.3.15 rtXmlNSLookupURIInList()**

```
OSXMLNamespace* rtXmlNSLookupURIInList (
    OSRTDList * pNSAttrs,
    const OSUTF8CHAR * uri )
```

This function looks up a namespace in the given list using URI as the key value.

**Parameters**

<i>pNSAttrs</i>	List of namespace records.
<i>uri</i>	Namespace URI to be found.

**Returns**

Pointer to namespace structure or NULL if not found.

### 8.6.3.16 rtXmlNSNewPrefix()

```
const OSUTF8CHAR* rtXmlNSNewPrefix (
    OSCTXT * pctxt,
    const OSUTF8CHAR * uri,
    OSRTDList * pNSAttrs )
```

This function returns the next unused prefix of the form "nsX" where X is a number.

The new namespace declaration is added to the list provided or the context list if a NULL pointer is passed for *pNSAttrs*.

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>uri</i>	Namespace URI. Must not be NULL or empty string.
<i>pNSAttrs</i>	Pointer to list of namespace attributes. If null, the namespace list in the context will be used.

#### Returns

New namespace prefix.

### 8.6.3.17 rtXmlNSRemoveAll()

```
void rtXmlNSRemoveAll (
    OSCTXT * pctxt )
```

This function removes all namespaces from the context namespace list and frees the dynamic memory used to hold the names.

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
--------------	-----------------------------

### 8.6.3.18 rtXmlNSSetNamespace()

```
OSXMLNamespace* rtXmlNSSetNamespace (
    OSCTXT * pctxt,
    OSRTDList * pNSAttrs,
    const OSUTF8CHAR * prefix,
    const OSUTF8CHAR * uri,
    OSBOOL override )
```

This function 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 action depends on the value of the override flag. If true, the value of the namespace prefix will be changed to the given prefix. If false, the existing namespace specification is not altered.

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>pNSAttrs</i>	Namespace attribute list to which namespace info is to be added.
<i>prefix</i>	Namespace prefix
<i>uri</i>	Namespace URI
<i>override</i>	Should existing definition be changed?

#### Returns

Pointer to namespace structure or NULL if not set.

## 8.7 rtXmlpCppDecFuncs.h File Reference

XML low-level C++ decode functions.

```
#include "rtxmlsrc/osrtxml.h"  
#include "rtxmlsrc/rtXmlPull.h"  
#include "rtxmlsrc/OSXSDComplexType.h"
```

### Classes

- class [OSXMLStringListParser](#)  
*Class enabling parsing of an XML Schema list into strings.*

#### 8.7.1 Detailed Description

XML low-level C++ decode functions.

These are overloaded versions of C XML encode functions for use with C++.

# Index

- DOM API functions., 152
  - domAddAttribute, 153
  - domAddCdata, 154
  - domAddContent, 154
  - domCreateChild, 154
  - domCreateDocument, 155
  - domFreeDoc, 156
  - domGetAttrData, 156
  - domGetChild, 156
  - domGetDoc, 157
  - domGetElementName, 157
  - domGetNext, 158
  - domGetNextAttr, 158
  - domGetNodeAttributesNum, 159
  - domGetNodeContent, 159
  - domGetNodeFirstAttribute, 160
  - domGetRootElement, 160
  - domParseFile, 160
  - domSaveDoc, 161
- DOM runtime encode/decode functions., 162
  - rtDomAddAttr, 162
  - rtDomAddNSAttrs, 164
  - rtDomAddNode, 163
  - rtDomAddSubTree, 164
  - rtDomDecodeDoc, 165
  - rtDomEncAny, 166
  - rtDomEncAnyAttr, 166
  - rtDomEncString, 167
  - rtDomEncStringValue, 167
  - rtDomEncXSIAttrs, 168
  - rtDomSetNode, 168
- domAddAttribute
  - DOM API functions., 153
- domAddCdata
  - DOM API functions., 154
- domAddContent
  - DOM API functions., 154
- domCreateChild
  - DOM API functions., 154
- domCreateDocument
  - DOM API functions., 155
- domFreeDoc
  - DOM API functions., 156
- domGetAttrData
  - DOM API functions., 156
- domGetChild
  - DOM API functions., 156
- domGetDoc
  - DOM API functions., 157
- domGetElementName
  - DOM API functions., 157
- domGetNext
  - DOM API functions., 158
- domGetNextAttr
  - DOM API functions., 158
- domGetNodeAttributesNum
  - DOM API functions., 159
- domGetNodeContent
  - DOM API functions., 159
- domGetNodeFirstAttribute
  - DOM API functions., 160
- domGetRootElement
  - DOM API functions., 160
- domParseFile
  - DOM API functions., 160
- domSaveDoc
  - DOM API functions., 161
- next
  - OSXMLStringListParser, 172
- OSXMLGroupDesc, 171
  - osrxml.h, 190
- OSXMLStringListParser, 171
  - next, 172
  - OSXMLStringListParser, 172
- osrtdom.h, 175
- osrxml.h, 176
  - OSXMLGroupDesc, 190
  - rtSaxGetAttrValue, 190
  - rtSaxGetElemID8, 191
  - rtSaxGetElemID, 190
  - rtSaxHasXMLNSAttrs, 192
  - rtSaxIsEmptyBuffer, 192
  - rtSaxSortAttrs, 192
  - rtSaxStrListMatch, 193
  - rtSaxStrListParse, 193
  - rtXmlCmpBase64Str, 194
  - rtXmlCmpHexStr, 194
  - rtXmlCreateFileInputSource, 195
  - rtXmlInitContext, 195

- rtXmlInitContextUsingKey, [195](#)
- rtXmlInitCtxtAppInfo, [196](#)
- rtXmlMatchBase64Str, [196](#)
- rtXmlMatchDate, [197](#)
- rtXmlMatchDateTime, [197](#)
- rtXmlMatchGDay, [198](#)
- rtXmlMatchGMonth, [198](#)
- rtXmlMatchGMonthDay, [198](#)
- rtXmlMatchGYear, [199](#)
- rtXmlMatchGYearMonth, [199](#)
- rtXmlMatchHexStr, [200](#)
- rtXmlMatchTime, [200](#)
- rtXmlMemFreeAnyAttrs, [201](#)
- rtXmlNewQName, [201](#)
- rtXmlPrepareContext, [202](#)
- rtXmlSetEncC14N, [202](#)
- rtXmlSetEncDocHdr, [202](#)
- rtXmlSetEncXSINamespace, [203](#)
- rtXmlSetEncXSINilAttr, [204](#)
- rtXmlSetEncodingStr, [203](#)
- rtXmlSetFormatting, [204](#)
- rtXmlSetIndent, [205](#)
- rtXmlSetIndentChar, [205](#)
- rtXmlSetNSPrefixLinks, [206](#)
- rtXmlSetNamespacesSet, [205](#)
- rtXmlSetNoNSSchemaLocation, [206](#)
- rtXmlSetSchemaLocation, [207](#)
- rtXmlSetSoapVersion, [207](#)
- rtXmlSetWriteBOM, [208](#)
- rtXmlSetXSITypeAttr, [208](#)

**RTXMLNSSETQNAME**

- rtXmlNamespace.h, [215](#)

**rtDomAddAttr**

- DOM runtime encode/decode functions., [162](#)

**rtDomAddNSAttrs**

- DOM runtime encode/decode functions., [164](#)

**rtDomAddNode**

- DOM runtime encode/decode functions., [163](#)

**rtDomAddSubTree**

- DOM runtime encode/decode functions., [164](#)

**rtDomDecodeDoc**

- DOM runtime encode/decode functions., [165](#)

**rtDomEncAny**

- DOM runtime encode/decode functions., [166](#)

**rtDomEncAnyAttr**

- DOM runtime encode/decode functions., [166](#)

**rtDomEncString**

- DOM runtime encode/decode functions., [167](#)

**rtDomEncStringValue**

- DOM runtime encode/decode functions., [167](#)

**rtDomEncXSIAAttrs**

- DOM runtime encode/decode functions., [168](#)

**rtDomSetNode**

- DOM runtime encode/decode functions., [168](#)

**rtSaxGetAttrValue**

- osrxml.h, [190](#)

**rtSaxGetElemID8**

- osrxml.h, [191](#)

**rtSaxGetElemID**

- osrxml.h, [190](#)

**rtSaxHasXMLNSAttrs**

- osrxml.h, [192](#)

**rtSaxIsEmptyBuffer**

- osrxml.h, [192](#)

**rtSaxSortAttrs**

- osrxml.h, [192](#)

**rtSaxStrListMatch**

- osrxml.h, [193](#)

**rtSaxStrListParse**

- osrxml.h, [193](#)

**rtXmlCmpBase64Str**

- osrxml.h, [194](#)

**rtXmlCmpHexStr**

- osrxml.h, [194](#)

**rtXmlCreateFileInputSource**

- osrxml.h, [195](#)

**rtXmlDecBase64Binary**

- XML decode functions., [13](#)

**rtXmlDecBase64Str**

- XML decode functions., [14](#)

**rtXmlDecBase64Str64**

- XML decode functions., [14](#)

**rtXmlDecBase64StrValue**

- XML decode functions., [15](#)

**rtXmlDecBase64StrValue64**

- XML decode functions., [16](#)

**rtXmlDecBigInt**

- XML decode functions., [16](#)

**rtXmlDecBool**

- XML decode functions., [17](#)

**rtXmlDecDate**

- XML decode functions., [17](#)

**rtXmlDecDateTime**

- XML decode functions., [18](#)

**rtXmlDecDecimal**

- XML decode functions., [18](#)

**rtXmlDecDouble**

- XML decode functions., [19](#)

**rtXmlDecDynBase64Str**

- XML decode functions., [19](#)

**rtXmlDecDynBase64Str64**

- XML decode functions., [20](#)

**rtXmlDecDynHexStr**

- XML decode functions., [20](#)

**rtXmlDecDynHexStr64**

- XML decode functions., [21](#)

**rtXmlDecDynUTF8Str**

- XML decode functions., 21
- rtXmlDecEmptyElement
  - XML decode functions., 22
- rtXmlDecGDay
  - XML decode functions., 22
- rtXmlDecGMonth
  - XML decode functions., 23
- rtXmlDecGMonthDay
  - XML decode functions., 23
- rtXmlDecGYear
  - XML decode functions., 24
- rtXmlDecGYearMonth
  - XML decode functions., 24
- rtXmlDecHexBinary
  - XML decode functions., 25
- rtXmlDecHexStr
  - XML decode functions., 25
- rtXmlDecHexStr64
  - XML decode functions., 26
- rtXmlDecInt
  - XML decode functions., 27
- rtXmlDecInt16
  - XML decode functions., 27
- rtXmlDecInt64
  - XML decode functions., 28
- rtXmlDecInt8
  - XML decode functions., 28
- rtXmlDecNSAttr
  - XML decode functions., 29
- rtXmlDecQName
  - XML decode functions., 29
- rtXmlDecTime
  - XML decode functions., 30
- rtXmlDecUInt
  - XML decode functions., 31
- rtXmlDecUInt16
  - XML decode functions., 31
- rtXmlDecUInt64
  - XML decode functions., 32
- rtXmlDecUInt8
  - XML decode functions., 32
- rtXmlDecUTF8Str
  - XML decode functions., 33
- rtXmlDecXSIAttr
  - XML decode functions., 34
- rtXmlDecXSIAttrS
  - XML decode functions., 34
- rtXmlDecXmlStr
  - XML decode functions., 33
- rtXmlEncAny
  - XML encode functions., 42
- rtXmlEncAnyAttr
  - XML encode functions., 43
- rtXmlEncAnyTypeValue

- XML encode functions., 43
- rtXmlEncAttrC14N
  - XML pull-parser decode functions., 97
- rtXmlEncBOM
  - XML encode functions., 49
- rtXmlEncBase64Binary
  - XML encode functions., 44
- rtXmlEncBase64BinaryAttr
  - XML encode functions., 44
- rtXmlEncBase64StrValue
  - XML encode functions., 45
- rtXmlEncBigInt
  - XML encode functions., 45
- rtXmlEncBigIntAttr
  - XML encode functions., 46
- rtXmlEncBigIntValue
  - XML encode functions., 47
- rtXmlEncBinStrValue
  - XML encode functions., 47
- rtXmlEncBitString
  - XML encode functions., 48
- rtXmlEncBitStringExt
  - XML encode functions., 48
- rtXmlEncBool
  - XML encode functions., 50
- rtXmlEncBoolAttr
  - XML encode functions., 50
- rtXmlEncBoolValue
  - XML encode functions., 51
- rtXmlEncCanonicalSort
  - XML encode functions., 51
- rtXmlEncComment
  - XML encode functions., 52
- rtXmlEncDate
  - XML encode functions., 52
- rtXmlEncDateTime
  - XML encode functions., 53
- rtXmlEncDateTimeValue
  - XML encode functions., 53
- rtXmlEncDateValue
  - XML encode functions., 54
- rtXmlEncDecimal
  - XML encode functions., 54
- rtXmlEncDecimalAttr
  - XML encode functions., 55
- rtXmlEncDecimalValue
  - XML encode functions., 55
- rtXmlEncDouble
  - XML encode functions., 56
- rtXmlEncDoubleAttr
  - XML encode functions., 56
- rtXmlEncDoubleNormalValue
  - XML encode functions., 57
- rtXmlEncDoubleValue

- XML encode functions., 58
- rtXmlEncEmptyElement
  - XML encode functions., 58
- rtXmlEncEndDocument
  - XML encode functions., 59
- rtXmlEncEndElement
  - XML encode functions., 59
- rtXmlEncEndSoapElements
  - XML encode functions., 60
- rtXmlEncEndSoapEnv
  - XML encode functions., 60
- rtXmlEncFloat
  - XML encode functions., 61
- rtXmlEncFloatAttr
  - XML encode functions., 61
- rtXmlEncGDay
  - XML encode functions., 62
- rtXmlEncGDayValue
  - XML encode functions., 62
- rtXmlEncGMonth
  - XML encode functions., 63
- rtXmlEncGMonthDay
  - XML encode functions., 64
- rtXmlEncGMonthDayValue
  - XML encode functions., 64
- rtXmlEncGMonthValue
  - XML encode functions., 65
- rtXmlEncGYear
  - XML encode functions., 65
- rtXmlEncGYearMonth
  - XML encode functions., 66
- rtXmlEncGYearMonthValue
  - XML encode functions., 66
- rtXmlEncGYearValue
  - XML encode functions., 67
- rtXmlEncHexBinary
  - XML encode functions., 67
- rtXmlEncHexBinaryAttr
  - XML encode functions., 68
- rtXmlEncHexStrValue
  - XML encode functions., 68
- rtXmlEncIndent
  - XML encode functions., 69
- rtXmlEncInt
  - XML encode functions., 69
- rtXmlEncInt64
  - XML encode functions., 70
- rtXmlEncInt64Attr
  - XML encode functions., 70
- rtXmlEncInt64Value
  - XML encode functions., 71
- rtXmlEncIntAttr
  - XML encode functions., 72
- rtXmlEncIntPattern

- XML encode functions., 72
- rtXmlEncIntValue
  - XML encode functions., 73
- rtXmlEncNSAttrs
  - XML encode functions., 74
- rtXmlEncNamedBits
  - XML encode functions., 73
- rtXmlEncReal10
  - XML encode functions., 74
- rtXmlEncSoapArrayTypeAttr
  - XML encode functions., 75
- rtXmlEncStartDocument
  - XML encode functions., 76
- rtXmlEncStartElement
  - XML encode functions., 76
- rtXmlEncStartSoapElements
  - XML encode functions., 77
- rtXmlEncStartSoapEnv
  - XML encode functions., 77
- rtXmlEncString
  - XML encode functions., 78
- rtXmlEncStringValue
  - XML encode functions., 78
- rtXmlEncStringValue2
  - XML encode functions., 79
- rtXmlEncTermStartElement
  - XML encode functions., 79
- rtXmlEncTime
  - XML encode functions., 80
- rtXmlEncTimeValue
  - XML encode functions., 80
- rtXmlEncUInt
  - XML encode functions., 81
- rtXmlEncUInt64
  - XML encode functions., 81
- rtXmlEncUInt64Attr
  - XML encode functions., 82
- rtXmlEncUInt64Value
  - XML encode functions., 83
- rtXmlEncUIntAttr
  - XML encode functions., 83
- rtXmlEncUIntValue
  - XML encode functions., 84
- rtXmlEncUTF8Attr
  - XML encode functions., 85
- rtXmlEncUTF8Attr2
  - XML encode functions., 85
- rtXmlEncUTF8Str
  - XML encode functions., 86
- rtXmlEncUnicodeStr
  - XML encode functions., 84
- rtXmlEncXSIAttrs
  - XML encode functions., 86
- rtXmlEncXSINilAttr



- XML encode functions., 87
- rtXmlEncXSISTypeAttr
  - XML encode functions., 87
- rtXmlEncXSISTypeAttr2
  - XML encode functions., 88
- rtXmlErrCodes.h, 208
- rtXmlExternDefs.h, 210
- rtXmlFreeInputSource
  - XML encode functions., 88
- rtXmlGetEncBufLen
  - XML encode functions., 41
- rtXmlGetEncBufPtr
  - XML encode functions., 42
- rtXmlGetIndent
  - XML encode functions., 89
- rtXmlGetIndentChar
  - XML encode functions., 89
- rtXmlGetWriteBOM
  - XML encode functions., 89
- rtXmlInitContext
  - osrtxml.h, 195
- rtXmlInitContextUsingKey
  - osrtxml.h, 195
- rtXmlInitCtxtAppInfo
  - osrtxml.h, 196
- rtXmlKeyArray.h, 210
  - rtXmlKeyArrayAdd, 211
  - rtXmlKeyArrayContains, 211
  - rtXmlKeyArrayInit, 211
  - rtXmlKeyArraySetDecimal, 212
  - rtXmlKeyArraySetInt, 212
  - rtXmlKeyArraySetString, 212
  - rtXmlKeyArraySetUInt, 212
- rtXmlKeyArrayAdd
  - rtXmlKeyArray.h, 211
- rtXmlKeyArrayContains
  - rtXmlKeyArray.h, 211
- rtXmlKeyArrayInit
  - rtXmlKeyArray.h, 211
- rtXmlKeyArraySetDecimal
  - rtXmlKeyArray.h, 212
- rtXmlKeyArraySetInt
  - rtXmlKeyArray.h, 212
- rtXmlKeyArraySetString
  - rtXmlKeyArray.h, 212
- rtXmlKeyArraySetUInt
  - rtXmlKeyArray.h, 212
- rtXmlMatchBase64Str
  - osrtxml.h, 196
- rtXmlMatchDate
  - osrtxml.h, 197
- rtXmlMatchDateTime
  - osrtxml.h, 197
- rtXmlMatchGDay
  - osrtxml.h, 198
- rtXmlMatchGMonth
  - osrtxml.h, 198
- rtXmlMatchGMonthDay
  - osrtxml.h, 198
- rtXmlMatchGYear
  - osrtxml.h, 199
- rtXmlMatchGYearMonth
  - osrtxml.h, 199
- rtXmlMatchHexStr
  - osrtxml.h, 200
- rtXmlMatchTime
  - osrtxml.h, 200
- rtXmlMemFreeAnyAttrs
  - osrtxml.h, 201
- rtXmlINSAddNamespace
  - rtXmlNamespace.h, 215
- rtXmlINSEqual
  - rtXmlNamespace.h, 216
- rtXmlINSFreeAttrList
  - rtXmlNamespace.h, 216
- rtXmlINSGetAttrPrefix
  - rtXmlNamespace.h, 217
- rtXmlINSGetAttrQName
  - rtXmlNamespace.h, 217
- rtXmlINSGetPrefix
  - rtXmlNamespace.h, 218
- rtXmlINSGetPrefixCount
  - rtXmlNamespace.h, 218
- rtXmlINSGetPrefixIndex
  - rtXmlNamespace.h, 219
- rtXmlINSGetPrefixUsingIndex
  - rtXmlNamespace.h, 219
- rtXmlINSGetQName
  - rtXmlNamespace.h, 220
- rtXmlINSLookupPrefix
  - rtXmlNamespace.h, 220
- rtXmlINSLookupPrefixForURI
  - rtXmlNamespace.h, 221
- rtXmlINSLookupPrefixFrag
  - rtXmlNamespace.h, 221
- rtXmlINSLookupURIInList
  - rtXmlNamespace.h, 222
- rtXmlINSLookupURI
  - rtXmlNamespace.h, 222
- rtXmlINSNewPrefix
  - rtXmlNamespace.h, 222
- rtXmlINSRemoveAll
  - rtXmlNamespace.h, 223
- rtXmlINSSetNamespace
  - rtXmlNamespace.h, 223
- rtXmlNamespace.h, 213
  - RTXMLNSSETQNAME, 215
  - rtXmlINSAddNamespace, 215

- rtXmlINSEqual, 216
- rtXmlINSFreeAttrList, 216
- rtXmlINSGetAttrPrefix, 217
- rtXmlINSGetAttrQName, 217
- rtXmlINSGetPrefix, 218
- rtXmlINSGetPrefixCount, 218
- rtXmlINSGetPrefixIndex, 219
- rtXmlINSGetPrefixUsingIndex, 219
- rtXmlINSGetQName, 220
- rtXmlINSLookupPrefix, 220
- rtXmlINSLookupPrefixForURI, 221
- rtXmlINSLookupPrefixFrag, 221
- rtXmlINSLookupURIInList, 222
- rtXmlINSLookupURI, 222
- rtXmlINSNewPrefix, 222
- rtXmlINSRemoveAll, 223
- rtXmlINSSetNamespace, 223
- rtXmlNewQName
  - osrtxml.h, 201
- rtXmlParseElemQName
  - XML decode functions., 35
- rtXmlParseElementName
  - XML decode functions., 35
- rtXmlPrepareContext
  - osrtxml.h, 202
- rtXmlPrintNSAttrs
  - XML encode functions., 90
- rtXmlSetEncBufPtr
  - XML encode functions., 90
- rtXmlSetEncC14N
  - osrtxml.h, 202
- rtXmlSetEncDocHdr
  - osrtxml.h, 202
- rtXmlSetEncXSINamespace
  - osrtxml.h, 203
- rtXmlSetEncXSINilAttr
  - osrtxml.h, 204
- rtXmlSetEncodingStr
  - osrtxml.h, 203
- rtXmlSetFormatting
  - osrtxml.h, 204
- rtXmlSetIndent
  - osrtxml.h, 205
- rtXmlSetIndentChar
  - osrtxml.h, 205
- rtXmlSetNSPrefixLinks
  - osrtxml.h, 206
- rtXmlSetNamespacesSet
  - osrtxml.h, 205
- rtXmlSetNoNSSchemaLocation
  - osrtxml.h, 206
- rtXmlSetSchemaLocation
  - osrtxml.h, 207
- rtXmlSetSoapVersion
  - osrtxml.h, 207
- rtXmlSetWriteBOM
  - osrtxml.h, 208
- rtXmlSetXSISchemaTypeAttr
  - osrtxml.h, 208
- rtXmlWriteToFile
  - XML utility functions., 92
- rtXmlpCountListItems
  - XML pull-parser decode functions., 97
- rtXmlpCppDecFuncs.h, 224
- rtXmlpCreateReader
  - XML pull-parser decode functions., 98
- rtXmlpDecAny
  - XML pull-parser decode functions., 98
- rtXmlpDecAny2
  - XML pull-parser decode functions., 99
- rtXmlpDecAnyAttrStr
  - XML pull-parser decode functions., 99
- rtXmlpDecAnyElem
  - XML pull-parser decode functions., 100
- rtXmlpDecBase64Str
  - XML pull-parser decode functions., 100
- rtXmlpDecBase64Str64
  - XML pull-parser decode functions., 101
- rtXmlpDecBigInt
  - XML pull-parser decode functions., 102
- rtXmlpDecBitString
  - XML pull-parser decode functions., 102
- rtXmlpDecBitString64
  - XML pull-parser decode functions., 103
- rtXmlpDecBitStringExt
  - XML pull-parser decode functions., 103
- rtXmlpDecBitStringExt64
  - XML pull-parser decode functions., 104
- rtXmlpDecBool
  - XML pull-parser decode functions., 105
- rtXmlpDecDate
  - XML pull-parser decode functions., 105
- rtXmlpDecDateTime
  - XML pull-parser decode functions., 106
- rtXmlpDecDecimal
  - XML pull-parser decode functions., 106
- rtXmlpDecDouble
  - XML pull-parser decode functions., 107
- rtXmlpDecDoubleExt
  - XML pull-parser decode functions., 108
- rtXmlpDecDynBase64Str
  - XML pull-parser decode functions., 108
- rtXmlpDecDynBase64Str64
  - XML pull-parser decode functions., 109
- rtXmlpDecDynBitString
  - XML pull-parser decode functions., 109
- rtXmlpDecDynHexStr
  - XML pull-parser decode functions., 110

rtXmIpDecDynHexStr64  
     XML pull-parser decode functions., 110  
 rtXmIpDecDynUTF8Str  
     XML pull-parser decode functions., 111  
 rtXmIpDecDynUnicodeStr  
     XML pull-parser decode functions., 111  
 rtXmIpDecGDay  
     XML pull-parser decode functions., 112  
 rtXmIpDecGMonth  
     XML pull-parser decode functions., 112  
 rtXmIpDecGMonthDay  
     XML pull-parser decode functions., 113  
 rtXmIpDecGYear  
     XML pull-parser decode functions., 113  
 rtXmIpDecGYearMonth  
     XML pull-parser decode functions., 114  
 rtXmIpDecHexStr  
     XML pull-parser decode functions., 114  
 rtXmIpDecHexStr64  
     XML pull-parser decode functions., 115  
 rtXmIpDecInt  
     XML pull-parser decode functions., 116  
 rtXmIpDecInt16  
     XML pull-parser decode functions., 116  
 rtXmIpDecInt64  
     XML pull-parser decode functions., 117  
 rtXmIpDecInt8  
     XML pull-parser decode functions., 117  
 rtXmIpDecNamedBits  
     XML pull-parser decode functions., 118  
 rtXmIpDecNamedBits64  
     XML pull-parser decode functions., 118  
 rtXmIpDecStrList  
     XML pull-parser decode functions., 119  
 rtXmIpDecTime  
     XML pull-parser decode functions., 119  
 rtXmIpDecUInt  
     XML pull-parser decode functions., 120  
 rtXmIpDecUInt16  
     XML pull-parser decode functions., 121  
 rtXmIpDecUInt64  
     XML pull-parser decode functions., 121  
 rtXmIpDecUInt8  
     XML pull-parser decode functions., 122  
 rtXmIpDecUTF8Str  
     XML pull-parser decode functions., 122  
 rtXmIpDecXSIAAttr  
     XML pull-parser decode functions., 124  
 rtXmIpDecXSIAAttrs  
     XML pull-parser decode functions., 124  
 rtXmIpDecXSISTypeAttr  
     XML pull-parser decode functions., 125  
 rtXmIpDecXmlStr  
     XML pull-parser decode functions., 123  
 rtXmIpDecXmlStrList  
     XML pull-parser decode functions., 123  
 rtXmIpForceDecodeAsGroup  
     XML pull-parser decode functions., 125  
 rtXmIpGetAttributeCount  
     XML pull-parser decode functions., 126  
 rtXmIpGetAttributeID  
     XML pull-parser decode functions., 126  
 rtXmIpGetCurrentLevel  
     XML pull-parser decode functions., 127  
 rtXmIpGetNextAllElemID16  
     XML pull-parser decode functions., 128  
 rtXmIpGetNextAllElemID32  
     XML pull-parser decode functions., 129  
 rtXmIpGetNextAllElemID  
     XML pull-parser decode functions., 127  
 rtXmIpGetNextElem  
     XML pull-parser decode functions., 129  
 rtXmIpGetNextElemID  
     XML pull-parser decode functions., 130  
 rtXmIpGetNextSeqElemID2  
     XML pull-parser decode functions., 131  
 rtXmIpGetNextSeqElemIDExt  
     XML pull-parser decode functions., 132  
 rtXmIpGetNextSeqElemID  
     XML pull-parser decode functions., 130  
 rtXmIpGetReader  
     XML pull-parser decode functions., 134  
 rtXmIpGetXSISTypeAttr  
     XML pull-parser decode functions., 136  
 rtXmIpGetXSISTypeIndex  
     XML pull-parser decode functions., 136  
 rtXmIpGetXmInsAttrs  
     XML pull-parser decode functions., 134  
 rtXmIpHasAttributes  
     XML pull-parser decode functions., 138  
 rtXmIpHideAttributes  
     XML pull-parser decode functions., 138  
 rtXmIplsDecodeAsGroup  
     XML pull-parser decode functions., 139  
 rtXmIplsEmptyElement  
     XML pull-parser decode functions., 139  
 rtXmIplsLastEventDone  
     XML pull-parser decode functions., 139  
 rtXmIplsUTF8Encoding  
     XML pull-parser decode functions., 140  
 rtXmIpListHasItem  
     XML pull-parser decode functions., 140  
 rtXmIpLookupXSISTypeIndex  
     XML pull-parser decode functions., 141  
 rtXmIpMarkLastEventActive  
     XML pull-parser decode functions., 141  
 rtXmIpMarkPos  
     XML pull-parser decode functions., 142

- rtXmlpMatchEndTag
  - XML pull-parser decode functions., 142
- rtXmlpMatchStartTag
  - XML pull-parser decode functions., 142
- rtXmlpNeedDecodeAttributes
  - XML pull-parser decode functions., 143
- rtXmlpReadBytes
  - XML pull-parser decode functions., 143
- rtXmlpResetMarkedPos
  - XML pull-parser decode functions., 144
- rtXmlpRewindToMarkedPos
  - XML pull-parser decode functions., 144
- rtXmlpSelectAttribute
  - XML pull-parser decode functions., 145
- rtXmlpSetListMode
  - XML pull-parser decode functions., 145
- rtXmlpSetMixedContentMode
  - XML pull-parser decode functions., 146
- rtXmlpSetNamespaceTable
  - XML pull-parser decode functions., 146
- rtXmlpSetWhiteSpaceMode
  - XML pull-parser decode functions., 146
- XML decode functions., 11
  - rtXmlDecBase64Binary, 13
  - rtXmlDecBase64Str, 14
  - rtXmlDecBase64Str64, 14
  - rtXmlDecBase64StrValue, 15
  - rtXmlDecBase64StrValue64, 16
  - rtXmlDecBigInt, 16
  - rtXmlDecBool, 17
  - rtXmlDecDate, 17
  - rtXmlDecDateTime, 18
  - rtXmlDecDecimal, 18
  - rtXmlDecDouble, 19
  - rtXmlDecDynBase64Str, 19
  - rtXmlDecDynBase64Str64, 20
  - rtXmlDecDynHexStr, 20
  - rtXmlDecDynHexStr64, 21
  - rtXmlDecDynUTF8Str, 21
  - rtXmlDecEmptyElement, 22
  - rtXmlDecGDay, 22
  - rtXmlDecGMonth, 23
  - rtXmlDecGMonthDay, 23
  - rtXmlDecGYear, 24
  - rtXmlDecGYearMonth, 24
  - rtXmlDecHexBinary, 25
  - rtXmlDecHexStr, 25
  - rtXmlDecHexStr64, 26
  - rtXmlDeclnt, 27
  - rtXmlDeclnt16, 27
  - rtXmlDeclnt64, 28
  - rtXmlDeclnt8, 28
  - rtXmlDecNSAttr, 29
  - rtXmlDecQName, 29
  - rtXmlDecTime, 30
  - rtXmlDecUInt, 31
  - rtXmlDecUInt16, 31
  - rtXmlDecUInt64, 32
  - rtXmlDecUInt8, 32
  - rtXmlDecUTF8Str, 33
  - rtXmlDecXSIAAttr, 34
  - rtXmlDecXSIAAttrs, 34
  - rtXmlDecXmlStr, 33
  - rtXmlParseElemQName, 35
  - rtXmlParseElementName, 35
- XML encode functions., 37
  - rtXmlEncAny, 42
  - rtXmlEncAnyAttr, 43
  - rtXmlEncAnyTypeValue, 43
  - rtXmlEncBOM, 49
  - rtXmlEncBase64Binary, 44
  - rtXmlEncBase64BinaryAttr, 44
  - rtXmlEncBase64StrValue, 45
  - rtXmlEncBigInt, 45
  - rtXmlEncBigIntAttr, 46
  - rtXmlEncBigIntValue, 47
  - rtXmlEncBinStrValue, 47
  - rtXmlEncBitString, 48
  - rtXmlEncBitStringExt, 48
  - rtXmlEncBool, 50
  - rtXmlEncBoolAttr, 50
  - rtXmlEncBoolValue, 51
  - rtXmlEncCanonicalSort, 51
  - rtXmlEncComment, 52
  - rtXmlEncDate, 52
  - rtXmlEncDateTime, 53
  - rtXmlEncDateTimeValue, 53
  - rtXmlEncDateValue, 54
  - rtXmlEncDecimal, 54
  - rtXmlEncDecimalAttr, 55
  - rtXmlEncDecimalValue, 55
  - rtXmlEncDouble, 56
  - rtXmlEncDoubleAttr, 56
  - rtXmlEncDoubleNormalValue, 57
  - rtXmlEncDoubleValue, 58
  - rtXmlEncEmptyElement, 58
  - rtXmlEncEndDocument, 59
  - rtXmlEncEndElement, 59
  - rtXmlEncEndSoapElems, 60
  - rtXmlEncEndSoapEnv, 60
  - rtXmlEncFloat, 61
  - rtXmlEncFloatAttr, 61
  - rtXmlEncGDay, 62
  - rtXmlEncGDayValue, 62
  - rtXmlEncGMonth, 63
  - rtXmlEncGMonthDay, 64
  - rtXmlEncGMonthDayValue, 64

rtXmlEncGMonthValue, 65  
 rtXmlEncGYear, 65  
 rtXmlEncGYearMonth, 66  
 rtXmlEncGYearMonthValue, 66  
 rtXmlEncGYearValue, 67  
 rtXmlEncHexBinary, 67  
 rtXmlEncHexBinaryAttr, 68  
 rtXmlEncHexStrValue, 68  
 rtXmlEncIndent, 69  
 rtXmlEncInt, 69  
 rtXmlEncInt64, 70  
 rtXmlEncInt64Attr, 70  
 rtXmlEncInt64Value, 71  
 rtXmlEncIntAttr, 72  
 rtXmlEncIntPattern, 72  
 rtXmlEncIntValue, 73  
 rtXmlEncNSAttrs, 74  
 rtXmlEncNamedBits, 73  
 rtXmlEncReal10, 74  
 rtXmlEncSoapArrayTypeAttr, 75  
 rtXmlEncStartDocument, 76  
 rtXmlEncStartElement, 76  
 rtXmlEncStartSoapElements, 77  
 rtXmlEncStartSoapEnv, 77  
 rtXmlEncString, 78  
 rtXmlEncStringValue, 78  
 rtXmlEncStringValue2, 79  
 rtXmlEncTermStartElement, 79  
 rtXmlEncTime, 80  
 rtXmlEncTimeValue, 80  
 rtXmlEncUInt, 81  
 rtXmlEncUInt64, 81  
 rtXmlEncUInt64Attr, 82  
 rtXmlEncUInt64Value, 83  
 rtXmlEncUIntAttr, 83  
 rtXmlEncUIntValue, 84  
 rtXmlEncUTF8Attr, 85  
 rtXmlEncUTF8Attr2, 85  
 rtXmlEncUTF8Str, 86  
 rtXmlEncUnicodeStr, 84  
 rtXmlEncXSIAttrs, 86  
 rtXmlEncXSINilAttr, 87  
 rtXmlEncXSITypeAttr, 87  
 rtXmlEncXSITypeAttr2, 88  
 rtXmlFreeInputSource, 88  
 rtXmlGetEncBufLen, 41  
 rtXmlGetEncBufPtr, 42  
 rtXmlGetIndent, 89  
 rtXmlGetIndentChar, 89  
 rtXmlGetWriteBOM, 89  
 rtXmlPrintNSAttrs, 90  
 rtXmlSetEncBufPtr, 90  
 XML pull-parser decode functions., 93  
 rtXmlEncAttrC14N, 97  
 rtXmIplCountListItems, 97  
 rtXmIplCreateReader, 98  
 rtXmIplDecAny, 98  
 rtXmIplDecAny2, 99  
 rtXmIplDecAnyAttrStr, 99  
 rtXmIplDecAnyElem, 100  
 rtXmIplDecBase64Str, 100  
 rtXmIplDecBase64Str64, 101  
 rtXmIplDecBigInt, 102  
 rtXmIplDecBitString, 102  
 rtXmIplDecBitString64, 103  
 rtXmIplDecBitStringExt, 103  
 rtXmIplDecBitStringExt64, 104  
 rtXmIplDecBool, 105  
 rtXmIplDecDate, 105  
 rtXmIplDecDateTime, 106  
 rtXmIplDecDecimal, 106  
 rtXmIplDecDouble, 107  
 rtXmIplDecDoubleExt, 108  
 rtXmIplDecDynBase64Str, 108  
 rtXmIplDecDynBase64Str64, 109  
 rtXmIplDecDynBitString, 109  
 rtXmIplDecDynHexStr, 110  
 rtXmIplDecDynHexStr64, 110  
 rtXmIplDecDynUTF8Str, 111  
 rtXmIplDecDynUnicodeStr, 111  
 rtXmIplDecGDay, 112  
 rtXmIplDecGMonth, 112  
 rtXmIplDecGMonthDay, 113  
 rtXmIplDecGYear, 113  
 rtXmIplDecGYearMonth, 114  
 rtXmIplDecHexStr, 114  
 rtXmIplDecHexStr64, 115  
 rtXmIplDeclInt, 116  
 rtXmIplDeclInt16, 116  
 rtXmIplDeclInt64, 117  
 rtXmIplDeclInt8, 117  
 rtXmIplDecNamedBits, 118  
 rtXmIplDecNamedBits64, 118  
 rtXmIplDecStrList, 119  
 rtXmIplDecTime, 119  
 rtXmIplDecUInt, 120  
 rtXmIplDecUInt16, 121  
 rtXmIplDecUInt64, 121  
 rtXmIplDecUInt8, 122  
 rtXmIplDecUTF8Str, 122  
 rtXmIplDecXSIAttr, 124  
 rtXmIplDecXSIAttrs, 124  
 rtXmIplDecXSITypeAttr, 125  
 rtXmIplDecXmlStr, 123  
 rtXmIplDecXmlStrList, 123  
 rtXmIplForceDecodeAsGroup, 125  
 rtXmIplGetAttributeCount, 126  
 rtXmIplGetAttributeID, 126

- rtXmlpGetCurrentLevel, [127](#)
- rtXmlpGetNextAllElemID16, [128](#)
- rtXmlpGetNextAllElemID32, [129](#)
- rtXmlpGetNextAllElemID, [127](#)
- rtXmlpGetNextElem, [129](#)
- rtXmlpGetNextElemID, [130](#)
- rtXmlpGetNextSeqElemID2, [131](#)
- rtXmlpGetNextSeqElemIDExt, [132](#)
- rtXmlpGetNextSeqElemID, [130](#)
- rtXmlpGetReader, [134](#)
- rtXmlpGetXSITypeAttr, [136](#)
- rtXmlpGetXSITypeIndex, [136](#)
- rtXmlpGetXmlnsAttrs, [134](#)
- rtXmlpHasAttributes, [138](#)
- rtXmlpHideAttributes, [138](#)
- rtXmlpIsDecodeAsGroup, [139](#)
- rtXmlpIsEmptyElement, [139](#)
- rtXmlpIsLastEventDone, [139](#)
- rtXmlpIsUTF8Encoding, [140](#)
- rtXmlpListHasItem, [140](#)
- rtXmlpLookupXSITypeIndex, [141](#)
- rtXmlpMarkLastEventActive, [141](#)
- rtXmlpMarkPos, [142](#)
- rtXmlpMatchEndTag, [142](#)
- rtXmlpMatchStartTag, [142](#)
- rtXmlpNeedDecodeAttributes, [143](#)
- rtXmlpReadBytes, [143](#)
- rtXmlpResetMarkedPos, [144](#)
- rtXmlpRewindToMarkedPos, [144](#)
- rtXmlpSelectAttribute, [145](#)
- rtXmlpSetListMode, [145](#)
- rtXmlpSetMixedContentMode, [146](#)
- rtXmlpSetNamespaceTable, [146](#)
- rtXmlpSetWhiteSpaceMode, [146](#)
- XML run-time error status codes., [148](#)
- XML\_E\_BASE, [149](#)
- XML\_E\_ELEMMISRQ, [149](#)
- XML\_E\_ELEMSISRQ, [149](#)
- XML\_E\_FLDABSENT, [150](#)
- XML\_E\_NOMATCH, [150](#)
- XML\_E\_NSURINOTFOU, [150](#)
- XML\_E\_TAGMISMATCH, [150](#)
- XML\_OK\_EOB, [151](#)
- XML\_OK\_FRAG, [151](#)
- XML utility functions., [92](#)
- rtXmlWriteToFile, [92](#)
- XML\_E\_BASE
  - XML run-time error status codes., [149](#)
- XML\_E\_ELEMMISRQ
  - XML run-time error status codes., [149](#)
- XML\_E\_ELEMSISRQ
  - XML run-time error status codes., [149](#)
- XML\_E\_FLDABSENT
  - XML run-time error status codes., [150](#)
- XML\_E\_NOMATCH
  - XML run-time error status codes., [150](#)
- XML\_E\_NSURINOTFOU
  - XML run-time error status codes., [150](#)
- XML\_E\_TAGMISMATCH
  - XML run-time error status codes., [150](#)
- XML\_OK\_EOB
  - XML run-time error status codes., [151](#)
- XML\_OK\_FRAG
  - XML run-time error status codes., [151](#)