

# **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>Module Index</b>	<b>3</b>
2.1	Modules . . . . .	3
<b>3</b>	<b>Hierarchical Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	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	<code>rtXmlDecBool()</code>	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	rtXmlDecUInt64()	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	rtXmlEncGYearMonth()	66
6.2.3.45	rtXmlEncGYearMonthValue()	66
6.2.3.46	rtXmlEncGYearValue()	67
6.2.3.47	rtXmlEncHexBinary()	67
6.2.3.48	rtXmlEncHexBinaryAttr()	68
6.2.3.49	rtXmlEncHexStrValue()	68
6.2.3.50	rtXmlEncIndent()	69
6.2.3.51	rtXmlEncInt()	69
6.2.3.52	rtXmlEncInt64()	70
6.2.3.53	rtXmlEncInt64Attr()	71
6.2.3.54	rtXmlEncInt64Value()	71
6.2.3.55	rtXmlEncIntAttr()	72
6.2.3.56	rtXmlEncIntPattern()	72
6.2.3.57	rtXmlEncIntValue()	73
6.2.3.58	rtXmlEncNamedBits()	73
6.2.3.59	rtXmlEncNSAttrs()	74
6.2.3.60	rtXmlEncReal10()	74
6.2.3.61	rtXmlEncSoapArrayTypeAttr()	75
6.2.3.62	rtXmlEncStartDocument()	76
6.2.3.63	rtXmlEncStartElement()	76
6.2.3.64	rtXmlEncStartSoapElems()	77
6.2.3.65	rtXmlEncStartSoapEnv()	77
6.2.3.66	rtXmlEncString()	78
6.2.3.67	rtXmlEncStringValue()	78
6.2.3.68	rtXmlEncStringValue2()	79
6.2.3.69	rtXmlEncTermStartElement()	79
6.2.3.70	rtXmlEncTime()	80

6.2.3.71	rtXmlEncTimeValue()	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	<code>rtXmlEncAttrC14N()</code>	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>rtXmlpGetNextAllElemID16()</code>	128
6.4.2.59	<code>rtXmlpGetNextAllElemID32()</code>	129
6.4.2.60	<code>rtXmlpGetNextElem()</code>	129
6.4.2.61	<code>rtXmlpGetNextElemID()</code>	130
6.4.2.62	<code>rtXmlpGetNextSeqElemID()</code>	131
6.4.2.63	<code>rtXmlpGetNextSeqElemID2()</code>	132
6.4.2.64	<code>rtXmlpGetNextSeqElemIDExt()</code>	132
6.4.2.65	<code>rtXmlpGetReader()</code>	134
6.4.2.66	<code>rtXmlpGetXmlnsAttrs()</code>	134
6.4.2.67	<code>rtXmlpGetXSITypeAttr()</code>	136
6.4.2.68	<code>rtXmlpGetXSITypeIndex()</code>	136
6.4.2.69	<code>rtXmlpHasAttributes()</code>	138
6.4.2.70	<code>rtXmlpHideAttributes()</code>	138
6.4.2.71	<code>rtXmlplsDecodeAsGroup()</code>	139
6.4.2.72	<code>rtXmlplsEmptyElement()</code>	139
6.4.2.73	<code>rtXmlplsLastEventDone()</code>	140
6.4.2.74	<code>rtXmlplsUTF8Encoding()</code>	140
6.4.2.75	<code>rtXmlpListHasItem()</code>	140
6.4.2.76	<code>rtXmlpLookupXSITypeIndex()</code>	141
6.4.2.77	<code>rtXmlpMarkLastEventActive()</code>	141
6.4.2.78	<code>rtXmlpMarkPos()</code>	142
6.4.2.79	<code>rtXmlpMatchEndTag()</code>	142
6.4.2.80	<code>rtXmlpMatchStartTag()</code>	143
6.4.2.81	<code>rtXmlpNeedDecodeAttributes()</code>	143
6.4.2.82	<code>rtXmlpReadBytes()</code>	144
6.4.2.83	<code>rtXmlpResetMarkedPos()</code>	144
6.4.2.84	<code>rtXmlpRewindToMarkedPos()</code>	144
6.4.2.85	<code>rtXmlpSelectAttribute()</code>	145
6.4.2.86	<code>rtXmlpSetListMode()</code>	145
6.4.2.87	<code>rtXmlpSetMixedContentMode()</code>	146
6.4.2.88	<code>rtXmlpSetNamespaceTable()</code>	146
6.4.2.89	<code>rtXmlpSetWhiteSpaceMode()</code>	146

<b>7</b>	<b>Class Documentation</b>	<b>149</b>
7.1	OSXMLContentHandler Class Reference	149
7.1.1	Detailed Description	149
7.1.2	Member Function Documentation	150
7.1.2.1	characters()	150
7.1.2.2	endElement()	150
7.1.2.3	startElement()	151
7.2	OSXMLDecodeBuffer Class Reference	151
7.2.1	Detailed Description	152
7.2.2	Constructor & Destructor Documentation	153
7.2.2.1	OSXMLDecodeBuffer() [1/3]	153
7.2.2.2	OSXMLDecodeBuffer() [2/3]	153
7.2.2.3	OSXMLDecodeBuffer() [3/3]	153
7.2.3	Member Function Documentation	154
7.2.3.1	decodeXML()	154
7.2.3.2	init()	154
7.2.3.3	isA()	155
7.2.3.4	isWellFormed()	155
7.2.3.5	parseElementName()	155
7.2.3.6	parseElemQName()	156
7.2.3.7	setMaxErrors()	156
7.2.4	Member Data Documentation	157
7.2.4.1	mbOwnStream	157
7.3	OSXMLDefaultHandler Class Reference	157
7.3.1	Detailed Description	158
7.3.2	Member Function Documentation	158
7.3.2.1	characters()	158
7.3.2.2	endElement()	158

7.3.2.3	getState()	159
7.3.2.4	startElement()	159
7.4	OSXMLDefaultHandlerIF Class Reference	160
7.4.1	Detailed Description	160
7.5	OSXMLEncodeBase Class Reference	161
7.5.1	Detailed Description	161
7.5.2	Constructor & Destructor Documentation	162
7.5.2.1	OSXMLEncodeBase()	162
7.5.3	Member Function Documentation	162
7.5.3.1	encodeAttr()	162
7.5.3.2	encodeText()	162
7.5.3.3	endElement()	163
7.5.3.4	startDocument()	163
7.5.3.5	startElement()	164
7.5.3.6	termStartElement()	164
7.6	OSXMLEncodeBuffer Class Reference	165
7.6.1	Detailed Description	166
7.6.2	Constructor & Destructor Documentation	166
7.6.2.1	OSXMLEncodeBuffer()	166
7.6.3	Member Function Documentation	166
7.6.3.1	addXMLHeader()	166
7.6.3.2	addXMLText()	167
7.6.3.3	getMsgLen()	167
7.6.3.4	init()	168
7.6.3.5	isA()	168
7.6.3.6	setFragment()	168
7.6.3.7	write() [1/2]	168
7.6.3.8	write() [2/2]	169

7.7	OSXMLEncodeStream Class Reference	169
7.7.1	Detailed Description	170
7.7.2	Constructor & Destructor Documentation	170
7.7.2.1	OSXMLEncodeStream() [1/2]	170
7.7.2.2	OSXMLEncodeStream() [2/2]	171
7.7.3	Member Function Documentation	171
7.7.3.1	getMsgPtr()	171
7.7.3.2	getStream()	172
7.7.3.3	init()	172
7.7.3.4	isA()	172
7.7.4	Member Data Documentation	172
7.7.4.1	mbOwnStream	173
7.7.4.2	mpStream	173
7.8	OSXMLGroupDesc Struct Reference	173
7.8.1	Detailed Description	173
7.9	OSXMLMessageBuffer Class Reference	174
7.9.1	Detailed Description	175
7.9.2	Constructor & Destructor Documentation	175
7.9.2.1	OSXMLMessageBuffer()	175
7.9.3	Member Function Documentation	175
7.9.3.1	getIndent()	175
7.9.3.2	getIndentChar()	176
7.9.3.3	getWriteBOM()	176
7.9.3.4	setAppInfo()	176
7.9.3.5	setFormatting()	176
7.9.3.6	setIndent()	177
7.9.3.7	setIndentChar()	177
7.9.3.8	setNamespace()	177



7.9.3.9	setWriteBOM()	178
7.10	OSXMLNamespaceClass Class Reference	178
7.10.1	Detailed Description	179
7.10.2	Constructor & Destructor Documentation	179
7.10.2.1	OSXMLNamespaceClass() [1/2]	179
7.10.2.2	OSXMLNamespaceClass() [2/2]	180
7.11	OSXMLStringListParser Class Reference	180
7.11.1	Detailed Description	180
7.11.2	Constructor & Destructor Documentation	181
7.11.2.1	OSXMLStringListParser()	181
7.11.3	Member Function Documentation	181
7.11.3.1	next()	181
7.12	OSXMLStrListHandler Class Reference	182
7.12.1	Detailed Description	182
7.13	OSXSDGlobalElement Class Reference	182
7.13.1	Detailed Description	183
7.13.2	Constructor & Destructor Documentation	184
7.13.2.1	OSXSDGlobalElement() [1/3]	184
7.13.2.2	OSXSDGlobalElement() [2/3]	184
7.13.2.3	OSXSDGlobalElement() [3/3]	184
7.13.2.4	~OSXSDGlobalElement()	185
7.13.3	Member Function Documentation	185
7.13.3.1	decodeFrom()	185
7.13.3.2	encodeTo()	185
7.13.3.3	getCtxtPtr()	186
7.13.3.4	memAlloc()	186
7.13.3.5	memFreePtr()	186
7.13.3.6	setDefaultNamespace()	187
7.13.3.7	setDiag()	187
7.13.3.8	setEncXSINameSpace()	187
7.13.3.9	setMsgBuf()	188
7.13.3.10	setNameSpace()	188
7.13.3.11	setNoNSSchemaLocation()	188
7.13.3.12	setSchemaLocation()	189
7.13.3.13	setXSISType()	189
7.13.3.14	validateFrom()	190
7.13.4	Member Data Documentation	190
7.13.4.1	mpContext	190

<b>8</b>	<b>File Documentation</b>	<b>191</b>
8.1	osrtxml.h File Reference	191
8.1.1	Detailed Description	204
8.1.2	Typedef Documentation	204
8.1.2.1	OSXMLGroupDesc	205
8.1.3	Function Documentation	205
8.1.3.1	rtSaxGetAttrValue()	205
8.1.3.2	rtSaxGetElemID()	205
8.1.3.3	rtSaxGetElemID8()	206
8.1.3.4	rtSaxHasXMLNSAttrs()	207
8.1.3.5	rtSaxIsEmptyBuffer()	207
8.1.3.6	rtSaxSortAttrs()	207
8.1.3.7	rtSaxStrListMatch()	208
8.1.3.8	rtSaxStrListParse()	208
8.1.3.9	rtXmlCmpBase64Str()	209
8.1.3.10	rtXmlCmpHexStr()	209
8.1.3.11	rtXmlCreateFileInputSource()	210
8.1.3.12	rtXmlInitContext()	210
8.1.3.13	rtXmlInitContextUsingKey()	210
8.1.3.14	rtXmlInitCtxtAppInfo()	211
8.1.3.15	rtXmlMatchBase64Str()	211
8.1.3.16	rtXmlMatchDate()	212
8.1.3.17	rtXmlMatchDateTime()	212
8.1.3.18	rtXmlMatchGDay()	213
8.1.3.19	rtXmlMatchGMonth()	213
8.1.3.20	rtXmlMatchGMonthDay()	214
8.1.3.21	rtXmlMatchGYear()	214
8.1.3.22	rtXmlMatchGYearMonth()	214

8.1.3.23	<a href="#">rtXmlMatchHexStr()</a>	215
8.1.3.24	<a href="#">rtXmlMatchTime()</a>	215
8.1.3.25	<a href="#">rtXmlMemFreeAnyAttrs()</a>	216
8.1.3.26	<a href="#">rtXmlNewQName()</a>	216
8.1.3.27	<a href="#">rtXmlPrepareContext()</a>	217
8.1.3.28	<a href="#">rtXmlSetEncC14N()</a>	217
8.1.3.29	<a href="#">rtXmlSetEncDocHdr()</a>	217
8.1.3.30	<a href="#">rtXmlSetEncodingStr()</a>	218
8.1.3.31	<a href="#">rtXmlSetEncXSINamespace()</a>	218
8.1.3.32	<a href="#">rtXmlSetEncXSINilAttr()</a>	219
8.1.3.33	<a href="#">rtXmlSetFormatting()</a>	219
8.1.3.34	<a href="#">rtXmlSetIndent()</a>	220
8.1.3.35	<a href="#">rtXmlSetIndentChar()</a>	220
8.1.3.36	<a href="#">rtXmlSetNamespacesSet()</a>	221
8.1.3.37	<a href="#">rtXmlSetNoNSSchemaLocation()</a>	221
8.1.3.38	<a href="#">rtXmlSetNSPrefixLinks()</a>	221
8.1.3.39	<a href="#">rtXmlSetSchemaLocation()</a>	222
8.1.3.40	<a href="#">rtXmlSetSoapVersion()</a>	222
8.1.3.41	<a href="#">rtXmlSetWriteBOM()</a>	223
8.1.3.42	<a href="#">rtXmlSetXSITypeAttr()</a>	223
8.2	<a href="#">OSXMLDecodeBuffer.h File Reference</a>	223
8.2.1	<a href="#">Detailed Description</a>	224
8.3	<a href="#">OSXMLEncodeBuffer.h File Reference</a>	224
8.3.1	<a href="#">Detailed Description</a>	224
8.4	<a href="#">OSXMLEncodeStream.h File Reference</a>	224
8.4.1	<a href="#">Detailed Description</a>	225
8.5	<a href="#">OSXMLMessageBuffer.h File Reference</a>	225
8.5.1	<a href="#">Detailed Description</a>	225

8.6	<a href="#">rtSaxCppAny.h File Reference</a>	225
8.7	<a href="#">rtSaxCppAnyType.h File Reference</a>	225
8.8	<a href="#">rtSaxCppSimpleType.h File Reference</a>	226
8.9	<a href="#">rtSaxCppSoap.h File Reference</a>	226
8.10	<a href="#">rtSaxCppStrList.h File Reference</a>	226
8.11	<a href="#">rtXmlCppEncFuncs.h File Reference</a>	226
8.11.1	Detailed Description	227
8.11.2	Function Documentation	227
8.11.2.1	<a href="#">rtXmlCppEncAnyAttr()</a>	227
8.11.2.2	<a href="#">rtXmlCppEncAnyTypeValue()</a>	228
8.11.2.3	<a href="#">rtXmlCppEncStartElement()</a>	228
8.11.2.4	<a href="#">rtXmlEncAny()</a>	229
8.11.2.5	<a href="#">rtXmlEncString()</a>	229
8.12	<a href="#">rtXmlCppMsgBuf.h File Reference</a>	230
8.12.1	Detailed Description	230
8.13	<a href="#">rtXmlCppNamespace.h File Reference</a>	230
8.13.1	Detailed Description	231
8.14	<a href="#">rtXmlCppXSDElement.h File Reference</a>	231
8.14.1	Detailed Description	231
8.15	<a href="#">rtXmlpCppDecFuncs.h File Reference</a>	231
8.15.1	Detailed Description	231
<b>Index</b>		<b>233</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

## Module Index

### 2.1 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





## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

OSXMLContentHandler . . . . .	149
OSXMLDefaultHandlerIF . . . . .	160
OSXMLDefaultHandler . . . . .	157
OSXMLGroupDesc . . . . .	173
OSXMLMessageBuffer . . . . .	174
OSXMLDecodeBuffer . . . . .	151
OSXMLEncodeBase . . . . .	161
OSXMLEncodeBuffer . . . . .	165
OSXMLEncodeStream . . . . .	169
OSXMLNamespaceClass . . . . .	178
OSXMLStringListParser . . . . .	180
OSXMLStrListHandler . . . . .	182
OSXSDGlobalElement . . . . .	182



# Chapter 4

## Class Index

### 4.1 Class List

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

<a href="#">OSXMLContentHandler</a>	Receive notification of general document events . . . . .	149
<a href="#">OSXMLDecodeBuffer</a>	Derived from the <a href="#">OSXMLMessageBuffer</a> base class . . . . .	151
<a href="#">OSXMLDefaultHandler</a>	This class is derived from the SAX class DefaultHandler base class . . . . .	157
<a href="#">OSXMLDefaultHandlerIF</a>	This class is derived from the SAX class DefaultHandler base class . . . . .	160
<a href="#">OSXMLEncodeBase</a>	<a href="#">OSXMLEncodeBase</a> is a base class for the XML encode buffer and stream classes, <a href="#">OSXMLEncodeBuffer</a> and <a href="#">OSXMLEncodeStream</a> . . . . .	161
<a href="#">OSXMLEncodeBuffer</a>	Derived from the <a href="#">OSXMLEncodeBase</a> class . . . . .	165
<a href="#">OSXMLEncodeStream</a>	Derived from the <a href="#">OSXMLEncodeBase</a> class . . . . .	169
<a href="#">OSXMLGroupDesc</a>	<a href="#">OSXMLGroupDesc</a> describes how entries in an OSXMLElemIDRec array make up a group . . . . .	173
<a href="#">OSXMLMessageBuffer</a>	The XML message buffer class is derived from the OSMMessageBuffer base class . . . . .	174
<a href="#">OSXMLNamespaceClass</a>	This class is used to hold an XML namespace prefix to URI mapping . . . . .	178
<a href="#">OSXMLStringListParser</a>	Class enabling parsing of an XML Schema list into strings . . . . .	180
<a href="#">OSXMLStrListHandler</a>	<a href="#">OSXMLStrListHandler</a> . . . . .	182
<a href="#">OSXSDGlobalElement</a>	XSD global element base class . . . . .	182



# Chapter 5

## File Index

### 5.1 File List

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

<a href="#">osrtxml.h</a>	XML low-level C encode/decode functions . . . . .	191
<a href="#">OSXMLDecodeBuffer.h</a>	XML decode buffer or stream class definition . . . . .	223
<a href="#">OSXMLEncodeBuffer.h</a>	XML encode message buffer class definition . . . . .	224
<a href="#">OSXMLEncodeStream.h</a>	XML encode stream class definition . . . . .	224
<a href="#">OSXMLMessageBuffer.h</a>	XML encode/decode buffer and stream base class . . . . .	225
<a href="#">rtSaxCppAny.h</a>		225
<a href="#">rtSaxCppAnyType.h</a>		225
<a href="#">rtSaxCppParser.h</a>		??
<a href="#">rtSaxCppParserIF.h</a>		??
<a href="#">rtSaxCppSimpleType.h</a>		226
<a href="#">rtSaxCppSoap.h</a>		226
<a href="#">rtSaxCppStrList.h</a>		226
<a href="#">rtXmlCppEncFuncs.h</a>	XML low-level C++ encode functions . . . . .	226
<a href="#">rtXmlCppMsgBuf.h</a>	This file is deprecated . . . . .	230
<a href="#">rtXmlCppNamespace.h</a>	XML namespace handling structures and function definitions . . . . .	230
<a href="#">rtXmlCppXSDElement.h</a>	C++ run-time XML schema global element class definition . . . . .	231
<a href="#">rtXmlpCppDecFuncs.h</a>	XML low-level C++ decode functions . . . . .	231



## 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>pnocets</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 * pnocets,
    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>pnocets</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 rtxXmlDecTime()

```
int rtxXmlDecTime (
    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 `rtXmlEncBinStrValue` (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 bit 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 <i>totalDigits</i> 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 <i>fractionDigits</i> 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 * pctxt )
```

This function returns current nesting level.

## Parameters

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

## Returns

Current nesting level.

### 6.4.2.57 rtXmlpGetNextAllElemID()

```
int rtXmlpGetNextAllElemID (
    OSCTXT * pctxt,
    const OSXMLElemIDRec * tab,
    OSSIZE nrows,
    const OSUINT8 * pOrder,
```

```

OSSIZE nOrder,
OSSIZE maxOrder,
int anyID )

```

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

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

**Parameters**

<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 rtXmIplGetNextElemID()

```
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.

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 rtXmlpGetNextSeqElemID.

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

## Parameters

<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 rtXmlpGetXlnsAttrs()

```
int rtXmlpGetXlnsAttrs (
    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 rtXmIplLookupXSITypeIndex()

```
int rtXmIplLookupXSITypeIndex (
    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 rtXmIplMarkLastEventActive()

```
int rtXmIplMarkLastEventActive (
    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 rtXmIpSetMixedContentMode()

```
OSBOOL rtXmIpSetMixedContentMode (
    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 rtXmIpSetNamespaceTable()

```
void rtXmIpSetNamespaceTable (
    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 rtXmIpSetWhiteSpaceMode()

```
void rtXmIpSetWhiteSpaceMode (
```

```
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.





## Chapter 7

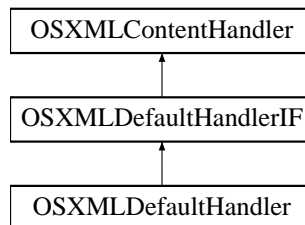
# Class Documentation

### 7.1 OSXMLContentHandler Class Reference

Receive notification of general document events.

```
#include <rtSaxCppParserIF.h>
```

Inheritance diagram for OSXMLContentHandler:



#### Public Member Functions

##### The virtual document handler interface

- virtual int [characters](#) (const OSUTF8CHAR \*const chars, unsigned int length)=0  
*Receive notification of character data.*
- virtual int [endElement](#) (const OSUTF8CHAR \*const uri, const OSUTF8CHAR \*const localname, const OSUTF8CHAR \*const qname)=0  
*Receive notification of the end of an element.*
- virtual int [startElement](#) (const OSUTF8CHAR \*const uri, const OSUTF8CHAR \*const localname, const OSUTF8CHAR \*const qname, const OSUTF8CHAR \*const \*attrs)=0  
*Receive notification of the beginning of an element.*

#### 7.1.1 Detailed Description

Receive notification of general document events.

Definition at line 112 of file rtSaxCppParserIF.h.

## 7.1.2 Member Function Documentation

### 7.1.2.1 characters()

```
virtual int OSXMLContentHandler::characters (
    const OSUTF8CHAR *const chars,
    unsigned int length ) [pure virtual]
```

Receive notification of character data.

The Parser will call this method to report each chunk of character data. SAX parsers may return all contiguous character data in a single chunk, or they may split it into several chunks; however, all of the characters in any single event must come from the same external entity, so that the Locator provides useful information.

#### Parameters

<i>chars</i>	The characters from the XML document.
<i>length</i>	The length of chars.

Implemented in [OSXMLDefaultHandler](#).

### 7.1.2.2 endElement()

```
virtual int OSXMLContentHandler::endElement (
    const OSUTF8CHAR *const uri,
    const OSUTF8CHAR *const localname,
    const OSUTF8CHAR *const qname ) [pure virtual]
```

Receive notification of the end of an element.

The SAX parser will invoke this method at the end of every element in the XML document; there will be a corresponding [startElement\(\)](#) event for every [endElement\(\)](#) event (even when the element is empty).

#### Parameters

<i>uri</i>	The URI of the associated namespace for this element
<i>localname</i>	The local part of the element name
<i>qname</i>	The QName of this element

Implemented in [OSXMLDefaultHandler](#).

### 7.1.2.3 startElement()

```
virtual int OSXMLContentHandler::startElement (
    const OSUTF8CHAR *const uri,
    const OSUTF8CHAR *const localname,
    const OSUTF8CHAR *const qname,
    const OSUTF8CHAR *const * attrs ) [pure virtual]
```

Receive notification of the beginning of an element.

The Parser will invoke this method at the beginning of every element in the XML document; there will be a corresponding [endElement\(\)](#) event for every [startElement\(\)](#) event (even when the element is empty). All of the element's content will be reported, in order, before the corresponding [endElement\(\)](#) event.

#### Parameters

<i>uri</i>	The URI of the associated namespace for this element
<i>localname</i>	The local part of the element name
<i>qname</i>	The QName of this element
<i>attrs</i>	The attributes name/value pairs attached to the element, if any.

#### See also

[endElement](#)

Implemented in [OSXMLDefaultHandler](#).

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

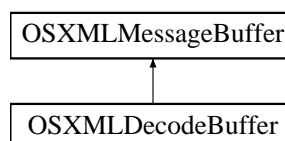
- [rtSaxCppParserIF.h](#)

## 7.2 OSXMLDecodeBuffer Class Reference

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

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

Inheritance diagram for OSXMLDecodeBuffer:



## Public Member Functions

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

## Protected Attributes

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

## Additional Inherited Members

### 7.2.1 Detailed Description

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

It contains variables and methods specific to decoding XML messages. It is used to manage an input buffer or stream containing a message to be decoded.

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

Definition at line 45 of file [OSXMLDecodeBuffer.h](#).

## 7.2.2 Constructor & Destructor Documentation

### 7.2.2.1 OSXMLDecodeBuffer() [1/3]

```
OSXMLDecodeBuffer::OSXMLDecodeBuffer (
    const char * xmlFile )
```

This version of the [OSXMLDecodeBuffer](#) constructor takes a name of a file that contains XML data to be decoded and constructs a buffer.

#### Parameters

<i>xmlFile</i>	A pointer to name of file to be decoded.
----------------	--

### 7.2.2.2 OSXMLDecodeBuffer() [2/3]

```
OSXMLDecodeBuffer::OSXMLDecodeBuffer (
    const OSOCTET * msgbuf,
    size_t bufsiz )
```

This version of the [OSXMLDecodeBuffer](#) constructor takes parameters describing a message in memory to be decoded and constructs a buffer.

#### Parameters

<i>msgbuf</i>	A pointer to a buffer containing an XML message.
<i>bufsiz</i>	Size of the message buffer.

### 7.2.2.3 OSXMLDecodeBuffer() [3/3]

```
OSXMLDecodeBuffer::OSXMLDecodeBuffer (
    OSRTInputStream & inputStream )
```

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

The stream is assumed to have been previously initialized to point at an encoded XML message.

## Parameters

<i>inputStream</i>	reference to the OSInputStream object
--------------------	---------------------------------------

## 7.2.3 Member Function Documentation

### 7.2.3.1 decodeXML()

```
EXTXMLMETHOD int OSXMLDecodeBuffer::decodeXML (
    OSXMLReaderClass * pReader )
```

This method decodes an XML message associated with this buffer.

#### Returns

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

#### Parameters

<i>pReader</i>	Pointer to OSXMLReaderClass object.
----------------	-------------------------------------

#### Returns

Completion status.

### 7.2.3.2 init()

```
virtual EXTMLMETHOD int OSXMLDecodeBuffer::init ( ) [virtual]
```

This method initializes the decode message buffer.

#### Returns

Completion status of operation:

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

### 7.2.3.3 isA()

```
virtual OSBOOL OSXMLDecodeBuffer::isA (
    Type bufferType ) [inline], [virtual]
```

This is a virtual method that must be overridden by derived classes to allow identification of the class.

The base class variant is abstract. This method matches an enumerated identifier defined in the base class. One identifier is declared for each of the derived classes.

#### Parameters

<i>bufferType</i>	Enumerated identifier specifying a derived class. This type is defined as a public access type in the OSRTMessageBufferIF base interface. Possible values include BEREncode, BERDecode, PEREncode, PERDecode, XMLEncode, and XMLDecode.
-------------------	---

#### Returns

Boolean result of the match operation. True if the *bufferType* argument is XMLDecode. argument.

Definition at line 169 of file OSXMLDecodeBuffer.h.

### 7.2.3.4 isWellFormed()

```
EXTXMLMETHOD OSBOOL OSXMLDecodeBuffer::isWellFormed ( )
```

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

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

#### Returns

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

### 7.2.3.5 parseElementName()

```
EXTXMLMETHOD int OSXMLDecodeBuffer::parseElementName (
    OSUTF8CHAR ** ppName )
```

This method parses the initial tag from an XML message.

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

#### Parameters

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

#### Returns

Completion status of operation:

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

#### 7.2.3.6 `parseElemQName()`

```
EXTXMLMETHOD int OSXMLDecodeBuffer::parseElemQName (  
    OSXMLQName * pQName )
```

This method parses the initial tag from an XML message.

#### Parameters

<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 <code>rtxMemAlloc</code> function.
---------------	---

#### Returns

Completion status of operation:

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

#### 7.2.3.7 `setMaxErrors()`

```
EXTXMLMETHOD OSUINT32 OSXMLDecodeBuffer::setMaxErrors (  
    OSUINT32 maxErrors )
```

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

#### Parameters

<i>maxErrors</i>	The desired number of maximum errors.
------------------	---------------------------------------



## Returns

The previously set maximum number of errors.

## 7.2.4 Member Data Documentation

### 7.2.4.1 mbOwnStream

```
OSBOOL OSXMLDecodeBuffer::mbOwnStream [protected]
```

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

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

Definition at line 57 of file OSXMLDecodeBuffer.h.

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

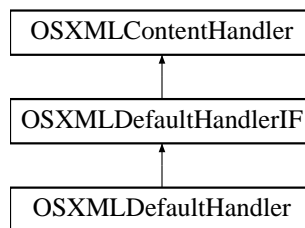
- [OSXMLDecodeBuffer.h](#)

## 7.3 OSXMLDefaultHandler Class Reference

This class is derived from the SAX class DefaultHandler base class.

```
#include <rtSaxCppParser.h>
```

Inheritance diagram for OSXMLDefaultHandler:



## Public Member Functions

- virtual EXTMETHOD int [startElement](#) (const OSUTF8CHAR \*const uri, const OSUTF8CHAR \*const local-name, const OSUTF8CHAR \*const qname, const OSUTF8CHAR \*const \*attrs)  
*Receive notification of the beginning of an element.*
- virtual EXTMETHOD int [characters](#) (const OSUTF8CHAR \*const chars, unsigned int length)  
*Receive notification of character data.*
- virtual EXTMETHOD int [endElement](#) (const OSUTF8CHAR \*const uri, const OSUTF8CHAR \*const local-name, const OSUTF8CHAR \*const qname)  
*Receive notification of the end of an element.*
- OSINT16 [getState](#) ()  
*This method returns the current state of the decoding process.*

### 7.3.1 Detailed Description

This class is derived from the SAX class `DefaultHandler` base class.

It contains variables and methods specific to decoding XML messages. It is used to intercept standard SAX parser events, such as `startElement`, `characters`, `endElement`. This class is used as a base class for `XBinder` generated global element control classes (`<elem>_CC`).

Definition at line 58 of file `rtSaxCppParser.h`.

### 7.3.2 Member Function Documentation

#### 7.3.2.1 `characters()`

```
virtual EXTXMLMETHOD int OSXMLDefaultHandler::characters (
    const OSUTF8CHAR *const chars,
    unsigned int length ) [virtual]
```

Receive notification of character data.

The Parser will call this method to report each chunk of character data. SAX parsers may return all contiguous character data in a single chunk, or they may split it into several chunks; however, all of the characters in any single event must come from the same external entity, so that the Locator provides useful information.

#### Parameters

<i>chars</i>	The characters from the XML document.
<i>length</i>	The length of chars.

Implements [OSXMLContentHandler](#).

#### 7.3.2.2 `endElement()`

```
virtual EXTXMLMETHOD int OSXMLDefaultHandler::endElement (
    const OSUTF8CHAR *const uri,
    const OSUTF8CHAR *const localname,
    const OSUTF8CHAR *const qname ) [virtual]
```

Receive notification of the end of an element.

The SAX parser will invoke this method at the end of every element in the XML document; there will be a corresponding [startElement\(\)](#) event for every [endElement\(\)](#) event (even when the element is empty).

#### Parameters

<i>uri</i>	The URI of the associated namespace for this element
<i>localname</i>	The local part of the element name
<i>qname</i>	The QName of this element

Implements [OSXMLContentHandler](#).

#### 7.3.2.3 getState()

```
OSINT16 OSXMLDefaultHandler::getState ( ) [inline]
```

This method returns the current state of the decoding process.

#### Returns

The state of the decoding process as type OSXMLState. Can be XMLINIT, XMLSTART, XMLDATA, or XMLEND.

Definition at line 124 of file rtSaxCppParser.h.

#### 7.3.2.4 startElement()

```
virtual EXTXMLMETHOD int OSXMLDefaultHandler::startElement (
    const OSUTF8CHAR *const uri,
    const OSUTF8CHAR *const localname,
    const OSUTF8CHAR *const qname,
    const OSUTF8CHAR *const * attrs ) [virtual]
```

Receive notification of the beginning of an element.

The Parser will invoke this method at the beginning of every element in the XML document; there will be a corresponding [endElement\(\)](#) event for every [startElement\(\)](#) event (even when the element is empty). All of the element's content will be reported, in order, before the corresponding [endElement\(\)](#) event.

#### Parameters

<i>uri</i>	The URI of the associated namespace for this element
<i>localname</i>	The local part of the element name
<i>qname</i>	The QName of this element
<i>attrs</i>	The attributes name/value pairs attached to the element, if any.

See also

[endElement](#)

Implements [OSXMLContentHandler](#).

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

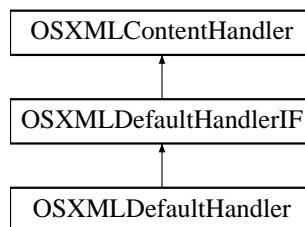
- [rtSaxCppParser.h](#)

## 7.4 OSXMLDefaultHandlerIF Class Reference

This class is derived from the SAX class DefaultHandler base class.

```
#include <rtSaxCppParserIF.h>
```

Inheritance diagram for OSXMLDefaultHandlerIF:



### Additional Inherited Members

#### 7.4.1 Detailed Description

This class is derived from the SAX class DefaultHandler base class.

It contains variables and methods specific to decoding XML messages. It is used to intercept standard SAX parser events, such as startElement, characters, endElement. This class is used as a base class for XBinder generated global element control classes (<elem>\_CC).

Definition at line 260 of file [rtSaxCppParserIF.h](#).

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

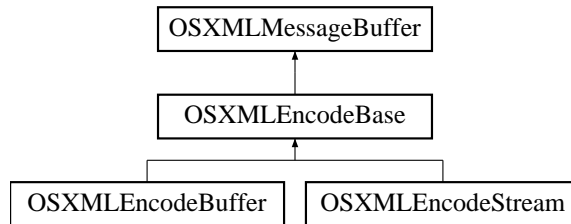
- [rtSaxCppParserIF.h](#)

## 7.5 OSXMLEncodeBase Class Reference

[OSXMLEncodeBase](#) is a base class for the XML encode buffer and stream classes, [OSXMLEncodeBuffer](#) and [OSXMLEncodeStream](#).

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

Inheritance diagram for OSXMLEncodeBase:



### Public Member Functions

- EXTXMLMETHOD int [encodeAttr](#) (const OSUTF8CHAR \*name, const OSUTF8CHAR \*value)  
*This function encodes an attribute in which the name and value are given as null-terminated UTF-8 strings.*
- EXTXMLMETHOD int [encodeText](#) (const OSUTF8CHAR \*value)  
*This method encodes XML textual content.*
- EXTXMLMETHOD int [endDocument](#) ()  
*This method ends an XML document by flushing any remaining data to the stream.*
- EXTXMLMETHOD int [endElement](#) (const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS=0)  
*This method encodes an end element tag value (</elemName>).*
- EXTXMLMETHOD int [startDocument](#) ()  
*This method writes information to start an XML document to the encode stream.*
- EXTXMLMETHOD int [startElement](#) (const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS=0, OSRTDList \*pNSAttrs=0, OSBOOL terminate=FALSE)  
*This method writes information to start an XML element to the encode stream.*
- EXTXMLMETHOD int [termStartElement](#) ()  
*This method terminates a currently open XML start element by adding either a '>' or '>' (if empty) terminator.*

### Protected Member Functions

- EXTXMLMETHOD [OSXMLEncodeBase](#) (OSRTContext \*pContext=0)  
*The protected constructor creates a new context and sets the buffer class type.*

### 7.5.1 Detailed Description

[OSXMLEncodeBase](#) is a base class for the XML encode buffer and stream classes, [OSXMLEncodeBuffer](#) and [OSXMLEncodeStream](#).

Definition at line 153 of file OSXMLMessageBuffer.h.

## 7.5.2 Constructor & Destructor Documentation

### 7.5.2.1 OSXMLEncodeBase()

```
EXTXMLMETHOD OSXMLEncodeBase::OSXMLEncodeBase (
    OSRTContext * pContext = 0 ) [protected]
```

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

#### Parameters

<i>pContext</i>	Pointer to a context to use. If NULL, new context will be allocated.
-----------------	--

## 7.5.3 Member Function Documentation

### 7.5.3.1 encodeAttr()

```
EXTXMLMETHOD int OSXMLEncodeBase::encodeAttr (
    const OSUTF8CHAR * name,
    const OSUTF8CHAR * value )
```

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

#### Parameters

<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.

### 7.5.3.2 encodeText()

```
EXTXMLMETHOD int OSXMLEncodeBase::encodeText (
    const OSUTF8CHAR * value )
```

This method encodes XML textual content.

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

#### Parameters

<i>value</i>	UTF-8 string value to be encoded.
--------------	-----------------------------------

#### Returns

Completion status of operation:

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

#### 7.5.3.3 endElement()

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

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

#### Parameters

<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.

#### 7.5.3.4 startDocument()

```
EXTXMLMETHOD int OSXMLEncodeBase::startDocument ( )
```

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

This includes the XML header declaration.

### 7.5.3.5 startElement()

```
EXTXMLMETHOD int OSXMLEncodeBase::startElement (
    const OSUTF8CHAR * elemName,
    OSXMLNamespace * pNS = 0,
    OSRTDList * pNSAttrs = 0,
    OSBOOL terminate = FALSE )
```

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

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

#### Parameters

<i>elemName</i>	XML element name.
<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.

### 7.5.3.6 termStartElement()

```
EXTXMLMETHOD int OSXMLEncodeBase::termStartElement ( )
```

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

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

#### Returns

Completion status of operation:

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

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

- [OSXMLMessageBuffer.h](#)

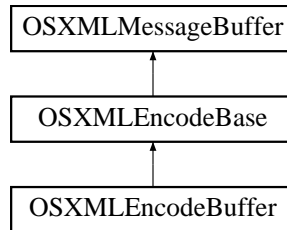


## 7.6 OSXMLEncodeBuffer Class Reference

The `OSXMLEncodeBuffer` class is derived from the `OSXMLEncodeBase` class.

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

Inheritance diagram for `OSXMLEncodeBuffer`:



### Public Member Functions

- EXTXMLMETHOD `OSXMLEncodeBuffer` ()  
*Default constructor.*
- EXTXMLMETHOD `OSXMLEncodeBuffer` (OSOCKET \*pMsgBuf, size\_t msgBufLen)  
*This constructor allows a static message buffer to be specified to receive the encoded message.*
- int `addXMLHeader` (const OSUTF8CHAR \*version=OSUTF8("1.0"), const OSUTF8CHAR \*encoding=OSUTF8(OSXMLHDRUTF8), OSBOOL newLine=TRUE)  
*This method adds XML header text to the output buffer with the given version number and encoding attributes.*
- EXTXMLMETHOD int `addXMLText` (const OSUTF8CHAR \*text)  
*This method adds encoded XML text to the encode buffer.*
- virtual size\_t `getMsgLen` ()  
*This method returns the length of a previously encoded XML message.*
- virtual EXTXMLMETHOD int `init` ()  
*This method reinitializes the encode buffer to allow a new message to be encoded.*
- virtual OSBOOL `isA` (Type bufferType)  
*This is a virtual method that must be overridden by derived classes to allow identification of the class.*
- void `nullTerminate` ()  
*This method adds a null-terminator character ('\0') at the current buffer position.*
- EXTXMLMETHOD void `setFragment` (OSBOOL value=TRUE)  
*This method sets a flag indicating that the data is to be encoded as an XML fragment instead of as a complete XML document (i.e.*
- virtual EXTXMLMETHOD long `write` (const char \*filename)  
*This method writes the encoded message to the given file.*
- virtual EXTXMLMETHOD long `write` (FILE \*fp)  
*This version of the write method writes to a file that is specified by a FILE pointer.*

## Additional Inherited Members

### 7.6.1 Detailed Description

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

It contains variables and methods specific to encoding XML messages. It is used to manage the buffer into which a message is to be encoded.

Definition at line 38 of file OSXMLEncodeBuffer.h.

### 7.6.2 Constructor & Destructor Documentation

#### 7.6.2.1 OSXMLEncodeBuffer()

```
EXTXMLMETHOD OSXMLEncodeBuffer::OSXMLEncodeBuffer (  
    OSOCKET * pMsgBuf,  
    size_t msgBufLen )
```

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

#### Parameters

<i>pMsgBuf</i>	A pointer to a fixed size message buffer to receive the encoded message.
<i>msgBufLen</i>	Size of the fixed-size message buffer.

### 7.6.3 Member Function Documentation

#### 7.6.3.1 addXMLHeader()

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

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

#### Parameters

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

#### Returns

Zero if success or negative error code.

#### 7.6.3.2 addXMLText()

```
EXTXMLMETHOD int OSXMLEncodeBuffer::addXMLText (
    const OSUTF8CHAR * text )
```

This method adds encoded XML text to the encode buffer.

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

#### Parameters

<i>text</i>	Encoded XML text to be added to the buffer.
-------------	---

#### Returns

Zero if success or negative error code.

#### 7.6.3.3 getMsgLen()

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

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

#### Returns

Length of the XML message encapsulated within this buffer object.

Definition at line 90 of file OSXMLEncodeBuffer.h.

#### 7.6.3.4 init()

```
virtual EXTXMLMETHOD int OSXMLEncodeBuffer::init ( ) [virtual]
```

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

This makes it possible to reuse one message buffer object in a loop to encode multiple messages. After this method is called, any previously encoded message in the buffer will be overwritten on the next encode call.

#### 7.6.3.5 isA()

```
virtual OSBOOL OSXMLEncodeBuffer::isA (
    Type bufferType ) [inline], [virtual]
```

This is a virtual method that must be overridden by derived classes to allow identification of the class.

The base class variant is abstract. This method matches an enumerated identifier defined in the base class. One identifier is declared for each of the derived classes.

##### Parameters

<i>bufferType</i>	Enumerated identifier specifying a derived class. This type is defined as a public access type in the OSRTMessageBufferIF base interface. Possible values include BEREncode, BERDecode, PEREncode, PERDecode, XMLEncode, and XMLDecode.
-------------------	---

##### Returns

Boolean result of the match operation. True if the *bufferType* argument is XMLEncode. argument.

Definition at line 118 of file OSXMLEncodeBuffer.h.

#### 7.6.3.6 setFragment()

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

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

an XML header will not be added).

#### 7.6.3.7 write() [1/2]

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

This method writes the encoded message to the given file.

#### Parameters

<i>filename</i>	The name of file to which the encoded message will be written.
-----------------	--

#### Returns

Number of octets actually written. This value may be less than the actual message length if an error occurs.

#### 7.6.3.8 write() [2/2]

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

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

#### Parameters

<i>fp</i>	Pointer to FILE structure to which the encoded message will be written.
-----------	---

#### Returns

Number of octets actually written. This value may be less than the actual message length if an error occurs.

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

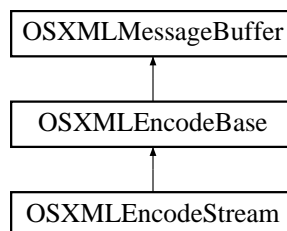
- [OSXMLEncodeBuffer.h](#)

## 7.7 OSXMLEncodeStream Class Reference

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

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

Inheritance diagram for OSXMLEncodeStream:



## Public Member Functions

- EXTXMLMETHOD [OSXMLEncodeStream](#) (OSRTOutputStream &outputStream)  
*This version of the [OSXMLEncodeStream](#) constructor takes a reference to the OSOutputStream object.*
- [OSXMLEncodeStream](#) (OSRTOutputStream \*pOutputStream, OSBOOL ownStream=TRUE)  
*This version of the [OSXMLEncodeStream](#) constructor takes a pointer to the OSRTOutputStream object.*
- virtual EXTXMLMETHOD int [init](#) ()  
*This method reinitializes the encode stream to allow a new message to be encoded.*
- virtual OSBOOL [isA](#) (Type bufferSize)  
*This is a virtual method that must be overridden by derived classes to allow identification of the class.*
- virtual const OSOCTET \* [getMsgPtr](#) ()  
*This is a virtual method that must be overridden by derived classes to allow access to the stored message.*
- OSRTOutputStream \* [getStream](#) () const  
*This method returns the output stream associated with the object.*

## Protected Attributes

- OSRTOutputStream \* [mpStream](#)  
*A pointer to an OSRTOutputStream object.*
- OSBOOL [mbOwnStream](#)  
*TRUE if the [OSXMLEncodeStream](#) object will close and free the stream in the destructor.*

## Additional Inherited Members

### 7.7.1 Detailed Description

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

It contains variables and methods specific to streaming encoding XML messages. It is used to manage the stream into which a message is to be encoded.

Definition at line 40 of file OSXMLEncodeStream.h.

### 7.7.2 Constructor & Destructor Documentation

#### 7.7.2.1 [OSXMLEncodeStream\(\)](#) [1/2]

```
EXTXMLMETHOD OSXMLEncodeStream::OSXMLEncodeStream (  
    OSRTOutputStream & outputStream )
```

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

The stream is assumed to have been previously initialized.

## Parameters

<i>outputStream</i>	reference to the OSOutputStream object
---------------------	--

### 7.7.2.2 OSXMLEncodeStream() [2/2]

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

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

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

## Parameters

<i>pOutputStream</i>	reference to the OSOutputStream object
<i>ownStream</i>	set ownership for the passed stream object.

## 7.7.3 Member Function Documentation

### 7.7.3.1 getMsgPtr()

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

This is a virtual method that must be overridden by derived classes to allow access to the stored message.

The base class implementation returns a null value.

## Returns

A pointer to the stored message.

Definition at line 106 of file OSXMLEncodeStream.h.

### 7.7.3.2 `getStream()`

```
OSRTOutputStream* OSXMLEncodeStream::getStream ( ) const [inline]
```

This method returns the output stream associated with the object.

#### Returns

A pointer to the output stream.

Definition at line 113 of file OSXMLEncodeStream.h.

### 7.7.3.3 `init()`

```
virtual EXTXMLMETHOD int OSXMLEncodeStream::init ( ) [virtual]
```

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

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

### 7.7.3.4 `isA()`

```
virtual OSBOOL OSXMLEncodeStream::isA (
    Type bufferType ) [inline], [virtual]
```

This is a virtual method that must be overridden by derived classes to allow identification of the class.

The base class variant is abstract. This method matches an enumerated identifier defined in the base class. One identifier is declared for each of the derived classes.

#### Parameters

<i>bufferType</i>	Enumerated identifier specifying a derived class. This type is defined as a public access type in the OSRTMessageBufferIF base interface. Possible values include BEREncode, BERDecode, PEREncode, PERDecode, XMLEncode, and XMLDecode.
-------------------	---

#### Returns

Boolean result of the match operation. True if the `bufferType` argument is `XMLEncode` or `Stream`.

Definition at line 95 of file OSXMLEncodeStream.h.

## 7.7.4 Member Data Documentation



#### 7.7.4.1 mbOwnStream

```
OSBOOL OSXMLEncodeStream::mbOwnStream [protected]
```

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

Definition at line 47 of file OSXMLEncodeStream.h.

#### 7.7.4.2 mpStream

```
OSRTOutputStream* OSXMLEncodeStream::mpStream [protected]
```

A pointer to an OSRTOutputStream object.

Definition at line 43 of file OSXMLEncodeStream.h.

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

- [OSXMLEncodeStream.h](#)

## 7.8 OSXMLGroupDesc Struct Reference

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

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

### 7.8.1 Detailed Description

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

Here, "group" means a set of elements, any of which may be matched next. This does not correspond directly to an XSD group.

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

Definition at line 140 of file osrtxml.h.

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

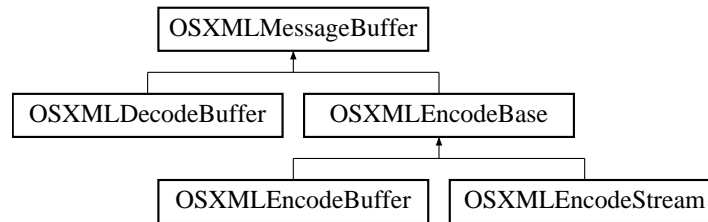
- [osrtxml.h](#)

## 7.9 OSXMLMessageBuffer Class Reference

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

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

Inheritance diagram for OSXMLMessageBuffer:



### Public Member Functions

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

### Protected Member Functions

- EXTXMLMETHOD [OSXMLMessageBuffer](#) (Type bufferType, OSRTContext \*pContext=0)  
*The protected constructor creates a new context and sets the buffer class type.*

## 7.9.1 Detailed Description

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

It is the base class for the `OSXMLEncodeBuffer` and `OSXMLDecodeBuffer` classes. It contains variables and methods specific to encoding or decoding XML messages. It is used to manage the buffer into which a message is to be encoded or decoded.

Definition at line 42 of file `OSXMLMessageBuffer.h`.

## 7.9.2 Constructor & Destructor Documentation

### 7.9.2.1 OSXMLMessageBuffer()

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

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

#### Parameters

<i>bufferType</i>	Type of message buffer that is being created (for example, <code>XMLEncode</code> or <code>XMLDecode</code> ).
<i>pContext</i>	Pointer to a context to use. If <code>NULL</code> , new context will be allocated.

## 7.9.3 Member Function Documentation

### 7.9.3.1 getIndent()

```
EXTXMLMETHOD int OSXMLMessageBuffer::getIndent ( )
```

This method returns current XML output indent value.

#### Returns

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

### 7.9.3.2 getIndentChar()

```
EXTXMLMETHOD int OSXMLMessageBuffer::getIndentChar ( )
```

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

#### Returns

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

### 7.9.3.3 getWriteBOM()

```
EXTXMLMETHOD OSBOOL OSXMLMessageBuffer::getWriteBOM ( )
```

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

#### Returns

TRUE if writing BOM is enabled, FALSE otherwise.

### 7.9.3.4 setAppInfo()

```
virtual EXTMLMETHOD void OSXMLMessageBuffer::setAppInfo (
    void * pXMLInfo ) [virtual]
```

This method sets application specific context information within the common context structure.

For XML encoding/decoding, this is a structure of type *OSXMLCtxtInfo*.

#### Parameters

<i>pXMLInfo</i>	Pointer to context information.
-----------------	---------------------------------

### 7.9.3.5 setFormatting()

```
EXTXMLMETHOD void OSXMLMessageBuffer::setFormatting (
    OSBOOL doFormatting )
```

This method sets XML output formatting to the given value.

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

#### Parameters

<i>doFormatting</i>	Boolean value indicating if formatting is to be done
---------------------	--

#### Returns

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

#### 7.9.3.6 setIndent()

```
EXTXMLMETHOD void OSXMLMessageBuffer::setIndent (
    OSUINT8 indent )
```

This method sets XML output indent to the given value.

#### Parameters

<i>indent</i>	Number of spaces per indent. Default is 3.
---------------	--

#### 7.9.3.7 setIndentChar()

```
EXTXMLMETHOD void OSXMLMessageBuffer::setIndentChar (
    char indentChar )
```

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

#### Parameters

<i>indentChar</i>	Indent character. Default is space.
-------------------	-------------------------------------

#### 7.9.3.8 setNamespace()

```
virtual EXTMLMETHOD void OSXMLMessageBuffer::setNamespace (
```

```

const OSUTF8CHAR * prefix,
const OSUTF8CHAR * uri,
OSRTDList * pNSAttrs = 0 ) [virtual]

```

This method sets a namespace in the context namespace list.

If the given namespace URI does not exist in the list, the namespace is added. If the URI is found, the value of the namespace prefix will be changed to the given prefix.

#### Parameters

<i>prefix</i>	Namespace prefix
<i>uri</i>	Namespace URI
<i>pNSAttrs</i>	Namespace list to which namespace is to be added

#### 7.9.3.9 setWriteBOM()

```

EXTXMLMETHOD void OSXMLMessageBuffer::setWriteBOM (
    OSBOOL write )

```

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

#### Parameters

<i>write</i>	TRUE if BOM should be written, FALSE otherwise.
--------------	---

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

- [OSXMLMessageBuffer.h](#)

## 7.10 OSXMLNamespaceClass Class Reference

This class is used to hold an XML namespace prefix to URI mapping.

```
#include <rtXmlCppNamespace.h>
```

Inherits OSXMLNamespace.

## Public Member Functions

- [OSXMLNamespaceClass](#) ()  
*The default constructor sets the namespace prefix and URI values to empty values.*
- [~OSXMLNamespaceClass](#) ()  
*The destructor deletes the prefix and uri string variables.*
- [OSXMLNamespaceClass](#) (const OSUTF8CHAR \*nsPrefix, const OSUTF8CHAR \*nsURI)  
*The parameterized constructor sets the namespace prefix and URI values to the given values.*
- [OSXMLNamespaceClass](#) (const OSUTF8CHAR \*nsPrefix, size\_t nsPrefixBytes, const OSUTF8CHAR \*nsURI, size\_t nsURIBytes)  
*The parameterized constructor sets the namespace prefix and URI values to the given values.*
- [OSXMLNamespaceClass](#) (const [OSXMLNamespaceClass](#) &o)  
*The copy constructor make a deep-copy of the prefix and URI values.*
- const OSUTF8CHAR \* [getPrefix](#) () const  
*This method is used to get the namespace prefix value.*
- const OSUTF8CHAR \* [getURI](#) () const  
*This method is used to get the namespace URI value.*
- void [setPrefix](#) (const OSUTF8CHAR \*nsPrefix)  
*This method is used to set the namespace prefix value.*
- void [setURI](#) (const OSUTF8CHAR \*nsURI)  
*This method is used to set the namespace URI value.*

### 7.10.1 Detailed Description

This class is used to hold an XML namespace prefix to URI mapping.

Definition at line 38 of file rtXmlCppNamespace.h.

### 7.10.2 Constructor & Destructor Documentation

#### 7.10.2.1 OSXMLNamespaceClass() [1/2]

```
OSXMLNamespaceClass::OSXMLNamespaceClass (  
    const OSUTF8CHAR * nsPrefix,  
    const OSUTF8CHAR * nsURI )
```

The parameterized constructor sets the namespace prefix and URI values to the given values.

A deep copy of the values is done.

#### Parameters

<i>nsPrefix</i>	Namespace prefix value.
<i>nsURI</i>	Namespace URI value.

### 7.10.2.2 OSXMLNamespaceClass() [2/2]

```
OSXMLNamespaceClass::OSXMLNamespaceClass (
    const OSUTF8CHAR * nsPrefix,
    size_t nsPrefixBytes,
    const OSUTF8CHAR * nsURI,
    size_t nsURIBytes )
```

The parameterized constructor sets the namespace prefix and URI values to the given values.

A deep copy of the values is done.

#### Parameters

<i>nsPrefix</i>	Namespace prefix value.
<i>nsPrefixBytes</i>	Namespace prefix value size in bytes.
<i>nsURI</i>	Namespace URI value.
<i>nsURIBytes</i>	Namespace URI value size in bytes.

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

- [rtXmlCppNamespace.h](#)

## 7.11 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.*

#### 7.11.1 Detailed Description

Class enabling parsing of an XML Schema list into strings.

Definition at line 47 of file `rtXmlpCppDecFuncs.h`.



## 7.11.2 Constructor & Destructor Documentation

### 7.11.2.1 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.
--------------	--

## 7.11.3 Member Function Documentation

### 7.11.3.1 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:

- [rtXmIpcppDecFuncs.h](#)

## 7.12 OSXMLStrListHandler Class Reference

[OSXMLStrListHandler](#).

```
#include <rtSaxCppStrList.h>
```

### 7.12.1 Detailed Description

[OSXMLStrListHandler](#).

Definition at line 41 of file `rtSaxCppStrList.h`.

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

- [rtSaxCppStrList.h](#)

## 7.13 OSXSDGlobalElement Class Reference

XSD global element base class.

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

### Public Member Functions

- [OSXSDGlobalElement](#) (OSRTMessageBufferIF &msgBuf)  
*This constructor sets the internal message buffer pointer to point at the given message buffer or stream object.*
- [OSXSDGlobalElement](#) (const [OSXSDGlobalElement](#) &o)  
*The copy constructor sets the internal message buffer pointer and context to point at the message buffer and context from the original OSCType object.*
- virtual [~OSXSDGlobalElement](#) ()  
*The virtual destructor does nothing.*
- int [decode](#) ()  
*The decode method decodes the message described by the encapsulated message buffer object.*
- virtual int [decodeFrom](#) (OSRTMessageBufferIF &)  
*The decodeFrom method decodes a message from the given message buffer or stream argument.*
- int [encode](#) ()  
*The encode method encodes a message using the encoding rules specified by the derived message buffer object.*
- virtual int [encodeTo](#) (OSRTMessageBufferIF &)  
*The encodeTo method encodes a message into the given message buffer or stream argument.*
- OSCTXT \* [getCtxtPtr](#) ()  
*The getCtxtPtr method returns the underlying C runtime context.*
- void \* [memAlloc](#) (size\_t numocts)  
*The memAlloc method allocates memory using the C runtime memory management functions.*
- void [memFreePtr](#) (void \*ptr)

- The memFreePtr method frees the memory at a specific location.*

  - void [setDefaultNamespace](#) (const OSUTF8CHAR \*uri)
 

*The setDefaultNamespace method sets the default namespace for the element to the given value.*
  - void [setDiag](#) (OSBOOL value=TRUE)
 

*The setDiag method turns diagnostic tracing on or off.*
  - void [setEncXSINamespace](#) (OSBOOL value=TRUE)
 

*The setEncXSINamespace method sets a flag in the internal context that indicates the xsi namespace attribute must be encoded.*
  - void [setNamespace](#) (const OSUTF8CHAR \*prefix, const OSUTF8CHAR \*uri)
 

*The setNamespace method adds or modifies the namespace with the given URI in the namespace list to contain the given prefix.*
  - void [setNoNSSchemaLocation](#) (const OSUTF8CHAR \*uri)
 

*The setNoNSSchemaLocation method adds an xsi:noNamespaceSchemaLocation attribute to the document.*
  - void [setSchemaLocation](#) (const OSUTF8CHAR \*uri)
 

*The setSchemaLocation method adds an xsi:schemaLocation attribute to the document.*
  - void [setXSISchemaType](#) (const OSUTF8CHAR \*typeName)
 

*The setXSISchemaType method sets a type name to be used in the xsi:type attribute in the top-level module element declaration.*
  - int [validate](#) ()
 

*The validate method validates the message described by the encapsulated message buffer object.*
  - virtual int [validateFrom](#) (OSRTMessageBufferIF &)
 

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

## Protected Member Functions

- [OSXSDGlobalElement](#) ()
 

*The default constructor sets the message pointer member variable to NULL and creates a new context object.*
- [OSXSDGlobalElement](#) (OSRTContext &ctxt)
 

*This constructor sets the message pointer member variable to NULL and initializes the context object to point at the given context value.*
- void [setMsgBuf](#) (OSRTMessageBufferIF &msgBuf)
 

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

## Protected Attributes

- OSRTCtxtPtr [mpContext](#)

*The mpContext member variable holds a reference-counted C runtime variable.*
- OSRTMessageBufferIF \* [mpMsgBuf](#)

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

### 7.13.1 Detailed Description

XSD global element base class.

This is the main base class for all generated global element control classes. It holds a variable of a generated data type as well as the associated message buffer or stream class to which a message will be encoded or from which a message will be decoded.

Definition at line 57 of file rtXmlCppXSDElement.h.

## 7.13.2 Constructor & Destructor Documentation

### 7.13.2.1 OSXSDGlobalElement() [1/3]

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

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

#### Parameters

<i>ctxt</i>	- Reference to a context object.
-------------	----------------------------------

Definition at line 85 of file rtXmlCppXSDElement.h.

### 7.13.2.2 OSXSDGlobalElement() [2/3]

```
OSXSDGlobalElement::OSXSDGlobalElement (
    OSRTMessageBufferIF & msgBuf ) [inline]
```

This constructor sets the internal message buffer pointer to point at the given message buffer or stream object.

The context is set to point at the context contained within the message buffer object. Thus, the message buffer and control class object share the context. It will not be released until both objects are destroyed.

#### Parameters

<i>msgBuf</i>	- Reference to a message buffer or stream object.
---------------	---

Definition at line 105 of file rtXmlCppXSDElement.h.

### 7.13.2.3 OSXSDGlobalElement() [3/3]

```
OSXSDGlobalElement::OSXSDGlobalElement (
    const OSXSDGlobalElement & o ) [inline]
```

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

## Parameters

<i>o</i>	- Reference to a global element object.
----------	---

Definition at line 116 of file rtXmlCppXSDElement.h.

### 7.13.2.4 ~OSXSDGlobalElement()

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

The virtual destructor does nothing.

It is overridden by derived versions of this class.

Definition at line 123 of file rtXmlCppXSDElement.h.

## 7.13.3 Member Function Documentation

### 7.13.3.1 decodeFrom()

```
virtual int OSXSDGlobalElement::decodeFrom (
    OSRTMessageBufferIF & ) [inline], [virtual]
```

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

## Parameters

-	Message buffer or stream containing message to decode.
---	--

Definition at line 138 of file rtXmlCppXSDElement.h.

### 7.13.3.2 encodeTo()

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

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

## Parameters

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

Definition at line 153 of file rtXmlCppXSDElement.h.

### 7.13.3.3 getCtxPtr()

```
OSCTXT* OSXSDGlobalElement::getCtxPtr ( ) [inline]
```

The getCtxPtr method returns the underlying C runtime context.

This context can be used in calls to C runtime functions.

Definition at line 159 of file rtXmlCppXSDElement.h.

### 7.13.3.4 memAlloc()

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

The memAlloc method allocates memory using the C runtime memory management functions.

The memory is tracked in the underlying context structure. When both this [OSXSDGlobalElement](#) derived control class object and the message buffer object are destroyed, this memory will be freed.

## Parameters

<i>numocts</i>	- Number of bytes of memory to allocate
----------------	---

Definition at line 172 of file rtXmlCppXSDElement.h.

### 7.13.3.5 memFreePtr()

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

The memFreePtr method frees the memory at a specific location.

This memory must have been allocated using the memAlloc method described earlier.

#### Parameters

<i>ptr</i>	- Pointer to a block of memory allocated with <code>memAlloc</code>
------------	---

Definition at line 200 of file `rtXmlCppXSDElement.h`.

#### 7.13.3.6 `setDefaultNamespace()`

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

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

#### Parameters

<i>uri</i>	- Default namespace URI
------------	-------------------------

Definition at line 210 of file `rtXmlCppXSDElement.h`.

#### 7.13.3.7 `setDiag()`

```
void OSXSDGlobalElement::setDiag (
    OSBOOL value = TRUE ) [inline]
```

The `setDiag` method turns diagnostic tracing on or off.

#### Parameters

<i>value</i>	- Boolean on/off value (default = on)
--------------	---------------------------------------

Definition at line 219 of file `rtXmlCppXSDElement.h`.

#### 7.13.3.8 `setEncXSINamespace()`

```
void OSXSDGlobalElement::setEncXSINamespace (
    OSBOOL value = TRUE ) [inline]
```

The `setEncXSINamespace` method sets a flag in the internal context that indicates the xsi namespace attribute must be encoded.

#### Parameters

<i>value</i>	- Boolean on/off value (default = on)
--------------	---------------------------------------

Definition at line 229 of file rtXmlCppXSDElement.h.

References rtXmlSetEncXSINamespace().

#### 7.13.3.9 setMsgBuf()

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

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

#### Parameters

<i>msgBuf</i>	- Reference to a message buffer or stream object.
---------------	---

#### 7.13.3.10 setNamespace()

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

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

#### Parameters

<i>prefix</i>	- Namespace prefix
<i>uri</i>	- Namespace URI

Definition at line 240 of file rtXmlCppXSDElement.h.

#### 7.13.3.11 setNoNSSchemaLocation()

```
void OSXSDGlobalElement::setNoNSSchemaLocation (
    const OSUTF8CHAR * uri ) [inline]
```



The setNoNSSchemaLocation method adds an xsi:noNamespaceSchemaLocation attribute to the document.

**Parameters**

<i>uri</i>	- URI for noNamespaceSchemaLocation.
------------	--------------------------------------

Definition at line 250 of file rtXmlCppXSDElement.h.

References rtXmlSetNoNSSchemaLocation().

**7.13.3.12 setSchemaLocation()**

```
void OSXSDGlobalElement::setSchemaLocation (
    const OSUTF8CHAR * uri ) [inline]
```

The setSchemaLocation method adds an xsi:schemaLocation attribute to the document.

**Parameters**

<i>uri</i>	- URI for schemaLocation.
------------	---------------------------

Definition at line 260 of file rtXmlCppXSDElement.h.

References rtXmlSetSchemaLocation().

**7.13.3.13 setXSIType()**

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

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

**Parameters**

<i>typeName</i>	- XSI type name
-----------------	-----------------

Definition at line 270 of file rtXmlCppXSDElement.h.

References rtXmlSetXSITypeAttr().

#### 7.13.3.14 validateFrom()

```
virtual int OSXSDGlobalElement::validateFrom (
    OSRTMessageBufferIF & ) [inline], [virtual]
```

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

##### Parameters

-	Message buffer or stream containing message to validate.
---	--

Definition at line 287 of file `rtXmlCppXSDElement.h`.

### 7.13.4 Member Data Documentation

#### 7.13.4.1 mpContext

```
OSRTCtxtPtr OSXSDGlobalElement::mpContext [protected]
```

The `mpContext` member variable holds a reference-counted C runtime variable.

This context is used in calls to all C run-time functions. The context pointed at by this smart-pointer object is shared with the message buffer object contained within this class.

Definition at line 65 of file `rtXmlCppXSDElement.h`.

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

- [rtXmlCppXSDElement.h](#)

## Chapter 8

# File Documentation

### 8.1 osrtxml.h File Reference

XML low-level C encode/decode functions.

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

#### Classes

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

#### 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.*

#### 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 it 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 \*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 \*pctx, OSDynOctStr \*pvalue)
 

*This function decodes the 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 \*pnocets, OSINT32 bufsize)
 

*This function decodes the contents of a hexBinary string into a static memory structure.*
  - int `rtXmlDecHexStr64` (OSCTXT \*pctx, 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 \*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 `rtXmlDecInt` (OSCTXT \*pctx, OSINT32 \*pvalue)
 

*This function decodes the contents of a 32-bit integer data type.*
  - int `rtXmlDecInt8` (OSCTXT \*pctx, OSINT8 \*pvalue)
 

*This function decodes the contents of an 8-bit integer data type (i.e.*
  - int `rtXmlDecInt16` (OSCTXT \*pctx, OSINT16 \*pvalue)
 

*This function decodes the contents of a 16-bit integer data type.*
  - int `rtXmlDecInt64` (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 \*pctx, OSUINT64 \*pvalue)
 

*This function decodes the contents of an unsigned 64-bit integer data type.*
  - int `rtXmlDecNSAttr` (OSCTXT \*pctx, 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, OSXML↵  
Namespace \*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 rtXmlEncBase64StringValue` (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, OSXML↵  
Namespace \*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 \*elem↵  
Name, 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 O↵  
SOCTET \*extValue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the ASN.1 BIT STRING type.*
- `int rtXmlEncBinStringValue` (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 \*p↵  
NS)  
*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 \*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).*
- 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 [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.*
- 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.*
- 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 \*pnocets, 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 \*pnocets, 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 \*pnocts, OSSIZE bufsize)  
*This function decodes the contents of a hexBinary string into a static memory structure.*
- int [rtXmlpDecHexStr64](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocts, OSSIZE bufsize)  
*This function is identical to rtXmlpDecHexStr except that it supports lengths up to 64-bits in size on 64-bit machines.*
- 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 OSXML↵AttrDescr 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 \*pctxt, const OSXMLElemIDRec \*tab, OSSIZE nrows, OSINT32 level, O↵SBOOL 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 curlID, 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 \*pctxt, 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 [rtXmIplGetNextSeqElemIDExt](#) (OSCTXT \*pctxt, 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 rtXmIplGetNextSeqElemID.*
- int [rtXmIplGetNextAllElemID](#) (OSCTXT \*pctxt, const OSXMLElemIDRec \*tab, OSSIZE nRows, const OSUINT8 \*pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)
 

*This function parses the next start tag and finds index of element name in descriptor table.*
- int [rtXmIplGetNextAllElemID16](#) (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.*
- int [rtXmIplGetNextAllElemID32](#) (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.*
- void [rtXmIplSetNamespaceTable](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*namespaceTable[], OSSIZE nNamespaces)
 

*Sets user namespace table.*
- int [rtXmIplCreateReader](#) (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 xsiTypeldx, 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.*

### 8.1.1 Detailed Description

XML low-level C encode/decode functions.

### 8.1.2 Typedef Documentation



### 8.1.2.1 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.

## 8.1.3 Function Documentation

### 8.1.3.1 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.

### 8.1.3.2 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.

If 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> .

#### 8.1.3.3 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> .

#### 8.1.3.4 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.

#### 8.1.3.5 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.

#### 8.1.3.6 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.

### 8.1.3.7 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.

### 8.1.3.8 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.

### 8.1.3.9 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.

### 8.1.3.10 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.

### 8.1.3.11 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.

### 8.1.3.12 rtXmlInitContext()

```
int rtXmlInitContext (
    OSCTXT * pctxt )
```

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

#### Parameters

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

### 8.1.3.13 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.

**8.1.3.14 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
--------------	-----------------------------

**8.1.3.15 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.

#### 8.1.3.16 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.

#### 8.1.3.17 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.

### 8.1.3.18 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.

### 8.1.3.19 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.

### 8.1.3.20 rtXmlMatchGMonthDay()

```
int rtXmlMatchGMonthDay (
    OSCTXT * pctxt )
```

This function tests the context buffer for containing a correct gMonthDay 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.

### 8.1.3.21 rtXmlMatchGYear()

```
int rtXmlMatchGYear (
    OSCTXT * pctxt )
```

This function tests the context buffer for containing a correct gYear 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.

### 8.1.3.22 rtXmlMatchGYearMonth()

```
int rtXmlMatchGYearMonth (
    OSCTXT * pctxt )
```

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.

### 8.1.3.23 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.

### 8.1.3.24 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.

### 8.1.3.25 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.

### 8.1.3.26 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.

### 8.1.3.27 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.

### 8.1.3.28 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.

### 8.1.3.29 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.

#### 8.1.3.30 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.

#### 8.1.3.31 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.

Referenced by OSXSDGlobalElement::setEncXSINamespace().

### 8.1.3.32 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.

### 8.1.3.33 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.

### 8.1.3.34 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.

### 8.1.3.35 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.



### 8.1.3.36 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.

### 8.1.3.37 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.

Referenced by OSXSDGlobalElement::setNoNSSchemaLocation().

### 8.1.3.38 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.

#### 8.1.3.39 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.

Referenced by OSXSDGlobalElement::setSchemaLocation().

#### 8.1.3.40 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.)

#### 8.1.3.41 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.

#### 8.1.3.42 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.

Referenced by OSXSDGlobalElement::setXSIType().

## 8.2 OSXMLDecodeBuffer.h File Reference

XML decode buffer or stream class definition.

```
#include "rtxsrc/OSRTInputStream.h"
#include "rtxmlsrc/OSXMLMessageBuffer.h"
#include "rtxmlsrc/rtSaxCppParserIF.h"
```

## Classes

- class [OSXMLDecodeBuffer](#)

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

### 8.2.1 Detailed Description

XML decode buffer or stream class definition.

## 8.3 OSXMLEncodeBuffer.h File Reference

XML encode message buffer class definition.

```
#include "rtxmlsrc/OSXMLMessageBuffer.h"
```

## Classes

- class [OSXMLEncodeBuffer](#)

*The [OSXMLEncodeBuffer](#) class is derived from the [OSXMLEncodeBase](#) class.*

### 8.3.1 Detailed Description

XML encode message buffer class definition.

## 8.4 OSXMLEncodeStream.h File Reference

XML encode stream class definition.

```
#include "rtxsrc/OSRTOutputStream.h"
#include "rtxmlsrc/OSXMLMessageBuffer.h"
```

## Classes

- class [OSXMLEncodeStream](#)

*The [OSXMLEncodeStream](#) class is derived from the [OSXMLEncodeBase](#) class.*

## 8.4.1 Detailed Description

XML encode stream class definition.

## 8.5 OSXMLMessageBuffer.h File Reference

XML encode/decode buffer and stream base class.

```
#include "rtxsrc/OSRTMsgBuf.h"
#include "rtxmlsrc/osrtxml.h"
```

### Classes

- class [OSXMLMessageBuffer](#)  
*The XML message buffer class is derived from the OSMessageBuffer base class.*
- class [OSXMLEncodeBase](#)  
*OSXMLEncodeBase is a base class for the XML encode buffer and stream classes, OSXMLEncodeBuffer and OSXML↔EncodeStream.*

## 8.5.1 Detailed Description

XML encode/decode buffer and stream base class.

## 8.6 rtSaxCppAny.h File Reference

```
#include "rtxsrc/OSRTContext.h"
#include "rtxmlsrc/osrtxml.h"
#include "rtxmlsrc/rtSaxCppParser.h"
#include "rtxmlsrc/rtXmlCppMsgBuf.h"
#include "rtxsrc/rtxCppXmlString.h"
#include "rtxmlsrc/OSXSDAnyTypeClass.h"
#include "rtxmlsrc/rtSaxCppAnyType.h"
```

## 8.7 rtSaxCppAnyType.h File Reference

```
#include "rtxsrc/OSRTContext.h"
#include "rtxmlsrc/osrtxml.h"
#include "rtxmlsrc/rtSaxCppParser.h"
#include "rtxmlsrc/rtXmlCppMsgBuf.h"
#include "rtxsrc/rtxCppXmlString.h"
#include "rtxmlsrc/OSXSDAnyTypeClass.h"
```

## 8.8 rtSaxCppSimpleType.h File Reference

```
#include "rtxmlsrc/osrtxml.h"  
#include "rtxmlsrc/rtSaxCppParser.h"
```

## 8.9 rtSaxCppSoap.h File Reference

```
#include "rtxsrc/rtxCppDynOctStr.h"  
#include "rtxsrc/OSRTContext.h"  
#include "rtxsrc/OSRTMemBuf.h"  
#include "rtxsrc/rtxCppXmlString.h"  
#include "rtxmlsrc/osrtxml.h"  
#include "rtxmlsrc/rtSaxCppParser.h"  
#include "rtxmlsrc/rtXmlCppMsgBuf.h"
```

## 8.10 rtSaxCppStrList.h File Reference

```
#include "rtxsrc/rtxCppDList.h"  
#include "rtxmlsrc/osrtxml.h"  
#include "rtxmlsrc/rtSaxCppParser.h"
```

### Classes

- class [OSXMLStrListHandler](#)  
*OSXMLStrListHandler.*

## 8.11 rtXmlCppEncFuncs.h File Reference

XML low-level C++ encode functions.

```
#include "rtxmlsrc/osrtxml.h"  
#include "rtxsrc/rtxCppDList.h"
```

## Functions

- int `rtXmlCppEncAnyAttr` (OSCTXT \*pctxt, OSRTObjListClass \*pAnyAttrList)  
*This function encodes a variable of the XSD any attribute type.*
- int `rtXmlEncAny` (OSCTXT \*pctxt, OSRTXMLString \*pxmlstr, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD any type.*
- int `rtXmlCppEncAnyTypeValue` (OSCTXT \*pctxt, OSXSDAnyTypeClass \*pvalue)  
*This function encodes a variable of the XSD anyType type.*
- int `rtXmlCppEncStartElement` (OSCTXT \*pctxt, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS, OSRTDListClass \*pNSAttrs, OSBOOL terminate)  
*This function encodes a start element tag value (<elemName>).*
- int `rtXmlEncString` (OSCTXT \*pctxt, OSRTXMLString \*pxmlstr, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD string type.*

### 8.11.1 Detailed Description

XML low-level C++ encode functions.

These are overloaded versions of C XML encode functions for use with C++.

### 8.11.2 Function Documentation

#### 8.11.2.1 `rtXmlCppEncAnyAttr()`

```
int rtXmlCppEncAnyAttr (  
    OSCTXT * pctxt,  
    OSRTObjListClass * pAnyAttrList )
```

This function encodes a variable of the XSD any attribute type.

This is expressed as list of name/value pairs.

#### Parameters

<code>pctxt</code>	Pointer to context block structure.
<code>pAnyAttrList</code>	List of name/value pair objects.

#### Returns

Completion status of operation:

- 0 = success,

- negative return value is error.

### 8.11.2.2 rtXmlCppEncAnyTypeValue()

```
int rtXmlCppEncAnyTypeValue (
    OSCTXT * pctxt,
    OSXSAnyTypeClass * pvalue )
```

This function encodes a variable of the XSD anyType type.

This is considered to be a fully-wrapped element of anyType 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 pointer to a OSXSAnyTypeClass containing the fully-encoded XML text to be copied to the output stream.

#### Returns

Completion status of operation:

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

### 8.11.2.3 rtXmlCppEncStartElement()

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

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

This function is for use with C++ which uses OSRTDListClass instead of OSRTList (as in the C version of this function).

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>elemName</i>	XML element name.
<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.

### 8.11.2.4 rtXmlEncAny()

```
int rtXmlEncAny (
    OSCTXT * pctxt,
    OSRTXMLString * pxmlstr,
    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>pxmlstr</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.

### 8.11.2.5 rtXmlEncString()

```
int rtXmlEncString (
    OSCTXT * pctxt,
    OSRTXMLString * 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. 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.

## 8.12 rtXmlCppMsgBuf.h File Reference

This file is deprecated.

```
#include "rtxmlsrc/OSXMLEncodeBuffer.h"
#include "rtxmlsrc/OSXMLEncodeStream.h"
#include "rtxmlsrc/OSXMLDecodeBuffer.h"
```

### 8.12.1 Detailed Description

This file is deprecated.

Users should use one or more of the individual headers files defined in the #include statements below.

## 8.13 rtXmlCppNamespace.h File Reference

XML namespace handling structures and function definitions.

```
#include "rtxmlsrc/osrtxml.h"
#include "rtxsrc/OSRTBaseType.h"
#include "rtxsrc/OSRTString.h"
```

## Classes

- class [OSXMLNamespaceClass](#)

*This class is used to hold an XML namespace prefix to URI mapping.*

### 8.13.1 Detailed Description

XML namespace handling structures and function definitions.

## 8.14 rtXmlCppXSDElement.h File Reference

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

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

### Classes

- class [OSXSDGlobalElement](#)  
*XSD global element base class.*

### 8.14.1 Detailed Description

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

## 8.15 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.*

### 8.15.1 Detailed Description

XML low-level C++ decode functions.

These are overloaded versions of C XML encode functions for use with C++.



# Index

- ~OSXSDGlobalElement
  - OSXSDGlobalElement, [185](#)
- addXMLHeader
  - OSXMLEncodeBuffer, [166](#)
- addXMLText
  - OSXMLEncodeBuffer, [167](#)
- characters
  - OSXMLContentHandler, [150](#)
  - OSXMLDefaultHandler, [158](#)
- decodeFrom
  - OSXSDGlobalElement, [185](#)
- decodeXML
  - OSXMLDecodeBuffer, [154](#)
- encodeAttr
  - OSXMLEncodeBase, [162](#)
- encodeText
  - OSXMLEncodeBase, [162](#)
- encodeTo
  - OSXSDGlobalElement, [185](#)
- endElement
  - OSXMLContentHandler, [150](#)
  - OSXMLDefaultHandler, [158](#)
  - OSXMLEncodeBase, [163](#)
- getCtxtPtr
  - OSXSDGlobalElement, [186](#)
- getIndent
  - OSXMLMessageBuffer, [175](#)
- getIndentChar
  - OSXMLMessageBuffer, [175](#)
- getMsgLen
  - OSXMLEncodeBuffer, [167](#)
- getMsgPtr
  - OSXMLEncodeStream, [171](#)
- getState
  - OSXMLDefaultHandler, [159](#)
- getStream
  - OSXMLEncodeStream, [171](#)
- getWriteBOM
  - OSXMLMessageBuffer, [176](#)
- init
  - OSXMLDecodeBuffer, [154](#)
  - OSXMLEncodeBuffer, [167](#)
  - OSXMLEncodeStream, [172](#)
- isWellFormed
  - OSXMLDecodeBuffer, [155](#)
- isA
  - OSXMLDecodeBuffer, [154](#)
  - OSXMLEncodeBuffer, [168](#)
  - OSXMLEncodeStream, [172](#)
- mbOwnStream
  - OSXMLDecodeBuffer, [157](#)
  - OSXMLEncodeStream, [172](#)
- memAlloc
  - OSXSDGlobalElement, [186](#)
- memFreePtr
  - OSXSDGlobalElement, [186](#)
- mpContext
  - OSXSDGlobalElement, [190](#)
- mpStream
  - OSXMLEncodeStream, [173](#)
- next
  - OSXMLStringListParser, [181](#)
- OSXMLContentHandler, [149](#)
  - characters, [150](#)
  - endElement, [150](#)
  - startElement, [150](#)
- OSXMLDecodeBuffer, [151](#)
  - decodeXML, [154](#)
  - init, [154](#)
  - isWellFormed, [155](#)
  - isA, [154](#)
  - mbOwnStream, [157](#)
  - OSXMLDecodeBuffer, [153](#)
  - parseElemQName, [156](#)
  - parseElementName, [155](#)
  - setMaxErrors, [156](#)
- OSXMLDecodeBuffer.h, [223](#)
- OSXMLDefaultHandler, [157](#)
  - characters, [158](#)
  - endElement, [158](#)
  - getState, [159](#)
  - startElement, [159](#)
- OSXMLDefaultHandlerIF, [160](#)
- OSXMLEncodeBase, [161](#)

- encodeAttr, 162
- encodeText, 162
- endElement, 163
- OSXMLEncodeBase, 162
- startDocument, 163
- startElement, 163
- termStartElement, 164
- OSXMLEncodeBuffer, 165
  - addXMLHeader, 166
  - addXMLText, 167
  - getMsgLen, 167
  - init, 167
  - isA, 168
  - OSXMLEncodeBuffer, 166
  - setFragment, 168
  - write, 168, 169
- OSXMLEncodeBuffer.h, 224
- OSXMLEncodeStream, 169
  - getMsgPtr, 171
  - getStream, 171
  - init, 172
  - isA, 172
  - mbOwnStream, 172
  - mpStream, 173
  - OSXMLEncodeStream, 170, 171
- OSXMLEncodeStream.h, 224
- OSXMLGroupDesc, 173
  - osrtxml.h, 204
- OSXMLMessageBuffer, 174
  - getIndent, 175
  - getIndentChar, 175
  - getWriteBOM, 176
  - OSXMLMessageBuffer, 175
  - setAppInfo, 176
  - setFormatting, 176
  - setIndent, 177
  - setIndentChar, 177
  - setNamespace, 177
  - setWriteBOM, 178
- OSXMLMessageBuffer.h, 225
- OSXMLNamespaceClass, 178
  - OSXMLNamespaceClass, 179, 180
- OSXMLStrListHandler, 182
- OSXMLStringListParser, 180
  - next, 181
  - OSXMLStringListParser, 181
- OSXSDGlobalElement, 182
  - ~OSXSDGlobalElement, 185
  - decodeFrom, 185
  - encodeTo, 185
  - getCtxtPtr, 186
  - memAlloc, 186
  - memFreePtr, 186
  - mpContext, 190
  - OSXSDGlobalElement, 184
  - setDefaultNamespace, 187
  - setDiag, 187
  - setEncXSINamespace, 187
  - setMsgBuf, 188
  - setNamespace, 188
  - setNoNSSchemaLocation, 188
  - setSchemaLocation, 189
  - setXSIType, 189
  - validateFrom, 189
- osrtxml.h, 191
  - OSXMLGroupDesc, 204
  - rtSaxGetAttrValue, 205
  - rtSaxGetElemID8, 206
  - rtSaxGetElemID, 205
  - rtSaxHasXMLNSAttrs, 207
  - rtSaxIsEmptyBuffer, 207
  - rtSaxSortAttrs, 207
  - rtSaxStrListMatch, 208
  - rtSaxStrListParse, 208
  - rtXmlCmpBase64Str, 209
  - rtXmlCmpHexStr, 209
  - rtXmlCreateFileInputSource, 210
  - rtXmlInitContext, 210
  - rtXmlInitContextUsingKey, 210
  - rtXmlInitCtxtAppInfo, 211
  - rtXmlMatchBase64Str, 211
  - rtXmlMatchDate, 212
  - rtXmlMatchDateTime, 212
  - rtXmlMatchGDay, 213
  - rtXmlMatchGMonth, 213
  - rtXmlMatchGMonthDay, 213
  - rtXmlMatchGYear, 214
  - rtXmlMatchGYearMonth, 214
  - rtXmlMatchHexStr, 215
  - rtXmlMatchTime, 215
  - rtXmlMemFreeAnyAttrs, 216
  - rtXmlNewQName, 216
  - rtXmlPrepareContext, 217
  - rtXmlSetEncC14N, 217
  - rtXmlSetEncDocHdr, 217
  - rtXmlSetEncXSINamespace, 218
  - rtXmlSetEncXSINilAttr, 219
  - rtXmlSetEncodingStr, 218
  - rtXmlSetFormatting, 219
  - rtXmlSetIndent, 220
  - rtXmlSetIndentChar, 220
  - rtXmlSetNSPrefixLinks, 221
  - rtXmlSetNamespacesSet, 220
  - rtXmlSetNoNSSchemaLocation, 221
  - rtXmlSetSchemaLocation, 222
  - rtXmlSetSoapVersion, 222
  - rtXmlSetWriteBOM, 223
  - rtXmlSetXSITypeAttr, 223

parseElemQName  
     OSXMLDecodeBuffer, 156  
 parseElementName  
     OSXMLDecodeBuffer, 155  
  
 rtSaxCppAny.h, 225  
 rtSaxCppAnyType.h, 225  
 rtSaxCppSimpleType.h, 226  
 rtSaxCppSoap.h, 226  
 rtSaxCppStrList.h, 226  
 rtSaxGetAttrValue  
     osrtxml.h, 205  
 rtSaxGetElemID8  
     osrtxml.h, 206  
 rtSaxGetElemID  
     osrtxml.h, 205  
 rtSaxHasXMLNSAttrs  
     osrtxml.h, 207  
 rtSaxIsEmptyBuffer  
     osrtxml.h, 207  
 rtSaxSortAttrs  
     osrtxml.h, 207  
 rtSaxStrListMatch  
     osrtxml.h, 208  
 rtSaxStrListParse  
     osrtxml.h, 208  
 rtXmlCmpBase64Str  
     osrtxml.h, 209  
 rtXmlCmpHexStr  
     osrtxml.h, 209  
 rtXmlCppEncAnyAttr  
     rtXmlCppEncFuncs.h, 227  
 rtXmlCppEncAnyTypeValue  
     rtXmlCppEncFuncs.h, 228  
 rtXmlCppEncFuncs.h, 226  
     rtXmlCppEncAnyAttr, 227  
     rtXmlCppEncAnyTypeValue, 228  
     rtXmlCppEncStartElement, 228  
     rtXmlEncAny, 229  
     rtXmlEncString, 229  
 rtXmlCppEncStartElement  
     rtXmlCppEncFuncs.h, 228  
 rtXmlCppMsgBuf.h, 230  
 rtXmlCppNamespace.h, 230  
 rtXmlCppXSDElement.h, 231  
 rtXmlCreateFileInputSource  
     osrtxml.h, 210  
 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  
 rtXmlDeclnt  
     XML decode functions., 27  
 rtXmlDeclnt16  
     XML decode functions., 27  
 rtXmlDeclnt64  
     XML decode functions., 28  
 rtXmlDeclnt8  
     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
  - rtXmlCppEncFuncs.h, 229
  - 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
- rtXmlEncEndSoapElems
  - 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  
 rtXmlEncStartSoapElems  
     XML encode functions., 77  
 rtXmlEncStartSoapEnv  
     XML encode functions., 77  
 rtXmlEncString  
     rtXmlCppEncFuncs.h, 229  
     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  
 rtXmlEncXSITypeAttr  
     XML encode functions., 87  
 rtXmlEncXSITypeAttr2  
     XML encode functions., 88  
 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, 210  
 rtXmlInitContextUsingKey  
     osrtxml.h, 210  
 rtXmlInitCtxtAppInfo  
     osrtxml.h, 211  
 rtXmlMatchBase64Str  
     osrtxml.h, 211  
 rtXmlMatchDate  
     osrtxml.h, 212  
 rtXmlMatchDateTime

- osrtxml.h, [212](#)
- rtXmlMatchGDay
  - osrtxml.h, [213](#)
- rtXmlMatchGMonth
  - osrtxml.h, [213](#)
- rtXmlMatchGMonthDay
  - osrtxml.h, [213](#)
- rtXmlMatchGYear
  - osrtxml.h, [214](#)
- rtXmlMatchGYearMonth
  - osrtxml.h, [214](#)
- rtXmlMatchHexStr
  - osrtxml.h, [215](#)
- rtXmlMatchTime
  - osrtxml.h, [215](#)
- rtXmlMemFreeAnyAttrs
  - osrtxml.h, [216](#)
- rtXmlNewQName
  - osrtxml.h, [216](#)
- rtXmlParseElemQName
  - XML decode functions., [35](#)
- rtXmlParseElementName
  - XML decode functions., [35](#)
- rtXmlPrepareContext
  - osrtxml.h, [217](#)
- rtXmlPrintNSAttrs
  - XML encode functions., [90](#)
- rtXmlSetEncBufPtr
  - XML encode functions., [90](#)
- rtXmlSetEncC14N
  - osrtxml.h, [217](#)
- rtXmlSetEncDocHdr
  - osrtxml.h, [217](#)
- rtXmlSetEncXSINamespace
  - osrtxml.h, [218](#)
- rtXmlSetEncXSINilAttr
  - osrtxml.h, [219](#)
- rtXmlSetEncodingStr
  - osrtxml.h, [218](#)
- rtXmlSetFormatting
  - osrtxml.h, [219](#)
- rtXmlSetIndent
  - osrtxml.h, [220](#)
- rtXmlSetIndentChar
  - osrtxml.h, [220](#)
- rtXmlSetNSPrefixLinks
  - osrtxml.h, [221](#)
- rtXmlSetNamespacesSet
  - osrtxml.h, [220](#)
- rtXmlSetNoNSSchemaLocation
  - osrtxml.h, [221](#)
- rtXmlSetSchemaLocation
  - osrtxml.h, [222](#)
- rtXmlSetSoapVersion

- osrtxml.h, [222](#)
- rtXmlSetWriteBOM
  - osrtxml.h, [223](#)
- rtXmlSetXSITypeAttr
  - osrtxml.h, [223](#)
- rtXmlWriteToFile
  - XML utility functions., [92](#)
- rtXmIplCountListItems
  - XML pull-parser decode functions., [97](#)
- rtXmIplCppDecFuncs.h, [231](#)
- rtXmIplCreateReader
  - XML pull-parser decode functions., [98](#)
- rtXmIplDecAny
  - XML pull-parser decode functions., [98](#)
- rtXmIplDecAny2
  - XML pull-parser decode functions., [99](#)
- rtXmIplDecAnyAttrStr
  - XML pull-parser decode functions., [99](#)
- rtXmIplDecAnyElem
  - XML pull-parser decode functions., [100](#)
- rtXmIplDecBase64Str
  - XML pull-parser decode functions., [100](#)
- rtXmIplDecBase64Str64
  - XML pull-parser decode functions., [101](#)
- rtXmIplDecBigInt
  - XML pull-parser decode functions., [102](#)
- rtXmIplDecBitString
  - XML pull-parser decode functions., [102](#)
- rtXmIplDecBitString64
  - XML pull-parser decode functions., [103](#)
- rtXmIplDecBitStringExt
  - XML pull-parser decode functions., [103](#)
- rtXmIplDecBitStringExt64
  - XML pull-parser decode functions., [104](#)
- rtXmIplDecBool
  - XML pull-parser decode functions., [105](#)
- rtXmIplDecDate
  - XML pull-parser decode functions., [105](#)
- rtXmIplDecDateTime
  - XML pull-parser decode functions., [106](#)
- rtXmIplDecDecimal
  - XML pull-parser decode functions., [106](#)
- rtXmIplDecDouble
  - XML pull-parser decode functions., [107](#)
- rtXmIplDecDoubleExt
  - XML pull-parser decode functions., [108](#)
- rtXmIplDecDynBase64Str
  - XML pull-parser decode functions., [108](#)
- rtXmIplDecDynBase64Str64
  - XML pull-parser decode functions., [109](#)
- rtXmIplDecDynBitString
  - XML pull-parser decode functions., [109](#)
- rtXmIplDecDynHexStr
  - XML pull-parser decode functions., [110](#)

rtXmlpDecDynHexStr64  
     XML pull-parser decode functions., 110  
 rtXmlpDecDynUTF8Str  
     XML pull-parser decode functions., 111  
 rtXmlpDecDynUnicodeStr  
     XML pull-parser decode functions., 111  
 rtXmlpDecGDay  
     XML pull-parser decode functions., 112  
 rtXmlpDecGMonth  
     XML pull-parser decode functions., 112  
 rtXmlpDecGMonthDay  
     XML pull-parser decode functions., 113  
 rtXmlpDecGYear  
     XML pull-parser decode functions., 113  
 rtXmlpDecGYearMonth  
     XML pull-parser decode functions., 114  
 rtXmlpDecHexStr  
     XML pull-parser decode functions., 114  
 rtXmlpDecHexStr64  
     XML pull-parser decode functions., 115  
 rtXmlpDecInt  
     XML pull-parser decode functions., 116  
 rtXmlpDecInt16  
     XML pull-parser decode functions., 116  
 rtXmlpDecInt64  
     XML pull-parser decode functions., 117  
 rtXmlpDecInt8  
     XML pull-parser decode functions., 117  
 rtXmlpDecNamedBits  
     XML pull-parser decode functions., 118  
 rtXmlpDecNamedBits64  
     XML pull-parser decode functions., 118  
 rtXmlpDecStrList  
     XML pull-parser decode functions., 119  
 rtXmlpDecTime  
     XML pull-parser decode functions., 119  
 rtXmlpDecUInt  
     XML pull-parser decode functions., 120  
 rtXmlpDecUInt16  
     XML pull-parser decode functions., 121  
 rtXmlpDecUInt64  
     XML pull-parser decode functions., 121  
 rtXmlpDecUInt8  
     XML pull-parser decode functions., 122  
 rtXmlpDecUTF8Str  
     XML pull-parser decode functions., 122  
 rtXmlpDecXSIAAttr  
     XML pull-parser decode functions., 124  
 rtXmlpDecXSIAAttrs  
     XML pull-parser decode functions., 124  
 rtXmlpDecXSISTypeAttr  
     XML pull-parser decode functions., 125  
 rtXmlpDecXmlStr  
     XML pull-parser decode functions., 123  
 rtXmlpDecXmlStrList  
     XML pull-parser decode functions., 123  
 rtXmlpForceDecodeAsGroup  
     XML pull-parser decode functions., 125  
 rtXmlpGetAttributeCount  
     XML pull-parser decode functions., 126  
 rtXmlpGetAttributeID  
     XML pull-parser decode functions., 126  
 rtXmlpGetCurrentLevel  
     XML pull-parser decode functions., 127  
 rtXmlpGetNextAllElemID16  
     XML pull-parser decode functions., 128  
 rtXmlpGetNextAllElemID32  
     XML pull-parser decode functions., 129  
 rtXmlpGetNextAllElemID  
     XML pull-parser decode functions., 127  
 rtXmlpGetNextElem  
     XML pull-parser decode functions., 129  
 rtXmlpGetNextElemID  
     XML pull-parser decode functions., 130  
 rtXmlpGetNextSeqElemID2  
     XML pull-parser decode functions., 131  
 rtXmlpGetNextSeqElemIDExt  
     XML pull-parser decode functions., 132  
 rtXmlpGetNextSeqElemID  
     XML pull-parser decode functions., 130  
 rtXmlpGetReader  
     XML pull-parser decode functions., 134  
 rtXmlpGetXSITypeAttr  
     XML pull-parser decode functions., 136  
 rtXmlpGetXSITypeIndex  
     XML pull-parser decode functions., 136  
 rtXmlpGetXmInsAttrs  
     XML pull-parser decode functions., 134  
 rtXmlpHasAttributes  
     XML pull-parser decode functions., 138  
 rtXmlpHideAttributes  
     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  
 rtXmlpListHasItem  
     XML pull-parser decode functions., 140  
 rtXmlpLookupXSISTypeIndex  
     XML pull-parser decode functions., 141  
 rtXmlpMarkLastEventActive  
     XML pull-parser decode functions., 141  
 rtXmlpMarkPos  
     XML pull-parser decode functions., 142

- rtXmIplMatchEndTag
  - XML pull-parser decode functions., 142
- rtXmIplMatchStartTag
  - XML pull-parser decode functions., 142
- rtXmIplNeedDecodeAttributes
  - XML pull-parser decode functions., 143
- rtXmIplReadBytes
  - XML pull-parser decode functions., 143
- rtXmIplResetMarkedPos
  - XML pull-parser decode functions., 144
- rtXmIplRewindToMarkedPos
  - XML pull-parser decode functions., 144
- rtXmIplSelectAttribute
  - XML pull-parser decode functions., 145
- rtXmIplSetListMode
  - XML pull-parser decode functions., 145
- rtXmIplSetMixedContentMode
  - XML pull-parser decode functions., 146
- rtXmIplSetNamespaceTable
  - XML pull-parser decode functions., 146
- rtXmIplSetWhiteSpaceMode
  - XML pull-parser decode functions., 146
  
- setAppInfo
  - OSXMLMessageBuffer, 176
- setDefaultNamespace
  - OSXSDGlobalElement, 187
- setDiag
  - OSXSDGlobalElement, 187
- setEncXSINamespace
  - OSXSDGlobalElement, 187
- setFormatting
  - OSXMLMessageBuffer, 176
- setFragment
  - OSXMLEncodeBuffer, 168
- setIndent
  - OSXMLMessageBuffer, 177
- setIndentChar
  - OSXMLMessageBuffer, 177
- setMaxErrors
  - OSXMLDecodeBuffer, 156
- setMsgBuf
  - OSXSDGlobalElement, 188
- setNamespace
  - OSXMLMessageBuffer, 177
  - OSXSDGlobalElement, 188
- setNoNSSchemaLocation
  - OSXSDGlobalElement, 188
- setSchemaLocation
  - OSXSDGlobalElement, 189
- setWriteBOM
  - OSXMLMessageBuffer, 178
- setXSIType
  - OSXSDGlobalElement, 189
  
- startDocument
  - OSXMLEncodeBase, 163
- startElement
  - OSXMLContentHandler, 150
  - OSXMLDefaultHandler, 159
  - OSXMLEncodeBase, 163
  
- termStartElement
  - OSXMLEncodeBase, 164
  
- validateFrom
  - OSXSDGlobalElement, 189
  
- write
  - OSXMLEncodeBuffer, 168, 169
  
- 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
  - rtXmlDecInt, 27
  - rtXmlDecInt16, 27
  - rtXmlDecInt64, 28
  - rtXmlDecInt8, 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
  - rtXmlEncEndSoapElements, 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
  - rtXmlEncNSAAttrs, 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
  - rtXmlEncXSIAAttrs, 86
  - rtXmlEncXSINilAttr, 87
  - rtXmlEncXSITypeAttr, 87
  - rtXmlEncXSITypeAttr2, 88
  - rtXmlFreeInputSource, 88
  - rtXmlGetEncBufLen, 41
  - rtXmlGetEncBufPtr, 42
  - rtXmlGetIndent, 89
  - rtXmlGetIndentChar, 89
  - rtXmlGetWriteBOM, 89
  - rtXmlPrintNSAAttrs, 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

[rtXmlpDecBigInt](#), 102  
[rtXmlpDecBitString](#), 102  
[rtXmlpDecBitString64](#), 103  
[rtXmlpDecBitStringExt](#), 103  
[rtXmlpDecBitStringExt64](#), 104  
[rtXmlpDecBool](#), 105  
[rtXmlpDecDate](#), 105  
[rtXmlpDecDateTime](#), 106  
[rtXmlpDecDecimal](#), 106  
[rtXmlpDecDouble](#), 107  
[rtXmlpDecDoubleExt](#), 108  
[rtXmlpDecDynBase64Str](#), 108  
[rtXmlpDecDynBase64Str64](#), 109  
[rtXmlpDecDynBitString](#), 109  
[rtXmlpDecDynHexStr](#), 110  
[rtXmlpDecDynHexStr64](#), 110  
[rtXmlpDecDynUTF8Str](#), 111  
[rtXmlpDecDynUnicodeStr](#), 111  
[rtXmlpDecGDay](#), 112  
[rtXmlpDecGMonth](#), 112  
[rtXmlpDecGMonthDay](#), 113  
[rtXmlpDecGYear](#), 113  
[rtXmlpDecGYearMonth](#), 114  
[rtXmlpDecHexStr](#), 114  
[rtXmlpDecHexStr64](#), 115  
[rtXmlpDeclnt](#), 116  
[rtXmlpDeclnt16](#), 116  
[rtXmlpDeclnt64](#), 117  
[rtXmlpDeclnt8](#), 117  
[rtXmlpDecNamedBits](#), 118  
[rtXmlpDecNamedBits64](#), 118  
[rtXmlpDecStrList](#), 119  
[rtXmlpDecTime](#), 119  
[rtXmlpDecUInt](#), 120  
[rtXmlpDecUInt16](#), 121  
[rtXmlpDecUInt64](#), 121  
[rtXmlpDecUInt8](#), 122  
[rtXmlpDecUTF8Str](#), 122  
[rtXmlpDecXSIAAttr](#), 124  
[rtXmlpDecXSIAAttrs](#), 124  
[rtXmlpDecXSISTypeAttr](#), 125  
[rtXmlpDecXmlStr](#), 123  
[rtXmlpDecXmlStrList](#), 123  
[rtXmlpForceDecodeAsGroup](#), 125  
[rtXmlpGetAttributeCount](#), 126  
[rtXmlpGetAttributeID](#), 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  
[rtXmlplsDecodeAsGroup](#), 139  
[rtXmlplsEmptyElement](#), 139  
[rtXmlplsLastEventDone](#), 139  
[rtXmlplsUTF8Encoding](#), 140  
[rtXmlpListHasItem](#), 140  
[rtXmlpLookupXSISTypeIndex](#), 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 utility functions., 92  
[rtXmlWriteToFile](#), 92