

# **XBinder**

---

XML Schema Compiler  
Version 2.4  
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 ©1997–2016 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>Main Page</b>	<b>1</b>
<b>2</b>	<b>C DOM Runtime Library Functions</b>	<b>2</b>
<b>3</b>	<b>Module Index</b>	<b>4</b>
3.1	Modules . . . . .	4
<b>4</b>	<b>Class Index</b>	<b>5</b>
4.1	Class List . . . . .	5
<b>5</b>	<b>File Index</b>	<b>6</b>
5.1	File List . . . . .	6
<b>6</b>	<b>Module Documentation</b>	<b>7</b>
6.1	XML decode functions. . . . .	7
6.1.1	Function Documentation . . . . .	9
6.1.1.1	rtXmlDecBase64Binary . . . . .	9
6.1.1.2	rtXmlDecBase64Str . . . . .	10
6.1.1.3	rtXmlDecBase64StrValue . . . . .	10
6.1.1.4	rtXmlDecBigInt . . . . .	11
6.1.1.5	rtXmlDecBool . . . . .	11
6.1.1.6	rtXmlDecDate . . . . .	11
6.1.1.7	rtXmlDecDateTime . . . . .	12
6.1.1.8	rtXmlDecDecimal . . . . .	12
6.1.1.9	rtXmlDecDouble . . . . .	12
6.1.1.10	rtXmlDecDynBase64Str . . . . .	13
6.1.1.11	rtXmlDecDynHexStr . . . . .	13
6.1.1.12	rtXmlDecDynUTF8Str . . . . .	13
6.1.1.13	rtXmlDecEmptyElement . . . . .	14
6.1.1.14	rtXmlDecGDay . . . . .	14

6.1.1.15	rtXmlDecGMonth	14
6.1.1.16	rtXmlDecGMonthDay	15
6.1.1.17	rtXmlDecGYear	15
6.1.1.18	rtXmlDecGYearMonth	15
6.1.1.19	rtXmlDecHexBinary	16
6.1.1.20	rtXmlDecHexStr	16
6.1.1.21	rtXmlDecInt	16
6.1.1.22	rtXmlDecInt16	17
6.1.1.23	rtXmlDecInt64	17
6.1.1.24	rtXmlDecInt8	17
6.1.1.25	rtXmlDecNSAttr	18
6.1.1.26	rtXmlDecQName	18
6.1.1.27	rtXmlDecTime	19
6.1.1.28	rtXmlDecUInt	19
6.1.1.29	rtXmlDecUInt16	19
6.1.1.30	rtXmlDecUInt64	20
6.1.1.31	rtXmlDecUInt8	20
6.1.1.32	rtXmlDecUTF8Str	20
6.1.1.33	rtXmlDecXmlStr	21
6.1.1.34	rtXmlDecXSIAttr	21
6.1.1.35	rtXmlDecXSIAttrs	22
6.1.1.36	rtXmlParseElementName	22
6.1.1.37	rtXmlParseElemQName	22
6.2	XML encode functions	23
6.2.1	Define Documentation	29
6.2.1.1	rtXmlGetEncBufLen	29
6.2.1.2	rtXmlGetEncBufPtr	29
6.2.2	Function Documentation	30
6.2.2.1	rtXmlEncAny	30
6.2.2.2	rtXmlEncAnyAttr	30
6.2.2.3	rtXmlEncAnyTypeValue	30
6.2.2.4	rtXmlEncBase64Binary	31
6.2.2.5	rtXmlEncBase64BinaryAttr	31
6.2.2.6	rtXmlEncBase64StrValue	32
6.2.2.7	rtXmlEncBigInt	32
6.2.2.8	rtXmlEncBigIntAttr	32
6.2.2.9	rtXmlEncBigIntValue	33

6.2.2.10	rtXmlEncBinStrValue	33
6.2.2.11	rtXmlEncBitString	34
6.2.2.12	rtXmlEncBOM	34
6.2.2.13	rtXmlEncBool	34
6.2.2.14	rtXmlEncBoolAttr	35
6.2.2.15	rtXmlEncBoolValue	35
6.2.2.16	rtXmlEncComment	36
6.2.2.17	rtXmlEncDate	36
6.2.2.18	rtXmlEncDateTime	36
6.2.2.19	rtXmlEncDateTimeValue	37
6.2.2.20	rtXmlEncDateValue	37
6.2.2.21	rtXmlEncDecimal	37
6.2.2.22	rtXmlEncDecimalAttr	38
6.2.2.23	rtXmlEncDecimalValue	38
6.2.2.24	rtXmlEncDouble	39
6.2.2.25	rtXmlEncDoubleAttr	39
6.2.2.26	rtXmlEncDoubleNormalValue	39
6.2.2.27	rtXmlEncDoubleValue	40
6.2.2.28	rtXmlEncEmptyElement	40
6.2.2.29	rtXmlEncEndDocument	41
6.2.2.30	rtXmlEncEndElement	41
6.2.2.31	rtXmlEncEndSoapElems	41
6.2.2.32	rtXmlEncEndSoapEnv	42
6.2.2.33	rtXmlEncFloat	42
6.2.2.34	rtXmlEncFloatAttr	42
6.2.2.35	rtXmlEncGDay	43
6.2.2.36	rtXmlEncGDayValue	43
6.2.2.37	rtXmlEncGMonth	43
6.2.2.38	rtXmlEncGMonthDay	44
6.2.2.39	rtXmlEncGMonthDayValue	44
6.2.2.40	rtXmlEncGMonthValue	44
6.2.2.41	rtXmlEncGYear	45
6.2.2.42	rtXmlEncGYearMonth	45
6.2.2.43	rtXmlEncGYearMonthValue	45
6.2.2.44	rtXmlEncGYearValue	46
6.2.2.45	rtXmlEncHexBinary	46
6.2.2.46	rtXmlEncHexBinaryAttr	46

6.2.2.47	rtXmlEncHexStrValue	47
6.2.2.48	rtXmlEncIndent	47
6.2.2.49	rtXmlEncInt	47
6.2.2.50	rtXmlEncInt64	48
6.2.2.51	rtXmlEncInt64Attr	48
6.2.2.52	rtXmlEncInt64Value	49
6.2.2.53	rtXmlEncIntAttr	49
6.2.2.54	rtXmlEncIntPattern	49
6.2.2.55	rtXmlEncIntValue	50
6.2.2.56	rtXmlEncNamedBits	50
6.2.2.57	rtXmlEncNSAttrs	51
6.2.2.58	rtXmlEncReal10	51
6.2.2.59	rtXmlEncSoapArrayTypeAttr	51
6.2.2.60	rtXmlEncStartDocument	52
6.2.2.61	rtXmlEncStartElement	52
6.2.2.62	rtXmlEncStartSoapElems	53
6.2.2.63	rtXmlEncStartSoapEnv	53
6.2.2.64	rtXmlEncString	53
6.2.2.65	rtXmlEncStringValue	54
6.2.2.66	rtXmlEncStringValue2	54
6.2.2.67	rtXmlEncTermStartElement	54
6.2.2.68	rtXmlEncTime	55
6.2.2.69	rtXmlEncTimeValue	55
6.2.2.70	rtXmlEncUInt	55
6.2.2.71	rtXmlEncUInt64	56
6.2.2.72	rtXmlEncUInt64Attr	56
6.2.2.73	rtXmlEncUInt64Value	57
6.2.2.74	rtXmlEncUIntAttr	57
6.2.2.75	rtXmlEncUIntValue	57
6.2.2.76	rtXmlEncUnicodeStr	58
6.2.2.77	rtXmlEncUTF8Attr	58
6.2.2.78	rtXmlEncUTF8Attr2	58
6.2.2.79	rtXmlEncUTF8Str	59
6.2.2.80	rtXmlEncXSAttrs	59
6.2.2.81	rtXmlEncXSINilAttr	60
6.2.2.82	rtXmlEncXSITypeAttr	60
6.2.2.83	rtXmlEncXSITypeAttr2	60



6.2.2.84	rtXmlFreeInputSource	61
6.2.2.85	rtXmlGetIndent	61
6.2.2.86	rtXmlGetIndentChar	61
6.2.2.87	rtXmlGetWriteBOM	61
6.2.2.88	rtXmlPrintNSAttrs	62
6.2.2.89	rtXmlSetEncBufPtr	62
6.3	XML utility functions.	63
6.3.1	Function Documentation	63
6.3.1.1	rtXmlWriteToFile	63
6.4	XML pull-parser decode functions.	64
6.4.1	Function Documentation	69
6.4.1.1	rtXmlEncAttrC14N	69
6.4.1.2	rtXmlpCountListItems	69
6.4.1.3	rtXmlpCreateReader	69
6.4.1.4	rtXmlpDecAny	70
6.4.1.5	rtXmlpDecAny2	70
6.4.1.6	rtXmlpDecAnyAttrStr	70
6.4.1.7	rtXmlpDecAnyElem	71
6.4.1.8	rtXmlpDecBase64Str	71
6.4.1.9	rtXmlpDecBigInt	72
6.4.1.10	rtXmlpDecBitString	72
6.4.1.11	rtXmlpDecBool	72
6.4.1.12	rtXmlpDecDate	73
6.4.1.13	rtXmlpDecDateTime	73
6.4.1.14	rtXmlpDecDecimal	73
6.4.1.15	rtXmlpDecDouble	74
6.4.1.16	rtXmlpDecDoubleExt	74
6.4.1.17	rtXmlpDecDynBase64Str	75
6.4.1.18	rtXmlpDecDynBitString	75
6.4.1.19	rtXmlpDecDynHexStr	75
6.4.1.20	rtXmlpDecDynUnicodeStr	76
6.4.1.21	rtXmlpDecDynUTF8Str	76
6.4.1.22	rtXmlpDecGDay	76
6.4.1.23	rtXmlpDecGMonth	77
6.4.1.24	rtXmlpDecGMonthDay	77
6.4.1.25	rtXmlpDecGYear	78
6.4.1.26	rtXmlpDecGYearMonth	78

6.4.1.27	rtXmlpDecHexStr	78
6.4.1.28	rtXmlpDecInt	79
6.4.1.29	rtXmlpDecInt16	79
6.4.1.30	rtXmlpDecInt64	79
6.4.1.31	rtXmlpDecInt8	80
6.4.1.32	rtXmlpDecNamedBits	80
6.4.1.33	rtXmlpDecStrList	81
6.4.1.34	rtXmlpDecTime	81
6.4.1.35	rtXmlpDecUInt	81
6.4.1.36	rtXmlpDecUInt16	82
6.4.1.37	rtXmlpDecUInt64	82
6.4.1.38	rtXmlpDecUInt8	82
6.4.1.39	rtXmlpDecUTF8Str	83
6.4.1.40	rtXmlpDecXmlStr	83
6.4.1.41	rtXmlpDecXmlStrList	83
6.4.1.42	rtXmlpDecXSIAAttr	84
6.4.1.43	rtXmlpDecXSIAAttrs	84
6.4.1.44	rtXmlpDecXSISTypeAttr	84
6.4.1.45	rtXmlpForceDecodeAsGroup	85
6.4.1.46	rtXmlpGetAttributeCount	85
6.4.1.47	rtXmlpGetAttributeID	85
6.4.1.48	rtXmlpGetCurrentLevel	86
6.4.1.49	rtXmlpGetNextAllElemID	86
6.4.1.50	rtXmlpGetNextAllElemID16	86
6.4.1.51	rtXmlpGetNextAllElemID32	87
6.4.1.52	rtXmlpGetNextElem	87
6.4.1.53	rtXmlpGetNextElemID	88
6.4.1.54	rtXmlpGetNextSeqElemID	88
6.4.1.55	rtXmlpGetNextSeqElemID2	89
6.4.1.56	rtXmlpGetNextSeqElemIDExt	89
6.4.1.57	rtXmlpGetReader	90
6.4.1.58	rtXmlpGetXmLnsAttrs	90
6.4.1.59	rtXmlpGetXSISTypeAttr	91
6.4.1.60	rtXmlpGetXSISTypeIndex	91
6.4.1.61	rtXmlpHasAttributes	91
6.4.1.62	rtXmlpHideAttributes	92
6.4.1.63	rtXmlpIsDecodeAsGroup	92

6.4.1.64	rtXmlpIsEmptyElement	92
6.4.1.65	rtXmlpIsLastEventDone	92
6.4.1.66	rtXmlpIsUTF8Encoding	93
6.4.1.67	rtXmlpListHasItem	93
6.4.1.68	rtXmlpLookupXSITypeIndex	93
6.4.1.69	rtXmlpMarkLastEventActive	94
6.4.1.70	rtXmlpMarkPos	94
6.4.1.71	rtXmlpMatchEndTag	94
6.4.1.72	rtXmlpMatchStartTag	94
6.4.1.73	rtXmlpNeedDecodeAttributes	95
6.4.1.74	rtXmlpReadBytes	95
6.4.1.75	rtXmlpResetMarkedPos	95
6.4.1.76	rtXmlpRewindToMarkedPos	96
6.4.1.77	rtXmlpSelectAttribute	96
6.4.1.78	rtXmlpSetListMode	96
6.4.1.79	rtXmlpSetMixedContentMode	96
6.4.1.80	rtXmlpSetNamespaceTable	97
6.4.1.81	rtXmlpSetWhiteSpaceMode	97
6.5	XML run-time error status codes.	98
6.5.1	Detailed Description	99
6.5.2	Define Documentation	99
6.5.2.1	XML_E_BASE	99
6.5.2.2	XML_E_ELEMMISRQ	100
6.5.2.3	XML_E_ELEMSISRQ	100
6.5.2.4	XML_E_FLDABSENT	100
6.5.2.5	XML_E_NOMATCH	100
6.5.2.6	XML_E_NSURINOTFOU	100
6.5.2.7	XML_E_TAGMISMATCH	100
6.5.2.8	XML_OK_EOB	100
6.5.2.9	XML_OK_FRAG	101
6.6	DOM API functions.	102
6.6.1	Function Documentation	103
6.6.1.1	domAddAttribute	103
6.6.1.2	domAddCdata	104
6.6.1.3	domAddContent	104
6.6.1.4	domCreateChild	104
6.6.1.5	domCreateDocument	105

6.6.1.6	domFreeDoc	105
6.6.1.7	domGetAttrData	105
6.6.1.8	domGetChild	106
6.6.1.9	domGetDoc	106
6.6.1.10	domGetElementName	106
6.6.1.11	domGetNext	106
6.6.1.12	domGetNextAttr	107
6.6.1.13	domGetNodeAttributesNum	107
6.6.1.14	domGetNodeContent	107
6.6.1.15	domGetNodeFirstAttribute	108
6.6.1.16	domGetRootElement	108
6.6.1.17	domParseFile	108
6.6.1.18	domSaveDoc	108
6.7	DOM runtime encode/decode functions.	110
6.7.1	Function Documentation	111
6.7.1.1	rtDomAddAttr	111
6.7.1.2	rtDomAddNode	111
6.7.1.3	rtDomAddNSAttrs	112
6.7.1.4	rtDomAddSubTree	112
6.7.1.5	rtDomDecodeDoc	112
6.7.1.6	rtDomEncAny	113
6.7.1.7	rtDomEncAnyAttr	113
6.7.1.8	rtDomEncString	114
6.7.1.9	rtDomEncStringValue	114
6.7.1.10	rtDomEncXSIAttrs	114
6.7.1.11	rtDomSetNode	115
<b>7</b>	<b>Class Documentation</b>	<b>116</b>
7.1	OSXMLGroupDesc Struct Reference	116
7.1.1	Detailed Description	116
<b>8</b>	<b>File Documentation</b>	<b>117</b>
8.1	osrtdom.h File Reference	117
8.1.1	Detailed Description	118
8.2	osrxml.h File Reference	119
8.2.1	Detailed Description	136
8.2.2	Typedef Documentation	136
8.2.2.1	OSXMLGroupDesc	136

8.2.3	Function Documentation	136
8.2.3.1	rtSaxGetAttrValue	136
8.2.3.2	rtSaxGetElemID	137
8.2.3.3	rtSaxGetElemID8	137
8.2.3.4	rtSaxHasXMLNSAttrs	137
8.2.3.5	rtSaxIsEmptyBuffer	138
8.2.3.6	rtSaxSortAttrs	138
8.2.3.7	rtSaxStrListMatch	138
8.2.3.8	rtSaxStrListParse	138
8.2.3.9	rtXmlCmpBase64Str	139
8.2.3.10	rtXmlCmpHexStr	139
8.2.3.11	rtXmlCreateFileInputSource	139
8.2.3.12	rtXmlInitContext	140
8.2.3.13	rtXmlInitContextUsingKey	140
8.2.3.14	rtXmlInitCtxtAppInfo	140
8.2.3.15	rtXmlMatchBase64Str	140
8.2.3.16	rtXmlMatchDate	141
8.2.3.17	rtXmlMatchDateTime	141
8.2.3.18	rtXmlMatchGDay	141
8.2.3.19	rtXmlMatchGMonth	141
8.2.3.20	rtXmlMatchGMonthDay	142
8.2.3.21	rtXmlMatchGYear	142
8.2.3.22	rtXmlMatchGYearMonth	142
8.2.3.23	rtXmlMatchHexStr	142
8.2.3.24	rtXmlMatchTime	143
8.2.3.25	rtXmlMemFreeAnyAttrs	143
8.2.3.26	rtXmlNewQName	143
8.2.3.27	rtXmlPrepareContext	144
8.2.3.28	rtXmlSetEncC14N	144
8.2.3.29	rtXmlSetEncDocHdr	144
8.2.3.30	rtXmlSetEncodingStr	144
8.2.3.31	rtXmlSetEncXSINamespace	145
8.2.3.32	rtXmlSetEncXSINilAttr	145
8.2.3.33	rtXmlSetFormatting	145
8.2.3.34	rtXmlSetIndent	145
8.2.3.35	rtXmlSetIndentChar	146
8.2.3.36	rtXmlSetNamespacesSet	146

8.2.3.37	<code>rtXmlSetNoNSSchemaLocation</code>	146
8.2.3.38	<code>rtXmlSetNSPrefixLinks</code>	147
8.2.3.39	<code>rtXmlSetSchemaLocation</code>	147
8.2.3.40	<code>rtXmlSetSoapVersion</code>	147
8.2.3.41	<code>rtXmlSetWriteBOM</code>	147
8.2.3.42	<code>rtXmlSetXSITypeAttr</code>	148
8.3	<code>rtXmlErrCodes.h</code> File Reference	149
8.3.1	Detailed Description	150
8.4	<code>rtXmlExternDefs.h</code> File Reference	151
8.4.1	Detailed Description	151
8.5	<code>rtXmlKeyArray.h</code> File Reference	152
8.5.1	Detailed Description	152
8.5.2	Function Documentation	152
8.5.2.1	<code>rtXmlKeyArrayAdd</code>	152
8.5.2.2	<code>rtXmlKeyArrayContains</code>	153
8.5.2.3	<code>rtXmlKeyArrayInit</code>	153
8.5.2.4	<code>rtXmlKeyArraySetDecimal</code>	153
8.5.2.5	<code>rtXmlKeyArraySetInt</code>	153
8.5.2.6	<code>rtXmlKeyArraySetString</code>	153
8.5.2.7	<code>rtXmlKeyArraySetUInt</code>	153
8.6	<code>rtXmlNamespace.h</code> File Reference	154
8.6.1	Detailed Description	156
8.6.2	Define Documentation	156
8.6.2.1	<code>RTXMLNSSETQNAME</code>	156
8.6.3	Function Documentation	156
8.6.3.1	<code>rtXmlNSAddNamespace</code>	156
8.6.3.2	<code>rtXmlNSEqual</code>	157
8.6.3.3	<code>rtXmlNSFreeAttrList</code>	157
8.6.3.4	<code>rtXmlNSGetAttrPrefix</code>	157
8.6.3.5	<code>rtXmlNSGetAttrQName</code>	157
8.6.3.6	<code>rtXmlNSGetPrefix</code>	158
8.6.3.7	<code>rtXmlNSGetPrefixCount</code>	158
8.6.3.8	<code>rtXmlNSGetPrefixIndex</code>	158
8.6.3.9	<code>rtXmlNSGetPrefixUsingIndex</code>	159
8.6.3.10	<code>rtXmlNSGetQName</code>	159
8.6.3.11	<code>rtXmlNSLookupPrefix</code>	159
8.6.3.12	<code>rtXmlNSLookupPrefixForURI</code>	160

8.6.3.13	<code>rtXmlNSLookupPrefixFrag</code>	160
8.6.3.14	<code>rtXmlNSLookupURI</code>	160
8.6.3.15	<code>rtXmlNSLookupURIInList</code>	160
8.6.3.16	<code>rtXmlNSNewPrefix</code>	161
8.6.3.17	<code>rtXmlNSRemoveAll</code>	161
8.6.3.18	<code>rtXmlNSSetNamespace</code>	161
8.7	<code>rtXmlpCppDecFuncs.h</code> File Reference	162
8.7.1	Detailed Description	162

# Chapter 1

## Main Page

### C XML Runtime Library Functions

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

The categories of functions provided are as follows:

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



## **Chapter 2**

# **C DOM Runtime Library Functions**

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

The categories of functions provided are as follows:

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

# Chapter 3

## Module Index

### 3.1 Modules

Here is a list of all modules:

XML decode functions. . . . .	7
XML encode functions. . . . .	23
XML utility functions. . . . .	63
XML pull-parser decode functions. . . . .	64
XML run-time error status codes. . . . .	98
DOM API functions. . . . .	102
DOM runtime encode/decode functions. . . . .	110

# Chapter 4

## Class Index

### 4.1 Class List

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

<a href="#">OSXMLGroupDesc</a> ( <a href="#">OSXMLGroupDesc</a> describes how entries in an OSXMLElemIDRec array make up a group) . . . . .	116
---	-----

# Chapter 5

## File Index

### 5.1 File List

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

<b>domAPI.h</b>	??
<a href="#">osrtdom.h</a> (DOM low-level C encode/decode functions )	117
<a href="#">osrtxml.h</a> (XML low-level C encode/decode functions )	119
<b>rtLx2Dom.h</b>	??
<b>rtxDomDefs.h</b>	??
<b>rtXmlCtxtAppInfo.h</b>	??
<b>rtXmlEncDateTmpl.h</b>	??
<a href="#">rtXmlErrCodes.h</a> (List of numeric status codes that can be returned by ASN1C run-time functions and generated code )	149
<a href="#">rtXmlExternDefs.h</a> (XML external definitions macro )	151
<a href="#">rtXmlKeyArray.h</a>	(
• Implementation of a dynamic pointer sorted array	
)	152
<a href="#">rtXmlNamespace.h</a> (XML namespace handling structures and function definitions )	154
<a href="#">rtXmlpCppDecFuncs.h</a> (XML low-level C++ decode functions )	162
<b>rtXmlPull.h</b>	??

# Chapter 6

## Module Documentation

### 6.1 XML decode functions.

#### Functions

- int [rtXmlDecBase64Binary](#) (OSRTMEMBUF \*pMemBuf, const OSUTF8CHAR \*inpdata, int length)  
*This function decodes the contents of a Base64-encoded binary data type into a memory buffer.*
- 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 [rtXmlDecBase64StrValue](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, size\_t bufSize, size\_t srcDataLen)  
*This function decodes a contents of a Base64-encode binary string into the specified octet array.*
- 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 [rtXmlDecDynHexStr](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)  
*This function decodes a contents of a hexBinary string.*
- int [rtXmlDecEmptyElement](#) (OSCTXT \*pctxt)  
*This function is used to enforce a requirement that an element be empty.*
- int [rtXmlDecUTF8Str](#) (OSCTXT \*pctxt, OSUTF8CHAR \*outdata, size\_t 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, int length)  
*This function decodes the contents of a hex-encoded binary data type into a memory buffer.*
- int [rtXmlDecHexStr](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, OSINT32 bufsize)  
*This function decodes the contents of a hexBinary string into a static memory structure.*
- 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 \*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 \*pctx, const OSUTF8CHAR \*qname, const OSUTF8CHAR \*\*prefix)  
*This function decodes an XML qualified name string (QName) type.*
- int `rtXmlDecXSIAttr` (OSCTXT \*pctx, const OSUTF8CHAR \*attrName, const OSUTF8CHAR \*attrValue)  
*This function decodes XML schema instance (XSI) attribute.*
- int `rtXmlDecXSIAttrs` (OSCTXT \*pctx, const OSUTF8CHAR \*const \*attrs, const char \*typeName)  
*This function decodes XML schema instance (XSI) attributes.*
- int `rtXmlDecXmlStr` (OSCTXT \*pctx, OSXMLSTRING \*outdata)  
*This function decodes the contents of an XML string data type.*
- int `rtXmlParseElementName` (OSCTXT \*pctx, OSUTF8CHAR \*\*ppName)  
*This function parses the initial tag from an XML message.*
- int `rtXmlParseElemQName` (OSCTXT \*pctx, OSXMLQName \*pQName)  
*This function parses the initial tag from an XML message.*

## 6.1.1 Function Documentation

### 6.1.1.1 int `rtXmlDecBase64Binary` (OSRTMEMBUF \* *pMemBuf*, const OSUTF8CHAR \* *inpdata*, int *length*)

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

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

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

#### Parameters

*pMemBuf* Memory buffer to which decoded binary data is to be appended.

*inpdata* Pointer to a source string to be decoded.

*length* Length of the source string (in characters).



## Returns

Completion status of operation:

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

### 6.1.1.2 `int rtXmlDecBase64Str (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocts, OSINT32 bufsize)`

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

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

## Parameters

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

*pvalue* A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the *bufsize* input parameter.

*pnocts* A pointer to an integer value to receive the decoded number of octets.

*bufsize* The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

## Returns

Completion status of operation:

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

### 6.1.1.3 `int rtXmlDecBase64StrValue (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocts, size_t bufSize, size_t srcDataLen)`

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

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

## Parameters

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

*pvalue* A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the *bufsize* input parameter.

*pnocts* A pointer to an integer value to receive the decoded number of octets.

*bufSize* A maximum size (in octets) of *pvalue* buffer. An error will occur if the number of octets in the decoded string is larger than this value.

*srcDataLen* An actual source data length (in octets) without whitespaces.

## Returns

Completion status of operation:

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

#### 6.1.1.4 `int rtXmlDecBigInt (OSCTXT * pctxt, const OSUTF8CHAR ** ppvalue)`

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

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

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

##### Parameters

*pctxt* Pointer to context block structure.

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

##### Returns

Completion status of operation:

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

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

This function decodes a variable of the boolean type.

##### Parameters

*pctxt* Pointer to context block structure.

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

##### Returns

Completion status of operation:

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

#### 6.1.1.6 `int rtXmlDecDate (OSCTXT * pctxt, OSXSDDateTime * pvalue)`

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

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

##### Parameters

*pctxt* Pointer to context block structure.

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

##### Returns

Completion status of operation:

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

#### 6.1.1.7 **int rtXmlDecDateTime (OSCTXT \* *pctxt*, OSXSDDateTime \* *pvalue*)**

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

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

##### **Parameters**

*pctxt* Pointer to context block structure.

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

##### **Returns**

Completion status of operation:

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

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

This function decodes the contents of a decimal data type.

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

##### **Parameters**

*pctxt* Pointer to context block structure.

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

##### **Returns**

Completion status of operation:

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

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

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

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

##### **Parameters**

*pctxt* Pointer to context block structure.

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

##### **Returns**

Completion status of operation:

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

#### 6.1.1.10 `int rtXmlDecDynBase64Str (OSCTXT * pctxt, OSDynOctStr * pvalue)`

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

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

##### Parameters

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

*pvalue* A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the `rtxMemAlloc` function.

##### Returns

Completion status of operation:

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

#### 6.1.1.11 `int rtXmlDecDynHexStr (OSCTXT * pctxt, OSDynOctStr * pvalue)`

This function decodes a contents of a hexBinary string.

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

##### Parameters

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

*pvalue* A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the `rtxMemAlloc` function.

##### Returns

Completion status of operation:

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

#### 6.1.1.12 `int rtXmlDecDynUTF8Str (OSCTXT * pctxt, const OSUTF8CHAR ** outdata)`

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

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

##### Parameters

*pctxt* Pointer to context block structure.

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

##### Returns

Completion status of operation:

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

#### 6.1.1.13 int rtXmlDecEmptyElement (OSCTXT \* *pctxt*)

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

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

##### Parameters

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

##### Returns

Completion status of operation:

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

#### 6.1.1.14 int rtXmlDecGDay (OSCTXT \* *pctxt*, OSXSDDateTime \* *pvalue*)

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

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

##### Parameters

*pctxt* Pointer to context block structure.

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

##### Returns

Completion status of operation:

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

#### 6.1.1.15 int rtXmlDecGMonth (OSCTXT \* *pctxt*, OSXSDDateTime \* *pvalue*)

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

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

##### Parameters

*pctxt* Pointer to context block structure.

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

##### Returns

Completion status of operation:

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

#### 6.1.1.16 **int rtXmlDecGMonthDay (OSCTXT \* *pctxt*, OSXSDDateTime \* *pvalue*)**

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

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

##### **Parameters**

*pctxt* Pointer to context block structure.

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

##### **Returns**

Completion status of operation:

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

#### 6.1.1.17 **int rtXmlDecGYear (OSCTXT \* *pctxt*, OSXSDDateTime \* *pvalue*)**

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

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

##### **Parameters**

*pctxt* Pointer to context block structure.

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

##### **Returns**

Completion status of operation:

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

#### 6.1.1.18 **int rtXmlDecGYearMonth (OSCTXT \* *pctxt*, OSXSDDateTime \* *pvalue*)**

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

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

##### **Parameters**

*pctxt* Pointer to context block structure.

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

##### **Returns**

Completion status of operation:

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

#### 6.1.1.19 `int rtXmlDecHexBinary (OSRTMEMBUF * pMemBuf, const OSUTF8CHAR * inpdata, int length)`

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

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

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

##### Parameters

*pMemBuf* Pointer to memory buffer onto which the decoded binary data will be appended.

*inpdata* Pointer to a source string to be decoded.

*length* Length of the source string (in characters).

##### Returns

Completion status of operation:

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

#### 6.1.1.20 `int rtXmlDecHexStr (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocets, OSINT32 bufsize)`

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

##### Parameters

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

*pvalue* A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.

*pnocets* A pointer to an integer value to receive the decoded number of octets.

*bufsize* The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

##### Returns

Completion status of operation:

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

#### 6.1.1.21 `int rtXmlDecInt (OSCTXT * pctxt, OSINT32 * pvalue)`

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 32-bit integer value to receive decoded result.

### Returns

Completion status of operation:

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

#### 6.1.1.22 `int rtXmlDecInt16 (OSCTXT * pctxt, OSINT16 * pvalue)`

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

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

### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 16-bit integer value to receive decoded result.

### Returns

Completion status of operation:

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

#### 6.1.1.23 `int rtXmlDecInt64 (OSCTXT * pctxt, OSINT64 * pvalue)`

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

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

### Parameters

*pctxt* Pointer to context block structure.

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

### Returns

Completion status of operation:

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

#### 6.1.1.24 `int rtXmlDecInt8 (OSCTXT * pctxt, OSINT8 * pvalue)`

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

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

### Parameters

*pctxt* Pointer to context block structure.



*pvalue* Pointer to 8-bit integer value to receive decoded result.

## Returns

Completion status of operation:

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

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

This function decodes an XML namespace attribute (xmlns).

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

## Parameters

*pctxt* Pointer to context structure.

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

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

*pNSAttrs* List to receive parsed namespace values.

*nsTable* Namespace URI's parsed from schema.

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

## Returns

Zero if success or negative error code.

**6.1.1.26** `const OSUTF8CHAR* rtXmlDecQName (OSCTXT * pctxt, const OSUTF8CHAR * qname, const OSUTF8CHAR ** prefix)`

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

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

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

[NS 6] Prefix ::= NCName

[NS 7] LocalPart ::= NCName

## Parameters

*pctxt* Pointer to context block structure.

*qname* String containing XML QName to be decoded.

*prefix* Pointer to string pointer to receive decoded prefix. This is optional. If NULL, the prefix will not be returned. If not-NULL, the prefix will be returned in memory allocated using `rtxMemAlloc` which must be freed using one of the `rtxMemFree` functions.

## Returns

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

### 6.1.1.27 `int rtXmlDecTime (OSCTXT * pctxt, OSXSDDateTime * pvalue)`

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

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

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

## Parameters

*pctxt* Pointer to context block structure.

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

## Returns

Completion status of operation:

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

### 6.1.1.28 `int rtXmlDecUInt (OSCTXT * pctxt, OSUINT32 * pvalue)`

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

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

## Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned 32-bit integer value to receive decoded result.

## Returns

Completion status of operation:

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

### 6.1.1.29 `int rtXmlDecUInt16 (OSCTXT * pctxt, OSUINT16 * pvalue)`

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

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

## Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned 16-bit integer value to receive decoded result.

#### Returns

Completion status of operation:

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

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

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

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

#### Parameters

*pctxt* Pointer to context block structure.

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

#### Returns

Completion status of operation:

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

#### 6.1.1.31 int rtXmlDecUInt8 (OSCTXT \* *pctxt*, OSUINT8 \* *pvalue*)

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

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

#### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned 8-bit integer value to receive decoded result.

#### Returns

Completion status of operation:

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

#### 6.1.1.32 int rtXmlDecUTF8Str (OSCTXT \* *pctxt*, OSUTF8CHAR \* *outdata*, size\_t *max\_len*)

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

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

#### Parameters

*pctxt* Pointer to context block structure.

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

*max\_len* Size of memory block.

### Returns

Completion status of operation:

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

#### 6.1.1.33 int rtXmlDecXmlStr (OSCTXT \* *pctxt*, OSXMLSTRING \* *outdata*)

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

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

### Parameters

*pctxt* Pointer to context block structure.

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

### Returns

Completion status of operation:

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

#### 6.1.1.34 int rtXmlDecXSIAttr (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *attrName*, const OSUTF8CHAR \* *attrValue*)

This function decodes XML schema instance (XSI) attribute.

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

### Parameters

*pctxt* Pointer to context block structure.

*attrName* Attribute's name to be decoded

*attrValue* Attribute's value to be decoded

### Returns

Completion status of operation:

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

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

This function decodes XML schema instance (XSI) attributes.

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

**Parameters**

*pctxt* Pointer to context block structure.

*attrs* Attributes-values array [attr, value]. Should be null-terminated.

*typeName* Name of parent type to add in error log, if would be necessary.

**Returns**

Completion status of operation:

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

**6.1.1.36 int rtXmlParseElementName (OSCTXT \* *pctxt*, OSUTF8CHAR \*\* *ppName*)**

This function parses the initial tag from an XML message.

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

**Parameters**

*pctxt* Pointer to OSCTXT structure

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

**Returns**

Completion status of operation:

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

**6.1.1.37 int rtXmlParseElemQName (OSCTXT \* *pctxt*, OSXMLQName \* *pQName*)**

This function parses the initial tag from an XML message.

**Parameters**

*pctxt* Pointer to OSCTXT structure

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

**Returns**

Completion status of operation:

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

## 6.2 XML encode functions.

### Defines

- #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, OSXMLNamespace \*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, OSUINT32 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, size\_t attrNameLen)  
*This function encodes a variable of the XSD base64Binary type as an attribute.*
- int `rtXmlEncBase64StrValue` (OSCTXT \*pctx, OSUINT32 nocts, const OSOCTET \*value)  
*This function encodes a variable of the XSD base64Binary type.*
- int `rtXmlEncBigInt` (OSCTXT \*pctx, const OSUTF8CHAR \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD integer type.*
- int `rtXmlEncBigIntAttr` (OSCTXT \*pctx, const OSUTF8CHAR \*value, const OSUTF8CHAR \*attrName, size\_t 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, OSUINT32 nbits, const OSOCTET \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the ASN.1 BIT STRING type.*
- int `rtXmlEncBinStrValue` (OSCTXT \*pctx, OSUINT32 nbits, const OSOCTET \*data)

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

- int [rtXmlEncBool](#) (OSCTXT \*pctxt, OSBOOL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD boolean type.*
- int [rtXmlEncBoolValue](#) (OSCTXT \*pctxt, OSBOOL value)  
*This function encodes a variable of the XSD boolean type.*
- int [rtXmlEncBoolAttr](#) (OSCTXT \*pctxt, OSBOOL value, const OSUTF8CHAR \*attrName, size\_t attrNameLen)  
*This function encodes an XSD boolean attribute value.*
- 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, size\_t 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, size\_t 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, size\_t 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, size\_t 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 \*pctx, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a numeric gMonthDay element into an XML string representation.*
- int `rtXmlEncGDay` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a numeric gDay element into an XML string representation.*
- int `rtXmlEncGYearValue` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gYear value into an XML string representation.*
- int `rtXmlEncGYearMonthValue` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gYearMonth value into an XML string representation.*
- int `rtXmlEncGMonthValue` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gMonth value into an XML string representation.*
- int `rtXmlEncGMonthDayValue` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gMonthDay value into an XML string representation.*
- int `rtXmlEncGDayValue` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gDay value into an XML string representation.*
- int `rtXmlEncHexBinary` (OSCTXT \*pctx, OSUINT32 noctx, const OSOCTET \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD hexBinary type.*
- int `rtXmlEncHexBinaryAttr` (OSCTXT \*pctx, OSUINT32 noctx, const OSOCTET \*value, const OSUTF8CHAR \*attrName, size\_t attrNameLen)  
*This function encodes a variable of the XSD hexBinary type as an attribute.*
- int `rtXmlEncHexStrValue` (OSCTXT \*pctx, OSUINT32 noctx, const OSOCTET \*data)  
*This function encodes a variable of the XSD hexBinary type.*
- int `rtXmlEncIndent` (OSCTXT \*pctx)  
*This function adds indentation whitespace to the output stream.*
- int `rtXmlEncInt` (OSCTXT \*pctx, OSINT32 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD integer type.*
- int `rtXmlEncIntValue` (OSCTXT \*pctx, OSINT32 value)  
*This function encodes a variable of the XSD integer type.*
- int `rtXmlEncIntAttr` (OSCTXT \*pctx, OSINT32 value, const OSUTF8CHAR \*attrName, size\_t attrNameLen)  
*This function encodes a variable of the XSD integer type as an attribute (name="value").*
- int `rtXmlEncIntPattern` (OSCTXT \*pctx, 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, size\_t attrNameLen)  
*This function encodes a variable of the XSD integer type as an attribute (name="value").*
- int [rtXmlEncNamedBits](#) (OSCTXT \*pctxt, const OSBitMapItem \*pBitMap, OSUINT32 nbits, const OSOCTET \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the ASN.1 BIT STRING type.*
- 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, size\_t 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 \*pctx, const OSUTF8CHAR \*value)  
*This function encodes a variable of the XSD string type.*
- int [rtXmlEncStringValue2](#) (OSCTXT \*pctx, const OSUTF8CHAR \*value, size\_t valueLen)  
*This function encodes a variable of the XSD string type.*
- int [rtXmlEncTermStartElement](#) (OSCTXT \*pctx)  
*This function terminates a currently open XML start element by adding either a '>' or '/>' (if empty) terminator.*
- int [rtXmlEncUnicodeStr](#) (OSCTXT \*pctx, const OSUNICHAR \*value, OSUINT32 nchars, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a Unicode string value.*
- int [rtXmlEncUTF8Attr](#) (OSCTXT \*pctx, const OSUTF8CHAR \*name, const OSUTF8CHAR \*value)  
*This function encodes an attribute in which the name and value are given as a null-terminated UTF-8 strings.*
- int [rtXmlEncUTF8Attr2](#) (OSCTXT \*pctx, const OSUTF8CHAR \*name, size\_t nameLen, const OSUTF8CHAR \*value, size\_t valueLen)  
*This function encodes an attribute in which the name and value are given as a UTF-8 strings with lengths.*
- int [rtXmlEncUTF8Str](#) (OSCTXT \*pctx, const OSUTF8CHAR \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a UTF-8 string value.*
- int [rtXmlEncUInt](#) (OSCTXT \*pctx, OSUINT32 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD unsigned integer type.*
- int [rtXmlEncUIntValue](#) (OSCTXT \*pctx, OSUINT32 value)  
*This function encodes a variable of the XSD unsigned integer type.*
- int [rtXmlEncUIntAttr](#) (OSCTXT \*pctx, OSUINT32 value, const OSUTF8CHAR \*attrName, size\_t attrNameLen)  
*This function encodes a variable of the XSD unsigned integer type as an attribute (name="value").*
- int [rtXmlEncUInt64](#) (OSCTXT \*pctx, OSUINT64 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD integer type.*
- int [rtXmlEncUInt64Value](#) (OSCTXT \*pctx, OSUINT64 value)  
*This function encodes a variable of the XSD integer type.*
- int [rtXmlEncUInt64Attr](#) (OSCTXT \*pctx, OSUINT64 value, const OSUTF8CHAR \*attrName, size\_t attrNameLen)  
*This function encodes a variable of the XSD integer type as an attribute (name="value").*
- int [rtXmlEncXSIAttrs](#) (OSCTXT \*pctx, OSBOOL needXSI)  
*This function encodes XML schema instance (XSI) attributes at the beginning of an XML document.*
- int [rtXmlEncXSITypeAttr](#) (OSCTXT \*pctx, const OSUTF8CHAR \*value)  
*This function encodes an XML schema instance (XSI) type attribute value (xsi:type="value").*

- int [rtXmlEncXSITypeAttr2](#) (OSCTXT \*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, size\_t 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 Define Documentation

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

This macro returns the length of the encoded XML message.

#### Parameters

*pctxt* Pointer to a context structure.

Definition at line 2492 of file osrtxml.h.

### 6.2.1.2 #define rtXmlGetEncBufPtr(pctxt) (pctxt)->buffer.data

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

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

#### Parameters

*pctxt* Pointer to a context structure.

Definition at line 2485 of file osrtxml.h.

## 6.2.2 Function Documentation

### 6.2.2.1 `int rtXmlEncAny (OSCTXT * pctxt, OSXMLSTRING * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)`

This function encodes a variable of the XSD any type.

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

#### Parameters

*pctxt* Pointer to context block structure.

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

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

*pNS* Pointer to namespace structure.

#### Returns

Completion status of operation:

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

### 6.2.2.2 `int rtXmlEncAnyAttr (OSCTXT * pctxt, OSRTDList * pAnyAttrList)`

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

#### Parameters

*pctxt* Pointer to context block structure.

*pAnyAttrList* List of attributes.

#### Returns

Completion status of operation:

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

### 6.2.2.3 `int rtXmlEncAnyTypeValue (OSCTXT * pctxt, const OSUTF8CHAR * pvalue)`

This function encodes a variable of the XSD anyType type.

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

#### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Value to be encoded.

## Returns

Completion status of operation:

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

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

This function encodes a variable of the XSD base64Binary type.

## Parameters

*pctxt* Pointer to context block structure.

*nocts* Number of octets in the value string.

*value* Value to be encoded.

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

*pNS* Pointer to namespace structure.

## Returns

Completion status of operation:

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

### 6.2.2.5 int rtXmlEncBase64BinaryAttr (OSCTXT \* *pctxt*, OSUINT32 *nocts*, const OSOCTET \* *value*, const OSUTF8CHAR \* *attrName*, size\_t *attrNameLen*)

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

## Parameters

*pctxt* Pointer to context block structure.

*nocts* Number of octets in the value string.

*value* Value to be encoded.

*attrName* XML attribute name. A name must be provided.

*attrNameLen* Length in bytes of the attribute name.

## Returns

Completion status of operation:

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

#### 6.2.2.6 **int rtXmlEncBase64StrValue (OSCTXT \* *pctxt*, OSUINT32 *nocts*, const OSOCTET \* *value*)**

This function encodes a variable of the XSD base64Binary type.

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

##### **Parameters**

*pctxt* Pointer to context block structure.

*nocts* Number of octets in the value string.

*value* Value to be encoded.

##### **Returns**

Completion status of operation:

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

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

This function encodes a variable of the XSD integer type.

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

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

##### **Parameters**

*pctxt* Pointer to context block structure.

*value* A pointer to a character string containing the value to be encoded.

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

*pNS* Pointer to namespace structure.

##### **Returns**

Completion status of operation:

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

#### 6.2.2.8 **int rtXmlEncBigIntAttr (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *value*, const OSUTF8CHAR \* *attrName*, size\_t *attrNameLen*)**

This function encodes an XSD integer attribute value.

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

##### **Parameters**

*pctxt* Pointer to context block structure.

*value* A pointer to a character string containing the value to be encoded.

*attrName* XML attribute name. A name must be provided.

*attrNameLen* Length in bytes of the attribute name.

### Returns

Completion status of operation:

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

#### 6.2.2.9 int rtXmlEncBigIntValue (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *value*)

This function encodes an XSD integer attribute value.

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

### Parameters

*pctxt* Pointer to context block structure.

*value* A pointer to a character string containing the value to be encoded.

### Returns

Completion status of operation:

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

#### 6.2.2.10 int rtXmlEncBinStrValue (OSCTXT \* *pctxt*, OSUINT32 *nbits*, const OSOCTET \* *data*)

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

### Parameters

*pctxt* Pointer to context block structure.

*nbits* Number of bits in the value string.

*data* Value to be encoded.

### Returns

Completion status of operation:

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



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

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

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

**See also**

[rtXmlEncNamedBits](#)).

**Parameters**

*pctxt* Pointer to context block structure.

*nbits* Number of bits in the bit string.

*value* Value to be encoded.

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

*pNS* Pointer to namespace structure.

**Returns**

Completion status of operation:

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

**6.2.2.12 int rtXmlEncBOM (OSCTXT \* *pctxt*)**

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

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

**Parameters**

*pctxt* Pointer to context block structure.

**Returns**

Completion status of operation:

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

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

This function encodes a variable of the XSD boolean type.

**Parameters**

*pctxt* Pointer to context block structure.

*value* Boolean value to be encoded.

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

*pNS* Pointer to namespace structure.

### Returns

Completion status of operation:

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

#### 6.2.2.14 **int rtXmlEncBoolAttr (OSCTXT \* *pctxt*, OSBOOL *value*, const OSUTF8CHAR \* *attrName*, size\_t *attrNameLen*)**

This function encodes an XSD boolean attribute value.

### Parameters

*pctxt* Pointer to context block structure.

*value* Boolean value to be encoded.

*attrName* XML attribute name. A name must be provided.

*attrNameLen* Length in bytes of the attribute name.

### Returns

Completion status of operation:

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

#### 6.2.2.15 **int rtXmlEncBoolValue (OSCTXT \* *pctxt*, OSBOOL *value*)**

This function encodes a variable of the XSD boolean type.

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

### Parameters

*pctxt* Pointer to context block structure.

*value* Boolean value to be encoded.

### Returns

Completion status of operation:

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

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

This function encodes an XML comment.

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

##### Parameters

*pctxt* Pointer to context block structure.

*comment* The comment text.

##### Returns

Completion status of operation:

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

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

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* OSXSDDateTime type pointer points to a OSXSDDateTime value to be encoded.

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

*pNS* Pointer to namespace structure.

##### Returns

Completion status of operation:

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

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

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

*pNS* Pointer to namespace structure.

## Returns

Completion status of operation:

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

### 6.2.2.19 int rtXmlEncDateTimeValue (OSCTXT \* *pctxt*, const OSXSDDateTime \* *pvalue*)

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

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

## Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

## Returns

Completion status of operation:

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

### 6.2.2.20 int rtXmlEncDateValue (OSCTXT \* *pctxt*, const OSXSDDateTime \* *pvalue*)

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

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

## Parameters

*pctxt* Pointer to context block structure.

*pvalue* OSXSDDateTime type pointer points to a OSXSDDateTime value to be encoded.

## Returns

Completion status of operation:

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

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

This function encodes a variable of the XSD decimal type.

## Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

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

*pNS* Pointer to namespace structure.

*pFmtSpec* Pointer to format specification structure.

### Returns

Completion status of operation:

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

#### 6.2.2.22 `int rtXmlEncDecimalAttr (OSCTXT * pctxt, OSREAL value, const OSUTF8CHAR * attrName, size_t attrNameLen, const OSDecimalFmt * pFmtSpec)`

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

### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

*attrName* XML attribute name. A name must be provided.

*attrNameLen* Length of XML attribute name.

*pFmtSpec* Pointer to format specification structure.

### Returns

Completion status of operation:

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

#### 6.2.2.23 `int rtXmlEncDecimalValue (OSCTXT * pctxt, OSREAL value, const OSDecimalFmt * pFmtSpec, char * pDestBuf, size_t destBufSize)`

This function encodes a value of the XSD decimal type.

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

### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

*pFmtSpec* Pointer to format specification structure.

*pDestBuf* Pointer to a destination buffer. If NULL (destBufSize should be 0) the encoded value will be put in pctxt->buffer or in stream associated with the pctxt.

*destBufSize* The size of the destination buffer. Must be 0, if pDestBuf is NULL.

### Returns

Completion status of operation:

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

**6.2.2.24 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**

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

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

*pNS* Pointer to namespace structure.

*pFmtSpec* Pointer to format specification structure.

**Returns**

Completion status of operation:

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

**6.2.2.25 int rtXmlEncDoubleAttr (OSCTXT \* *pctxt*, OSREAL *value*, const OSUTF8CHAR \* *attrName*, size\_t *attrNameLen*, const OSDoubleFmt \* *pFmtSpec*)**

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

**Parameters**

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

*attrName* XML attribute name. A name must be provided.

*attrNameLen* Length of XML attribute name.

*pFmtSpec* Pointer to format specification structure.

**Returns**

Completion status of operation:

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

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

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

*pFmtSpec* Pointer to format specification structure.

*defaultPrecision* Default precision of the value. For float, it is 6, for double it is 15.

## Returns

Completion status of operation:

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

### 6.2.2.27 `int rtXmlEncDoubleValue (OSCTXT * pctxt, OSREAL value, const OSDoubleFmt * pFmtSpec, int defaultPrecision)`

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

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

## Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

*pFmtSpec* Pointer to format specification structure.

*defaultPrecision* Default precision of the value. For float, it is 6, for double it is 15.

## Returns

Completion status of operation:

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

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

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

## Parameters

*pctxt* Pointer to context block structure.

*elemName* XML element name.

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

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

*terminate* Add closing '>' character.

## Returns

Completion status of operation:

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

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

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

#### Parameters

*pctxt* Pointer to context block structure.

#### Returns

Completion status of operation:

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

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

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

#### Parameters

*pctxt* Pointer to context block structure.

*elemName* XML element name.

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

#### Returns

Completion status of operation:

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

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

This function encodes SOAP end element tags.

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

#### Parameters

*pctxt* Pointer to context block structure.

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

#### Returns

Completion status of operation:

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



### 6.2.2.32 int rtXmlEncEndSoapEnv (OSCTXT \* *pctxt*)

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

#### Parameters

*pctxt* Pointer to context block structure.

#### Returns

Completion status of operation:

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

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

This function encodes a variable of the XSD float type.

#### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

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

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

*pFmtSpec* Pointer to format specification structure.

#### Returns

Completion status of operation:

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

### 6.2.2.34 int rtXmlEncFloatAttr (OSCTXT \* *pctxt*, OSREAL *value*, const OSUTF8CHAR \* *attrName*, size\_t *attrNameLen*, const OSDoubleFmt \* *pFmtSpec*)

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

#### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

*attrName* XML attribute name. A name must be provided.

*attrNameLen* Length of XML attribute name.

*pFmtSpec* Pointer to format specification structure.

#### Returns

Completion status of operation:

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

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

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

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

**Returns**

Completion status of operation:

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

**6.2.2.36** `int rtXmlEncGDayValue (OSCTXT * pctxt, const OSXSDDateTime * pvalue)`

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

**Returns**

Completion status of operation:

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

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

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

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

**Returns**

Completion status of operation:

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

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

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

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

**Returns**

Completion status of operation:

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

**6.2.2.39 int rtXmlEncGMonthDayValue (OSCTXT \* *pctxt*, const OSXSDDateTime \* *pvalue*)**

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

**Returns**

Completion status of operation:

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

**6.2.2.40 int rtXmlEncGMonthValue (OSCTXT \* *pctxt*, const OSXSDDateTime \* *pvalue*)**

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

**Returns**

Completion status of operation:

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

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

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

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

**Returns**

Completion status of operation:

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

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

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

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

**Returns**

Completion status of operation:

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

**6.2.2.43 int rtXmlEncGYearMonthValue (OSCTXT \* *pctxt*, const OSXSDDateTime \* *pvalue*)**

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

**Returns**

Completion status of operation:

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

#### 6.2.2.44 **int rtXmlEncGYearValue (OSCTXT \* *pctxt*, const OSXSDDateTime \* *pvalue*)**

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

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

##### **Parameters**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to value to be encoded.

##### **Returns**

Completion status of operation:

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

#### 6.2.2.45 **int rtXmlEncHexBinary (OSCTXT \* *pctxt*, OSUINT32 *nocts*, const OSOCTET \* *value*, const OSUTF8CHAR \* *elemName*, OSXMLNamespace \* *pNS*)**

This function encodes a variable of the XSD hexBinary type.

##### **Parameters**

*pctxt* Pointer to context block structure.

*nocts* Number of octets in the value string.

*value* Value to be encoded.

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

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

##### **Returns**

Completion status of operation:

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

#### 6.2.2.46 **int rtXmlEncHexBinaryAttr (OSCTXT \* *pctxt*, OSUINT32 *nocts*, const OSOCTET \* *value*, const OSUTF8CHAR \* *attrName*, size\_t *attrNameLen*)**

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

##### **Parameters**

*pctxt* Pointer to context block structure.

*nocts* Number of octets in the value string.

*value* Value to be encoded.

*attrName* XML attribute name. A name must be provided.

*attrNameLen* Length of XML attribute name.

## Returns

Completion status of operation:

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

### 6.2.2.47 `int rtXmlEncHexStrValue (OSCTXT * pctxt, OSUINT32 nocts, const OSOCTET * data)`

This function encodes a variable of the XSD hexBinary type.

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

## Parameters

*pctxt* Pointer to context block structure.

*nocts* Number of octets in the value string.

*data* Value to be encoded.

## Returns

Completion status of operation:

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

### 6.2.2.48 `int rtXmlEncIndent (OSCTXT * pctxt)`

This function adds indentation whitespace to the output stream.

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

## Parameters

*pctxt* Pointer to context block structure.

## Returns

Completion status of operation:

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

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

This function encodes a variable of the XSD integer type.

## Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

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

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

### Returns

Completion status of operation:

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

#### 6.2.2.50 `int rtXmlEncInt64 (OSCTXT * pctxt, OSINT64 value, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)`

This function encodes a variable of the XSD integer type.

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

### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

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

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

### Returns

Completion status of operation:

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

#### 6.2.2.51 `int rtXmlEncInt64Attr (OSCTXT * pctxt, OSINT64 value, const OSUTF8CHAR * attrName, size_t attrNameLen)`

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

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

### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

*attrName* XML attribute name.

*attrNameLen* Length (in bytes) of the attribute name.

### Returns

Completion status of operation:

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

#### 6.2.2.52 `int rtXmlEncInt64Value (OSCTXT * pctxt, OSINT64 value)`

This function encodes a variable of the XSD integer type.

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

##### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

##### Returns

Completion status of operation:

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

#### 6.2.2.53 `int rtXmlEncIntAttr (OSCTXT * pctxt, OSINT32 value, const OSUTF8CHAR * attrName, size_t attrNameLen)`

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

##### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

*attrName* XML attribute name.

*attrNameLen* Length (in bytes) of the attribute name.

##### Returns

Completion status of operation:

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

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

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

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

*pattern* Pattern of the encoded value.



## Returns

Completion status of operation:

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

### 6.2.2.55 `int rtXmlEncIntValue (OSCTXT * pctxt, OSINT32 value)`

This function encodes a variable of the XSD integer type.

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

## Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

## Returns

Completion status of operation:

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

### 6.2.2.56 `int rtXmlEncNamedBits (OSCTXT * pctxt, const OSBitMapItem * pBitMap, OSUINT32 nbits, const OSOCTET * pvalue, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)`

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

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

## Parameters

*pctxt* Pointer to context block structure.

*pBitMap* Bit map equating symbolic bit names to bit numbers.

*nbits* Number of bits in the bit string value.

*pvalue* Bit string value to be encoded.

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

*pNS* Pointer to namespace structure.

## Returns

Completion status of operation:

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

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

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

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

#### **Parameters**

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

#### **Returns**

Completion status of operation:

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

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

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

#### **Parameters**

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

#### **Returns**

Completion status of operation:

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

### 6.2.2.59 **int rtXmlEncSoapArrayTypeAttr (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *name*, const OSUTF8CHAR \* *value*, size\_t *itemCount*)**

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

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

#### **Parameters**

- pctxt* Pointer to context block structure.
- name* Attribute name (NS prefix + arrayType)
- value* UTF-8 string value to be encoded.

*itemCount* Count of the number of elements in the array.

### Returns

Completion status of operation:

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

### 6.2.2.60 int rtXmlEncStartDocument (OSCTXT \* *pctxt*)

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

This is normally the following:

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

### Parameters

*pctxt* Pointer to context block structure.

### Returns

Completion status of operation:

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

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

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

### Parameters

*pctxt* Pointer to context block structure.

*elemName* XML element name. Empty string and null are treated equivalently.

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

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

*terminate* Add closing '>' character.

### Returns

Completion status of operation:

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

### 6.2.2.62 `int rtXmlEncStartSoapElems (OSCTXT * pctxt, OSXMLSOAPMsgType msgtype)`

This function encodes a SOAP envelope start element tag and an optional SOAP body or fault tag. This includes all of the standard SOAP namespace attributes.

#### Parameters

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

#### Returns

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

### 6.2.2.63 `int rtXmlEncStartSoapEnv (OSCTXT * pctxt, OSRTDList * pNSAttrs)`

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

#### Parameters

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

#### Returns

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

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

This function encodes a variable of the XSD string type.

#### Parameters

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

#### Returns

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

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

This function encodes a variable of the XSD string type.

##### Parameters

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

##### Returns

Completion status of operation:

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

#### 6.2.2.66 `int rtXmlEncStringValue2 (OSCTXT * pctxt, const OSUTF8CHAR * value, size_t valueLen)`

This function encodes a variable of the XSD string type.

##### Parameters

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

##### Returns

Completion status of operation:

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

#### 6.2.2.67 `int rtXmlEncTermStartElement (OSCTXT * pctxt)`

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

##### Parameters

- pctxt* Pointer to context block structure.

##### Returns

Completion status of operation:

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

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

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*pvalue* OSXSDDateTime type pointer points to a OSXSDDateTime value to be encoded.

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

*pNS* Pointer to namespace structure.

**Returns**

Completion status of operation:

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

**6.2.2.69 int rtXmlEncTimeValue (OSCTXT \* *pctxt*, const OSXSDDateTime \* *pvalue*)**

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*pvalue* OSXSDDateTime type pointer points to a OSXSDDateTime value to be encoded.

**Returns**

Completion status of operation:

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

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

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

**Parameters**

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

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

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

### Returns

Completion status of operation:

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

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

This function encodes a variable of the XSD integer type.

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

### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

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

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

### Returns

Completion status of operation:

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

**6.2.2.72** `int rtXmlEncUInt64Attr (OSCTXT * pctxt, OSUINT64 value, const OSUTF8CHAR * attrName, size_t attrNameLen)`

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

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

### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

*attrName* XML attribute name.

*attrNameLen* Length (in bytes) of the attribute name.

### Returns

Completion status of operation:

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

### 6.2.2.73 `int rtXmlEncUInt64Value (OSCTXT * pctxt, OSUINT64 value)`

This function encodes a variable of the XSD integer type.

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

#### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

#### Returns

Completion status of operation:

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

### 6.2.2.74 `int rtXmlEncUIntAttr (OSCTXT * pctxt, OSUINT32 value, const OSUTF8CHAR * attrName, size_t attrNameLen)`

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

#### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

*attrName* XML attribute name.

*attrNameLen* Length (in bytes) of the attribute name.

#### Returns

Completion status of operation:

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

### 6.2.2.75 `int rtXmlEncUIntValue (OSCTXT * pctxt, OSUINT32 value)`

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

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

#### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

#### Returns

Completion status of operation:

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



**6.2.2.76** `int rtXmlEncUnicodeStr (OSCTXT * pctxt, const OSUNICHAR * value, OSUINT32 nchars, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)`

This function encodes a Unicode string value.

#### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded. This is a pointer to an array of 16-bit integer values.

*nchars* Number of characters in value array.

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

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

#### Returns

Completion status of operation:

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

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

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

#### Parameters

*pctxt* Pointer to context block structure.

*name* Attribute name.

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

#### Returns

Completion status of operation:

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

**6.2.2.78** `int rtXmlEncUTF8Attr2 (OSCTXT * pctxt, const OSUTF8CHAR * name, size_t nameLen, const OSUTF8CHAR * value, size_t valueLen)`

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

#### Parameters

*pctxt* Pointer to context block structure.

*name* Attribute name.

*nameLen* Attribute name length (in octets).

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

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

## Returns

Completion status of operation:

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

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

This function encodes a UTF-8 string value.

## Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

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

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

## Returns

Completion status of operation:

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

**6.2.2.80** `int rtXmlEncXSIAttrs (OSCTXT * pctxt, OSBOOL needXSI)`

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

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

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

## Parameters

*pctxt* Pointer to context block structure.

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

## Returns

Completion status of operation:

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

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

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

#### Parameters

*pctxt* Pointer to context block structure.

#### Returns

Completion status of operation:

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

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

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

#### Parameters

*pctxt* Pointer to context block structure.

*value* XSI type attribute value.

#### Returns

Completion status of operation:

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

### 6.2.2.83 `int rtXmlEncXSITypeAttr2 (OSCTXT * pctxt, const OSUTF8CHAR * typeNsUri, const OSUTF8CHAR * typeName)`

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

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

#### Parameters

*pctxt* Pointer to context block structure.

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

*typeName* The type's name.

#### Returns

Completion status of operation:

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

#### **6.2.2.84 int rtXmlFreeInputSource (OSCTXT \* *pctxt*)**

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

##### **Parameters**

*pctxt* Pointer to context block structure.

##### **Returns**

Completion status of operation:

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

#### **6.2.2.85 int rtXmlGetIndent (OSCTXT \* *pctxt*)**

This function returns current XML output indent value.

##### **Parameters**

*pctxt* Pointer to OSCTXT structure

##### **Returns**

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

#### **6.2.2.86 int rtXmlGetIndentChar (OSCTXT \* *pctxt*)**

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

##### **Parameters**

*pctxt* Pointer to OSCTXT structure

##### **Returns**

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

#### **6.2.2.87 OSBOOL rtXmlGetWriteBOM (OSCTXT \* *pctxt*)**

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

##### **Parameters**

*pctxt* Pointer to OSCTXT structure.

##### **Returns**

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

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

This function prints a list of namespace attributes.

#### Parameters

*name* Name to print.

*data* List of namespace attributes.

#### Returns

Completion status of operation:

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

### 6.2.2.89 int rtXmlSetEncBufPtr (OSCTXT \* ptxt, OSOCTET \* bufaddr, size\_t bufsiz)

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

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

#### Parameters

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

*bufaddr* A pointer to a memory buffer to use to encode a message. The buffer should be declared as an array of unsigned characters (OCTETs). This parameter can be set to NULL to specify dynamic encoding (i.e., the encode functions will dynamically allocate a buffer to hold the encoded message).

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

## 6.3 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 Function Documentation

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

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

#### Parameters

*pctxt* Pointer to OSCTXT structure.

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

#### Returns

0 - if success, negative value if error.

## 6.4 XML pull-parser decode functions.

### Functions

- 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, size\_t 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, OSINT32 bufsize)  
*This function decodes a contents of a Base64-encode binary string into a static memory structure.*
- 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 [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 [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 [rtXmlpDecDynUnicodeStr](#) (OSCTXT \*pctxt, const OSUNICHAR \*\*ppdata, OSUINT32 \*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, size\_t 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, OSINT32 bufsize)

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

- 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 [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 OS-UTF8CHAR \*\*ppAttrValue)  
*This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).*
- int [rtXmlpGetAttributeID](#) (const OSXMLStrFragment \*attrName, OSINT16 nsidx, size\_t nAttr, const OSXMLAttrDescr attrNames[ ], OSUINT32 attrPresent[ ])  
*This function finds an attribute in the descriptor table.*
- int [rtXmlpGetNextElem](#) (OSCTXT \*pctxt, OSXMLElemDescr \*pElem, OSINT32 level)  
*This function parse the next element start tag.*
- int [rtXmlpGetNextElemID](#) (OSCTXT \*pctxt, const OSXMLElemIDRec \*tab, size\_t nrows, OSINT32 level, OSBOOL continueParse)  
*This function parses the next start tag and finds the index of the element name in the descriptor table.*
- int [rtXmlpMarkLastEventActive](#) (OSCTXT \*pctxt)  
*This function marks current tag as unprocessed.*
- int [rtXmlpMatchStartTag](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*elemLocalName, OSINT16 nsidx)  
*This function parses the next start tag that matches with given name.*
- int [rtXmlpMatchEndTag](#) (OSCTXT \*pctxt, OSINT32 level)  
*This function parse next end tag that matches with given name.*
- OSBOOL [rtXmlpHasAttributes](#) (OSCTXT \*pctxt)  
*This function checks accessibility of attributes.*
- int [rtXmlpGetAttributeCount](#) (OSCTXT \*pctxt)

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

- int [rtXmIplSelectAttribute](#) (OSCTXT \*pctxt, OSXMLNameFragments \*pAttr, OSINT16 \*nsidx, size\_t 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 \*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.*
- int [rtXmIplGetNextSeqElemIDExt](#) (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 rtXmIplGetNextSeqElemID.*
- int [rtXmIplGetNextAllElemID](#) (OSCTXT \*pctxt, const OSXMLElemIDRec \*tab, size\_t nRows, const OSUINT8 \*pOrder, OSUINT32 nOrder, OSUINT32 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, size\_t nRows, const OSUINT16 \*pOrder, OSUINT32 nOrder, OSUINT32 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, size\_t nRows, const OSUINT32 \*pOrder, OSUINT32 nOrder, OSUINT32 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[], size\_t nNamespaces)

*Sets user namespace table.*

- int [rtXmlpCreateReader](#) (OSCTXT \*pctx)  
*Creates pull parser reader structure within the context.*
- void [rtXmlpHideAttributes](#) (OSCTXT \*pctx)  
*Disable access to attributes.*
- OSBOOL [rtXmlpNeedDecodeAttributes](#) (OSCTXT \*pctx)  
*This function checks if attributes were previously decoded.*
- void [rtXmlpMarkPos](#) (OSCTXT \*pctx)  
*Save current decode position.*
- void [rtXmlpRewindToMarkedPos](#) (OSCTXT \*pctx)  
*Rewind to saved decode position.*
- void [rtXmlpResetMarkedPos](#) (OSCTXT \*pctx)  
*Reset saved decode position.*
- int [rtXmlpGetXSITypeAttr](#) (OSCTXT \*pctx, const OSUTF8CHAR \*\*ppAttrValue, OSINT16 \*nsidx, size\_t \*pLocalOffs)  
*This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).*
- int [rtXmlpGetXmlnsAttrs](#) (OSCTXT \*pctx, OSRTDList \*pNSAttrs)  
*This function decodes namespace attributes from start tag and adds them to the given list.*
- int [rtXmlpDecXSIAttrs](#) (OSCTXT \*pctx)  
*This function decodes XSI (XML Schema Instance) that may be present in any arbitrary XML element within a document.*
- OSBOOL [rtXmlpIsEmptyElement](#) (OSCTXT \*pctx)  
*Check element content: empty or not.*
- int [rtXmlEncAttrC14N](#) (OSCTXT \*pctx)  
*This function used only in C14 mode.*
- struct OSXMLReader \* [rtXmlpGetReader](#) (OSCTXT \*pctx)  
*This function fetches the XML reader structure from the context for use in low-level pull parser calls.*
- OSBOOL [rtXmlpIsLastEventDone](#) (OSCTXT \*pctx)  
*Check processing status of current tag.*
- int [rtXmlpGetXSITypeIndex](#) (OSCTXT \*pctx, const OSXMLItemDescr typetab[ ], size\_t 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[ ], size\_t 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 [rtXmlIsDecodeAsGroup](#) (OSCTXT \*pctxt)  
*This function checks if "decode as group" mode was forced.*
- OSBOOL [rtXmlIsUTF8Encoding](#) (OSCTXT \*pctxt)  
*This function checks if the encoding specified in XML header is UTF-8.*
- int [rtXmlReadBytes](#) (OSCTXT \*pctxt, OSOCTET \*pbuf, size\_t nbytes)  
*This function reads the specified number of bytes directly from the underlying XML parser stream.*

## 6.4.1 Function Documentation

### 6.4.1.1 int rtXmlEncAttrC14N (OSCTXT \* pctxt)

This function used only in C14 mode.

It provide attributes sorting.

#### Parameters

*pctxt* Pointer to context block structure.

#### Returns

Completion status of operation:

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

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

Count tokens in list.

#### Parameters

*pctxt* Pointer to context block structure.

*itemCnt* Pointer to number of elements in list.

#### Returns

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

### 6.4.1.3 int rtXmlpCreateReader (OSCTXT \* pctxt)

Creates pull parser reader structure within the context.

#### Parameters

*pctxt* Pointer to context block structure.

## Returns

Completion status of operation:

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

### 6.4.1.4 `int rtXmlpDecAny (OSCTXT * pctxt, const OSUTF8CHAR ** pvalue)`

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

The decoded XML fragment is returned as a string in the form as it appears in the document. Memory is allocated for the string using the `rtxMemAlloc` function.

## Parameters

*pctxt* Pointer to context block structure.

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

## Returns

Completion status of operation:

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

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

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

## Parameters

*pctxt* Pointer to context block structure.

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

## Returns

Completion status of operation:

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

### 6.4.1.6 `int rtXmlpDecAnyAttrStr (OSCTXT * pctxt, const OSUTF8CHAR ** ppAttrStr, size_t attrIndex)`

This function decodes an any attribute string.

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

## Parameters

*pctxt* Pointer to context block structure.

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

*attrIndex* Index of attribute.

## Returns

Completion status of operation:

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

### 6.4.1.7 int rtXmlpDecAnyElem (OSCTXT \* *pctxt*, const OSUTF8CHAR \*\* *pvalue*)

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

The decoded XML fragment is returned as a string in the form as it appears in the document. Memory is allocated for the string using the rtxMemAlloc function. The difference between this function and rtXmlpDecAny is that this function preserves the full encoded XML fragment including the start and end elements tags and attributes. rtXmlpDecAny decodes the contents within the start and end tags.

## Parameters

*pctxt* Pointer to context block structure.

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

## Returns

Completion status of operation:

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

### 6.4.1.8 int rtXmlpDecBase64Str (OSCTXT \* *pctxt*, OSOCTET \* *pvalue*, OSUINT32 \* *pnocets*, OSINT32 *bufsize*)

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

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

## Parameters

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

*pvalue* A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.

*pnocets* A pointer to an integer value to receive the decoded number of octets.

*bufsize* The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

## Returns

Completion status of operation:

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

#### 6.4.1.9 int rtXmlpDecBigInt (OSCTXT \* *pctxt*, const OSUTF8CHAR \*\* *pvalue*)

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

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

These variables are stored in character string constant variables. The radix should be 10. If it is necessary to convert to another radix, then use `rtxBigIntSetStr` or `rtxBigIntToString` functions. Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

##### Parameters

*pctxt* Pointer to context block structure.

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

##### Returns

Completion status of operation:

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

#### 6.4.1.10 int rtXmlpDecBitString (OSCTXT \* *pctxt*, OSOCTET \* *pvalue*, OSUINT32 \* *pnbits*, OSUINT32 *bufsize*)

This function decodes a bit string value.

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

##### Parameters

*pctxt* Pointer to context block structure.

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

*pnbits* Pointer to hold decoded number of bits.

*bufsize* Size of buffer passed in *pvalue* argument.

##### Returns

Completion status of operation:

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

#### 6.4.1.11 int rtXmlpDecBool (OSCTXT \* *pctxt*, OSBOOL \* *pvalue*)

This function decodes a variable of the boolean type.

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

##### Parameters

*pctxt* Pointer to context block structure.

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

#### Returns

Completion status of operation:

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

#### 6.4.1.12 int rtXmlpDecDate (OSCTXT \* *pctxt*, OSXSDDateTime \* *pvalue*)

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

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

#### Parameters

*pctxt* Pointer to context block structure.

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

#### Returns

Completion status of operation:

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

#### 6.4.1.13 int rtXmlpDecDateTime (OSCTXT \* *pctxt*, OSXSDDateTime \* *pvalue*)

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

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

#### Parameters

*pctxt* Pointer to context block structure.

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

#### Returns

Completion status of operation:

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

#### 6.4.1.14 int rtXmlpDecDecimal (OSCTXT \* *pctxt*, OSREAL \* *pvalue*, int *totalDigits*, int *fractionDigits*)

This function decodes the contents of a decimal data type.

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

#### Parameters

*pctxt* Pointer to context block structure.



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

*totalDigits* Number of total digits in the decimal number from XSD totalDigits facet value. Argument should be set to -1 if facet not given.

*fractionDigits* Number of fraction digits in the decimal number from XSD fractionDigits facet value. Argument should be set to -1 if facet not given.

## Returns

Completion status of operation:

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

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

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

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

## Parameters

*pctxt* Pointer to context block structure.

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

## Returns

Completion status of operation:

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

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

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

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

## Parameters

*pctxt* Pointer to context block structure.

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

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

## Returns

Completion status of operation:

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

#### 6.4.1.17 `int rtXmlpDecDynBase64Str (OSCTXT * pctxt, OSDynOctStr * pvalue)`

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

The octet string must be Base64 encoded. This function will allocate dynamic memory to store the decoded result. Input is expected to be a string of OSUTF8CHAR characters returned by an XML pull parser.

##### Parameters

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

*pvalue* A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated for the string using the `rtxMemAlloc` function.

##### Returns

Completion status of operation:

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

#### 6.4.1.18 `int rtXmlpDecDynBitString (OSCTXT * pctxt, OSDynOctStr * pvalue)`

This function decodes a bit string value.

The string consists of a series of '1' and '0' characters. This is the dynamic version in which memory is allocated for the returned binary string variable. Bits are stored from MSB to LSB order.

##### Parameters

*pctxt* Pointer to context block structure.

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

##### Returns

Completion status of operation:

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

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

This function decodes a contents of a hexBinary string.

This function will allocate dynamic memory to store the decoded result. Input is expected to be a string of OS-UTF8CHAR characters returned by an XML pull parser.

##### Parameters

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

*pvalue* A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the `rtxMemAlloc` function.

## Returns

Completion status of operation:

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

### 6.4.1.20 `int rtXmlpDecDynUnicodeStr (OSCTXT * pctxt, const OSUNICHAR ** ppdata, OSUINT32 * pchars)`

This function decodes a Unicode string data type.

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

## Parameters

*pctxt* Pointer to context block structure.

*ppdata* Pointer to Unicode character string. A Unicode character string is represented as an array of unsigned 16-bit integers in C. Memory is allocated for the string using the run-time memory manager.

*pchars* 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.

### 6.4.1.21 `int rtXmlpDecDynUTF8Str (OSCTXT * pctxt, const OSUTF8CHAR ** outdata)`

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

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

## Parameters

*pctxt* Pointer to context block structure.

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

## Returns

Completion status of operation:

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

### 6.4.1.22 `int rtXmlpDecGDay (OSCTXT * pctxt, OSXSDDateTime * pvalue)`

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

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

## Parameters

*pctxt* Pointer to context block structure.

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

## Returns

Completion status of operation:

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

### 6.4.1.23 int rtXmlpDecGMonth (OSCTXT \* *pctxt*, OSXSDDateTime \* *pvalue*)

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

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

## Parameters

*pctxt* Pointer to context block structure.

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

## Returns

Completion status of operation:

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

### 6.4.1.24 int rtXmlpDecGMonthDay (OSCTXT \* *pctxt*, OSXSDDateTime \* *pvalue*)

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

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

## Parameters

*pctxt* Pointer to context block structure.

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

## Returns

Completion status of operation:

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

#### 6.4.1.25 **int rtXmlpDecGYear (OSCTXT \* *pctxt*, OSXSDDateTime \* *pvalue*)**

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

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

##### **Parameters**

*pctxt* Pointer to context block structure.

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

##### **Returns**

Completion status of operation:

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

#### 6.4.1.26 **int rtXmlpDecGYearMonth (OSCTXT \* *pctxt*, OSXSDDateTime \* *pvalue*)**

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

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

##### **Parameters**

*pctxt* Pointer to context block structure.

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

##### **Returns**

Completion status of operation:

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

#### 6.4.1.27 **int rtXmlpDecHexStr (OSCTXT \* *pctxt*, OSOCTET \* *pvalue*, OSUINT32 \* *pnocets*, OSINT32 *bufsize*)**

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

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

##### **Parameters**

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

*pvalue* A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.

*pnocets* A pointer to an integer value to receive the decoded number of octets.

*bufsize* The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

## Returns

Completion status of operation:

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

### 6.4.1.28 int rtXmlpDecInt (OSCTXT \* *pctxt*, OSINT32 \* *pvalue*)

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

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

## Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 32-bit integer value to receive decoded result.

## Returns

Completion status of operation:

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

### 6.4.1.29 int rtXmlpDecInt16 (OSCTXT \* *pctxt*, OSINT16 \* *pvalue*)

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

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

## Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 16-bit integer value to receive decoded result.

## Returns

Completion status of operation:

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

### 6.4.1.30 int rtXmlpDecInt64 (OSCTXT \* *pctxt*, OSINT64 \* *pvalue*)

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

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

## Parameters

*pctxt* Pointer to context block structure.

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

## Returns

Completion status of operation:

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

### 6.4.1.31 `int rtXmlpDecInt8 (OSCTXT * pctxt, OSINT8 * pvalue)`

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

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

## Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 8-bit integer value to receive decoded result.

## Returns

Completion status of operation:

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

### 6.4.1.32 `int rtXmlpDecNamedBits (OSCTXT * pctxt, const OSBitMapItem * pBitMap, OSOCTET * pvalue, OSUINT32 * pnbits, OSUINT32 bufsize)`

This function decodes the contents of a named bit field.

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

## Parameters

*pctxt* Pointer to context block structure.

*pBitMap* Pointer to bit map structure that defined token to bit mappings.

*pvalue* Pointer to buffer to receive decoded bit map.

*pnbits* Number of bits in decoded bit map.

*bufsize* Size of buffer passed in to received decoded bit values.

## Returns

Completion status of operation:

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

#### 6.4.1.33 `int rtXmlpDecStrList (OSCTXT * pctxt, OSRTDList * plist)`

This function decodes a list of space-separated tokens and returns each token as a separate item on the given list. Memory is allocated for the list nodes and token values using the rtx memory management functions.

##### Parameters

- pctxt* A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
- plist* A pointer to a linked list structure to which the parsed token values will be added.

##### Returns

Completion status of operation:

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

#### 6.4.1.34 `int rtXmlpDecTime (OSCTXT * pctxt, OSXSDDateTime * pvalue)`

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

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

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

##### Parameters

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

##### Returns

Completion status of operation:

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

#### 6.4.1.35 `int rtXmlpDecUInt (OSCTXT * pctxt, OSUINT32 * pvalue)`

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

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

##### Parameters

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

##### Returns

Completion status of operation:

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



#### 6.4.1.36 `int rtXmlpDecUInt16 (OSCTXT * pctxt, OSUINT16 * pvalue)`

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned 16-bit integer value to receive decoded result.

##### Returns

Completion status of operation:

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

#### 6.4.1.37 `int rtXmlpDecUInt64 (OSCTXT * pctxt, OSUINT64 * pvalue)`

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

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

##### Parameters

*pctxt* Pointer to context block structure.

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

##### Returns

Completion status of operation:

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

#### 6.4.1.38 `int rtXmlpDecUInt8 (OSCTXT * pctxt, OSOCTET * pvalue)`

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned 8-bit integer value to receive decoded result.

##### Returns

Completion status of operation:

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

#### 6.4.1.39 `int rtXmlpDecUTF8Str (OSCTXT * pctxt, OSUTF8CHAR * out, size_t max_len)`

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

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

##### Parameters

*pctxt* Pointer to context block structure.

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

*max\_len* Length of out array.

##### Returns

Completion status of operation:

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

#### 6.4.1.40 `int rtXmlpDecXmlStr (OSCTXT * pctxt, OSXMLSTRING * outdata)`

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

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

##### Parameters

*pctxt* Pointer to context block structure.

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

##### Returns

Completion status of operation:

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

#### 6.4.1.41 `int rtXmlpDecXmlStrList (OSCTXT * pctxt, OSRTDList * plist)`

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

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

##### Parameters

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

*plist* A pointer to a linked list structure to which the parsed token values will be added.

##### Returns

Completion status of operation:

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

#### 6.4.1.42 `int rtXmlpDecXSIAttr (OSCTXT * pctxt, const OSXMLNameFragments * attrName)`

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

##### Parameters

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

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

##### Returns

Completion status of operation:

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

#### 6.4.1.43 `int rtXmlpDecXSIAttrs (OSCTXT * pctxt)`

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

##### Parameters

*pctxt* Pointer to context block structure.

##### Returns

Completion status of operation:

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

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

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

##### Parameters

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

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

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

#### Returns

Completion status of operation:

- 0 = success,
- 1 = OK, but attrName was not xsi:type (i.e. no attribute match)
- negative return value is error.

#### 6.4.1.45 void rtXmlpForceDecodeAsGroup (OSCTXT \* *pctxt*)

Disable skipping of unknown elements in optional sequence tail.

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

#### Parameters

*pctxt* Pointer to context block structure.

#### 6.4.1.46 int rtXmlpGetAttributeCount (OSCTXT \* *pctxt*)

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

#### Parameters

*pctxt* Pointer to context block structure.

#### Returns

Completion status of operation:

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

#### 6.4.1.47 int rtXmlpGetAttributeID (const OSXMLStrFragment \* *attrName*, OSINT16 *nsidx*, size\_t *nAttr*, const OSXMLAttrDescr *attrNames*[], OSUINT32 *attrPresent*[])

This function finds an attribute in the descriptor table.

#### Parameters

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

*nsidx* Namespace index:

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

*nAttr* Number of descriptors in table.

*attrNames* Attributes descriptor table.

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

#### Returns

Completion status of operation:

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

#### 6.4.1.48 OSINT32 rtXmlpGetCurrentLevel (OSCTXT \* *pctxt*)

This function returns current nesting level.

#### Parameters

*pctxt* Pointer to context block structure.

#### Returns

Current nesting level.

#### 6.4.1.49 int rtXmlpGetNextAllElemID (OSCTXT \* *pctxt*, const OSXMLElemIDRec \* *tab*, size\_t *nrows*, const OSUINT8 \* *pOrder*, OSUINT32 *nOrder*, OSUINT32 *maxOrder*, int *anyID*)

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

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

#### Parameters

*pctxt* Pointer to context block structure.

*tab* Elements descriptor table.

*nrows* Number of descriptors in table.

*pOrder* Pointer to array to receive elements order.

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

*maxOrder* Size of order array.

*anyID* Identifier of xsd:any element.

#### Returns

Completion status of operation:

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

#### 6.4.1.50 int rtXmlpGetNextAllElemID16 (OSCTXT \* *pctxt*, const OSXMLElemIDRec \* *tab*, size\_t *nrows*, const OSUINT16 \* *pOrder*, OSUINT32 *nOrder*, OSUINT32 *maxOrder*, int *anyID*)

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

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

## Parameters

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

## Returns

Completion status of operation:

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

**6.4.1.51** `int rtXmlpGetNextAllElemID32 (OSCTXT * pctxt, const OSXMLElemIDRec * tab, size_t nrows, const OSUINT32 * pOrder, OSUINT32 nOrder, OSUINT32 maxOrder, int anyID)`

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

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

## Parameters

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

## Returns

Completion status of operation:

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

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

This function parse the next element start tag.

## Parameters

- pctxt* Pointer to context block structure.
- pElem* Pointer to a structure to receive the decoded element descriptor.
- level* Nesting level of parsed start tag. When value equal -1 parsed next start tag.

## Returns

Completion status of operation:

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

### 6.4.1.53 `int rtXmlpGetNextElemID (OSCTXT * pctxt, const OSXMLElemIDRec * tab, size_t nrows, OSINT32 level, OSBOOL continueParse)`

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

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

## Parameters

*pctxt* Pointer to context block structure.

*tab* Elements descriptor table.

*nrows* Number of descriptors in table.

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

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

## Returns

Completion status of operation:

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

### 6.4.1.54 `int rtXmlpGetNextSeqElemID (OSCTXT * pctxt, const OSXMLElemIDRec * tab, const OSXMLGroupDesc * pGroup, int curID, int lastMandatoryID, OSBOOL groupMode)`

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

It is used to decode sequences in strict mode.

Handling of unexpected elements:

- If the current decode group includes an any case, unexpected elements will be matched against it (the any case id will be returned).
- Otherwise, if groupMode is true, and the element identified by lastMandatoryID has been reached, XML\_OK\_EOB is returned. The unexpected element may belong to something following the elements in tab.
- If neither of the above applies, unknown elements will be logged and skipped over, with this method continuing as if the unexpected element were not present. Handling of the case of reaching closing tag for current level:
- If the last mandatory element is reached, return XML\_OK\_EOB.
- Otherwise, log and return XML\_E\_ELEMSMISRQ.

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

## Parameters

*pctxt* Pointer to context block structure.

*tab* Elements descriptor table.

*pGroup* Decode groups table.

*curID* Current decode group.

*lastMandatoryID* Identifier of last mandatory element.

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

## Returns

Completion status of operation:

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

**6.4.1.55** `int rtXmlpGetNextSeqElemID2 (OSCTXT * pctxt, const OSXMLElemIDRec * tab, const OSXMLGroupDesc * pGroup, int groups, int curID, int lastMandatoryID, OSBOOL groupMode, OSBOOL checkRepeat)`

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

It is used to decode sequences in strict mode.

## Parameters

*pctxt* Pointer to context block structure.

*tab* Elements descriptor table.

*pGroup* Decode groups table.

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

*curID* Current decode group.

*lastMandatoryID* Identifier of last mandatory element.

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

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

## Returns

Completion status of operation:

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

**6.4.1.56** `int rtXmlpGetNextSeqElemIDExt (OSCTXT * pctxt, const OSXMLElemIDRec * tab, const OSXMLGroupDesc * ppGroup, const OSBOOL * extRequired, int postExtRootID, int curID, int lastMandatoryID, OSBOOL groupMode)`

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

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



## Parameters

*pctxt* Pointer to context block structure.

*tab* Elements descriptor table.

*pGroup* Decode groups table.

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

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

*curID* Current decode group.

*lastMandatoryID* Identifier of last mandatory element.

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

## Returns

Completion status of operation:

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

### 6.4.1.57 `struct OSXMLReader* rtXmlpGetReader (OSCTXT * pctxt) [read]`

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

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

## Parameters

*pctxt* Pointer to context block structure.

## Returns

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

### 6.4.1.58 `int rtXmlpGetXmlnsAttrs (OSCTXT * pctxt, OSRTDList * pNSAttrs)`

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

## Parameters

*pctxt* Pointer to context block structure.

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

## Returns

Completion status of operation:

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

**6.4.1.59 int rtXmlpGetXSITypeAttr (OSCTXT \* *pctxt*, const OSUTF8CHAR \*\* *ppAttrValue*, OSINT16 \* *nsidx*, size\_t \* *pLocalOffs*)**

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

**Parameters**

*pctxt* Pointer to context block structure.

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

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

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

**Returns**

Completion status of operation:

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

**6.4.1.60 int rtXmlpGetXSITypeIndex (OSCTXT \* *pctxt*, const OSXMLItemDescr *typetab*[], size\_t *typetabsiz*)**

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

**Parameters**

*pctxt* Pointer to context block structure.

*typetab* XSI types descriptor table.

*typetabsiz* Number of descriptors in table.

**Returns**

Completion status of operation:

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

**6.4.1.61 OSBOOL rtXmlpHasAttributes (OSCTXT \* *pctxt*)**

This function checks accessibility of attributes.

**Parameters**

*pctxt* Pointer to context block structure.

**Returns**

Completion status of operation:

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

#### **6.4.1.62 void rtXmlpHideAttributes (OSCTXT \* *pctxt*)**

Disable access to attributes.

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

##### **Parameters**

*pctxt* Pointer to context block structure.

#### **6.4.1.63 OSBOOL rtXmlpIsDecodeAsGroup (OSCTXT \* *pctxt*)**

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

##### **Parameters**

*pctxt* Pointer to context block structure.

##### **Returns**

State of mode:

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

#### **6.4.1.64 OSBOOL rtXmlpIsEmptyElement (OSCTXT \* *pctxt*)**

Check element content: empty or not.

##### **Parameters**

*pctxt* Pointer to context block structure.

##### **Returns**

Status of element content:

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

#### **6.4.1.65 OSBOOL rtXmlpIsLastEventDone (OSCTXT \* *pctxt*)**

Check processing status of current tag.

##### **Parameters**

*pctxt* Pointer to context block structure.

##### **Returns**

Status of element content:

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

#### 6.4.1.66 OSBOOL rtXmlpIsUTF8Encoding (OSCTXT \* *pctxt*)

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

##### Parameters

*pctxt* Pointer to context block structure.

##### Returns

State of mode:

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

#### 6.4.1.67 OSBOOL rtXmlpListHasItem (OSCTXT \* *pctxt*)

Check for end of decoded token list.

##### Parameters

*pctxt* Pointer to context block structure.

##### Returns

State of decoded list:

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

#### 6.4.1.68 int rtXmlpLookupXSITypeIndex (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *pXsiType*, OSINT16 *xsiTypeIdx*, const OSXMLItemDescr *typetab*[], size\_t *typetabsiz*)

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

##### Parameters

*pctxt* Pointer to context block structure.

*pXsiType* A pointer to XSI type name.

*xsiTypeIdx* A XSI type namespace index.

*typetab* XSI types descriptor table.

*typetabsiz* Number of descriptors in table.

##### Returns

Completion status of operation:

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

#### 6.4.1.69 int rtXmlpMarkLastEventActive (OSCTXT \* *pctxt*)

This function marks current tag as unprocessed.

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

##### Parameters

*pctxt* Pointer to context block structure.

##### Returns

Completion status of operation:

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

#### 6.4.1.70 void rtXmlpMarkPos (OSCTXT \* *pctxt*)

Save current decode position.

##### Parameters

*pctxt* Pointer to context block structure.

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

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

##### Parameters

*pctxt* Pointer to context block structure.

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

##### Returns

Completion status of operation:

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

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*elemLocalName* Name of parsed element.

*nsidx* Namespace index:

- 0 = attribute is unqualified,

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

### Returns

Completion status of operation:

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

#### 6.4.1.73 OSBOOL rtXmlpNeedDecodeAttributes (OSCTXT \* *pctxt*)

This function checks if attributes were previously decoded.

### Parameters

*pctxt* Pointer to context block structure.

### Returns

State of attributes:

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

#### 6.4.1.74 int rtXmlpReadBytes (OSCTXT \* *pctxt*, OSOCTET \* *pbuf*, size\_t *nbytes*)

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

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

### Parameters

*pctxt* Pointer to context block structure.

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

*nbytes* Number of bytes to read.

### Returns

State of mode:

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

#### 6.4.1.75 void rtXmlpResetMarkedPos (OSCTXT \* *pctxt*)

Reset saved decode position.

### Parameters

*pctxt* Pointer to context block structure.

#### 6.4.1.76 void rtXmlpRewindToMarkedPos (OSCTXT \* *pctxt*)

Rewind to saved decode position.

##### Parameters

*pctxt* Pointer to context block structure.

#### 6.4.1.77 int rtXmlpSelectAttribute (OSCTXT \* *pctxt*, OSXMLNameFragments \* *pAttr*, OSINT16 \* *nsidx*, size\_t *attrIndex*)

This function selects attribute to decode.

##### Parameters

*pctxt* Pointer to context block structure.

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

*nsidx* Pointer to value to receive namespace index:

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

*attrIndex* Index of selected attribute.

##### Returns

Completion status of operation:

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

#### 6.4.1.78 void rtXmlpSetListMode (OSCTXT \* *pctxt*)

Sets list mode.

Attribute value or element content is decoded by tokens.

##### Parameters

*pctxt* Pointer to context block structure.

#### 6.4.1.79 OSBOOL rtXmlpSetMixedContentMode (OSCTXT \* *pctxt*, OSBOOL *mixedContentMode*)

Sets mixed content mode.

##### Parameters

*pctxt* Pointer to context block structure.

*mixedContentMode* Enable/disable mixed mode.

##### Returns

Previously state of mixed mode.

**6.4.1.80** void rtXmlpSetNamespaceTable (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *namespaceTable*[], size\_t *nmNamespaces*)

Sets user namespace table.

**Parameters**

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

**6.4.1.81** void rtXmlpSetWhiteSpaceMode (OSCTXT \* *pctxt*, OSXMLWhiteSpaceMode *whiteSpaceMode*)

Sets the whitespace treatment mode.

This mode affects the content and attribute values reading. For example, if OSXMLWSM\_COLLAPSE mode is set then all spaces in returned data will be already collapsed.

**Parameters**

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

**Returns**

Previously set whitespace mode.



## 6.5 XML run-time error status codes.

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

### Defines

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

*Unexpected start tag.*

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

## 6.5.1 Detailed Description

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

## 6.5.2 Define Documentation

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

Error base.

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

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

#### **6.5.2.2 #define XML\_E\_ELEMMISRQ (XML\_E\_BASE-8)**

Missing required element.

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

Definition at line 121 of file rtXmlErrCodes.h.

#### **6.5.2.3 #define XML\_E\_ELEMSISRQ (XML\_E\_BASE-9)**

Missing required elements.

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

Definition at line 128 of file rtXmlErrCodes.h.

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

Some key has no full set of fields.

It is not valid for key constraint.

Definition at line 199 of file rtXmlErrCodes.h.

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

Current tag is not matched to specified one.

Informational code.

Definition at line 115 of file rtXmlErrCodes.h.

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

Namespace URI not defined for given prefix.

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

Definition at line 166 of file rtXmlErrCodes.h.

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

Start/end tag mismatch.

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

Definition at line 90 of file rtXmlErrCodes.h.

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

End of block marker.

Definition at line 51 of file rtXmlErrCodes.h.

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

Maintained for backward compatibility.

Definition at line 54 of file rtXmlErrCodes.h.

## 6.6 DOM API functions.

### Typedefs

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

### Functions

- EXTERNDOM void [domParserInit](#) ()  
*Init parser.*
- EXTERNDOM void [domParserShutdown](#) ()  
*Shutdowns parser.*
- EXTERNDOM [OSRTDOMError](#) [domParseFile](#) (const char \*fileSpec, OSRTDOMDocPtr \*pXmlDoc)  
*Parse the file into a DOM document.*
- EXTERNDOM [OSRTDOMError](#) [domGetRootElement](#) (OSRTDOMDocPtr xmlDoc, OSRTDOMNodePtr \*pNode)  
*Returns root node for the document.*
- EXTERNDOM [OSRTDOMError](#) [domGetDoc](#) (OSRTDOMNodePtr node, OSRTDOMDocPtr \*pXmlDoc)  
*Returns the document for the given element.*
- EXTERNDOM void [domFreeDoc](#) (OSRTDOMDocPtr xmlDoc)  
*Frees document.*
- EXTERNDOM [OSRTDOMError](#) [domGetNext](#) (const OSRTDOMNodePtr curNode, OSRTDOMNodePtr \*pNextNode)  
*Returns next element node (down sibling).*
- EXTERNDOM [OSRTDOMError](#) [domGetChild](#) (const OSRTDOMNodePtr curNode, OSRTDOMNodePtr \*pChildNode)  
*Returns first (top) child node.*
- EXTERNDOM [OSRTDOMError](#) [domGetElementName](#) (const OSRTDOMNodePtr curNode, const OS-UTF8CHAR \*\*ppName, const OSUTF8CHAR \*\*ppNsPrefix, const OSUTF8CHAR \*\*ppNsUri)  
*Returns node's name and prefix (if any).*
- EXTERNDOM int [domGetNodeContent](#) (OSRTDOMContCtxtPtr \*ppCtxt, const OSRTDOMNodePtr curNode, const OSUTF8CHAR \*\*ppValue, size\_t \*pValueLen, OSBOOL \*pCdataProcessed)  
*Gets the node's content.*
- EXTERNDOM [OSRTDOMError](#) [domGetNodeFirstAttribute](#) (OSRTDOMAttrCtxtPtr \*ppCtxt, const OSRTDOMNodePtr curNode, OSRTDOMAttrPtr \*pTopAttrNode)  
*Returns the pointer to the first (top) attribute of the node.*
- EXTERNDOM int [domGetNodeAttributesNum](#) (const OSRTDOMNodePtr curNode)

Returns number of attributes in the node.

- EXTERNDOM [OSRTError domGetNextAttr](#) (OSRTDOMAttrCtxtPtr \*ppCtxt, const OSRTDOMAttrPtr curAttrNode, OSRTDOMAttrPtr \*pAttrNode)

Returns the next attribute.

- EXTERNDOM [OSRTError domGetAttrData](#) (const OSRTDOMAttrPtr curAttrNode, const OSUTF8CHAR \*\*ppAttrName, const OSUTF8CHAR \*\*ppAttrValue, const OSUTF8CHAR \*\*ppAttrPrefix, const OSUTF8CHAR \*\*ppAttrUri)

Returns the attribute's name and value.

- EXTERNDOM [OSRTError domSaveDoc](#) (OSRTDOMDocPtr xmlDoc, const char \*filename)

Saves DOM document to a file.

- EXTERNDOM [OSRTError domCreateDocument](#) (OSCTXT \*pctxt, OSRTDOMDocPtr \*pXmlDoc, const OSUTF8CHAR \*pNodeName, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs)

Creates a new document and a root element.

- EXTERNDOM [OSRTError domCreateChild](#) (OSCTXT \*pctxt, OSRTDOMNodePtr parentNode, const OSUTF8CHAR \*pNodeName, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs, OSRTDOMNodePtr \*pNewNode)

Creates new child node.

- EXTERNDOM [OSRTError domAddAttribute](#) (OSCTXT \*pctxt, OSRTDOMNodePtr node, const OSUTF8CHAR \*pAttrName, const OSUTF8CHAR \*pAttrValue, OSXMLNamespace \*pNS, OSRTDList \*pNSAttrs)

Creates new attribute and adds it to the node.

- EXTERNDOM [OSRTError domAddContent](#) (OSRTDOMNodePtr node, const OSUTF8CHAR \*pContent, size\_t contentLen)

Adds content (text) to the node.

- EXTERNDOM [OSRTError domAddCdata](#) (OSRTDOMNodePtr node, const OSUTF8CHAR \*pContent, size\_t contentLen)

Adds CDATA section to the node.

## 6.6.1 Function Documentation

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

Creates new attribute and adds it to the node.

#### Parameters

**pctxt** Pointer to context block structure.

**node** Pointer to the node where attribute to be added.

**pAttrName** Pointer to the null-terminated string containing name of the attribute.

**pAttrValue** Pointer to the null-terminated string containing name of the value.

*pNS* XML namespace information (prefix and URI).  
*pNSAttrs* List of namespace attributes to be added to element.

#### Returns

0 - success, otherwise error code.

#### 6.6.1.2 EXTERNDOM OSRTDOMError domAddCdata (OSRTDOMNodePtr *node*, const OSUTF8CHAR \**pContent*, size\_t *contentLen*)

Adds CDATA section to the node.

#### Parameters

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

#### Returns

0 - success, otherwise error code.

#### 6.6.1.3 EXTERNDOM OSRTDOMError domAddContent (OSRTDOMNodePtr *node*, const OSUTF8CHAR \**pContent*, size\_t *contentLen*)

Adds content (text) to the node.

#### Parameters

*node* Pointer to the node where content to be added.  
*pContent* Pointer to UTF-8 encoded content.  
*contentLen* Length of the content

#### Returns

0 - success, otherwise error code.

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

Creates new child node.

#### Parameters

*pctxt* Pointer to context block structure.  
*parentNode* Pointer to the parent node  
*pNodeName* Name of the node  
*pNS* XML namespace information (prefix and URI).  
*pNSAttrs* List of namespace attributes to be added to element.

*pNewNode* Pointer to pointer to newly created node.

#### Returns

0 - success, otherwise error code.

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

Creates a new document and a root element.

#### Parameters

*pctxt* Pointer to context block structure.

*pXmlDoc* Pointer to pointer to the newly created empty DOM document.

*pNodeName* Name of the node

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

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

#### Returns

0 - success, otherwise error code.

#### 6.6.1.6 EXTERNDOM void domFreeDoc (OSRTDOMDocPtr *xmlDoc*)

Frees document.

#### Parameters

*xmlDoc* Pointer to the DOM document context

#### 6.6.1.7 EXTERNDOM OSRTDOMError domGetAttrData (const OSRTDOMAttrPtr *curAttrNode*, const OSUTF8CHAR \*\* *ppAttrName*, const OSUTF8CHAR \*\* *ppAttrValue*, const OSUTF8CHAR \*\* *ppAttrPrefix*, const OSUTF8CHAR \*\* *ppAttrUri*)

Returns the attribute's name and value.

#### Parameters

*curAttrNode* Pointer to current attribute node,

*ppAttrName* Pointer to pointer to receive attribute's name.

*ppAttrValue* Pointer to pointer to receive attribute's value.

*ppAttrPrefix* Pointer to pointer to receive namespace's prefix.

*ppAttrUri* Pointer to pointer to receive namespace's uri.

#### Returns

0 - success, otherwise error code.



**6.6.1.8 EXTERNDOM OSRTDOMError domGetChild (const OSRTDOMNodePtr *curNode*, OSRTDOMNodePtr \* *pChildNode*)**

Returns first (top) child node.

**Parameters**

*curNode* Pointer to current node,  
*pChildNode* Pointer to pointer to receive the child node.

**Returns**

0 - success, otherwise error code.

**6.6.1.9 EXTERNDOM OSRTDOMError domGetDoc (OSRTDOMNodePtr *node*, OSRTDOMDocPtr \* *pXmlDoc*)**

Returns the document for the given element.

**Parameters**

*node* Pointer to a node.  
*pXmlDoc* Pointer to a pointer to the DOM document context

**Returns**

0 - success, otherwise error code.

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

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

**Parameters**

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

**Returns**

0 - success, otherwise error code.

**6.6.1.11 EXTERNDOM OSRTDOMError domGetNext (const OSRTDOMNodePtr *curNode*, OSRTDOMNodePtr \* *pNextNode*)**

Returns next element node (down sibling).

**Parameters**

*curNode* Pointer to current node,  
*pNextNode* Pointer to pointer to receive the next node.

**Returns**

0 - success, otherwise error code.

**6.6.1.12 EXTERNDOM OSRTDOMError domGetNextAttr (OSRTDOMAttrCtxtPtr \* ppCtxt, const OSRTDOMAttrPtr curAttrNode, OSRTDOMAttrPtr \* pAttrNode)**

Returns the next attribute.

**Parameters**

*ppCtxt* Pointer to an implementation-specific context pointer,  
*curAttrNode* Pointer to current attribute node,  
*pAttrNode* Pointer to pointer to receive the next attribute node.

**Returns**

0 - success, otherwise error code.

**6.6.1.13 EXTERNDOM int domGetNodeAttributesNum (const OSRTDOMNodePtr curNode)**

Returns number of attributes in the node.

**Parameters**

*curNode* Pointer to current node,

**Returns**

Number of attributes in the node.

**6.6.1.14 EXTERNDOM int domGetNodeContent (OSRTDOMContCtxtPtr \* ppCtxt, const OSRTDOMNodePtr curNode, const OSUTF8CHAR \*\* ppValue, size\_t \* pValueLen, OSBOOL \* pCdataProcessed)**

Gets the node's content.

**Parameters**

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

**Returns**

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

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

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

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

**Parameters**

*ppCtxt* Pointer to an implementation-specific context pointer,

*curNode* Pointer to current node,

*pTopAttrNode* Pointer to pointer to receive the first attribute node. domGetNextAttr/domGetPrevAttr functions may be used to get next/previous attributes.

**Returns**

0 - success, otherwise error code.

**6.6.1.16 EXTERNDOM OSRTDOMError domGetRootElement (OSRTDOMDocPtr xmlDoc, OSRTDOMNodePtr \* pNode)**

Returns root node for the document.

**Parameters**

*xmlDoc* Pointer to the DOM document context

*pNode* Pointer to pointer to receive the root node.

**Returns**

0 - success, otherwise error code.

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

Parse the file into a DOM document.

**Parameters**

*fileSpec* File name

*pXmlDoc* Pointer to a pointer to newly created DOM document.

**Returns**

0 - success, otherwise error code.

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

Saves DOM document to a file.

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

**Parameters**

*xmlDoc* Pointer to the DOM document context.

*filename* Pointer to file name.

**Returns**

0 - success, otherwise error code.

## 6.7 DOM runtime encode/decode functions.

### Functions

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

## 6.7.1 Function Documentation

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

Adds attribute to the node.

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

### Parameters

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

*domDoc* A pointer to a DOM-document.

*node* A pointer to a node where attribute to be added.

*attrName* A pointer to attribute's name.

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

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

### Returns

Completion status of operation:

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

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

Adds child node to the parent's node.

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

### Parameters

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

*domDoc* A pointer to a DOM-document.

*parentNode* A pointer to a parent node.

*elemName* A pointer to element's name.

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

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

### Returns

Completion status of operation:

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

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

Adds namespace attributes to the root node.

**Parameters**

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

*domDoc* A pointer to a DOM-document.

*rootNode* A pointer to a root node.

*pNSAttrs* A pointer to a list of namespace attrs.

**Returns**

Completion status of operation:

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

**6.7.1.4 EXTERNDOM int rtDomAddSubTree (OSCTXT \* *pctxt*, OSRTDOMDocPtr *xmlDoc*, OSRTDOMNodePtr *node*, const OSUTF8CHAR \* *pXmlData*, size\_t *dataLen*)**

This function adds the subtree to the specified node.

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

**Parameters**

*pctxt* Pointer to the context structure. This context might be used for memory allocations, if necessary.

*xmlDoc* Pointer to the document

*node* Pointer to the node where sub-tree to be inserted.

*pXmlData* Pointer to UTF-8 string representing fragment of XML data. For example, "<elem>data</elem>".

*dataLen* Length of XML data.

**Returns**

Completion status of operation:

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

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

This function starts decoding of the DOM document.

**Parameters**

*domDoc* A pointer to a DOM-document.

*pSaxBase* A pointer to SAX handler's base.

## Returns

Completion status of operation:

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

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

This function encodes a variable of the XSD any type.

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

## Parameters

*pctxt* Pointer to context block structure.

*domDoc* A pointer to a DOM-document.

*parentNode* A pointer to a parent node.

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

*elemName* XML element name. A name must be provided. If NULL pointer is passed, no element tag is added to the encoded value.

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

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

## Returns

Completion status of operation:

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

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

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

## Parameters

*pctxt* Pointer to context block structure.

*domDoc* A pointer to a DOM-document.

*node* A pointer to a node where attributes to be added.

*pAnyAttrList* List of attributes.

## Returns

Completion status of operation:

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



#### 6.7.1.8 EXTERNDOM int rtDomEncString (OSCTXT \* *pctxt*, OSXMLSTRING \* *pvalue*)

This function encodes a variable of the XSD string type.

##### Parameters

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

##### Returns

Completion status of operation:

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

#### 6.7.1.9 EXTERNDOM int rtDomEncStringValue (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *pvalue*)

This function encodes a variable of the XSD string type.

##### Parameters

- pctxt* Pointer to context block structure.
- pvalue* Null-terminated XML string value to be encoded.

##### Returns

Completion status of operation:

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

#### 6.7.1.10 EXTERNDOM int rtDomEncXSIAttrs (OSCTXT \* *pctxt*, OSRTDOMDocPtr *domDoc*, OSRTDOMNodePtr *node*, OSBOOL *needXSI*)

Adds XSI attributes to the node.

##### Parameters

- pctxt* A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
- domDoc* A pointer to a DOM-document.
- node* A pointer to a node.
- needXSI* Determines whether XSI namespace declaration needed.

##### Returns

Completion status of operation:

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

**6.7.1.11 EXTERNDOM int rtDomSetNode (OSCTXT \* *pctxt*, OSRTDOMDocPtr *domDoc*, OSRTDOMNodePtr *curNode*)**

Adds content or CDATA section to the node.

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

**Parameters**

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

*domDoc* A pointer to a DOM-document.

*curNode* A pointer to a current node.

**Returns**

Completion status of operation:

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

# Chapter 7

## Class Documentation

### 7.1 OSXMLGroupDesc Struct Reference

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

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

#### 7.1.1 Detailed Description

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

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

Definition at line 142 of file osrtxml.h.

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

- [osrtxml.h](#)

# Chapter 8

## File Documentation

### 8.1 osrtdom.h File Reference

DOM low-level C encode/decode functions.

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

#### Functions

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

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

### 8.1.1 Detailed Description

DOM low-level C encode/decode functions.

Definition in file [osrtdom.h](#).

## 8.2 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.

### Defines

- #define [rtXmlGetEncBufPtr](#)(pctxt) (pctxt)->buffer.data  
*This macro returns the start address of the encoded XML message.*
- #define [rtXmlGetEncBufLen](#)(pctxt) (pctxt)->buffer.byteIndex  
*This macro returns the length of the encoded XML message.*

### Typedefs

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

### Enumerations

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

### Functions

- int [rtXmlInitContext](#) (OSCTXT \*pctxt)  
*This function initializes a context variable for XML encoding or decoding.*
- int [rtXmlInitContextUsingKey](#) (OSCTXT \*pctxt, const OSOCTET \*key, size\_t 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 OSXSDDAnyType structure.*
- int [rtXmlDecBase64Binary](#) (OSRTMEMBUF \*pMemBuf, const OSUTF8CHAR \*inpdata, int length)  
*This function decodes the contents of a Base64-encoded binary data type into a memory buffer.*
- 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 [rtXmlDecBase64StrValue](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, size\_t bufSize, size\_t srcDataLen)  
*This function decodes a contents of a Base64-encode binary string into the specified octet array.*
- 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 [rtXmlDecDynHexStr](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)  
*This function decodes a contents of a hexBinary string.*
- int [rtXmlDecEmptyElement](#) (OSCTXT \*pctxt)  
*This function is used to enforce a requirement that an element be empty.*

- int [rtXmlDecUTF8Str](#) (OSCTXT \*pctxt, OSUTF8CHAR \*outdata, size\_t 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, int length)  
*This function decodes the contents of a hex-encoded binary data type into a memory buffer.*
- int [rtXmlDecHexStr](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, OSINT32 bufsize)  
*This function decodes the contents of a hexBinary string into a static memory structure.*
- 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.*
- int [rtXmlEncAny](#) (OSCTXT \*pctxt, OSXMLSTRING \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD any type.*
- int [rtXmlEncAnyTypeValue](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*pvalue)  
*This function encodes a variable of the XSD anyType type.*
- int [rtXmlEncAnyAttr](#) (OSCTXT \*pctxt, OSRTDList \*pAnyAttrList)  
*This function encodes a list of OSAnyAttr attributes in which the name and value are given as a UTF-8 string.*
- int [rtXmlEncBase64Binary](#) (OSCTXT \*pctxt, OSUINT32 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, size\_t attrNameLen)  
*This function encodes a variable of the XSD base64Binary type as an attribute.*
- int [rtXmlEncBase64StrValue](#) (OSCTXT \*pctxt, OSUINT32 nocts, const OSOCTET \*value)  
*This function encodes a variable of the XSD base64Binary type.*
- int [rtXmlEncBigInt](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD integer type.*
- int [rtXmlEncBigIntAttr](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*value, const OSUTF8CHAR \*attrName, size\_t 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, OSUINT32 nbits, const OSOCTET \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the ASN.1 BIT STRING type.*
- int **rtXmlEncBinStrValue** (OSCTXT \*pctxt, OSUINT32 nbits, const OSOCTET \*data)  
*This function encodes a binary string value as a sequence of '1's and '0's.*
- int **rtXmlEncBool** (OSCTXT \*pctxt, OSBOOL value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD boolean type.*
- int **rtXmlEncBoolValue** (OSCTXT \*pctxt, OSBOOL value)  
*This function encodes a variable of the XSD boolean type.*
- int **rtXmlEncBoolAttr** (OSCTXT \*pctxt, OSBOOL value, const OSUTF8CHAR \*attrName, size\_t attrNameLen)  
*This function encodes an XSD boolean attribute value.*
- 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, size\_t 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, size\_t 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, size\_t 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, size\_t 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 \*pctx, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a numeric gYearMonth element into an XML string representation.*
- int [rtXmlEncGMonth](#) (OSCTXT \*pctx, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a numeric gMonth element into an XML string representation.*
- int [rtXmlEncGMonthDay](#) (OSCTXT \*pctx, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a numeric gMonthDay element into an XML string representation.*
- int [rtXmlEncGDay](#) (OSCTXT \*pctx, const OSXSDDateTime \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a numeric gDay element into an XML string representation.*
- int [rtXmlEncGYearValue](#) (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gYear value into an XML string representation.*
- int [rtXmlEncGYearMonthValue](#) (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gYearMonth value into an XML string representation.*
- int [rtXmlEncGMonthValue](#) (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gMonth value into an XML string representation.*
- int [rtXmlEncGMonthDayValue](#) (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gMonthDay value into an XML string representation.*
- int [rtXmlEncGDayValue](#) (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gDay value into an XML string representation.*
- int [rtXmlEncHexBinary](#) (OSCTXT \*pctx, OSUINT32 noctx, const OSOCTET \*value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD hexBinary type.*
- int [rtXmlEncHexBinaryAttr](#) (OSCTXT \*pctx, OSUINT32 noctx, const OSOCTET \*value, const OSUTF8CHAR \*attrName, size\_t attrNameLen)  
*This function encodes a variable of the XSD hexBinary type as an attribute.*
- int [rtXmlEncHexStrValue](#) (OSCTXT \*pctx, OSUINT32 noctx, const OSOCTET \*data)  
*This function encodes a variable of the XSD hexBinary type.*
- int [rtXmlEncIndent](#) (OSCTXT \*pctx)  
*This function adds indentation whitespace to the output stream.*
- int [rtXmlEncInt](#) (OSCTXT \*pctx, OSINT32 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a variable of the XSD integer type.*
- int [rtXmlEncIntValue](#) (OSCTXT \*pctx, OSINT32 value)

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

- int [rtXmlEncIntAttr](#) (OSCTXT \*pctxt, OSINT32 value, const OSUTF8CHAR \*attrName, size\_t 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, size\_t attrNameLen)

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

- int [rtXmlEncNamedBits](#) (OSCTXT \*pctxt, const OSBitMapItem \*pBitMap, OSUINT32 nbits, const OSOCTET \*pvalue, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)

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

- 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, size\_t 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, size\_t 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, OSUINT32 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, size\_t nameLen, const OSUTF8CHAR \*value, size\_t valueLen)  
*This function encodes an attribute in which the name and value are given as a UTF-8 strings with lengths.*
- 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, size\_t 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 \*pctx, OSUINT64 value, const OSUTF8CHAR \*attrName, size\_t attrNameLen)  
*This function encodes a variable of the XSD integer type as an attribute (name="value").*
- int [rtXmlEncXSIAttrs](#) (OSCTXT \*pctx, OSBOOL needXSI)  
*This function encodes XML schema instance (XSI) attributes at the beginning of an XML document.*
- int [rtXmlEncXSITypeAttr](#) (OSCTXT \*pctx, const OSUTF8CHAR \*value)  
*This function encodes an XML schema instance (XSI) type attribute value (xsi:type="value").*
- int [rtXmlEncXSITypeAttr2](#) (OSCTXT \*pctx, const OSUTF8CHAR \*typeNsUri, const OSUTF8CHAR \*typeName)  
*This function encodes an XML schema instance (XSI) type attribute value (xsi:type="pfx:value").*
- int [rtXmlEncXSINilAttr](#) (OSCTXT \*pctx)  
*This function encodes an XML nil attribute (xsi:nil="true").*
- int [rtXmlFreeInputSource](#) (OSCTXT \*pctx)  
*This function closes an input source that was previously created with one of the create input source functions such as 'rtXmlCreateFileInputSource'.*
- int [rtXmlSetEncBufPtr](#) (OSCTXT \*pctx, OSOCTET \*bufaddr, size\_t bufsiz)  
*This function is used to set the internal buffer within the run-time library encoding context.*
- int [rtXmlGetIndent](#) (OSCTXT \*pctx)  
*This function returns current XML output indent value.*
- OSBOOL [rtXmlGetWriteBOM](#) (OSCTXT \*pctx)  
*This function returns whether the Unicode byte order mark will be encoded.*
- int [rtXmlGetIndentChar](#) (OSCTXT \*pctx)  
*This function returns current XML output indent character value (default is space).*
- int [rtXmlPrepareContext](#) (OSCTXT \*pctx)  
*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 \*pctx, OSBOOL value)  
*This function sets the option to encode in C14N mode.*
- int [rtXmlSetEncXSINamespace](#) (OSCTXT \*pctx, 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 \*pctx, 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 \*pctx, 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, size\_t minLength, size\_t maxLength)  
*This function tests the context buffer for containing a correct hexadecimal string.*
- int [rtXmlMatchBase64Str](#) (OSCTXT \*pctxt, size\_t minLength, size\_t 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 mathes 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, size\_t attrIndex)  
*This function decodes an any attribute string.*
- int [rtXmlpDecAnyElem](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*pvalue)  
*This function decodes an arbitrary XML section of code as defined by the XSD any type (xsd:any).*
- int [rtXmlpDecBase64Str](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, OSINT32 bufsize)  
*This function decodes a contents of a Base64-encode binary string into a static memory structure.*
- 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 [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 [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 [rtXmlpDecDynUnicodeStr](#) (OSCTXT \*pctxt, const OSUNICHAR \*\*ppdata, OSUINT32 \*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, size\_t 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, OSINT32 bufsize)  
*This function decodes the contents of a hexBinary string into a static memory structure.*
- 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 [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 \*pctx, OSUINT32 \*pvalue)  
*This function decodes the contents of an unsigned 32-bit integer data type.*
- int [rtXmlpDecUInt8](#) (OSCTXT \*pctx, OSOCTET \*pvalue)  
*This function decodes the contents of an unsigned 8-bit integer data type (i.e.*
- int [rtXmlpDecUInt16](#) (OSCTXT \*pctx, OSUINT16 \*pvalue)  
*This function decodes the contents of an unsigned 16-bit integer data type.*
- int [rtXmlpDecUInt64](#) (OSCTXT \*pctx, OSUINT64 \*pvalue)  
*This function decodes the contents of an unsigned 64-bit integer data type.*
- int [rtXmlpDecXmlStr](#) (OSCTXT \*pctx, OSXMLSTRING \*outdata)  
*This function decodes the contents of an XML string data type.*
- int [rtXmlpDecXmlStrList](#) (OSCTXT \*pctx, OSRTDList \*plist)  
*This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.*
- int [rtXmlpDecXSIAttr](#) (OSCTXT \*pctx, const OSXMLNameFragments \*attrName)  
*This function decodes XSI (XML Schema Instance) attributes that may be present in any arbitrary XML element within a document.*
- int [rtXmlpDecXSITypeAttr](#) (OSCTXT \*pctx, const OSXMLNameFragments \*attrName, const OS-UTF8CHAR \*\*ppAttrValue)  
*This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).*
- int [rtXmlpGetAttributeID](#) (const OSXMLStrFragment \*attrName, OSINT16 nsidx, size\_t nAttr, const OSXMLAttrDescr attrNames[ ], OSUINT32 attrPresent[ ])  
*This function finds an attribute in the descriptor table.*
- int [rtXmlpGetNextElem](#) (OSCTXT \*pctx, OSXMLElemDescr \*pElem, OSINT32 level)  
*This function parse the next element start tag.*
- int [rtXmlpGetNextElemID](#) (OSCTXT \*pctx, const OSXMLElemIDRec \*tab, size\_t nrows, OSINT32 level, OSBOOL continueParse)  
*This function parses the next start tag and finds the index of the element name in the descriptor table.*
- int [rtXmlpMarkLastEventActive](#) (OSCTXT \*pctx)  
*This function marks current tag as unprocessed.*
- int [rtXmlpMatchStartTag](#) (OSCTXT \*pctx, const OSUTF8CHAR \*elemLocalName, OSINT16 nsidx)  
*This function parses the next start tag that matches with given name.*
- int [rtXmlpMatchEndTag](#) (OSCTXT \*pctx, OSINT32 level)  
*This function parse next end tag that matches with given name.*
- OSBOOL [rtXmlpHasAttributes](#) (OSCTXT \*pctx)  
*This function checks accessibility of attributes.*

- int [rtXmlpGetAttributeCount](#) (OSCTXT \*pctxt)  
*This function returns number of attributes in last processed start tag.*
- int [rtXmlpSelectAttribute](#) (OSCTXT \*pctxt, OSXMLNameFragments \*pAttr, OSINT16 \*nsidx, size\_t attrIndex)  
*This function selects attribute to decode.*
- OSINT32 [rtXmlpGetCurrentLevel](#) (OSCTXT \*pctxt)  
*This function returns current nesting level.*
- void [rtXmlpSetWhiteSpaceMode](#) (OSCTXT \*pctxt, OSXMLWhiteSpaceMode whiteSpaceMode)  
*Sets the whitespace treatment mode.*
- OSBOOL [rtXmlpSetMixedContentMode](#) (OSCTXT \*pctxt, OSBOOL mixedContentMode)  
*Sets mixed content mode.*
- void [rtXmlpSetListMode](#) (OSCTXT \*pctxt)  
*Sets list mode.*
- OSBOOL [rtXmlpListHasItem](#) (OSCTXT \*pctxt)  
*Check for end of decoded token list.*
- void [rtXmlpCountListItems](#) (OSCTXT \*pctxt, OSSIZE \*itemCnt)  
*Count tokens in list.*
- 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.*
- 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.*
- 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.*
- int [rtXmlpGetNextAllElemID](#) (OSCTXT \*pctxt, const OSXMLElemIDRec \*tab, size\_t nRows, const OSUINT8 \*pOrder, OSUINT32 nOrder, OSUINT32 maxOrder, int anyID)  
*This function parses the next start tag and finds index of element name in descriptor table.*
- int [rtXmlpGetNextAllElemID16](#) (OSCTXT \*pctxt, const OSXMLElemIDRec \*tab, size\_t nRows, const OSUINT16 \*pOrder, OSUINT32 nOrder, OSUINT32 maxOrder, int anyID)  
*This function parses the next start tag and finds index of element name in descriptor table.*
- int [rtXmlpGetNextAllElemID32](#) (OSCTXT \*pctxt, const OSXMLElemIDRec \*tab, size\_t nRows, const OSUINT32 \*pOrder, OSUINT32 nOrder, OSUINT32 maxOrder, int anyID)  
*This function parses the next start tag and finds index of element name in descriptor table.*

- void `rtXmlpSetNamespaceTable` (OSCTXT \*pctxt, const OSUTF8CHAR \*namespaceTable[], size\_t nm-Namespaces)  
*Sets user namespace table.*
- int `rtXmlpCreateReader` (OSCTXT \*pctxt)  
*Creates pull parser reader structure within the context.*
- void `rtXmlpHideAttributes` (OSCTXT \*pctxt)  
*Disable access to attributes.*
- OSBOOL `rtXmlpNeedDecodeAttributes` (OSCTXT \*pctxt)  
*This function checks if attributes were previously decoded.*
- void `rtXmlpMarkPos` (OSCTXT \*pctxt)  
*Save current decode position.*
- void `rtXmlpRewindToMarkedPos` (OSCTXT \*pctxt)  
*Rewind to saved decode position.*
- void `rtXmlpResetMarkedPos` (OSCTXT \*pctxt)  
*Reset saved decode position.*
- int `rtXmlpGetXSITypeAttr` (OSCTXT \*pctxt, const OSUTF8CHAR \*\*ppAttrValue, OSINT16 \*nsidx, size\_t \*pLocalOffs)  
*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 `rtXmlpIsEmptyElement` (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 `rtXmlpIsLastEventDone` (OSCTXT \*pctxt)  
*Check processing status of current tag.*
- int `rtXmlpGetXSITypeIndex` (OSCTXT \*pctxt, const OSXMLItemDescr typetab[], size\_t typetabsiz)  
*This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type) and find type index in descriptor table.*
- int `rtXmlpLookupXSITypeIndex` (OSCTXT \*pctxt, const OSUTF8CHAR \*pXsiType, OSINT16 xsiTypeIdx, const OSXMLItemDescr typetab[], size\_t 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 **rtXmlpIsDecodeAsGroup** (OSCTXT \*pctxt)  
*This function checks if "decode as group" mode was forced.*
- OSBOOL **rtXmlpIsUTF8Encoding** (OSCTXT \*pctxt)  
*This function checks if the encoding specified in XML header is UTF-8.*
- int **rtXmlpReadBytes** (OSCTXT \*pctxt, OSOCTET \*pbuf, size\_t nbytes)  
*This function reads the specified number of bytes directly from the underlying XML parser stream.*

## 8.2.1 Detailed Description

XML low-level C encode/decode functions.

Definition in file [osrxml.h](#).

## 8.2.2 Typedef Documentation

### 8.2.2.1 typedef struct OSXMLGroupDesc OSXMLGroupDesc

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

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

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

## 8.2.3 Function Documentation

### 8.2.3.1 **const OSUTF8CHAR\* rtSaxGetAttrValue (const OSUTF8CHAR \* attrName, const OSUTF8CHAR \*const \* attrs)**

This function looks up an attribute in the attribute array returned by SAX to the startElement function.

#### Parameters

*attrName* Name of the attribute to find.

*attrs* Attribute array returned in SAX startElement function. This is an array of character strings containing name1, value1, name2, value2, ... List is terminated by a null name.

#### Returns

Pointer to character string containing attribute value or NULL if attrName not found.

**8.2.3.2 OSINT16 rtSaxGetElemID (OSINT16 \* pState, OSINT16 prevElemIdx, const OSUTF8CHAR \* localName, OSINT32 nsidx, const OSSAXElemTableRec idtab[], const OSINT16 \* fstab, OSINT16 fstabRows, OSINT16 fstabCols)**

This function looks up a sequence element name in the given element info array.

If ensures elements are received in the correct order and also sets the required element count variable.

**Parameters**

*pState* The pointer to state variable to be changed.

*prevElemIdx* Previous index of element. The search will be started from this element for better performance.

*localName* Local name of XML element

*nsidx* Namespace index

*idtab* Element ID table

*fstab* Finite state table

*fstabRows* Number of rows in *fstab*.

*fstabCols* Number of columns in *fstab*.

**8.2.3.3 OSINT16 rtSaxGetElemID8 (OSINT16 \* pState, OSINT16 prevElemIdx, const OSUTF8CHAR \* localName, OSINT32 nsidx, const OSSAXElemTableRec idtab[], const OSINT8 \* fstab, OSINT16 fstabRows, OSINT16 fstabCols)**

This function is a space optimized version of `rtSaxGetElemID`.

It operates with array of 8-bit integers (OSINT8) instead of 32-bit integers (int).

**Parameters**

*pState* The pointer to state variable to be changed.

*prevElemIdx* Previous index of element. The search will be started from this element + 1 for better performance.

*localName* Local name of XML element

*nsidx* Namespace index

*idtab* Element ID table

*fstab* Finite state table (array of 8-bit integers)

*fstabRows* Number of rows in *fstab*.

*fstabCols* Number of columns in *fstab*.

**8.2.3.4 OSBOOL rtSaxHasXMLNSAttrs (const OSUTF8CHAR \*const \* attrs)**

This function checks if the given attribute list contains one or more XML namespace attributes (xmlns).

**Parameters**

*attrs* Attribute list in form passed by parser into SAX `startElement` function.

**Returns**

TRUE, if xmlns attribute found in list.



### 8.2.3.5 OSBOOL rtSaxIsEmptyBuffer (OSCTXT \* *pctxt*)

This function checks if the buffer in the context is empty or not.

#### Parameters

*pctxt* Pointer to OSCTXT structure

#### Returns

TRUE, if the buffer contains empty string.

### 8.2.3.6 int rtSaxSortAttrs (OSCTXT \* *pctxt*, const OSUTF8CHAR \*const \* *attrs*, OSUINT16 \*\* *order*)

This function sorts a SAX attribute list in ascending order based on attribute name.

It currently only supports unqualified attributes.

#### Parameters

*pctxt* Pointer to OSCTXT structure.

*attrs* Standard SAX attribute list. Entry *i* is attribute name and *i*+1 is value. List is terminated by a null name.

*order* Order array containing the order of sorted attributes. This array is allocated using `rtxMemAlloc`, it can be freed using `rtxMemFreePtr` or will be freed when the context is freed. The list holds indices to name items in the attribute list that is passed in.

#### Returns

If success, positive value contains number of attributes in *attrs*; if failure, negative status code.

### 8.2.3.7 int rtSaxStrListMatch (OSCTXT \* *pctxt*)

This function matches the list of strings.

It is used for matching NMTOKENS, IDREFS, NMENTITIES.

#### Parameters

*pctxt* Pointer to OSCTXT structure

#### Returns

0 - if success, negative value is error.

### 8.2.3.8 int rtSaxStrListParse (OSCTXT \* *pctxt*, OSRTMEMBUF \* *pMemBuf*, OSRTDList \* *pvalue*)

This function parses the list of strings.

It is used for parsing NMTOKENS, IDREFS, NMENTITIES.

#### Parameters

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

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

*pvalue* Doubly-linked list for parsed strings.

#### Returns

0 - if success, negative value is error.

#### 8.2.3.9 OSBOOL rtXmlCmpBase64Str (OSUINT32 *noct1*, const OSOCTET \* *data1*, const OSUTF8CHAR \* *data2*)

This function compares an array of octets to a base64 string.

#### Parameters

*noct1* Number of octets in *data1*.

*data1* Pointer to array of OSOCTET.

*data2* Pointer to null-terminated array of OSUTF8CHAR.

#### Returns

TRUE if *data2* is a base64 string representation of *data1*, false otherwise.

#### 8.2.3.10 OSBOOL rtXmlCmpHexStr (OSUINT32 *noct1*, const OSOCTET \* *data1*, const OSUTF8CHAR \* *data2*)

This function compares an array of octets to a hex string.

#### Parameters

*noct1* Number of octets in *data1*.

*data1* Pointer to array of OSOCTET.

*data2* Pointer to null-terminated array of OSUTF8CHAR.

#### Returns

TRUE if *data2* is a hex string representation of *data1*, false otherwise.

#### 8.2.3.11 int rtXmlCreateFileInputSource (OSCTXT \* *pctxt*, const char \* *filepath*)

This function creates an XML document file input source.

The document can then be decoded by invoking an XML decode function.

#### Parameters

*pctxt* Pointer to context block structure.

*filepath* Full pathname of XML document file to open.

#### Returns

Completion status of operation:

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

### 8.2.3.12 int rtXmlInitContext (OSCTXT \* *pctxt*)

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

#### Parameters

*pctxt* Pointer to OSCTXT structure

### 8.2.3.13 int rtXmlInitContextUsingKey (OSCTXT \* *pctxt*, const OSOCTET \* *key*, size\_t *keylen*)

This function initializes a context using a run-time key.

This form is required for evaluation and limited distribution software. The compiler will generate a macro for rtXmlInitContext in the rtkey.h file that will invoke this function with the generated run-time key.

#### Parameters

*pctxt* The pointer to the context structure variable to be initialized.

*key* Key data generated by ASN1C compiler.

*keylen* Key data field length.

#### Returns

Completion status of operation:

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

### 8.2.3.14 int rtXmlInitCtxtAppInfo (OSCTXT \* *pctxt*)

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

#### Parameters

*pctxt* Pointer to OSCTXT structure

### 8.2.3.15 int rtXmlMatchBase64Str (OSCTXT \* *pctxt*, size\_t *minLength*, size\_t *maxLength*)

This function tests the context buffer for containing a correct base64 string.

It does not decode the value.

#### Parameters

*pctxt* Pointer to OSCTXT structure

*minLength* A minimal length of expected string.

*maxLength* A maximal length of expected string.

#### Returns

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

### 8.2.3.16 int rtXmlMatchDate (OSCTXT \* *pctxt*)

This function tests the context buffer for containing a correct date string.

It does not decode the value.

#### Parameters

*pctxt* Pointer to OSCTXT structure

#### Returns

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

### 8.2.3.17 int rtXmlMatchDateTime (OSCTXT \* *pctxt*)

This function tests the context buffer for containing a correct dateTime string.

It does not decode the value.

#### Parameters

*pctxt* Pointer to OSCTXT structure

#### Returns

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

### 8.2.3.18 int rtXmlMatchGDay (OSCTXT \* *pctxt*)

This function tests the context buffer for containing a correct gDay string.

It does not decode the value.

#### Parameters

*pctxt* Pointer to OSCTXT structure

#### Returns

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

### 8.2.3.19 int rtXmlMatchGMonth (OSCTXT \* *pctxt*)

This function tests the context buffer for containing a correct gMonth string.

It does not decode the value.

#### Parameters

*pctxt* Pointer to OSCTXT structure

#### Returns

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

### 8.2.3.20 int rtXmlMatchGMonthDay (OSCTXT \* *pctxt*)

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

It does not decode the value.

#### Parameters

*pctxt* Pointer to OSCTXT structure

#### Returns

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

### 8.2.3.21 int rtXmlMatchGYear (OSCTXT \* *pctxt*)

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

It does not decode the value.

#### Parameters

*pctxt* Pointer to OSCTXT structure

#### Returns

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

### 8.2.3.22 int rtXmlMatchGYearMonth (OSCTXT \* *pctxt*)

This function tests the context buffer for containing a correct gYearMonth string.

It does not decode the value.

#### Parameters

*pctxt* Pointer to OSCTXT structure

#### Returns

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

### 8.2.3.23 int rtXmlMatchHexStr (OSCTXT \* *pctxt*, *size\_t minLength*, *size\_t maxLength*)

This function tests the context buffer for containing a correct hexadecimal string.

It does not decode the value.

#### Parameters

*pctxt* Pointer to OSCTXT structure

*minLength* A minimal length of expected string.

*maxLength* A maximal length of expected string.

#### Returns

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

#### 8.2.3.24 int rtXmlMatchTime (OSCTXT \* *pctxt*)

This function tests the context buffer for containing a correct time string.

It does not decode the value.

#### Parameters

*pctxt* Pointer to OSCTXT structure

#### Returns

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

#### 8.2.3.25 void rtXmlMemFreeAnyAttrs (OSCTXT \* *pctxt*, OSRTDList \* *pAnyAttrList*)

This function frees a list of anyAttribute that is a member of OSXSDAnyType structure.

#### Parameters

*pctxt* Pointer to context block structure.

*pAnyAttrList* Pointer to list of anyAttribute that is to be freed.

#### 8.2.3.26 OSUTF8CHAR\* rtXmlNewQName (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *localName*, const OSUTF8CHAR \* *prefix*)

This function creates a new QName given the localName and prefix parts.

#### Parameters

*pctxt* Pointer to a context structure.

*localName* Element local name.

*prefix* Namespace prefix.

#### Returns

QName value. Memory for the value will have been allocated by rtxMemAlloc and thus must be freed using one of the rtxMemFree functions. The value will be NULL if no dynamic memory was available.

### 8.2.3.27 **int rtXmlPrepareContext (OSCTXT \* *pctxt*)**

This function prepares the context for another encode by setting the state back to OSXMLINIT and moving the buffer's cursor back to the beginning of the buffer.

#### **Parameters**

*pctxt* Pointer to OSCTXT structure

#### **Returns**

0 if OK, negative status code if error.

### 8.2.3.28 **int rtXmlSetEncC14N (OSCTXT \* *pctxt*, OSBOOL *value*)**

This function sets the option to encode in C14N mode.

#### **Parameters**

*pctxt* Pointer to OSCTXT structure

*value* Boolean value: true = C14N mode enabled.

#### **Returns**

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

### 8.2.3.29 **int rtXmlSetEncDocHdr (OSCTXT \* *pctxt*, OSBOOL *value*)**

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

#### **Parameters**

*pctxt* Pointer to OSCTXT structure

*value* Boolean value: true = add document header

#### **Returns**

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

### 8.2.3.30 **int rtXmlSetEncodingStr (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *encodingStr*)**

This function sets the XML output encoding to the given value. Currently, UTF-8/UTF-16/ISO-8859-1 encodings are supported.

#### **Parameters**

*pctxt* Pointer to OSCTXT structure

*encodingStr* XML output encoding format

#### **Returns**

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

### 8.2.3.31 **int rtXmlSetEncXSINamespace (OSCTXT \* *pctxt*, OSBOOL *value*)**

This function sets a flag in the context that indicates the XSI namespace declaration (xmlns:xsi) should be added to the encoded XML instance.

#### **Parameters**

*pctxt* Pointer to OSCTXT structure

*value* Boolean value: true = encode XSI namespace attribute.

#### **Returns**

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

### 8.2.3.32 **int rtXmlSetEncXSINilAttr (OSCTXT \* *pctxt*, OSBOOL *value*)**

This function sets a flag in the context that indicates the XSI attribute declaration (xmlns:xsi) should be added to the encoded XML instance.

#### **Parameters**

*pctxt* Pointer to OSCTXT structure

*value* Boolean value: true = encode xsi:nil attribute.

#### **Returns**

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

### 8.2.3.33 **int rtXmlSetFormatting (OSCTXT \* *pctxt*, OSBOOL *doFormatting*)**

This function sets XML output formatting to the given value.

If TRUE (the default), the XML document is formatted with indentation and newlines. If FALSE, all whitespace between elements is suppressed. Turning formatting off can provide more compressed documents and also a more canonical representation which is important for security applications. Also the function 'rtXmlSetIndent' might be used to set the exact size of indentation.

#### **Parameters**

*pctxt* Pointer to OSCTXT structure

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

#### **Returns**

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

### 8.2.3.34 **int rtXmlSetIndent (OSCTXT \* *pctxt*, OSUINT8 *indent*)**

This function sets XML output indent to the given value.

#### **Parameters**

*pctxt* Pointer to OSCTXT structure



*indent* Number of spaces per indent. Default is 3.

#### Returns

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

#### 8.2.3.35 int rtXmlSetIndentChar (OSCTXT \* *pctxt*, char *indentChar*)

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

#### Parameters

*pctxt* Pointer to OSCTXT structure

*indentChar* Indent character. Default is space.

#### Returns

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

#### 8.2.3.36 void rtXmlSetNamespacesSet (OSCTXT \* *pctxt*, OSBOOL *value*)

This function sets the context 'namespaces are set' flag.

This indicates that namespace declarations have been set either by the decoder or externally by the end user. It is used by the encoder to know not to set the default namespaces specified in the schema before starting encoding.

#### Parameters

*pctxt* Pointer to OSCTXT structure.

*value* Boolean value to which flag is to be set.

#### 8.2.3.37 int rtXmlSetNoNSSchemaLocation (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *schemaLocation*)

This function sets the XML Schema Instance (xsi) no namespace schema location attribute to be added to an encoded document.

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

#### Parameters

*pctxt* Pointer to OSCTXT structure

*schemaLocation* Schema location attribute value

#### Returns

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

### 8.2.3.38 int rtXmlSetNSPrefixLinks (OSCTXT \* *pctxt*, OSRTDList \* *pNSAttrs*)

This function sets namespace prefix/URI links in the namespace prefix stack in the context structure.

#### Parameters

*pctxt* Pointer to OSCTXT structure.

*pNSAttrs* List of namespace attributes.

#### Returns

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

### 8.2.3.39 int rtXmlSetSchemaLocation (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *schemaLocation*)

This function sets the XML Schema Instance (xsi) schema location attribute to be added to an encoded document.

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

#### Parameters

*pctxt* Pointer to OSCTXT structure

*schemaLocation* Schema location attribute value

#### Returns

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

### 8.2.3.40 void rtXmlSetSoapVersion (OSCTXT \* *pctxt*, OSUINT8 *version*)

This function sets the SOAP version number.

#### Parameters

*pctxt* Pointer to OSCTXT structure

*version* SOAP version number as 2 digit integer (for example, 11 is SOAP version 1.1, 12 is version 1.2, etc.)

### 8.2.3.41 int rtXmlSetWriteBOM (OSCTXT \* *pctxt*, OSBOOL *write*)

This function sets whether the Unicode byte order mark is encoded.

#### Parameters

*pctxt* Pointer to OSCTXT structure

*write* TRUE to encode BOM, FALSE to not encode BOM.

#### Returns

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

#### **8.2.3.42 int rtXmlSetXSITypeAttr (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *xsiType*)**

This function sets the XML Schema Instance (xsi) type attribute value.

This will cause an xsi:type attribute to be added to the top level element in an encoded XML instance.

#### **Parameters**

*pctxt* Pointer to OSCTXT structure

*xsiType* xsi:type attribute value

#### **Returns**

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

## 8.3 rtXmlErrCodes.h File Reference

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

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

### Defines

- #define [XML\\_OK\\_EOB](#) 0x7ffffff  
*End of block marker.*
- #define [XML\\_OK\\_FRAG](#) XML\_OK\_EOB  
*Maintained for backward compatibility.*
- #define [XML\\_E\\_BASE](#) -200  
*Error base.*
- #define [XML\\_E\\_GENERR](#) (XML\_E\_BASE)  
*General error.*
- #define [XML\\_E\\_INVSYMBOL](#) (XML\_E\_BASE-1)  
*An invalid XML symbol (character) was detected at the given point in the parse stream.*
- #define [XML\\_E\\_TAGMISMATCH](#) (XML\_E\_BASE-2)  
*Start/end tag mismatch.*
- #define [XML\\_E\\_DUPLATTR](#) (XML\_E\_BASE-3)  
*Duplicate attribute found.*
- #define [XML\\_E\\_BADCHARREF](#) (XML\_E\_BASE-4)  
*Bad character reference found.*
- #define [XML\\_E\\_INVMODE](#) (XML\_E\_BASE-5)  
*Invalid mode.*
- #define [XML\\_E\\_UNEXPEOF](#) (XML\_E\_BASE-6)  
*Unexpected end of file (document).*
- #define [XML\\_E\\_NOMATCH](#) (XML\_E\_BASE-7)  
*Current tag is not matched to specified one.*
- #define [XML\\_E\\_ELEMMISRQ](#) (XML\_E\_BASE-8)  
*Missing required element.*
- #define [XML\\_E\\_ELEMSISRQ](#) (XML\_E\_BASE-9)  
*Missing required elements.*
- #define [XML\\_E\\_TOOFWELEMS](#) (XML\_E\_BASE-10)  
*The number of elements in a repeating collection was less than the number of elements specified in the XSD minOccurs facet for this type or element.*

- #define [XML\\_E\\_UNEXPSTARTTAG](#) (XML\_E\_BASE-11)  
*Unexpected start tag.*
- #define [XML\\_E\\_UNEXPENDTAG](#) (XML\_E\_BASE-12)  
*Unexpected end tag.*
- #define [XML\\_E\\_IDNOTFOU](#) (XML\_E\_BASE-13)  
*Expected identifier not found.*
- #define [XML\\_E\\_INVTYPEINFO](#) (XML\_E\_BASE-14)  
*Unknown xsi:type.*
- #define [XML\\_E\\_NSURINOTFOU](#) (XML\_E\_BASE-15)  
*Namespace URI not defined for given prefix.*
- #define [XML\\_E\\_KEYNOTFOU](#) (XML\_E\_BASE-16)  
*Keyref constraint has some key that not present in refered constraint.*
- #define [XML\\_E\\_DUPLKEY](#) (XML\_E\_BASE-17)  
*Key or unique constraint has duplicated key.*
- #define [XML\\_E\\_FLDABSENT](#) (XML\_E\_BASE-18)  
*Some key has no full set of fields.*
- #define [XML\\_E\\_DUPLFLD](#) (XML\_E\_BASE-19)  
*Some key has more than one value for field.*
- #define [XML\\_E\\_NOTEMPTY](#) (XML\_E\_BASE-20)  
*An element was not empty when expected.*

### 8.3.1 Detailed Description

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

Definition in file [rtXmlErrCodes.h](#).

## 8.4 rtXmlExternDefs.h File Reference

XML external definitions macro.

### 8.4.1 Detailed Description

XML external definitions macro. This is used for Windows to properly declare function scope within DLL's.

Definition in file [rtXmlExternDefs.h](#).

## 8.5 rtXmlKeyArray.h File Reference

- Implementation of a dynamic pointer sorted array.

```
#include "rtxsrc/rtxContext.h"
#include "rtxmlsrc/rtXmlExternDefs.h"
```

### Functions

- int [rtXmlKeyArrayInit](#) (OSCTXT \*pctxt, OSXSDKeyArray \*pArray, OSUINT32 nmFields, OSBOOL key, const char \*name)  
*Initialize the array by allocating memory for it.*
- void [rtXmlKeyArraySetString](#) (OSXSDKeyArray \*pArray, const OSUTF8CHAR \*pValue, OSUINT32 fldNum)  
*Set the given field's value to the given string value.*
- void [rtXmlKeyArraySetInt](#) (OSXSDKeyArray \*pArray, OSINT32 value, OSUINT32 fldNum)  
*Set the given field's value to the given integer value.*
- void [rtXmlKeyArraySetUInt](#) (OSXSDKeyArray \*pArray, OSUINT32 value, OSUINT32 fldNum)  
*Set the given field's value to the given unsigned integer value.*
- void [rtXmlKeyArraySetDecimal](#) (OSXSDKeyArray \*pArray, const OSREAL \*value, OSUINT32 fldNum)  
*Set the given field's value to the given decimal value.*
- int [rtXmlKeyArrayAdd](#) (OSCTXT \*pctxt, OSXSDKeyArray \*pArray)  
*Once all the fields for a key are set, invoke this method to add the key.*
- int [rtXmlKeyArrayContains](#) (OSCTXT \*pctxt, OSXSDKeyArray \*pArray)  
*Once all the fields for a key are set, this method is used to check whether the key is already present in the array of keys.*

### 8.5.1 Detailed Description

- Implementation of a dynamic pointer sorted array.

Definition in file [rtXmlKeyArray.h](#).

### 8.5.2 Function Documentation

#### 8.5.2.1 int [rtXmlKeyArrayAdd](#) (OSCTXT \* *pctxt*, OSXSDKeyArray \* *pArray*)

Once all the fields for a key are set, invoke this method to add the key.

It will report an error if any of the following are true:

- a field was skipped and the constraint is a key constraint
- a field was set more than once This method adds the "new" key (pointed to by pArray) into the correct place within the array of keys (and their fields), also pointed to by pArray.

### 8.5.2.2 int rtXmlKeyArrayContains (OSCTXT \* *pctxt*, OSXSDKeyArray \* *pArray*)

Once all the fields for a key are set, this method is used to check whether the key is already present in the array of keys.

Thus, to check a key ref, you set all the fields, then invoke this method. You do NOT invoke rtXmlKeyArrayAdd.

#### Returns

0 if key is found as expected

### 8.5.2.3 int rtXmlKeyArrayInit (OSCTXT \* *pctxt*, OSXSDKeyArray \* *pArray*, OSUINT32 *nmFields*, OSBOOL *key*, const char \* *name*)

Initialize the array by allocating memory for it.

#### Parameters

*pctxt*

*pArray* The array whose data should be initialized.

*nmFields* The number of fields in the key whose data will be held in *pArray*->data.

*key* TRUE if the data is for a key; FALSE if for a unique identity constraint.

*name* of the identity constraint

### 8.5.2.4 void rtXmlKeyArraySetDecimal (OSXSDKeyArray \* *pArray*, const OSREAL \* *value*, OSUINT32 *fldNum*)

Set the given field's value to the given decimal value.

see comment on set\* methods above.

### 8.5.2.5 void rtXmlKeyArraySetInt (OSXSDKeyArray \* *pArray*, OSINT32 *value*, OSUINT32 *fldNum*)

Set the given field's value to the given integer value.

see comment on set\* methods above.

### 8.5.2.6 void rtXmlKeyArraySetString (OSXSDKeyArray \* *pArray*, const OSUTF8CHAR \* *pValue*, OSUINT32 *fldNum*)

Set the given field's value to the given string value.

see comment on set\* methods above.

### 8.5.2.7 void rtXmlKeyArraySetUInt (OSXSDKeyArray \* *pArray*, OSUINT32 *value*, OSUINT32 *fldNum*)

Set the given field's value to the given unsigned integer value.

see comment on set\* methods above.



## 8.6 rtXmlNamespace.h File Reference

XML namespace handling structures and function definitions.

```
#include "rtxsrc/rtxContext.h"
#include "rtxsrc/rtxDynPtrArray.h"
#include "rtxsrc/rtxXmlQName.h"
#include "rtxmlsrc/rtXmlExternDefs.h"
```

### Defines

- #define [RTXMLNSSETQNAME](#)(qname, pNS)  
*This macro populates the given QName structure with information from the given namespace structure (namespace URI and prefix).*

### Functions

- OSXMLNamespace \* [rtXmlNSAddNamespace](#) (OSCTXT \*pctxt, OSRTDList \*pNSAttrs, const OSUTF8CHAR \*prefix, const OSUTF8CHAR \*uri)  
*This function adds a namespace to the context namespace list.*
- OSBOOL [rtXmlNSEqual](#) (OSXMLNamespace \*pNS1, OSXMLNamespace \*pNS2)  
*This function checks if two namespace records are equal.*
- void [rtXmlNSFreeAttrList](#) (OSCTXT \*pctxt, OSRTDList \*pNSAttrs)  
*This function frees dynamic memory used to hold namespace attribute values.*
- const OSUTF8CHAR \* [rtXmlNSGetPrefix](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*uri)  
*This function gets a namespace prefix assigned to the given URI.*
- const OSUTF8CHAR \* [rtXmlNSGetPrefixUsingIndex](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*uri, OSUINT32 idx)  
*This function gets a namespace prefix assigned to the given URI using the given index to select a specific prefix from the URI/prefix map.*
- OSUINT32 [rtXmlNSGetPrefixCount](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*uri)  
*This function returns the total number of prefixes currently assigned to the given URI.*
- int [rtXmlNSGetPrefixIndex](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*prefix, OSUINT32 \*pcount)  
*This function gets the index of a given prefix in the internal list of prefixes maintained for a given URI in the namespace stack.*
- const OSUTF8CHAR \* [rtXmlNSGetQName](#) (OSCTXT \*pctxt, OSUTF8CHAR \*buf, size\_t bufsiz, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*localName)  
*This function creates a QName in the given fixed-size buffer.*
- EXTMLMETHOD const OSUTF8CHAR \* [rtXmlNSGetAttrPrefix](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*namespaceURI, OSRTDList \*pNSAttrs)

*This function returns a namespace prefix for use with attributes in the given namespace.*

- `const OSUTF8CHAR * rtXmlNSGetAttrQName` (`OSCTXT *pctxt`, `OSUTF8CHAR *buf`, `size_t bufsiz`, `OSXMLNamespace *pNS`, `const OSUTF8CHAR *localName`, `OSRTDList *pNSAttrs`)

*This function creates a QName for a qualified attribute.*

- `OSXMLNamespace * rtXmlNSLookupURI` (`OSCTXT *pctxt`, `const OSUTF8CHAR *uri`)

*This function looks up a namespace in the context namespace stack using URI as the key value.*

- `OSXMLNamespace * rtXmlNSLookupURIInList` (`OSRTDList *pNSAttrs`, `const OSUTF8CHAR *uri`)

*This function looks up a namespace in the given list using URI as the key value.*

- `const OSUTF8CHAR * rtXmlNSLookupPrefixForURI` (`OSCTXT *pctxt`, `const OSUTF8CHAR *uri`)

*This function looks up a namespace in the context namespace stack using URI as the key value and returns a non-empty prefix, if one has been defined.*

- `const OSUTF8CHAR * rtXmlNSLookupPrefix` (`OSCTXT *pctxt`, `const OSUTF8CHAR *prefix`)

*This function looks up a namespace in the context namespace list using the prefix as the key value.*

- `const OSUTF8CHAR * rtXmlNSLookupPrefixFrag` (`OSCTXT *pctxt`, `const OSUTF8CHAR *prefix`, `size_t prefixLen`)

*This function looks up a namespace in the context namespace list using the prefix as the key value.*

- `void rtXmlNSRemoveAll` (`OSCTXT *pctxt`)

*This function removes all namespaces from the context namespace list and frees the dynamic memory used to hold the names.*

- `OSXMLNamespace * rtXmlNSSetNamespace` (`OSCTXT *pctxt`, `OSRTDList *pNSAttrs`, `const OSUTF8CHAR *prefix`, `const OSUTF8CHAR *uri`, `OSBOOL override`)

*This function sets a namespace in the context namespace list.*

- `const OSUTF8CHAR * rtXmlNSNewPrefix` (`OSCTXT *pctxt`, `const OSUTF8CHAR *uri`, `OSRTDList *pNSAttrs`)

*This function returns the next unused prefix of the form "nsX" where X is a number.*

- `int rtXmlNSAddPrefixLink` (`OSCTXT *pctxt`, `const OSUTF8CHAR *prefix`, `const OSUTF8CHAR *uri`, `const OSUTF8CHAR *nsTable[ ]`, `OSUINT32 nsTableRowCount`)

*Add a prefix link at the current stack level.*

- `int rtXmlNSFreeAllPrefixLinks` (`OSCTXT *pctxt`, `OSXMLNSPfxLinkStackNode *pStackNode`)

*Free all prefixes links in the given namespace stack entry.*

- `int rtXmlNSFreePrefixLink` (`OSCTXT *pctxt`, `OSXMLNSPfxLink *plink`)

*Free all data within a given namespace prefix link structure.*

- `int rtXmlNSGetIndex` (`OSCTXT *pctxt`, `const OSUTF8CHAR *prefix`)

*Get namespace index for a given namespace prefix based on current namespace stack in context.*

- `int rtXmlNSPush` (`OSCTXT *pctxt`)

*Push new namespace prefix mapping level onto stack.*

- int `rtXmlNSPop` (OSCTXT \*pctxt)  
Remove top namespace prefix mapping level from stack.
- void `rtXmlNSSetURITable` (OSCTXT \*pctxt, const OSUTF8CHAR \*data[], OSUINT32 nrows)  
Set namespace URI table in context.

## 8.6.1 Detailed Description

XML namespace handling structures and function definitions.

Definition in file [rtXmlNamespace.h](#).

## 8.6.2 Define Documentation

### 8.6.2.1 #define RTXMLNSSETQNAME(qname, pNS)

**Value:**

```
if (0 != pNS) { qname.nsPrefix = pNS->prefix; qname.nsURI = pNS->uri; } \
else { qname.nsPrefix = qname.nsURI = 0; }
```

This macro populates the given QName structure with information from the given namespace structure (namespace URI and prefix).

**Parameters**

*qname* Reference to QName structure to be populated. Pointers to items in pNS are assigned directly to fields in qname. No copies of data are made.

*pNS* Pointer to namespace structure.

Definition at line 330 of file [rtXmlNamespace.h](#).

## 8.6.3 Function Documentation

### 8.6.3.1 OSXMLNamespace\* rtXmlNSAddNamespace (OSCTXT \*pctxt, OSRTDList \*pNSAttrs, const OSUTF8CHAR \*prefix, const OSUTF8CHAR \*uri)

This function adds a namespace to the context namespace list.

**Parameters**

*pctxt* Pointer to OSCTXT structure

*pNSAttrs* Namespace attribute list to which namespace info is to be added.

*prefix* Namespace prefix to be added

*uri* Namespace URI to be added

**Returns**

Pointer to namespace structure or NULL if not added.

### 8.6.3.2 OSBOOL rtXmlNSEqual (OSXMLNamespace \* *pNS1*, OSXMLNamespace \* *pNS2*)

This function checks if two namespace records are equal.

This does a deep compare in that it will first check if the pointers are equal and then it will check if the contents are equal (same prefix and URI).

#### Parameters

*pNS1* Pointer to first namespace records to check.

*pNS2* Pointer to second record.

#### Returns

True if records are equal, false otherwise.

### 8.6.3.3 void rtXmlNSFreeAttrList (OSCTXT \* *pctxt*, OSRTDList \* *pNSAttrs*)

This function frees dynamic memory used to hold namespace attribute values.

#### Parameters

*pctxt* Pointer to OSCTXT structure

*pNSAttrs* Pointer to namespace attribute list to be freed.

### 8.6.3.4 EXTXMLMETHOD const OSUTF8CHAR\* rtXmlNSGetAttrPrefix (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *namespaceURI*, OSRTDList \* *pNSAttrs*)

This function returns a namespace prefix for use with attributes in the given namespace.

If a prefix is not found for the namespace, a new namespace entry is created with a generated prefix.

#### Parameters

*pctxt* Pointer to OSCTXT structure

*namespaceURI* Pointer to namespace URI

*pNSAttrs* List of namespace records. If null, the namespace list in the context will be used.

#### Returns

The prefix.

### 8.6.3.5 const OSUTF8CHAR\* rtXmlNSGetAttrQName (OSCTXT \* *pctxt*, OSUTF8CHAR \* *buf*, size\_t *bufsiz*, OSXMLNamespace \* *pNS*, const OSUTF8CHAR \* *localName*, OSRTDList \* *pNSAttrs*)

This function creates a QName for a qualified attribute.

If a prefix is not found for the name, a new namespace entry is created with a generated prefix.

#### Parameters

*pctxt* Pointer to OSCTXT structure

*buf* Buffer into which QName will be written.

*bufsiz* Size of the buffer.

*pNS* Pointer to namespace URI and prefix structure.

*localName* Local name of the item.

*pNSAttrs* List of namespace records. If null, the namespace list in the context will be used.

#### Returns

Pointer to QName buffer (*buf*).

#### 8.6.3.6 `const OSUTF8CHAR* rtXmlNSGetPrefix (OSCTXT * pctxt, const OSUTF8CHAR * uri)`

This function gets a namespace prefix assigned to the given URI.

This gives preference to empty prefixes.

#### Parameters

*pctxt* Pointer to OSCTXT structure

*uri* Namespace URI to be searched for

#### Returns

Pointer to namespace prefix string. If a NULL or empty prefix was assigned to the URI, an empty string is returned. Otherwise, any assigned prefix is returned. If no prefix was assigned, null is returned.

#### 8.6.3.7 `OSUINT32 rtXmlNSGetPrefixCount (OSCTXT * pctxt, const OSUTF8CHAR * uri)`

This function returns the total number of prefixes currently assigned to the given URI.

#### Parameters

*pctxt* Pointer to OSCTXT structure

*uri* Namespace URI to be searched for

#### Returns

Count of prefixes assigned to the URI.

#### 8.6.3.8 `int rtXmlNSGetPrefixIndex (OSCTXT * pctxt, const OSUTF8CHAR * uri, const OSUTF8CHAR * prefix, OSUINT32 * pcount)`

This function gets the index of a given prefix in the internal list of prefixes maintained for a given URI in the namespace stack.

It also may return the total number of prefixes currently assigned to the URI.

#### Parameters

*pctxt* Pointer to OSCTXT structure

*uri* Namespace URI to be searched for

*prefix* Namespace prefix for which to get index.

*pcount* Optional pointer to an integer count variable. If provided, the total number of prefixes currently assigned to the URI will be returned.

#### Returns

Index to namespace prefix or -1 if the prefix is not assigned to the given URI.

#### 8.6.3.9 `const OSUTF8CHAR* rtXmlNSGetPrefixUsingIndex (OSCTXT * pctxt, const OSUTF8CHAR * uri, OSUINT32 idx)`

This function gets a namespace prefix assigned to the given URI using the given index to select a specific prefix from the URI/prefix map.

#### Parameters

*pctxt* Pointer to OSCTXT structure

*uri* Namespace URI to be searched for

*idx* Index to prefix in map. This only has meaning when multiple prefixes have been assigned to the given URI.

#### Returns

Pointer to namespace prefix string

#### 8.6.3.10 `const OSUTF8CHAR* rtXmlNSGetQName (OSCTXT * pctxt, OSUTF8CHAR * buf, size_t bufsiz, const OSUTF8CHAR * uri, const OSUTF8CHAR * localName)`

This function creates a QName in the given fixed-size buffer.

If the name will not fit in the buffer, it is truncated.

#### Parameters

*pctxt* Pointer to OSCTXT structure

*buf* Buffer into which qname will be written.

*bufsiz* Size of the buffer.

*uri* Namespace URI.

*localName* Local name of the item.

#### Returns

Pointer to QName buffer (*buf*).

#### 8.6.3.11 `const OSUTF8CHAR* rtXmlNSLookupPrefix (OSCTXT * pctxt, const OSUTF8CHAR * prefix)`

This function looks up a namespace in the context namespace list using the prefix as the key value.

#### Parameters

*pctxt* Pointer to OSCTXT structure

*prefix* Namespace Prefix to be found.

#### Returns

Pointer to namespace URI or NULL if not found.

**8.6.3.12** `const OSUTF8CHAR* rtXmlNSLookupPrefixForURI (OSCTXT * pctxt, const OSUTF8CHAR * uri)`

This function looks up a namespace in the context namespace stack using URI as the key value and returns a non-empty prefix, if one has been defined.

**Parameters**

*pctxt* Pointer to OSCTXT structure  
*uri* Namespace URI to be found.

**Returns**

Pointer to non-empty prefix. NULL if URI is not found or URI has no associated non-empty prefix.

**8.6.3.13** `const OSUTF8CHAR* rtXmlNSLookupPrefixFrag (OSCTXT * pctxt, const OSUTF8CHAR * prefix, size_t prefixLen)`

This function looks up a namespace in the context namespace list using the prefix as the key value.

**Parameters**

*pctxt* Pointer to OSCTXT structure  
*prefix* Namespace Prefix to be found.  
*prefixLen* Namespace Prefix length.

**Returns**

Pointer to namespace URI or NULL if not found.

**8.6.3.14** `OSXMLNamespace* rtXmlNSLookupURI (OSCTXT * pctxt, const OSUTF8CHAR * uri)`

This function looks up a namespace in the context namespace stack using URI as the key value.

**Parameters**

*pctxt* Pointer to OSCTXT structure  
*uri* Namespace URI to be found.

**Returns**

Pointer to namespace structure or NULL if not found.

**8.6.3.15** `OSXMLNamespace* rtXmlNSLookupURIInList (OSRTDList * pNSAttrs, const OSUTF8CHAR * uri)`

This function looks up a namespace in the given list using URI as the key value.

**Parameters**

*pNSAttrs* List of namespace records.

*uri* Namespace URI to be found.

#### Returns

Pointer to namespace structure or NULL if not found.

#### 8.6.3.16 `const OSUTF8CHAR* rtXmlNSNewPrefix (OSCTXT * pctxt, const OSUTF8CHAR * uri, OSRTDList * pNSAttrs)`

This function returns the next unused prefix of the form "nsX" where X is a number.

The new namespace declaration is added to the list provided or the context list if a NULL pointer is passed for *pNSAttrs*.

#### Parameters

*pctxt* Pointer to OSCTXT structure

*uri* Namespace URI. Must not be NULL or empty string.

*pNSAttrs* Pointer to list of namespace attributes. If null, the namespace list in the context will be used.

#### Returns

New namespace prefix.

#### 8.6.3.17 `void rtXmlNSRemoveAll (OSCTXT * pctxt)`

This function removes all namespaces from the context namespace list and frees the dynamic memory used to hold the names.

#### Parameters

*pctxt* Pointer to OSCTXT structure

#### 8.6.3.18 `OSXMLNamespace* rtXmlNSSetNamespace (OSCTXT * pctxt, OSRTDList * pNSAttrs, const OSUTF8CHAR * prefix, const OSUTF8CHAR * uri, OSBOOL override)`

This function sets a namespace in the context namespace list.

If the given namespace URI does not exist in the list, the namespace is added. If the URI is found, the action depends on the value of the override flag. If true, the value of the namespace prefix will be changed to the given prefix. If false, the existing namespace specification is not altered.

#### Parameters

*pctxt* Pointer to OSCTXT structure

*pNSAttrs* Namespace attribute list to which namespace info is to be added.

*prefix* Namespace prefix

*uri* Namespace URI

*override* Should existing definition be changed?

#### Returns

Pointer to namespace structure or NULL if not set.



## 8.7 rtXmlpCppDecFuncs.h File Reference

XML low-level C++ decode functions.

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

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

### 8.7.1 Detailed Description

XML low-level C++ decode functions. These are overloaded versions of C XML encode functions for use with C++.

Definition in file [rtXmlpCppDecFuncs.h](#).

# Index

- dom
  - domAddAttribute, 103
  - domAddCdata, 104
  - domAddContent, 104
  - domCreateChild, 104
  - domCreateDocument, 105
  - domFreeDoc, 105
  - domGetAttrData, 105
  - domGetChild, 105
  - domGetDoc, 106
  - domGetElementName, 106
  - domGetNext, 106
  - domGetNextAttr, 107
  - domGetNodeAttributesNum, 107
  - domGetNodeContent, 107
  - domGetNodeFirstAttribute, 107
  - domGetRootElement, 108
  - domParseFile, 108
  - domSaveDoc, 108
- DOM API functions., 102
- DOM runtime encode/decode functions., 110
- domAddAttribute
  - dom, 103
- domAddCdata
  - dom, 104
- domAddContent
  - dom, 104
- domCreateChild
  - dom, 104
- domCreateDocument
  - dom, 105
- domFreeDoc
  - dom, 105
- domGetAttrData
  - dom, 105
- domGetChild
  - dom, 105
- domGetDoc
  - dom, 106
- domGetElementName
  - dom, 106
- domGetNext
  - dom, 106
- domGetNextAttr
  - dom, 107
- domGetNodeAttributesNum
  - dom, 107
- domGetNodeContent
  - dom, 107
- domGetNodeFirstAttribute
  - dom, 107
- domGetRootElement
  - dom, 108
- domParseFile
  - dom, 108
- domSaveDoc
  - dom, 108
- osrtdom.h, 117
- osrtdom.h, 117
  - OSXMLGroupDesc, 136
  - rtSaxGetAttrValue, 136
  - rtSaxGetElemID, 136
  - rtSaxGetElemID8, 137
  - rtSaxHasXMLNSAttrs, 137
  - rtSaxIsEmptyBuffer, 137
  - rtSaxSortAttrs, 138
  - rtSaxStrListMatch, 138
  - rtSaxStrListParse, 138
  - rtXmlCmpBase64Str, 139
  - rtXmlCmpHexStr, 139
  - rtXmlCreateFileInputSource, 139
  - rtXmlInitContext, 139
  - rtXmlInitContextUsingKey, 140
  - rtXmlInitCtxAppInfo, 140
  - rtXmlMatchBase64Str, 140
  - rtXmlMatchDate, 140
  - rtXmlMatchDateTime, 141
  - rtXmlMatchGDay, 141
  - rtXmlMatchGMonth, 141
  - rtXmlMatchGMonthDay, 141
  - rtXmlMatchGYear, 142
  - rtXmlMatchGYearMonth, 142
  - rtXmlMatchHexStr, 142
  - rtXmlMatchTime, 143
  - rtXmlMemFreeAnyAttrs, 143
  - rtXmlNewQName, 143
  - rtXmlPrepareContext, 143
  - rtXmlSetEncC14N, 144
  - rtXmlSetEncDocHdr, 144

- rtXmlSetEncodingStr, [144](#)
- rtXmlSetEncXSINamespace, [144](#)
- rtXmlSetEncXSINilAttr, [145](#)
- rtXmlSetFormatting, [145](#)
- rtXmlSetIndent, [145](#)
- rtXmlSetIndentChar, [146](#)
- rtXmlSetNamespacesSet, [146](#)
- rtXmlSetNoNSSchemaLocation, [146](#)
- rtXmlSetNSPrefixLinks, [146](#)
- rtXmlSetSchemaLocation, [147](#)
- rtXmlSetSoapVersion, [147](#)
- rtXmlSetWriteBOM, [147](#)
- rtXmlSetXSISchemaTypeAttr, [147](#)
- OSXMLGroupDesc, [116](#)
- osrtxml.h, [136](#)
- rtDom
  - rtDomAddAttr, [111](#)
  - rtDomAddNode, [111](#)
  - rtDomAddNSAttrs, [111](#)
  - rtDomAddSubTree, [112](#)
  - rtDomDecodeDoc, [112](#)
  - rtDomEncAny, [113](#)
  - rtDomEncAnyAttr, [113](#)
  - rtDomEncString, [113](#)
  - rtDomEncStringValue, [114](#)
  - rtDomEncXSIAttrs, [114](#)
  - rtDomSetNode, [114](#)
- rtDomAddAttr
  - rtDom, [111](#)
- rtDomAddNode
  - rtDom, [111](#)
- rtDomAddNSAttrs
  - rtDom, [111](#)
- rtDomAddSubTree
  - rtDom, [112](#)
- rtDomDecodeDoc
  - rtDom, [112](#)
- rtDomEncAny
  - rtDom, [113](#)
- rtDomEncAnyAttr
  - rtDom, [113](#)
- rtDomEncString
  - rtDom, [113](#)
- rtDomEncStringValue
  - rtDom, [114](#)
- rtDomEncXSIAttrs
  - rtDom, [114](#)
- rtDomSetNode
  - rtDom, [114](#)
- rtSaxGetAttrValue
  - osrtxml.h, [136](#)
- rtSaxGetElemID
  - osrtxml.h, [136](#)
- rtSaxGetElemID8
  - osrtxml.h, [137](#)
- rtSaxHasXMLNSAttrs
  - osrtxml.h, [137](#)
- rtSaxIsEmptyBuffer
  - osrtxml.h, [137](#)
- rtSaxSortAttrs
  - osrtxml.h, [138](#)
- rtSaxStrListMatch
  - osrtxml.h, [138](#)
- rtSaxStrListParse
  - osrtxml.h, [138](#)
- rtXmlCmpBase64Str
  - osrtxml.h, [139](#)
- rtXmlCmpHexStr
  - osrtxml.h, [139](#)
- rtXmlCreateFileInputSource
  - osrtxml.h, [139](#)
- rtXmlDec
  - rtXmlDecBase64Binary, [9](#)
  - rtXmlDecBase64Str, [10](#)
  - rtXmlDecBase64StrValue, [10](#)
  - rtXmlDecBigInt, [10](#)
  - rtXmlDecBool, [11](#)
  - rtXmlDecDate, [11](#)
  - rtXmlDecDateTime, [11](#)
  - rtXmlDecDecimal, [12](#)
  - rtXmlDecDouble, [12](#)
  - rtXmlDecDynBase64Str, [12](#)
  - rtXmlDecDynHexStr, [13](#)
  - rtXmlDecDynUTF8Str, [13](#)
  - rtXmlDecEmptyElement, [13](#)
  - rtXmlDecGDay, [14](#)
  - rtXmlDecGMonth, [14](#)
  - rtXmlDecGMonthDay, [14](#)
  - rtXmlDecGYear, [15](#)
  - rtXmlDecGYearMonth, [15](#)
  - rtXmlDecHexBinary, [15](#)
  - rtXmlDecHexStr, [16](#)
  - rtXmlDecInt, [16](#)
  - rtXmlDecInt16, [17](#)
  - rtXmlDecInt64, [17](#)
  - rtXmlDecInt8, [17](#)
  - rtXmlDecNSAttr, [18](#)
  - rtXmlDecQName, [18](#)
  - rtXmlDecTime, [19](#)
  - rtXmlDecUInt, [19](#)
  - rtXmlDecUInt16, [19](#)
  - rtXmlDecUInt64, [20](#)
  - rtXmlDecUInt8, [20](#)
  - rtXmlDecUTF8Str, [20](#)
  - rtXmlDecXmlStr, [21](#)
  - rtXmlDecXSIAttr, [21](#)
  - rtXmlDecXSIAttrs, [21](#)

- rtXmlParseElementName, [22](#)
- rtXmlParseElemQName, [22](#)
- rtXmlDecBase64Binary
  - rtXmlDec, [9](#)
- rtXmlDecBase64Str
  - rtXmlDec, [10](#)
- rtXmlDecBase64StrValue
  - rtXmlDec, [10](#)
- rtXmlDecBigInt
  - rtXmlDec, [10](#)
- rtXmlDecBool
  - rtXmlDec, [11](#)
- rtXmlDecDate
  - rtXmlDec, [11](#)
- rtXmlDecDateTime
  - rtXmlDec, [11](#)
- rtXmlDecDecimal
  - rtXmlDec, [12](#)
- rtXmlDecDouble
  - rtXmlDec, [12](#)
- rtXmlDecDynBase64Str
  - rtXmlDec, [12](#)
- rtXmlDecDynHexStr
  - rtXmlDec, [13](#)
- rtXmlDecDynUTF8Str
  - rtXmlDec, [13](#)
- rtXmlDecEmptyElement
  - rtXmlDec, [13](#)
- rtXmlDecGDay
  - rtXmlDec, [14](#)
- rtXmlDecGMonth
  - rtXmlDec, [14](#)
- rtXmlDecGMonthDay
  - rtXmlDec, [14](#)
- rtXmlDecGYear
  - rtXmlDec, [15](#)
- rtXmlDecGYearMonth
  - rtXmlDec, [15](#)
- rtXmlDecHexBinary
  - rtXmlDec, [15](#)
- rtXmlDecHexStr
  - rtXmlDec, [16](#)
- rtXmlDecInt
  - rtXmlDec, [16](#)
- rtXmlDecInt16
  - rtXmlDec, [17](#)
- rtXmlDecInt64
  - rtXmlDec, [17](#)
- rtXmlDecInt8
  - rtXmlDec, [17](#)
- rtXmlDecNSAttr
  - rtXmlDec, [18](#)
- rtXmlDecQName
  - rtXmlDec, [18](#)
- rtXmlDecTime
  - rtXmlDec, [19](#)
- rtXmlDecUInt
  - rtXmlDec, [19](#)
- rtXmlDecUInt16
  - rtXmlDec, [19](#)
- rtXmlDecUInt64
  - rtXmlDec, [20](#)
- rtXmlDecUInt8
  - rtXmlDec, [20](#)
- rtXmlDecUTF8Str
  - rtXmlDec, [20](#)
- rtXmlDecXmlStr
  - rtXmlDec, [21](#)
- rtXmlDecXSIAAttr
  - rtXmlDec, [21](#)
- rtXmlDecXSIAAttrs
  - rtXmlDec, [21](#)
- rtXmlEnc
  - rtXmlEncAny, [30](#)
  - rtXmlEncAnyAttr, [30](#)
  - rtXmlEncAnyTypeValue, [30](#)
  - rtXmlEncBase64Binary, [31](#)
  - rtXmlEncBase64BinaryAttr, [31](#)
  - rtXmlEncBase64StrValue, [31](#)
  - rtXmlEncBigInt, [32](#)
  - rtXmlEncBigIntAttr, [32](#)
  - rtXmlEncBigIntValue, [33](#)
  - rtXmlEncBinStrValue, [33](#)
  - rtXmlEncBitString, [33](#)
  - rtXmlEncBOM, [34](#)
  - rtXmlEncBool, [34](#)
  - rtXmlEncBoolAttr, [35](#)
  - rtXmlEncBoolValue, [35](#)
  - rtXmlEncComment, [35](#)
  - rtXmlEncDate, [36](#)
  - rtXmlEncDateTime, [36](#)
  - rtXmlEncDateTimeValue, [37](#)
  - rtXmlEncDateValue, [37](#)
  - rtXmlEncDecimal, [37](#)
  - rtXmlEncDecimalAttr, [38](#)
  - rtXmlEncDecimalValue, [38](#)
  - rtXmlEncDouble, [38](#)
  - rtXmlEncDoubleAttr, [39](#)
  - rtXmlEncDoubleNormalValue, [39](#)
  - rtXmlEncDoubleValue, [40](#)
  - rtXmlEncEmptyElement, [40](#)
  - rtXmlEncEndDocument, [40](#)
  - rtXmlEncEndElement, [41](#)
  - rtXmlEncEndSoapElems, [41](#)
  - rtXmlEncEndSoapEnv, [41](#)
  - rtXmlEncFloat, [42](#)
  - rtXmlEncFloatAttr, [42](#)
  - rtXmlEncGDay, [42](#)

[rtXmlEncGDayValue](#), [43](#)  
[rtXmlEncGMonth](#), [43](#)  
[rtXmlEncGMonthDay](#), [43](#)  
[rtXmlEncGMonthDayValue](#), [44](#)  
[rtXmlEncGMonthValue](#), [44](#)  
[rtXmlEncGYear](#), [44](#)  
[rtXmlEncGYearMonth](#), [45](#)  
[rtXmlEncGYearMonthValue](#), [45](#)  
[rtXmlEncGYearValue](#), [45](#)  
[rtXmlEncHexBinary](#), [46](#)  
[rtXmlEncHexBinaryAttr](#), [46](#)  
[rtXmlEncHexStrValue](#), [47](#)  
[rtXmlEncIndent](#), [47](#)  
[rtXmlEncInt](#), [47](#)  
[rtXmlEncInt64](#), [48](#)  
[rtXmlEncInt64Attr](#), [48](#)  
[rtXmlEncInt64Value](#), [48](#)  
[rtXmlEncIntAttr](#), [49](#)  
[rtXmlEncIntPattern](#), [49](#)  
[rtXmlEncIntValue](#), [50](#)  
[rtXmlEncNamedBits](#), [50](#)  
[rtXmlEncNSAttrs](#), [50](#)  
[rtXmlEncReal10](#), [51](#)  
[rtXmlEncSoapArrayTypeAttr](#), [51](#)  
[rtXmlEncStartDocument](#), [52](#)  
[rtXmlEncStartElement](#), [52](#)  
[rtXmlEncStartSoapElems](#), [52](#)  
[rtXmlEncStartSoapEnv](#), [53](#)  
[rtXmlEncString](#), [53](#)  
[rtXmlEncStringValue](#), [53](#)  
[rtXmlEncStringValue2](#), [54](#)  
[rtXmlEncTermStartElement](#), [54](#)  
[rtXmlEncTime](#), [54](#)  
[rtXmlEncTimeValue](#), [55](#)  
[rtXmlEncUInt](#), [55](#)  
[rtXmlEncUInt64](#), [56](#)  
[rtXmlEncUInt64Attr](#), [56](#)  
[rtXmlEncUInt64Value](#), [56](#)  
[rtXmlEncUIntAttr](#), [57](#)  
[rtXmlEncUIntValue](#), [57](#)  
[rtXmlEncUnicodeStr](#), [57](#)  
[rtXmlEncUTF8Attr](#), [58](#)  
[rtXmlEncUTF8Attr2](#), [58](#)  
[rtXmlEncUTF8Str](#), [59](#)  
[rtXmlEncXSIAAttrs](#), [59](#)  
[rtXmlEncXSINilAttr](#), [59](#)  
[rtXmlEncXSITypeAttr](#), [60](#)  
[rtXmlEncXSITypeAttr2](#), [60](#)  
[rtXmlFreeInputSource](#), [60](#)  
[rtXmlGetEncBufLen](#), [29](#)  
[rtXmlGetEncBufPtr](#), [29](#)  
[rtXmlGetIndent](#), [61](#)  
[rtXmlGetIndentChar](#), [61](#)  
[rtXmlGetWriteBOM](#), [61](#)  
[rtXmlPrintNSAttrs](#), [61](#)  
[rtXmlSetEncBufPtr](#), [62](#)  
[rtXmlEncAny](#)  
    [rtXmlEnc](#), [30](#)  
[rtXmlEncAnyAttr](#)  
    [rtXmlEnc](#), [30](#)  
[rtXmlEncAnyTypeValue](#)  
    [rtXmlEnc](#), [30](#)  
[rtXmlEncAttrC14N](#)  
    [rtXm1pDec](#), [69](#)  
[rtXmlEncBase64Binary](#)  
    [rtXmlEnc](#), [31](#)  
[rtXmlEncBase64BinaryAttr](#)  
    [rtXmlEnc](#), [31](#)  
[rtXmlEncBase64StrValue](#)  
    [rtXmlEnc](#), [31](#)  
[rtXmlEncBigInt](#)  
    [rtXmlEnc](#), [32](#)  
[rtXmlEncBigIntAttr](#)  
    [rtXmlEnc](#), [32](#)  
[rtXmlEncBigIntValue](#)  
    [rtXmlEnc](#), [33](#)  
[rtXmlEncBinStrValue](#)  
    [rtXmlEnc](#), [33](#)  
[rtXmlEncBitString](#)  
    [rtXmlEnc](#), [33](#)  
[rtXmlEncBOM](#)  
    [rtXmlEnc](#), [34](#)  
[rtXmlEncBool](#)  
    [rtXmlEnc](#), [34](#)  
[rtXmlEncBoolAttr](#)  
    [rtXmlEnc](#), [35](#)  
[rtXmlEncBoolValue](#)  
    [rtXmlEnc](#), [35](#)  
[rtXmlEncComment](#)  
    [rtXmlEnc](#), [35](#)  
[rtXmlEncDate](#)  
    [rtXmlEnc](#), [36](#)  
[rtXmlEncDateTime](#)  
    [rtXmlEnc](#), [36](#)  
[rtXmlEncDateTimeValue](#)  
    [rtXmlEnc](#), [37](#)  
[rtXmlEncDateValue](#)  
    [rtXmlEnc](#), [37](#)  
[rtXmlEncDecimal](#)  
    [rtXmlEnc](#), [37](#)  
[rtXmlEncDecimalAttr](#)  
    [rtXmlEnc](#), [38](#)  
[rtXmlEncDecimalValue](#)  
    [rtXmlEnc](#), [38](#)  
[rtXmlEncDouble](#)  
    [rtXmlEnc](#), [38](#)  
[rtXmlEncDoubleAttr](#)  
    [rtXmlEnc](#), [39](#)

rtXmlEncDoubleNormalValue	rtXmlEncIntAttr
rtXmlEnc, 39	rtXmlEnc, 49
rtXmlEncDoubleValue	rtXmlEncIntPattern
rtXmlEnc, 40	rtXmlEnc, 49
rtXmlEncEmptyElement	rtXmlEncIntValue
rtXmlEnc, 40	rtXmlEnc, 50
rtXmlEncEndDocument	rtXmlEncNamedBits
rtXmlEnc, 40	rtXmlEnc, 50
rtXmlEncEndElement	rtXmlEncNSAttrs
rtXmlEnc, 41	rtXmlEnc, 50
rtXmlEncEndSoapElems	rtXmlEncReal10
rtXmlEnc, 41	rtXmlEnc, 51
rtXmlEncEndSoapEnv	rtXmlEncSoapArrayTypeAttr
rtXmlEnc, 41	rtXmlEnc, 51
rtXmlEncFloat	rtXmlEncStartDocument
rtXmlEnc, 42	rtXmlEnc, 52
rtXmlEncFloatAttr	rtXmlEncStartElement
rtXmlEnc, 42	rtXmlEnc, 52
rtXmlEncGDay	rtXmlEncStartSoapElems
rtXmlEnc, 42	rtXmlEnc, 52
rtXmlEncGDayValue	rtXmlEncStartSoapEnv
rtXmlEnc, 43	rtXmlEnc, 53
rtXmlEncGMonth	rtXmlEncString
rtXmlEnc, 43	rtXmlEnc, 53
rtXmlEncGMonthDay	rtXmlEncStringValue
rtXmlEnc, 43	rtXmlEnc, 53
rtXmlEncGMonthDayValue	rtXmlEncStringValue2
rtXmlEnc, 44	rtXmlEnc, 54
rtXmlEncGMonthValue	rtXmlEncTermStartElement
rtXmlEnc, 44	rtXmlEnc, 54
rtXmlEncGYear	rtXmlEncTime
rtXmlEnc, 44	rtXmlEnc, 54
rtXmlEncGYearMonth	rtXmlEncTimeValue
rtXmlEnc, 45	rtXmlEnc, 55
rtXmlEncGYearMonthValue	rtXmlEncUInt
rtXmlEnc, 45	rtXmlEnc, 55
rtXmlEncGYearValue	rtXmlEncUInt64
rtXmlEnc, 45	rtXmlEnc, 56
rtXmlEncHexBinary	rtXmlEncUInt64Attr
rtXmlEnc, 46	rtXmlEnc, 56
rtXmlEncHexBinaryAttr	rtXmlEncUInt64Value
rtXmlEnc, 46	rtXmlEnc, 56
rtXmlEncHexStringValue	rtXmlEncUIntAttr
rtXmlEnc, 47	rtXmlEnc, 57
rtXmlEncIndent	rtXmlEncUIntValue
rtXmlEnc, 47	rtXmlEnc, 57
rtXmlEncInt	rtXmlEncUnicodeStr
rtXmlEnc, 47	rtXmlEnc, 57
rtXmlEncInt64	rtXmlEncUTF8Attr
rtXmlEnc, 48	rtXmlEnc, 58
rtXmlEncInt64Attr	rtXmlEncUTF8Attr2
rtXmlEnc, 48	rtXmlEnc, 58
rtXmlEncInt64Value	rtXmlEncUTF8Str
rtXmlEnc, 48	rtXmlEnc, 59

- rtXmlEncXSIAttrs
  - rtXmlEnc, 59
- rtXmlEncXSINilAttr
  - rtXmlEnc, 59
- rtXmlEncXSITypeAttr
  - rtXmlEnc, 60
- rtXmlEncXSITypeAttr2
  - rtXmlEnc, 60
- rtXmlErrCodes.h, 149
- rtXmlExternDefs.h, 151
- rtXmlFreeInputSource
  - rtXmlEnc, 60
- rtXmlGetEncBufLen
  - rtXmlEnc, 29
- rtXmlGetEncBufPtr
  - rtXmlEnc, 29
- rtXmlGetIndent
  - rtXmlEnc, 61
- rtXmlGetIndentChar
  - rtXmlEnc, 61
- rtXmlGetWriteBOM
  - rtXmlEnc, 61
- rtXmlInitContext
  - osrtxml.h, 139
- rtXmlInitContextUsingKey
  - osrtxml.h, 140
- rtXmlInitCtxtAppInfo
  - osrtxml.h, 140
- rtXmlKeyArray.h, 152
  - rtXmlKeyArrayAdd, 152
  - rtXmlKeyArrayContains, 152
  - rtXmlKeyArrayInit, 153
  - rtXmlKeyArraySetDecimal, 153
  - rtXmlKeyArraySetInt, 153
  - rtXmlKeyArraySetString, 153
  - rtXmlKeyArraySetUInt, 153
- rtXmlKeyArrayAdd
  - rtXmlKeyArray.h, 152
- rtXmlKeyArrayContains
  - rtXmlKeyArray.h, 152
- rtXmlKeyArrayInit
  - rtXmlKeyArray.h, 153
- rtXmlKeyArraySetDecimal
  - rtXmlKeyArray.h, 153
- rtXmlKeyArraySetInt
  - rtXmlKeyArray.h, 153
- rtXmlKeyArraySetString
  - rtXmlKeyArray.h, 153
- rtXmlKeyArraySetUInt
  - rtXmlKeyArray.h, 153
- rtXmlMatchBase64Str
  - osrtxml.h, 140
- rtXmlMatchDate
  - osrtxml.h, 140
- rtXmlMatchDateTime
  - osrtxml.h, 141
- rtXmlMatchGDay
  - osrtxml.h, 141
- rtXmlMatchGMonth
  - osrtxml.h, 141
- rtXmlMatchGMonthDay
  - osrtxml.h, 141
- rtXmlMatchGYear
  - osrtxml.h, 142
- rtXmlMatchGYearMonth
  - osrtxml.h, 142
- rtXmlMatchHexStr
  - osrtxml.h, 142
- rtXmlMatchTime
  - osrtxml.h, 143
- rtXmlMemFreeAnyAttrs
  - osrtxml.h, 143
- rtXmlNamespace.h, 154
  - rtXmlNSAddNamespace, 156
  - rtXmlNSEqual, 156
  - rtXmlNSFreeAttrList, 157
  - rtXmlNSGetAttrPrefix, 157
  - rtXmlNSGetAttrQName, 157
  - rtXmlNSGetPrefix, 158
  - rtXmlNSGetPrefixCount, 158
  - rtXmlNSGetPrefixIndex, 158
  - rtXmlNSGetPrefixUsingIndex, 159
  - rtXmlNSGetQName, 159
  - rtXmlNSLookupPrefix, 159
  - rtXmlNSLookupPrefixForURI, 159
  - rtXmlNSLookupPrefixFrag, 160
  - rtXmlNSLookupURI, 160
  - rtXmlNSLookupURIInList, 160
  - rtXmlNSNewPrefix, 161
  - rtXmlNSRemoveAll, 161
  - rtXmlNSSetNamespace, 161
  - RTXMLNSSETQNAME, 156
- rtXmlNewQName
  - osrtxml.h, 143
- rtXmlNSAddNamespace
  - rtXmlNamespace.h, 156
- rtXmlNSEqual
  - rtXmlNamespace.h, 156
- rtXmlNSFreeAttrList
  - rtXmlNamespace.h, 157
- rtXmlNSGetAttrPrefix
  - rtXmlNamespace.h, 157
- rtXmlNSGetAttrQName
  - rtXmlNamespace.h, 157
- rtXmlNSGetPrefix
  - rtXmlNamespace.h, 158
- rtXmlNSGetPrefixCount
  - rtXmlNamespace.h, 158

- rtXmlNSGetPrefixIndex
  - rtXmlNamespace.h, 158
- rtXmlNSGetPrefixUsingIndex
  - rtXmlNamespace.h, 159
- rtXmlNSGetQName
  - rtXmlNamespace.h, 159
- rtXmlNSLookupPrefix
  - rtXmlNamespace.h, 159
- rtXmlNSLookupPrefixForURI
  - rtXmlNamespace.h, 159
- rtXmlNSLookupPrefixFrag
  - rtXmlNamespace.h, 160
- rtXmlNSLookupURI
  - rtXmlNamespace.h, 160
- rtXmlNSLookupURIInList
  - rtXmlNamespace.h, 160
- rtXmlNSNewPrefix
  - rtXmlNamespace.h, 161
- rtXmlNSRemoveAll
  - rtXmlNamespace.h, 161
- rtXmlNSSetNamespace
  - rtXmlNamespace.h, 161
- RTXMLNSSETQNAME
  - rtXmlNamespace.h, 156
- rtXmlParseElementName
  - rtXmlDec, 22
- rtXmlParseElemQName
  - rtXmlDec, 22
- rtXmlpCountListItems
  - rtXmlpDec, 69
- rtXmlpCppDecFuncs.h, 162
- rtXmlpCreateReader
  - rtXmlpDec, 69
- rtXmlpDec
  - rtXmlEncAttrC14N, 69
  - rtXmlpCountListItems, 69
  - rtXmlpCreateReader, 69
  - rtXmlpDecAny, 70
  - rtXmlpDecAny2, 70
  - rtXmlpDecAnyAttrStr, 70
  - rtXmlpDecAnyElem, 71
  - rtXmlpDecBase64Str, 71
  - rtXmlpDecBigInt, 71
  - rtXmlpDecBitString, 72
  - rtXmlpDecBool, 72
  - rtXmlpDecDate, 73
  - rtXmlpDecDateTime, 73
  - rtXmlpDecDecimal, 73
  - rtXmlpDecDouble, 74
  - rtXmlpDecDoubleExt, 74
  - rtXmlpDecDynBase64Str, 74
  - rtXmlpDecDynBitString, 75
  - rtXmlpDecDynHexStr, 75
  - rtXmlpDecDynUnicodeStr, 76
  - rtXmlpDecDynUTF8Str, 76
  - rtXmlpDecGDay, 76
  - rtXmlpDecGMonth, 77
  - rtXmlpDecGMonthDay, 77
  - rtXmlpDecGYear, 77
  - rtXmlpDecGYearMonth, 78
  - rtXmlpDecHexStr, 78
  - rtXmlpDecInt, 79
  - rtXmlpDecInt16, 79
  - rtXmlpDecInt64, 79
  - rtXmlpDecInt8, 80
  - rtXmlpDecNamedBits, 80
  - rtXmlpDecStrList, 80
  - rtXmlpDecTime, 81
  - rtXmlpDecUInt, 81
  - rtXmlpDecUInt16, 81
  - rtXmlpDecUInt64, 82
  - rtXmlpDecUInt8, 82
  - rtXmlpDecUTF8Str, 82
  - rtXmlpDecXmlStr, 83
  - rtXmlpDecXmlStrList, 83
  - rtXmlpDecXSIAAttr, 84
  - rtXmlpDecXSIAAttrs, 84
  - rtXmlpDecXSITypeAttr, 84
  - rtXmlpForceDecodeAsGroup, 85
  - rtXmlpGetAttributeCount, 85
  - rtXmlpGetAttributeID, 85
  - rtXmlpGetCurrentLevel, 86
  - rtXmlpGetNextAllElemID, 86
  - rtXmlpGetNextAllElemID16, 86
  - rtXmlpGetNextAllElemID32, 87
  - rtXmlpGetNextElem, 87
  - rtXmlpGetNextElemID, 88
  - rtXmlpGetNextSeqElemID, 88
  - rtXmlpGetNextSeqElemID2, 89
  - rtXmlpGetNextSeqElemIDExt, 89
  - rtXmlpGetReader, 90
  - rtXmlpGetXmInsAttrs, 90
  - rtXmlpGetXSITypeAttr, 90
  - rtXmlpGetXSITypeIndex, 91
  - rtXmlpHasAttributes, 91
  - rtXmlpHideAttributes, 91
  - rtXmlpIsDecodeAsGroup, 92
  - rtXmlpIsEmptyElement, 92
  - rtXmlpIsLastEventDone, 92
  - rtXmlpIsUTF8Encoding, 92
  - rtXmlpListHasItem, 93
  - rtXmlpLookupXSITypeIndex, 93
  - rtXmlpMarkLastEventActive, 93
  - rtXmlpMarkPos, 94
  - rtXmlpMatchEndTag, 94
  - rtXmlpMatchStartTag, 94
  - rtXmlpNeedDecodeAttributes, 95
  - rtXmlpReadBytes, 95



- rtXmlpResetMarkedPos, [95](#)
- rtXmlpRewindToMarkedPos, [95](#)
- rtXmlpSelectAttribute, [96](#)
- rtXmlpSetListMode, [96](#)
- rtXmlpSetMixedContentMode, [96](#)
- rtXmlpSetNamespaceTable, [96](#)
- rtXmlpSetWhiteSpaceMode, [97](#)
- rtXmlpDecAny
  - rtXmlpDec, [70](#)
- rtXmlpDecAny2
  - rtXmlpDec, [70](#)
- rtXmlpDecAnyAttrStr
  - rtXmlpDec, [70](#)
- rtXmlpDecAnyElem
  - rtXmlpDec, [71](#)
- rtXmlpDecBase64Str
  - rtXmlpDec, [71](#)
- rtXmlpDecBigInt
  - rtXmlpDec, [71](#)
- rtXmlpDecBitString
  - rtXmlpDec, [72](#)
- rtXmlpDecBool
  - rtXmlpDec, [72](#)
- rtXmlpDecDate
  - rtXmlpDec, [73](#)
- rtXmlpDecDateTime
  - rtXmlpDec, [73](#)
- rtXmlpDecDecimal
  - rtXmlpDec, [73](#)
- rtXmlpDecDouble
  - rtXmlpDec, [74](#)
- rtXmlpDecDoubleExt
  - rtXmlpDec, [74](#)
- rtXmlpDecDynBase64Str
  - rtXmlpDec, [74](#)
- rtXmlpDecDynBitString
  - rtXmlpDec, [75](#)
- rtXmlpDecDynHexStr
  - rtXmlpDec, [75](#)
- rtXmlpDecDynUnicodeStr
  - rtXmlpDec, [76](#)
- rtXmlpDecDynUTF8Str
  - rtXmlpDec, [76](#)
- rtXmlpDecGDay
  - rtXmlpDec, [76](#)
- rtXmlpDecGMonth
  - rtXmlpDec, [77](#)
- rtXmlpDecGMonthDay
  - rtXmlpDec, [77](#)
- rtXmlpDecGYear
  - rtXmlpDec, [77](#)
- rtXmlpDecGYearMonth
  - rtXmlpDec, [78](#)
- rtXmlpDecHexStr
  - rtXmlpDec, [78](#)
- rtXmlpDecInt
  - rtXmlpDec, [79](#)
- rtXmlpDecInt16
  - rtXmlpDec, [79](#)
- rtXmlpDecInt64
  - rtXmlpDec, [79](#)
- rtXmlpDecInt8
  - rtXmlpDec, [80](#)
- rtXmlpDecNamedBits
  - rtXmlpDec, [80](#)
- rtXmlpDecStrList
  - rtXmlpDec, [80](#)
- rtXmlpDecTime
  - rtXmlpDec, [81](#)
- rtXmlpDecUInt
  - rtXmlpDec, [81](#)
- rtXmlpDecUInt16
  - rtXmlpDec, [81](#)
- rtXmlpDecUInt64
  - rtXmlpDec, [82](#)
- rtXmlpDecUInt8
  - rtXmlpDec, [82](#)
- rtXmlpDecUTF8Str
  - rtXmlpDec, [82](#)
- rtXmlpDecXmlStr
  - rtXmlpDec, [83](#)
- rtXmlpDecXmlStrList
  - rtXmlpDec, [83](#)
- rtXmlpDecXSIAAttr
  - rtXmlpDec, [84](#)
- rtXmlpDecXSIAAttrs
  - rtXmlpDec, [84](#)
- rtXmlpDecXSITypeAttr
  - rtXmlpDec, [84](#)
- rtXmlpForceDecodeAsGroup
  - rtXmlpDec, [85](#)
- rtXmlpGetAttributeCount
  - rtXmlpDec, [85](#)
- rtXmlpGetAttributeID
  - rtXmlpDec, [85](#)
- rtXmlpGetCurrentLevel
  - rtXmlpDec, [86](#)
- rtXmlpGetNextAllElemID
  - rtXmlpDec, [86](#)
- rtXmlpGetNextAllElemID16
  - rtXmlpDec, [86](#)
- rtXmlpGetNextAllElemID32
  - rtXmlpDec, [87](#)
- rtXmlpGetNextElem
  - rtXmlpDec, [87](#)
- rtXmlpGetNextElemID
  - rtXmlpDec, [88](#)
- rtXmlpGetNextSeqElemID
  - rtXmlpDec, [88](#)

- rtXmlpDec, [88](#)
- rtXmlpGetNextSeqElemID2
  - rtXmlpDec, [89](#)
- rtXmlpGetNextSeqElemIDExt
  - rtXmlpDec, [89](#)
- rtXmlpGetReader
  - rtXmlpDec, [90](#)
- rtXmlpGetXmInsAttrs
  - rtXmlpDec, [90](#)
- rtXmlpGetXSITypeAttr
  - rtXmlpDec, [90](#)
- rtXmlpGetXSITypeIndex
  - rtXmlpDec, [91](#)
- rtXmlpHasAttributes
  - rtXmlpDec, [91](#)
- rtXmlpHideAttributes
  - rtXmlpDec, [91](#)
- rtXmlpIsDecodeAsGroup
  - rtXmlpDec, [92](#)
- rtXmlpIsEmptyElement
  - rtXmlpDec, [92](#)
- rtXmlpIsLastEventDone
  - rtXmlpDec, [92](#)
- rtXmlpIsUTF8Encoding
  - rtXmlpDec, [92](#)
- rtXmlpListHasItem
  - rtXmlpDec, [93](#)
- rtXmlpLookupXSITypeIndex
  - rtXmlpDec, [93](#)
- rtXmlpMarkLastEventActive
  - rtXmlpDec, [93](#)
- rtXmlpMarkPos
  - rtXmlpDec, [94](#)
- rtXmlpMatchEndTag
  - rtXmlpDec, [94](#)
- rtXmlpMatchStartTag
  - rtXmlpDec, [94](#)
- rtXmlpNeedDecodeAttributes
  - rtXmlpDec, [95](#)
- rtXmlpReadBytes
  - rtXmlpDec, [95](#)
- rtXmlPrepareContext
  - osrtxml.h, [143](#)
- rtXmlpResetMarkedPos
  - rtXmlpDec, [95](#)
- rtXmlpRewindToMarkedPos
  - rtXmlpDec, [95](#)
- rtXmlPrintNSAttrs
  - rtXmlEnc, [61](#)
- rtXmlpSelectAttribute
  - rtXmlpDec, [96](#)
- rtXmlpSetListMode
  - rtXmlpDec, [96](#)
- rtXmlpSetMixedContentMode
  - rtXmlpDec, [96](#)
- rtXmlSetNamespaceTable
  - rtXmlpDec, [96](#)
- rtXmlpSetWhiteSpaceMode
  - rtXmlpDec, [97](#)
- rtXmlSetEncBufPtr
  - rtXmlEnc, [62](#)
- rtXmlSetEncC14N
  - osrtxml.h, [144](#)
- rtXmlSetEncDocHdr
  - osrtxml.h, [144](#)
- rtXmlSetEncodingStr
  - osrtxml.h, [144](#)
- rtXmlSetEncXSINamespace
  - osrtxml.h, [144](#)
- rtXmlSetEncXSINilAttr
  - osrtxml.h, [145](#)
- rtXmlSetFormatting
  - osrtxml.h, [145](#)
- rtXmlSetIndent
  - osrtxml.h, [145](#)
- rtXmlSetIndentChar
  - osrtxml.h, [146](#)
- rtXmlSetNamespacesSet
  - osrtxml.h, [146](#)
- rtXmlSetNoNSSchemaLocation
  - osrtxml.h, [146](#)
- rtXmlSetNSPrefixLinks
  - osrtxml.h, [146](#)
- rtXmlSetSchemaLocation
  - osrtxml.h, [147](#)
- rtXmlSetSoapVersion
  - osrtxml.h, [147](#)
- rtXmlSetWriteBOM
  - osrtxml.h, [147](#)
- rtXmlSetXSITypeAttr
  - osrtxml.h, [147](#)
- rtXmlUtil
  - rtXmlWriteToFile, [63](#)
- rtXmlWriteToFile
  - rtXmlUtil, [63](#)
- XML decode functions., [7](#)
- XML encode functions., [23](#)
- XML pull-parser decode functions., [64](#)
- XML run-time error status codes., [98](#)
- XML utility functions., [63](#)
- XML\_E\_BASE
  - xmlErrCodes, [99](#)
- XML\_E\_ELEMMISRQ
  - xmlErrCodes, [99](#)
- XML\_E\_ELEMSMISRQ
  - xmlErrCodes, [100](#)
- XML\_E\_FLDABSENT

- xmlErrCodes, [100](#)
- XML\_E\_NOMATCH
  - xmlErrCodes, [100](#)
- XML\_E\_NSURINOTFOU
  - xmlErrCodes, [100](#)
- XML\_E\_TAGMISMATCH
  - xmlErrCodes, [100](#)
- XML\_OK\_EOB
  - xmlErrCodes, [100](#)
- XML\_OK\_FRAG
  - xmlErrCodes, [100](#)
- xmlErrCodes
  - XML\_E\_BASE, [99](#)
  - XML\_E\_ELEMMISRQ, [99](#)
  - XML\_E\_ELEMSMISRQ, [100](#)
  - XML\_E\_FLDABSENT, [100](#)
  - XML\_E\_NOMATCH, [100](#)
  - XML\_E\_NSURINOTFOU, [100](#)
  - XML\_E\_TAGMISMATCH, [100](#)
  - XML\_OK\_EOB, [100](#)
  - XML\_OK\_FRAG, [100](#)