

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.

Contents

1	Main Page	1
2	Module Index	2
2.1	Modules	2
3	Class Index	3
3.1	Class Hierarchy	3
4	Class Index	4
4.1	Class List	4
5	File Index	5
5.1	File List	5
6	Module Documentation	6
6.1	XML decode functions.	6
6.1.1	Function Documentation	8
6.1.1.1	rtXmlDecBase64Binary	8
6.1.1.2	rtXmlDecBase64Str	9
6.1.1.3	rtXmlDecBase64StrValue	9
6.1.1.4	rtXmlDecBigInt	10
6.1.1.5	rtXmlDecBool	10
6.1.1.6	rtXmlDecDate	10
6.1.1.7	rtXmlDecDateTime	11
6.1.1.8	rtXmlDecDecimal	11
6.1.1.9	rtXmlDecDouble	11
6.1.1.10	rtXmlDecDynBase64Str	12
6.1.1.11	rtXmlDecDynHexStr	12
6.1.1.12	rtXmlDecDynUTF8Str	12
6.1.1.13	rtXmlDecEmptyElement	13

6.1.1.14	rtXmlDecGDay	13
6.1.1.15	rtXmlDecGMonth	13
6.1.1.16	rtXmlDecGMonthDay	14
6.1.1.17	rtXmlDecGYear	14
6.1.1.18	rtXmlDecGYearMonth	14
6.1.1.19	rtXmlDecHexBinary	15
6.1.1.20	rtXmlDecHexStr	15
6.1.1.21	rtXmlDecInt	15
6.1.1.22	rtXmlDecInt16	16
6.1.1.23	rtXmlDecInt64	16
6.1.1.24	rtXmlDecInt8	16
6.1.1.25	rtXmlDecNSAttr	17
6.1.1.26	rtXmlDecQName	17
6.1.1.27	rtXmlDecTime	18
6.1.1.28	rtXmlDecUInt	18
6.1.1.29	rtXmlDecUInt16	18
6.1.1.30	rtXmlDecUInt64	19
6.1.1.31	rtXmlDecUInt8	19
6.1.1.32	rtXmlDecUTF8Str	19
6.1.1.33	rtXmlDecXmlStr	20
6.1.1.34	rtXmlDecXSIAAttr	20
6.1.1.35	rtXmlDecXSIAAttrs	21
6.1.1.36	rtXmlParseElementName	21
6.1.1.37	rtXmlParseElemQName	21
6.2	XML encode functions	22
6.2.1	Define Documentation	28
6.2.1.1	rtXmlGetEncBufLen	28
6.2.1.2	rtXmlGetEncBufPtr	28
6.2.2	Function Documentation	29
6.2.2.1	rtXmlEncAny	29
6.2.2.2	rtXmlEncAnyAttr	29
6.2.2.3	rtXmlEncAnyTypeValue	29
6.2.2.4	rtXmlEncBase64Binary	30
6.2.2.5	rtXmlEncBase64BinaryAttr	30
6.2.2.6	rtXmlEncBase64StrValue	31
6.2.2.7	rtXmlEncBigInt	31
6.2.2.8	rtXmlEncBigIntAttr	31

6.2.2.9	rtXmlEncBigIntValue	32
6.2.2.10	rtXmlEncBinStrValue	32
6.2.2.11	rtXmlEncBitString	33
6.2.2.12	rtXmlEncBOM	33
6.2.2.13	rtXmlEncBool	33
6.2.2.14	rtXmlEncBoolAttr	34
6.2.2.15	rtXmlEncBoolValue	34
6.2.2.16	rtXmlEncComment	35
6.2.2.17	rtXmlEncDate	35
6.2.2.18	rtXmlEncDateTime	35
6.2.2.19	rtXmlEncDateTimeValue	36
6.2.2.20	rtXmlEncDateValue	36
6.2.2.21	rtXmlEncDecimal	36
6.2.2.22	rtXmlEncDecimalAttr	37
6.2.2.23	rtXmlEncDecimalValue	37
6.2.2.24	rtXmlEncDouble	38
6.2.2.25	rtXmlEncDoubleAttr	38
6.2.2.26	rtXmlEncDoubleNormalValue	38
6.2.2.27	rtXmlEncDoubleValue	39
6.2.2.28	rtXmlEncEmptyElement	39
6.2.2.29	rtXmlEncEndDocument	40
6.2.2.30	rtXmlEncEndElement	40
6.2.2.31	rtXmlEncEndSoapElems	40
6.2.2.32	rtXmlEncEndSoapEnv	41
6.2.2.33	rtXmlEncFloat	41
6.2.2.34	rtXmlEncFloatAttr	41
6.2.2.35	rtXmlEncGDay	42
6.2.2.36	rtXmlEncGDayValue	42
6.2.2.37	rtXmlEncGMonth	42
6.2.2.38	rtXmlEncGMonthDay	43
6.2.2.39	rtXmlEncGMonthDayValue	43
6.2.2.40	rtXmlEncGMonthValue	43
6.2.2.41	rtXmlEncGYear	44
6.2.2.42	rtXmlEncGYearMonth	44
6.2.2.43	rtXmlEncGYearMonthValue	44
6.2.2.44	rtXmlEncGYearValue	45
6.2.2.45	rtXmlEncHexBinary	45

6.2.2.46	rtXmlEncHexBinaryAttr	45
6.2.2.47	rtXmlEncHexStrValue	46
6.2.2.48	rtXmlEncIndent	46
6.2.2.49	rtXmlEncInt	46
6.2.2.50	rtXmlEncInt64	47
6.2.2.51	rtXmlEncInt64Attr	47
6.2.2.52	rtXmlEncInt64Value	48
6.2.2.53	rtXmlEncIntAttr	48
6.2.2.54	rtXmlEncIntPattern	48
6.2.2.55	rtXmlEncIntValue	49
6.2.2.56	rtXmlEncNamedBits	49
6.2.2.57	rtXmlEncNSAttrs	50
6.2.2.58	rtXmlEncReal10	50
6.2.2.59	rtXmlEncSoapArrayTypeAttr	50
6.2.2.60	rtXmlEncStartDocument	51
6.2.2.61	rtXmlEncStartElement	51
6.2.2.62	rtXmlEncStartSoapElems	52
6.2.2.63	rtXmlEncStartSoapEnv	52
6.2.2.64	rtXmlEncString	52
6.2.2.65	rtXmlEncStringValue	53
6.2.2.66	rtXmlEncStringValue2	53
6.2.2.67	rtXmlEncTermStartElement	53
6.2.2.68	rtXmlEncTime	54
6.2.2.69	rtXmlEncTimeValue	54
6.2.2.70	rtXmlEncUInt	54
6.2.2.71	rtXmlEncUInt64	55
6.2.2.72	rtXmlEncUInt64Attr	55
6.2.2.73	rtXmlEncUInt64Value	56
6.2.2.74	rtXmlEncUIntAttr	56
6.2.2.75	rtXmlEncUIntValue	56
6.2.2.76	rtXmlEncUnicodeStr	57
6.2.2.77	rtXmlEncUTF8Attr	57
6.2.2.78	rtXmlEncUTF8Attr2	57
6.2.2.79	rtXmlEncUTF8Str	58
6.2.2.80	rtXmlEncXSAttrs	58
6.2.2.81	rtXmlEncXSINilAttr	59
6.2.2.82	rtXmlEncXSITypeAttr	59

6.2.2.83	rtXmlEncXSITypeAttr2	59
6.2.2.84	rtXmlFreeInputSource	60
6.2.2.85	rtXmlGetIndent	60
6.2.2.86	rtXmlGetIndentChar	60
6.2.2.87	rtXmlGetWriteBOM	60
6.2.2.88	rtXmlPrintNSAttrs	61
6.2.2.89	rtXmlSetEncBufPtr	61
6.3	XML utility functions.	62
6.3.1	Function Documentation	62
6.3.1.1	rtXmlWriteToFile	62
6.4	XML pull-parser decode functions.	63
6.4.1	Function Documentation	68
6.4.1.1	rtXmlEncAttrC14N	68
6.4.1.2	rtXmlpCountListItems	68
6.4.1.3	rtXmlpCreateReader	68
6.4.1.4	rtXmlpDecAny	69
6.4.1.5	rtXmlpDecAny2	69
6.4.1.6	rtXmlpDecAnyAttrStr	69
6.4.1.7	rtXmlpDecAnyElem	70
6.4.1.8	rtXmlpDecBase64Str	70
6.4.1.9	rtXmlpDecBigInt	71
6.4.1.10	rtXmlpDecBitString	71
6.4.1.11	rtXmlpDecBool	71
6.4.1.12	rtXmlpDecDate	72
6.4.1.13	rtXmlpDecDateTime	72
6.4.1.14	rtXmlpDecDecimal	72
6.4.1.15	rtXmlpDecDouble	73
6.4.1.16	rtXmlpDecDoubleExt	73
6.4.1.17	rtXmlpDecDynBase64Str	74
6.4.1.18	rtXmlpDecDynBitString	74
6.4.1.19	rtXmlpDecDynHexStr	74
6.4.1.20	rtXmlpDecDynUnicodeStr	75
6.4.1.21	rtXmlpDecDynUTF8Str	75
6.4.1.22	rtXmlpDecGDay	75
6.4.1.23	rtXmlpDecGMonth	76
6.4.1.24	rtXmlpDecGMonthDay	76
6.4.1.25	rtXmlpDecGYear	77

6.4.1.26	rtXmlpDecYearMonth	77
6.4.1.27	rtXmlpDecHexStr	77
6.4.1.28	rtXmlpDecInt	78
6.4.1.29	rtXmlpDecInt16	78
6.4.1.30	rtXmlpDecInt64	78
6.4.1.31	rtXmlpDecInt8	79
6.4.1.32	rtXmlpDecNamedBits	79
6.4.1.33	rtXmlpDecStrList	80
6.4.1.34	rtXmlpDecTime	80
6.4.1.35	rtXmlpDecUInt	80
6.4.1.36	rtXmlpDecUInt16	81
6.4.1.37	rtXmlpDecUInt64	81
6.4.1.38	rtXmlpDecUInt8	81
6.4.1.39	rtXmlpDecUTF8Str	82
6.4.1.40	rtXmlpDecXmlStr	82
6.4.1.41	rtXmlpDecXmlStrList	82
6.4.1.42	rtXmlpDecXSIAAttr	83
6.4.1.43	rtXmlpDecXSIAAttrs	83
6.4.1.44	rtXmlpDecXSISTypeAttr	83
6.4.1.45	rtXmlpForceDecodeAsGroup	84
6.4.1.46	rtXmlpGetAttributeCount	84
6.4.1.47	rtXmlpGetAttributeID	84
6.4.1.48	rtXmlpGetCurrentLevel	85
6.4.1.49	rtXmlpGetNextAllElemID	85
6.4.1.50	rtXmlpGetNextAllElemID16	85
6.4.1.51	rtXmlpGetNextAllElemID32	86
6.4.1.52	rtXmlpGetNextElem	86
6.4.1.53	rtXmlpGetNextElemID	87
6.4.1.54	rtXmlpGetNextSeqElemID	87
6.4.1.55	rtXmlpGetNextSeqElemID2	88
6.4.1.56	rtXmlpGetNextSeqElemIDExt	88
6.4.1.57	rtXmlpGetReader	89
6.4.1.58	rtXmlpGetXmInsAttrs	89
6.4.1.59	rtXmlpGetXSISTypeAttr	90
6.4.1.60	rtXmlpGetXSISTypeIndex	90
6.4.1.61	rtXmlpHasAttributes	90
6.4.1.62	rtXmlpHideAttributes	91

6.4.1.63	rtXmlpIsDecodeAsGroup	91
6.4.1.64	rtXmlpIsEmptyElement	91
6.4.1.65	rtXmlpIsLastEventDone	91
6.4.1.66	rtXmlpIsUTF8Encoding	92
6.4.1.67	rtXmlpListHasItem	92
6.4.1.68	rtXmlpLookupXSITypeIndex	92
6.4.1.69	rtXmlpMarkLastEventActive	93
6.4.1.70	rtXmlpMarkPos	93
6.4.1.71	rtXmlpMatchEndTag	93
6.4.1.72	rtXmlpMatchStartTag	93
6.4.1.73	rtXmlpNeedDecodeAttributes	94
6.4.1.74	rtXmlpReadBytes	94
6.4.1.75	rtXmlpResetMarkedPos	94
6.4.1.76	rtXmlpRewindToMarkedPos	95
6.4.1.77	rtXmlpSelectAttribute	95
6.4.1.78	rtXmlpSetListMode	95
6.4.1.79	rtXmlpSetMixedContentMode	95
6.4.1.80	rtXmlpSetNamespaceTable	96
6.4.1.81	rtXmlpSetWhiteSpaceMode	96
7	Class Documentation	97
7.1	OSXMLContentHandler Class Reference	97
7.1.1	Detailed Description	97
7.1.2	Member Function Documentation	98
7.1.2.1	characters	98
7.1.2.2	endElement	98
7.1.2.3	startElement	98
7.2	OSXMLDecodeBuffer Class Reference	100
7.2.1	Detailed Description	101
7.2.2	Constructor & Destructor Documentation	101
7.2.2.1	OSXMLDecodeBuffer	101
7.2.2.2	OSXMLDecodeBuffer	101
7.2.2.3	OSXMLDecodeBuffer	101
7.2.3	Member Function Documentation	102
7.2.3.1	decodeXML	102
7.2.3.2	init	102
7.2.3.3	isA	102

7.2.3.4	isWellFormed	102
7.2.3.5	parseElementName	103
7.2.3.6	parseElemQName	103
7.2.3.7	setMaxErrors	103
7.2.4	Member Data Documentation	103
7.2.4.1	mbOwnStream	103
7.3	OSXMLDefaultHandler Class Reference	104
7.3.1	Detailed Description	104
7.3.2	Member Function Documentation	104
7.3.2.1	characters	104
7.3.2.2	endElement	105
7.3.2.3	getState	105
7.3.2.4	startElement	105
7.4	OSXMLDefaultHandlerIF Class Reference	107
7.4.1	Detailed Description	107
7.5	OSXMLEncodeBuffer Class Reference	108
7.5.1	Detailed Description	109
7.5.2	Constructor & Destructor Documentation	109
7.5.2.1	OSXMLEncodeBuffer	109
7.5.3	Member Function Documentation	109
7.5.3.1	addXMLHeader	109
7.5.3.2	addXMLText	109
7.5.3.3	getMsgLen	110
7.5.3.4	init	110
7.5.3.5	isA	110
7.5.3.6	setFragment	110
7.5.3.7	write	110
7.5.3.8	write	111
7.6	OSXMLEncodeStream Class Reference	112
7.6.1	Detailed Description	113
7.6.2	Constructor & Destructor Documentation	113
7.6.2.1	OSXMLEncodeStream	113
7.6.2.2	OSXMLEncodeStream	113
7.6.3	Member Function Documentation	114
7.6.3.1	encodeAttr	114
7.6.3.2	encodeText	114
7.6.3.3	endElement	114

7.6.3.4	getMsgPtr	115
7.6.3.5	getStream	115
7.6.3.6	init	115
7.6.3.7	isA	115
7.6.3.8	startDocument	115
7.6.3.9	startElement	116
7.6.3.10	termStartElement	116
7.6.4	Member Data Documentation	116
7.6.4.1	mbOwnStream	116
7.6.4.2	mpCtxt	116
7.6.4.3	mpStream	117
7.7	OSXMLGroupDesc Struct Reference	118
7.7.1	Detailed Description	118
7.8	OSXMLMessageBuffer Class Reference	119
7.8.1	Detailed Description	120
7.8.2	Constructor & Destructor Documentation	120
7.8.2.1	OSXMLMessageBuffer	120
7.8.3	Member Function Documentation	120
7.8.3.1	getIndent	120
7.8.3.2	getIndentChar	120
7.8.3.3	getWriteBOM	120
7.8.3.4	setAppInfo	121
7.8.3.5	setFormatting	121
7.8.3.6	setIndent	121
7.8.3.7	setIndentChar	121
7.8.3.8	setNamespace	121
7.8.3.9	setWriteBOM	122
7.9	OSXMLNamespaceClass Class Reference	123
7.9.1	Detailed Description	123
7.9.2	Constructor & Destructor Documentation	123
7.9.2.1	OSXMLNamespaceClass	123
7.9.2.2	OSXMLNamespaceClass	124
7.10	OSXMLStrListHandler Class Reference	125
7.10.1	Detailed Description	125
7.11	OSXSDDGlobalElement Class Reference	126
7.11.1	Detailed Description	127
7.11.2	Constructor & Destructor Documentation	128

7.11.2.1	OSXSDGlobalElement	128
7.11.2.2	OSXSDGlobalElement	128
7.11.2.3	OSXSDGlobalElement	128
7.11.2.4	~OSXSDGlobalElement	128
7.11.3	Member Function Documentation	128
7.11.3.1	decodeFrom	128
7.11.3.2	encodeTo	129
7.11.3.3	getCtxtPtr	129
7.11.3.4	memAlloc	129
7.11.3.5	memFreePtr	129
7.11.3.6	setDefaultNamespace	129
7.11.3.7	setDiag	130
7.11.3.8	setEncXSINamespace	130
7.11.3.9	setMsgBuf	130
7.11.3.10	setNamespace	130
7.11.3.11	setNoNSSchemaLocation	131
7.11.3.12	setSchemaLocation	131
7.11.3.13	setXSIType	131
7.11.3.14	validateFrom	131
7.11.4	Member Data Documentation	131
7.11.4.1	mpContext	131
8	File Documentation	133
8.1	osrtxml.h File Reference	133
8.1.1	Detailed Description	150
8.1.2	Typedef Documentation	150
8.1.2.1	OSXMLGroupDesc	150
8.1.3	Function Documentation	150
8.1.3.1	rtSaxGetAttrValue	150
8.1.3.2	rtSaxGetElemID	151
8.1.3.3	rtSaxGetElemID8	151
8.1.3.4	rtSaxHasXMLNSAttrs	152
8.1.3.5	rtSaxIsEmptyBuffer	152
8.1.3.6	rtSaxSortAttrs	152
8.1.3.7	rtSaxStrListMatch	152
8.1.3.8	rtSaxStrListParse	153
8.1.3.9	rtXmlCmpBase64Str	153

8.1.3.10	rtXmlCmpHexStr	153
8.1.3.11	rtXmlCreateFileInputSource	153
8.1.3.12	rtXmlInitContext	154
8.1.3.13	rtXmlInitContextUsingKey	154
8.1.3.14	rtXmlInitCtxtAppInfo	154
8.1.3.15	rtXmlMatchBase64Str	155
8.1.3.16	rtXmlMatchDate	155
8.1.3.17	rtXmlMatchDateTime	155
8.1.3.18	rtXmlMatchGDay	155
8.1.3.19	rtXmlMatchGMonth	156
8.1.3.20	rtXmlMatchGMonthDay	156
8.1.3.21	rtXmlMatchGYear	156
8.1.3.22	rtXmlMatchGYearMonth	157
8.1.3.23	rtXmlMatchHexStr	157
8.1.3.24	rtXmlMatchTime	157
8.1.3.25	rtXmlMemFreeAnyAttrs	157
8.1.3.26	rtXmlNewQName	158
8.1.3.27	rtXmlPrepareContext	158
8.1.3.28	rtXmlSetEncC14N	158
8.1.3.29	rtXmlSetEncDocHdr	158
8.1.3.30	rtXmlSetEncodingStr	159
8.1.3.31	rtXmlSetEncXSINamespace	159
8.1.3.32	rtXmlSetEncXSINilAttr	159
8.1.3.33	rtXmlSetFormatting	159
8.1.3.34	rtXmlSetIndent	160
8.1.3.35	rtXmlSetIndentChar	160
8.1.3.36	rtXmlSetNamespacesSet	160
8.1.3.37	rtXmlSetNoNSSchemaLocation	161
8.1.3.38	rtXmlSetNSPrefixLinks	161
8.1.3.39	rtXmlSetSchemaLocation	161
8.1.3.40	rtXmlSetSoapVersion	161
8.1.3.41	rtXmlSetWriteBOM	162
8.1.3.42	rtXmlSetXSITypeAttr	162
8.2	OSXMLDecodeBuffer.h File Reference	163
8.2.1	Detailed Description	163
8.3	OSXMLEncodeBuffer.h File Reference	164
8.3.1	Detailed Description	164

8.4	OSXMLEncodeStream.h File Reference	165
8.4.1	Detailed Description	165
8.5	OSXMLMessageBuffer.h File Reference	166
8.5.1	Detailed Description	166
8.6	rtSaxCppAny.h File Reference	167
8.6.1	Detailed Description	167
8.7	rtSaxCppAnyType.h File Reference	168
8.7.1	Detailed Description	168
8.8	rtSaxCppSimpleType.h File Reference	169
8.8.1	Detailed Description	169
8.9	rtSaxCppSoap.h File Reference	170
8.9.1	Detailed Description	170
8.10	rtSaxCppStrList.h File Reference	171
8.10.1	Detailed Description	171
8.11	rtXmlCppEncFuncs.h File Reference	172
8.11.1	Detailed Description	172
8.11.2	Function Documentation	172
8.11.2.1	rtXmlCppEncAnyAttr	172
8.11.2.2	rtXmlCppEncAnyTypeValue	173
8.11.2.3	rtXmlCppEncStartElement	173
8.11.2.4	rtXmlEncAny	173
8.11.2.5	rtXmlEncString	174
8.12	rtXmlCppMsgBuf.h File Reference	175
8.12.1	Detailed Description	175
8.13	rtXmlCppNamespace.h File Reference	176
8.13.1	Detailed Description	176
8.14	rtXmlCppXSDElement.h File Reference	177
8.14.1	Detailed Description	177
8.15	rtXmlpCppDecFuncs.h File Reference	178
8.15.1	Detailed Description	178

Chapter 1

Main Page

C XML Runtime Library Functions

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

The categories of functions provided are as follows:

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

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

XML decode functions.	6
XML encode functions.	22
XML utility functions.	62
XML pull-parser decode functions.	63

Chapter 3

Class Index

3.1 Class Hierarchy

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

OSXMLContentHandler	97
OSXMLDefaultHandlerIF	107
OSXMLDefaultHandler	104
OSXMLGroupDesc	118
OSXMLMessageBuffer	119
OSXMLDecodeBuffer	100
OSXMLEncodeBuffer	108
OSXMLEncodeStream	112
OSXMLNamespaceClass	123
OSXMLStrListHandler	125
OSXSDGlobalElement	126

Chapter 4

Class Index

4.1 Class List

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

OSXMLContentHandler (Receive notification of general document events)	97
OSXMLDecodeBuffer (Derived from the OSXMLMessageBuffer base class)	100
OSXMLDefaultHandler (This class is derived from the SAX class DefaultHandler base class)	104
OSXMLDefaultHandlerIF (This class is derived from the SAX class DefaultHandler base class)	107
OSXMLEncodeBuffer (Derived from the OSXMLMessageBuffer base class)	108
OSXMLEncodeStream (Derived from the OSXMLMessageBuffer base class)	112
OSXMLGroupDesc (OSXMLGroupDesc describes how entries in an OSXMLElemIDRec array make up a group)	118
OSXMLMessageBuffer (The XML message buffer class is derived from the OSMessageBuffer base class)	119
OSXMLNamespaceClass (This class is used to hold an XML namespace prefix to URI mapping)	123
OSXMLStrListHandler (OSXMLStrListHandler)	125
OSXSDGlobalElement (XSD global element base class)	126

Chapter 5

File Index

5.1 File List

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

osrxml.h (XML low-level C encode/decode functions)	133
OSXMLDecodeBuffer.h (XML decode buffer or stream class definition)	163
OSXMLEncodeBuffer.h (XML encode message buffer class definition)	164
OSXMLEncodeStream.h (XML encode stream class definition)	165
OSXMLMessageBuffer.h (XML encode/decode buffer and stream base class)	166
rtSaxCppAny.h	167
rtSaxCppAnyType.h	168
rtSaxCppParser.h	??
rtSaxCppParserIF.h	??
rtSaxCppSimpleType.h	169
rtSaxCppSoap.h	170
rtSaxCppStrList.h	171
rtXmlCppEncFuncs.h (XML low-level C++ encode functions)	172
rtXmlCppMsgBuf.h (This file is deprecated)	175
rtXmlCppNamespace.h (XML namespace handling structures and function definitions)	176
rtXmlCppXSDElement.h (C++ run-time XML schema global element class definition)	177
rtXmlpCppDecFuncs.h (XML low-level C++ decode functions)	178

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 noctx, const OSOCTET *value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)
This function encodes a variable of the XSD base64Binary type.
- int `rtXmlEncBase64BinaryAttr` (OSCTXT *pctx, OSUINT32 noctx, 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 noctx, 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 (≥ 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 * pctx, 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

pctx 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 *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 *pctx)

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 nNamespaces)

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 [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 * *pnocts*, OSINT32 *bufsize*)

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

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

Parameters

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

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

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

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

Returns

Completion status of operation:

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

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.

Chapter 7

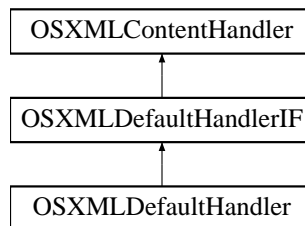
Class Documentation

7.1 OSXMLContentHandler Class Reference

Receive notification of general document events.

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

Inheritance diagram for OSXMLContentHandler:



Public Member Functions

The virtual document handler interface

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

7.1.1 Detailed Description

Receive notification of general document events.

Definition at line 112 of file rtSaxCppParserIF.h.

7.1.2 Member Function Documentation

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

Receive notification of character data.

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

Parameters

chars The characters from the XML document.

length The length of chars.

Implemented in [OSXMLDefaultHandler](#).

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

Receive notification of the end of an element.

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

Parameters

uri The URI of the associated namespace for this element

localname The local part of the element name

qname The QName of this element

Implemented in [OSXMLDefaultHandler](#).

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

Receive notification of the beginning of an element.

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

Parameters

uri The URI of the associated namespace for this element

localname The local part of the element name

qname The QName of this element

attrs The attributes name/value pairs attached to the element, if any.

See also

[endElement](#)

Implemented in [OSXMLDefaultHandler](#).

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

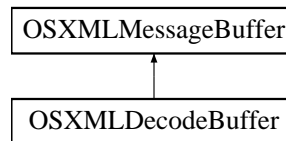
- [rtSaxCppParserIF.h](#)

7.2 OSXMLDecodeBuffer Class Reference

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

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

Inheritance diagram for OSXMLDecodeBuffer:



Public Member Functions

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

Protected Attributes

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

7.2.1 Detailed Description

The [OSXMLDecodeBuffer](#) class is derived from the [OSXMLMessageBuffer](#) base class. It contains variables and methods specific to decoding XML messages. It is used to manage an input buffer or stream containing a message to be decoded.

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

Definition at line 45 of file OSXMLDecodeBuffer.h.

7.2.2 Constructor & Destructor Documentation

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

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

Parameters

xmlFile A pointer to name of file to be decoded.

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

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

Parameters

msgbuf A pointer to a buffer containing an XML message.

bufsiz Size of the message buffer.

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

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

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

Parameters

inputStream reference to the OSInputStream object

7.2.3 Member Function Documentation

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

This method decodes an XML message associated with this buffer.

Returns

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

Parameters

pReader Pointer to OSXMLReaderClass object.

Returns

Completion status.

7.2.3.2 virtual EXTXMLMETHOD int OSXMLDecodeBuffer::init () [virtual]

This method initializes the decode message buffer.

Returns

Completion status of operation:

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

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

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

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

Parameters

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

Returns

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

Definition at line 169 of file OSXMLDecodeBuffer.h.

7.2.3.4 EXTXMLMETHOD OSBOOL OSXMLDecodeBuffer::isWellFormed ()

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

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

Returns

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

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

This method parses the initial tag from an XML message.

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

Parameters

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

Returns

Completion status of operation:

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

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

This method parses the initial tag from an XML message.

Parameters

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

Returns

Completion status of operation:

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

7.2.3.7 EXTXMLMETHOD OSUINT32 OSXMLDecodeBuffer::setMaxErrors (OSUINT32 *maxErrors*)

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

Parameters

maxErrors The desired number of maximum errors.

Returns

The previously set maximum number of errors.

7.2.4 Member Data Documentation

7.2.4.1 OSBOOL OSXMLDecodeBuffer::mbOwnStream [protected]

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

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

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

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

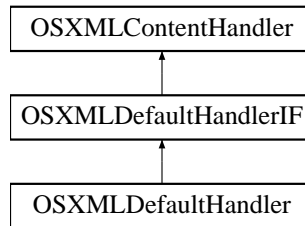
- [OSXMLDecodeBuffer.h](#)

7.3 OSXMLDefaultHandler Class Reference

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

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

Inheritance diagram for OSXMLDefaultHandler:



Public Member Functions

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

7.3.1 Detailed Description

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

Definition at line 58 of file rtSaxCppParser.h.

7.3.2 Member Function Documentation

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

Receive notification of character data.

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

Parameters

chars The characters from the XML document.

length The length of chars.

Implements [OSXMLContentHandler](#).

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

Receive notification of the end of an element.

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

Parameters

uri The URI of the associated namespace for this element

localname The local part of the element name

qname The QName of this element

Implements [OSXMLContentHandler](#).

7.3.2.3 OSINT16 OSXMLDefaultHandler::getState () [inline]

This method returns the current state of the decoding process.

Returns

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

Definition at line 124 of file rtSaxCppParser.h.

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

Receive notification of the beginning of an element.

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

Parameters

uri The URI of the associated namespace for this element

localname The local part of the element name

qname The QName of this element

attrs The attributes name/value pairs attached to the element, if any.

See also

[endElement](#)

Implements [OSXMLContentHandler](#).

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

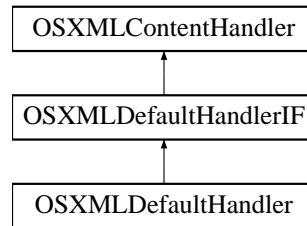
- `rtSaxCppParser.h`

7.4 OSXMLDefaultHandlerIF Class Reference

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

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

Inheritance diagram for OSXMLDefaultHandlerIF:



7.4.1 Detailed Description

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

Definition at line 260 of file rtSaxCppParserIF.h.

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

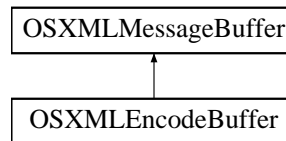
- rtSaxCppParserIF.h

7.5 OSXMLEncodeBuffer Class Reference

The `OSXMLEncodeBuffer` class is derived from the `OSXMLMessageBuffer` base class.

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

Inheritance diagram for `OSXMLEncodeBuffer`:



Public Member Functions

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

7.5.1 Detailed Description

The `OSXMLEncodeBuffer` class is derived from the `OSXMLMessageBuffer` base class. It contains variables and methods specific to encoding XML messages. It is used to manage the buffer into which a message is to be encoded.

Definition at line 38 of file `OSXMLEncodeBuffer.h`.

7.5.2 Constructor & Destructor Documentation

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

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

Parameters

pMsgBuf A pointer to a fixed size message buffer to receive the encoded message.

msgBufLen Size of the fixed-size message buffer.

7.5.3 Member Function Documentation

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

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

Parameters

version XML version (default is 1.0)

encoding Character encoding (default is UTF-8)

newLine Add newline char at end of header

Returns

Zero if success or negative error code.

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

This method adds encoded XML text to the encode buffer.

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

Parameters

text Encoded XML text to be added to the buffer.

Returns

Zero if success or negative error code.

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

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

Returns

Length of the XML message encapsulated within this buffer object.

Definition at line 90 of file OSXMLEncodeBuffer.h.

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

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

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

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

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

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

Parameters

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

Returns

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

Definition at line 118 of file OSXMLEncodeBuffer.h.

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

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

an XML header will not be added).

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

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

Parameters

fp Pointer to FILE structure to which the encoded message will be written.

Returns

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

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

This method writes the encoded message to the given file.

Parameters

filename The name of file to which the encoded message will be written.

Returns

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

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

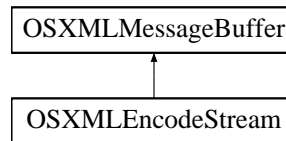
- [OSXMLEncodeBuffer.h](#)

7.6 OSXMLEncodeStream Class Reference

The `OSXMLEncodeStream` class is derived from the `OSXMLMessageBuffer` base class.

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

Inheritance diagram for `OSXMLEncodeStream`:



Public Member Functions

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

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

- EXTXMLMETHOD int [termStartElement](#) ()

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

Protected Attributes

- OSRTOutputStream * [mpStream](#)

A pointer to an OSRTOutputStream object.

- OSBOOL [mbOwnStream](#)

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

- OSCTXT * [mpCtxt](#)

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

7.6.1 Detailed Description

The [OSXMLEncodeStream](#) class is derived from the [OSXMLMessageBuffer](#) base class. It contains variables and methods specific to streaming encoding XML messages. It is used to manage the stream into which a message is to be encoded.

Definition at line 40 of file OSXMLEncodeStream.h.

7.6.2 Constructor & Destructor Documentation

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

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

The stream is assumed to have been previously initialized.

Parameters

outputStream reference to the OSOutputStream object

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

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

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

Parameters

pOutputStream reference to the OSOutputStream object

ownStream set ownership for the passed stream object.

7.6.3 Member Function Documentation

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

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

Parameters

name Attribute name.

value UTF-8 string value to be encoded.

Returns

Completion status of operation:

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

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

This method encodes XML textual content.

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

Parameters

value UTF-8 string value to be encoded.

Returns

Completion status of operation:

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

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

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

Parameters

elemName XML element name.

pNS XML namespace information (prefix and URI).

Returns

Completion status of operation:

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

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

This is a virtual method that must be overridden by derived classes to allow access to the stored message. The base class implementation returns a null value.

Returns

A pointer to the stored message.

Definition at line 154 of file OSXMLEncodeStream.h.

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

This method returns the output stream associated with the object.

Returns

A pointer to the output stream.

Definition at line 161 of file OSXMLEncodeStream.h.

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

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

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

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

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

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

Parameters

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

Returns

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

Definition at line 143 of file OSXMLEncodeStream.h.

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

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

This includes the XML header declaration.

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

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

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

Parameters

elemName XML element name.

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

pNSAttrs List of namespace attributes to be added to element.

terminate Add closing '>' character.

Returns

Completion status of operation:

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

7.6.3.10 EXTXMLMETHOD int OSXMLEncodeStream::termStartElement ()

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

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

Returns

Completion status of operation:

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

7.6.4 Member Data Documentation

7.6.4.1 OSBOOL OSXMLEncodeStream::mbOwnStream [protected]

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

Definition at line 47 of file OSXMLEncodeStream.h.

7.6.4.2 OSCTXT* OSXMLEncodeStream::mpCtxt [protected]

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

Definition at line 51 of file OSXMLEncodeStream.h.

7.6.4.3 OSRTOutputStream* OSXMLEncodeStream::mpStream [protected]

A pointer to an OSRTOutputStream object.

Definition at line 43 of file OSXMLEncodeStream.h.

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

- [OSXMLEncodeStream.h](#)

7.7 OSXMLGroupDesc Struct Reference

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

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

7.7.1 Detailed Description

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

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

Definition at line 142 of file osrtxml.h.

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

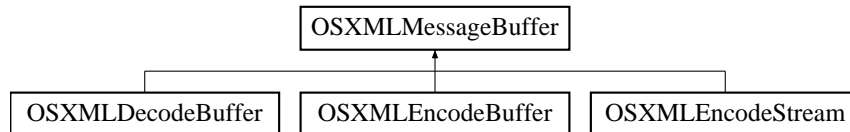
- [osrtxml.h](#)

7.8 OSXMLMessageBuffer Class Reference

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

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

Inheritance diagram for OSXMLMessageBuffer:



Public Member Functions

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

Protected Member Functions

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

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

7.8.1 Detailed Description

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

Definition at line 42 of file OSXMLMessageBuffer.h.

7.8.2 Constructor & Destructor Documentation

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

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

Parameters

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

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

7.8.3 Member Function Documentation

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

This method returns current XML output indent value.

Returns

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

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

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

Returns

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

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

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

Returns

TRUE if writing BOM is enabled, FALSE otherwise.

**7.8.3.4 virtual EXTXMLMETHOD void OSXMLMessageBuffer::setAppInfo (void * *pXMLInfo*)
[virtual]**

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

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

Parameters

pXMLInfo Pointer to context information.

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

This method sets XML output formatting to the given value.

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

Parameters

doFormatting Boolean value indicating if formatting is to be done

Returns

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

7.8.3.6 EXTXMLMETHOD void OSXMLMessageBuffer::setIndent (OSUINT8 *indent*)

This method sets XML output indent to the given value.

Parameters

indent Number of spaces per indent. Default is 3.

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

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

Parameters

indentChar Indent character. Default is space.

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

This method sets a namespace in the context namespace list.

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

Parameters

prefix Namespace prefix

uri Namespace URI

pNSAttrs Namespace list to which namespace is to be added

7.8.3.9 EXTXMLMETHOD void OSXMLMessageBuffer::setWriteBOM (OSBOOL *write*)

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

Parameters

write TRUE if BOM should be written, FALSE otherwise.

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

- [OSXMLMessageBuffer.h](#)

7.9 OSXMLNamespaceClass Class Reference

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

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

Public Member Functions

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

7.9.1 Detailed Description

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

Definition at line 38 of file `rtXmlCppNamespace.h`.

7.9.2 Constructor & Destructor Documentation

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

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

A deep copy of the values is done.

Parameters

nsPrefix Namespace prefix value.

nsURI Namespace URI value.

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

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

A deep copy of the values is done.

Parameters

nsPrefix Namespace prefix value.

nsPrefixBytes Namespace prefix value size in bytes.

nsURI Namespace URI value.

nsURIBytes Namespace URI value size in bytes.

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

- [rtXmlCppNamespace.h](#)

7.10 OSXMLStrListHandler Class Reference

[OSXMLStrListHandler](#).

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

7.10.1 Detailed Description

[OSXMLStrListHandler](#).

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

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

- [rtSaxCppStrList.h](#)

7.11 OSXSDGlobalElement Class Reference

XSD global element base class.

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

Public Member Functions

- [OSXSDGlobalElement](#) (OSRTMessageBufferIF &msgBuf)
This constructor sets the internal message buffer pointer to point at the given message buffer or stream object.
- [OSXSDGlobalElement](#) (const [OSXSDGlobalElement](#) &o)
The copy constructor sets the internal message buffer pointer and context to point at the message buffer and context from the original OSCType object.
- virtual [~OSXSDGlobalElement](#) ()
The virtual destructor does nothing.
- int [decode](#) ()
The decode method decodes the message described by the encapsulated message buffer object.
- virtual int [decodeFrom](#) (OSRTMessageBufferIF &)
The decodeFrom method decodes a message from the given message buffer or stream argument.
- int [encode](#) ()
The encode method encodes a message using the encoding rules specified by the derived message buffer object.
- virtual int [encodeTo](#) (OSRTMessageBufferIF &)
The encodeTo method encodes a message into the given message buffer or stream argument.
- OSCTXT * [getCtxtPtr](#) ()
The getCtxtPtr method returns the underlying C runtime context.
- void * [memAlloc](#) (size_t numocts)
The memAlloc method allocates memory using the C runtime memory management functions.
- void [memFreePtr](#) (void *ptr)
The memFreePtr method frees the memory at a specific location.
- void [setDefaultNamespace](#) (const OSUTF8CHAR *uri)
The setDefaultNamespace method sets the default namespace for the element to the given value.
- void [setDiag](#) (OSBOOL value=TRUE)
The setDiag method turns diagnostic tracing on or off.
- void [setEncXSINamespace](#) (OSBOOL value=TRUE)
The setEncXSINamespace method sets a flag in the internal context that indicates the xsi namespace attribute must be encoded.
- void [setNamespace](#) (const OSUTF8CHAR *prefix, const OSUTF8CHAR *uri)

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

- void [setNoNSSchemaLocation](#) (const OSUTF8CHAR *uri)
The setNoNSSchemaLocation method adds an xsi:noNamespaceSchemaLocation attribute to the document.
- void [setSchemaLocation](#) (const OSUTF8CHAR *uri)
The setSchemaLocation method adds an xsi:schemaLocation attribute to the document.
- void [setXSIType](#) (const OSUTF8CHAR *typeName)
The setXSIType method sets a type name to be used in the xsi:type attribute in the top-level module element declaration.
- int [validate](#) ()
The validate method validates the message described by the encapsulated message buffer object.
- virtual int [validateFrom](#) (OSRTMessageBufferIF &)
The validateFrom method validates a message from the given message buffer or stream argument.

Protected Member Functions

- [OSXSDGlobalElement](#) ()
The default constructor sets the message pointer member variable to NULL and creates a new context object.
- [OSXSDGlobalElement](#) (OSRTContext &ctxt)
This constructor sets the message pointer member variable to NULL and initializes the context object to point at the given context value.
- void [setMsgBuf](#) (OSRTMessageBufferIF &msgBuf)
The setMsgBuf method is used to set the internal message buffer pointer to point at the given message buffer or stream object.

Protected Attributes

- OSRTCtxtPtr [mpContext](#)
The mpContext member variable holds a reference-counted C runtime variable.
- OSRTMessageBufferIF * [mpMsgBuf](#)
The mpMsgBuf member variable is a pointer to a derived message buffer or stream class that will manage the message being encoded or decoded.

7.11.1 Detailed Description

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

Definition at line 57 of file rtXmlCppXSDElement.h.

7.11.2 Constructor & Destructor Documentation

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

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

Parameters

ctxt - Reference to a context object.

Definition at line 85 of file rtXmlCppXSDElement.h.

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

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

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

Parameters

msgBuf - Reference to a message buffer or stream object.

Definition at line 105 of file rtXmlCppXSDElement.h.

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

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

Parameters

o - Reference to a global element object.

Definition at line 116 of file rtXmlCppXSDElement.h.

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

The virtual destructor does nothing.

It is overridden by derived versions of this class.

Definition at line 123 of file rtXmlCppXSDElement.h.

7.11.3 Member Function Documentation

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

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

Parameters

- Message buffer or stream containing message to decode.

Definition at line 138 of file rtXmlCppXSDElement.h.

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

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

Parameters

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

Definition at line 153 of file rtXmlCppXSDElement.h.

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

The `getCtxtPtr` method returns the underlying C runtime context.

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

Definition at line 159 of file rtXmlCppXSDElement.h.

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

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

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

Parameters

- numocts* - Number of bytes of memory to allocate

Definition at line 172 of file rtXmlCppXSDElement.h.

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

The `memFreePtr` method frees the memory at a specific location.

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

Parameters

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

Definition at line 200 of file rtXmlCppXSDElement.h.

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

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

Parameters

uri - Default namespace URI

Definition at line 210 of file rtXmlCpXSDElement.h.

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

The setDiag method turns diagnostic tracing on or off.

Parameters

value - Boolean on/off value (default = on)

Definition at line 219 of file rtXmlCpXSDElement.h.

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

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

Parameters

value - Boolean on/off value (default = on)

Definition at line 229 of file rtXmlCpXSDElement.h.

References rtXmlSetEncXSINamespace().

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

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

Parameters

msgBuf - Reference to a message buffer or stream object.

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

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

Parameters

prefix - Namespace prefix

uri - Namespace URI

Definition at line 240 of file rtXmlCpXSDElement.h.

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

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

Parameters

uri - URI for noNamespaceSchemaLocation.

Definition at line 250 of file rtXmlCppXSDElement.h.

References rtXmlSetNoNSSchemaLocation().

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

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

Parameters

uri - URI for schemaLocation.

Definition at line 260 of file rtXmlCppXSDElement.h.

References rtXmlSetSchemaLocation().

7.11.3.13 void OSXSDGlobalElement::setXSISiteType (const OSUTF8CHAR * typeName) [inline]

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

Parameters

typeName - XSI type name

Definition at line 270 of file rtXmlCppXSDElement.h.

References rtXmlSetXSISiteTypeAttr().

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

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

Parameters

- Message buffer or stream containing message to validate.

Definition at line 287 of file rtXmlCppXSDElement.h.

7.11.4 Member Data Documentation

7.11.4.1 OSRTCtxtPtr OSXSDGlobalElement::mpContext [protected]

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

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

Definition at line 65 of file rtXmlCppXSDElement.h.

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

- [rtXmlCppXSDElement.h](#)

Chapter 8

File Documentation

8.1 osrxml.h File Reference

XML low-level C encode/decode functions.

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

Classes

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

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.

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 *pnocets, 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 *pnocets, 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 noctx, 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 noctx, 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 *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- int [rtXmlEncHexBinary](#) (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

This function encodes a variable of the XSD hexBinary type.

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

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

- int [rtXmlEncHexStrValue](#) (OSCTXT *pctxt, OSUINT32 noctx, const OSOCTET *data)

This function encodes a variable of the XSD hexBinary type.

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

This function adds indentation whitespace to the output stream.

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

This function encodes a variable of the XSD integer type.

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

This function encodes a variable of the XSD integer type.

- int [rtXmlEncIntAttr](#) (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *attrName, 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 *pctxt, OSUINT64 value, const OSUTF8CHAR *attrName, size_t attrNameLen)
This function encodes a variable of the XSD integer type as an attribute (name="value").
- int **rtXmlEncXSISchema** (OSCTXT *pctxt, OSBOOL needXSI)
This function encodes XML schema instance (XSI) attributes at the beginning of an XML document.
- int **rtXmlEncXSISchemaTypeAttr** (OSCTXT *pctxt, const OSUTF8CHAR *value)
This function encodes an XML schema instance (XSI) type attribute value (xsi:type="value").
- int **rtXmlEncXSISchemaTypeAttr2** (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).
- int **rtXmlPrepareContext** (OSCTXT *pctxt)
This function prepares the context for another encode by setting the state back to OSXMLINIT and moving the buffer's cursor back to the beginning of the buffer.

- int [rtXmlSetEncC14N](#) (OSCTXT *pctxt, OSBOOL value)
This function sets the option to encode in C14N mode.
- int [rtXmlSetEncXSINamespace](#) (OSCTXT *pctxt, OSBOOL value)
This function sets a flag in the context that indicates the XSI namespace declaration (xmlns:xsi) should be added to the encoded XML instance.
- int [rtXmlSetEncXSINilAttr](#) (OSCTXT *pctxt, OSBOOL value)
This function sets a flag in the context that indicates the XSI attribute declaration (xmlns:xsi) should be added to the encoded XML instance.
- int [rtXmlSetEncDocHdr](#) (OSCTXT *pctxt, OSBOOL value)
This function sets the option to add the XML document header (i.e.
- int [rtXmlSetEncodingStr](#) (OSCTXT *pctxt, const OSUTF8CHAR *encodingStr)
This function sets the XML output encoding to the given value.
- int [rtXmlSetFormatting](#) (OSCTXT *pctxt, OSBOOL doFormatting)
This function sets XML output formatting to the given value.
- int [rtXmlSetIndent](#) (OSCTXT *pctxt, OSUINT8 indent)
This function sets XML output indent to the given value.
- int [rtXmlSetIndentChar](#) (OSCTXT *pctxt, char indentChar)
This function sets XML output indent character to the given value.
- void [rtXmlSetNamespacesSet](#) (OSCTXT *pctxt, OSBOOL value)
This function sets the context 'namespaces are set' flag.
- int [rtXmlSetNSPrefixLinks](#) (OSCTXT *pctxt, OSRTDList *pNSAttrs)
This function sets namespace prefix/URI links in the namespace prefix stack in the context structure.
- int [rtXmlSetSchemaLocation](#) (OSCTXT *pctxt, const OSUTF8CHAR *schemaLocation)
This function sets the XML Schema Instance (xsi) schema location attribute to be added to an encoded document.
- int [rtXmlSetNoNSSchemaLocation](#) (OSCTXT *pctxt, const OSUTF8CHAR *schemaLocation)
This function sets the XML Schema Instance (xsi) no namespace schema location attribute to be added to an encoded document.
- void [rtXmlSetSoapVersion](#) (OSCTXT *pctxt, OSUINT8 version)
This function sets the SOAP version number.
- int [rtXmlSetXSITypeAttr](#) (OSCTXT *pctxt, const OSUTF8CHAR *xsiType)
This function sets the XML Schema Instance (xsi) type attribute value.
- int [rtXmlSetWriteBOM](#) (OSCTXT *pctxt, OSBOOL write)
This function sets whether the Unicode byte order mark is encoded.
- int [rtXmlMatchHexStr](#) (OSCTXT *pctxt, size_t minLength, size_t maxLength)
This function tests the context buffer for containing a correct hexadecimal string.

- int [rtXmlMatchBase64Str](#) (OSCTXT *pctx, size_t minLength, size_t maxLength)
This function tests the context buffer for containing a correct base64 string.
- int [rtXmlMatchDate](#) (OSCTXT *pctx)
This function tests the context buffer for containing a correct date string.
- int [rtXmlMatchTime](#) (OSCTXT *pctx)
This function tests the context buffer for containing a correct time string.
- int [rtXmlMatchDateTime](#) (OSCTXT *pctx)
This function tests the context buffer for containing a correct dateTime string.
- int [rtXmlMatchGYear](#) (OSCTXT *pctx)
This function tests the context buffer for containing a correct gYear string.
- int [rtXmlMatchGYearMonth](#) (OSCTXT *pctx)
This function tests the context buffer for containing a correct gYearMonth string.
- int [rtXmlMatchGMonth](#) (OSCTXT *pctx)
This function tests the context buffer for containing a correct gMonth string.
- int [rtXmlMatchGMonthDay](#) (OSCTXT *pctx)
This function tests the context buffer for containing a correct gMonthDay string.
- int [rtXmlMatchGDay](#) (OSCTXT *pctx)
This function tests the context buffer for containing a correct gDay string.
- OSUTF8CHAR * [rtXmlNewQName](#) (OSCTXT *pctx, const OSUTF8CHAR *localName, const OSUTF8CHAR *prefix)
This function creates a new QName given the localName and prefix parts.
- OSBOOL [rtXmlCmpBase64Str](#) (OSUINT32 noctx, const OSOCTET *data1, const OSUTF8CHAR *data2)
This function compares an array of octets to a base64 string.
- OSBOOL [rtXmlCmpHexStr](#) (OSUINT32 noctx, const OSOCTET *data1, const OSUTF8CHAR *data2)
This function compares an array of octets to a hex string.
- const OSUTF8CHAR * [rtSaxGetAttrValue](#) (const OSUTF8CHAR *attrName, const OSUTF8CHAR *const *attrs)
This function looks up an attribute in the attribute array returned by SAX to the startElement function.
- OSINT16 [rtSaxGetElemID](#) (OSINT16 *pState, OSINT16 prevElemIdx, const OSUTF8CHAR *localName, OSINT32 nsidx, const OSSAXElemTableRec idtab[], const OSINT16 *fstab, OSINT16 fstabRows, OSINT16 fstabCols)
This function looks up a sequence element name in the given element info array.
- OSINT16 [rtSaxGetElemID8](#) (OSINT16 *pState, OSINT16 prevElemIdx, const OSUTF8CHAR *localName, OSINT32 nsidx, const OSSAXElemTableRec idtab[], const OSINT8 *fstab, OSINT16 fstabRows, OSINT16 fstabCols)
This function is a space optimized version of `rtSaxGetElemID`.

- OSBOOL [rtSaxHasXMLNSAttrs](#) (const OSUTF8CHAR *const *attrs)
This function checks if the given attribute list contains one or more XML namespace attributes (xmlns).
- OSBOOL [rtSaxIsEmptyBuffer](#) (OSCTXT *pctxt)
This function checks if the buffer in the context is empty or not.
- int [rtSaxStrListParse](#) (OSCTXT *pctxt, OSRTMEMBUF *pMemBuf, OSRTDList *pvalue)
This function parses the list of strings.
- int [rtSaxSortAttrs](#) (OSCTXT *pctxt, const OSUTF8CHAR *const *attrs, OSUINT16 **order)
This function sorts a SAX attribute list in ascending order based on attribute name.
- int [rtSaxStrListMatch](#) (OSCTXT *pctxt)
This function matches the list of strings.
- int [rtXmlWriteToFile](#) (OSCTXT *pctxt, const char *filename)
This function writes the encoded XML message stored in the context message buffer out to a file.
- int [rtXmlpDecAny](#) (OSCTXT *pctxt, const OSUTF8CHAR **pvalue)
This function decodes an arbitrary XML section of code as defined by the XSD any type (xsd:any).
- int [rtXmlpDecAny2](#) (OSCTXT *pctxt, OSUTF8CHAR **pvalue)
This version of the rtXmlpDecAny function returns the string in a mutable buffer.
- int [rtXmlpDecAnyAttrStr](#) (OSCTXT *pctxt, const OSUTF8CHAR **ppAttrStr, 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 *pnocts, 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 *pctx, OSINT64 *pvalue)

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

- int [rtXmlpDecNamedBits](#) (OSCTXT *pctx, const OSBitMapItem *pBitMap, OSOCTET *pvalue, OSUINT32 *pnbits, OSUINT32 bufsize)

This function decodes the contents of a named bit field.

- int [rtXmlpDecStrList](#) (OSCTXT *pctx, OSRTDList *plist)

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

- int [rtXmlpDecTime](#) (OSCTXT *pctx, 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 *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 [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 nmNamespaces)
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.

8.1.1 Detailed Description

XML low-level C encode/decode functions.

Definition in file [osrxml.h](#).

8.1.2 Typedef Documentation

8.1.2.1 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.

8.1.3 Function Documentation

8.1.3.1 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.

8.1.3.2 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.

It ensures elements are received in the correct order and also sets the required element count variable.

Parameters

pState The pointer to state variable to be changed.

prevElemIdx Previous index of element. The search will be started from this element for better performance.

localName Local name of XML element

nsidx Namespace index

idtab Element ID table

fstab Finite state table

fstabRows Number of rows in fstab.

fstabCols Number of columns in fstab.

8.1.3.3 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.

8.1.3.4 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.

8.1.3.5 OSBOOL rtSaxIsEmptyBuffer (OSCTXT * ptxt)

This function checks if the buffer in the context is empty or not.

Parameters

ptxt Pointer to OSCTXT structure

Returns

TRUE, if the buffer contains empty string.

8.1.3.6 int rtSaxSortAttrs (OSCTXT * ptxt, 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

ptxt Pointer to OSCTXT structure.

attrs Standard SAX attribute list. Entry *i* is attribute name and *i*+1 is value. List is terminated by a null name.

order Order array containing the order of sorted attributes. This array is allocated using `rtxMemAlloc`, it can be freed using `rtxMemFreePtr` or will be freed when the context is freed. The list holds indices to name items in the attribute list that is passed in.

Returns

If success, positive value contains number of attributes in *attrs*; if failure, negative status code.

8.1.3.7 int rtSaxStrListMatch (OSCTXT * ptxt)

This function matches the list of strings.

It is used for matching NMTOKENS, IDREFS, NMENTITIES.

Parameters

ptxt Pointer to OSCTXT structure

Returns

0 - if success, negative value is error.

8.1.3.8 int rtSaxStrListParse (OSCTXT * *pctx*, OSRTMEMBUF * *pMemBuf*, OSRTDList * *pvalue*)

This function parses the list of strings.

It is used for parsing NMTOKENS, IDREFS, NMENTITIES.

Parameters

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

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

pvalue Doubly-linked list for parsed strings.

Returns

0 - if success, negative value is error.

8.1.3.9 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.

8.1.3.10 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.

8.1.3.11 int rtXmlCreateFileInputSource (OSCTXT * *pctx*, const char * *filepath*)

This function creates an XML document file input source.

The document can then be decoded by invoking an XML decode function.

Parameters

pctxt Pointer to context block structure.

filepath Full pathname of XML document file to open.

Returns

Completion status of operation:

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

8.1.3.12 int rtXmlInitContext (OSCTXT * *pctxt*)

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

Parameters

pctxt Pointer to OSCTXT structure

8.1.3.13 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.

8.1.3.14 int rtXmlInitCtxAppInfo (OSCTXT * *pctxt*)

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

Parameters

pctxt Pointer to OSCTXT structure

8.1.3.15 `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.

8.1.3.16 `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.

8.1.3.17 `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.

8.1.3.18 `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.

8.1.3.19 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.

8.1.3.20 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.

8.1.3.21 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.

8.1.3.22 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.

8.1.3.23 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.

8.1.3.24 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.

8.1.3.25 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.

8.1.3.26 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.

8.1.3.27 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.

8.1.3.28 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.

8.1.3.29 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.

8.1.3.30 `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.

8.1.3.31 `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.

Referenced by `OSXSDGlobalElement::setEncXSINamespace()`.

8.1.3.32 `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.

8.1.3.33 `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.

8.1.3.34 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.

8.1.3.35 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.

8.1.3.36 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.

8.1.3.37 **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.

Referenced by OSXSDGlobalElement::setNoNSSchemaLocation().

8.1.3.38 **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.

8.1.3.39 **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.

Referenced by OSXSDGlobalElement::setSchemaLocation().

8.1.3.40 **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.)

8.1.3.41 `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.

8.1.3.42 `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.

Referenced by OSXSDGlobalElement::setXSIType().

8.2 OSXMLDecodeBuffer.h File Reference

XML decode buffer or stream class definition.

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

Classes

- class [OSXMLDecodeBuffer](#)

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

8.2.1 Detailed Description

XML decode buffer or stream class definition.

Definition in file [OSXMLDecodeBuffer.h](#).

8.3 OSXMLEncodeBuffer.h File Reference

XML encode message buffer class definition.

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

Classes

- class [OSXMLEncodeBuffer](#)

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

8.3.1 Detailed Description

XML encode message buffer class definition.

Definition in file [OSXMLEncodeBuffer.h](#).

8.4 OSXMLEncodeStream.h File Reference

XML encode stream class definition.

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

Classes

- class [OSXMLEncodeStream](#)

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

8.4.1 Detailed Description

XML encode stream class definition.

Definition in file [OSXMLEncodeStream.h](#).

8.5 OSXMLMessageBuffer.h File Reference

XML encode/decode buffer and stream base class.

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

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

Classes

- class [OSXMLMessageBuffer](#)

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

8.5.1 Detailed Description

XML encode/decode buffer and stream base class.

Definition in file [OSXMLMessageBuffer.h](#).

8.6 rtSaxCppAny.h File Reference

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

8.6.1 Detailed Description

Definition in file [rtSaxCppAny.h](#).

8.7 rtSaxCppAnyType.h File Reference

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

8.7.1 Detailed Description

Definition in file [rtSaxCppAnyType.h](#).

8.8 rtSaxCppSimpleType.h File Reference

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

8.8.1 Detailed Description

Definition in file [rtSaxCppSimpleType.h](#).

8.9 rtSaxCppSoap.h File Reference

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

8.9.1 Detailed Description

Definition in file [rtSaxCppSoap.h](#).

8.10 rtSaxCppStrList.h File Reference

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

Classes

- class [OSXMLStrListHandler](#)
OSXMLStrListHandler.

8.10.1 Detailed Description

Definition in file [rtSaxCppStrList.h](#).

8.11 rtXmlCppEncFuncs.h File Reference

XML low-level C++ encode functions.

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

Functions

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

8.11.1 Detailed Description

XML low-level C++ encode functions. These are overloaded versions of C XML encode functions for use with C++. Definition in file [rtXmlCppEncFuncs.h](#).

8.11.2 Function Documentation

8.11.2.1 int [rtXmlCppEncAnyAttr](#) (OSCTXT * *pctxt*, OSRTObjListClass * *pAnyAttrList*)

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

This is expressed as list of name/value pairs.

Parameters

- pctxt* Pointer to context block structure.
- pAnyAttrList* List of name/value pair objects.

Returns

Completion status of operation:

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

8.11.2.2 `int rtXmlCppEncAnyTypeValue (OSCTXT * pctxt, OSXSDAnyTypeClass * pvalue)`

This function encodes a variable of the XSD anyType type.

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

Parameters

pctxt Pointer to context block structure.

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

Returns

Completion status of operation:

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

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

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

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

Parameters

pctxt Pointer to context block structure.

elemName XML element name.

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

pNSAttrs List of namespace attributes to be added to element.

terminate Add closing '>' character.

Returns

Completion status of operation:

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

8.11.2.4 `int rtXmlEncAny (OSCTXT * pctxt, OSXMLStringClass * pxmlstr, const OSUTF8CHAR * elemName, OSXMLNamespace * pNS)`

This function encodes a variable of the XSD any type.

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

Parameters

pctxt Pointer to context block structure.

pxmlstr 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.

8.11.2.5 `int rtXmlEncString (OSCTXT * pctxt, OSXMLStringClass * 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. 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.

8.12 rtXmlCppMsgBuf.h File Reference

This file is deprecated.

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

8.12.1 Detailed Description

This file is deprecated. Users should use one or more of the individual headers files defined in the include statements below.

Definition in file [rtXmlCppMsgBuf.h](#).

8.13 rtXmlCppNamespace.h File Reference

XML namespace handling structures and function definitions.

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

Classes

- class [OSXMLNamespaceClass](#)

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

8.13.1 Detailed Description

XML namespace handling structures and function definitions.

Definition in file [rtXmlCppNamespace.h](#).

8.14 rtXmlCppXSDElement.h File Reference

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

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

Classes

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

8.14.1 Detailed Description

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

Definition in file [rtXmlCppXSDElement.h](#).

8.15 rtXmlpCppDecFuncs.h File Reference

XML low-level C++ decode functions.

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

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

8.15.1 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

- ~OSXSDGlobalElement
 - OSXSDGlobalElement, [128](#)
- addXMLHeader
 - OSXMLEncodeBuffer, [109](#)
- addXMLText
 - OSXMLEncodeBuffer, [109](#)
- characters
 - OSXMLContentHandler, [98](#)
 - OSXMLDefaultHandler, [104](#)
- decodeFrom
 - OSXSDGlobalElement, [128](#)
- decodeXML
 - OSXMLDecodeBuffer, [102](#)
- encodeAttr
 - OSXMLEncodeStream, [114](#)
- encodeText
 - OSXMLEncodeStream, [114](#)
- encodeTo
 - OSXSDGlobalElement, [129](#)
- endElement
 - OSXMLContentHandler, [98](#)
 - OSXMLDefaultHandler, [105](#)
 - OSXMLEncodeStream, [114](#)
- getCtxtPtr
 - OSXSDGlobalElement, [129](#)
- getIndent
 - OSXMLMessageBuffer, [120](#)
- getIndentChar
 - OSXMLMessageBuffer, [120](#)
- getMsgLen
 - OSXMLEncodeBuffer, [109](#)
- getMsgPtr
 - OSXMLEncodeStream, [114](#)
- getState
 - OSXMLDefaultHandler, [105](#)
- getStream
 - OSXMLEncodeStream, [115](#)
- getWriteBOM
 - OSXMLMessageBuffer, [120](#)
- init
 - OSXMLDecodeBuffer, [102](#)
 - OSXMLEncodeBuffer, [110](#)
 - OSXMLEncodeStream, [115](#)
- isA
 - OSXMLDecodeBuffer, [102](#)
 - OSXMLEncodeBuffer, [110](#)
 - OSXMLEncodeStream, [115](#)
- isWellFormed
 - OSXMLDecodeBuffer, [102](#)
- mbOwnStream
 - OSXMLDecodeBuffer, [103](#)
 - OSXMLEncodeStream, [116](#)
- memAlloc
 - OSXSDGlobalElement, [129](#)
- memFreePtr
 - OSXSDGlobalElement, [129](#)
- mpContext
 - OSXSDGlobalElement, [131](#)
- mpCtxt
 - OSXMLEncodeStream, [116](#)
- mpStream
 - OSXMLEncodeStream, [116](#)
- osrtxml.h, [133](#)
 - OSXMLGroupDesc, [150](#)
 - rtSaxGetAttrValue, [150](#)
 - rtSaxGetElemID, [151](#)
 - rtSaxGetElemID8, [151](#)
 - rtSaxHasXMLNSAttrs, [151](#)
 - rtSaxIsEmptyBuffer, [152](#)
 - rtSaxSortAttrs, [152](#)
 - rtSaxStrListMatch, [152](#)
 - rtSaxStrListParse, [152](#)
 - rtXmlCmpBase64Str, [153](#)
 - rtXmlCmpHexStr, [153](#)
 - rtXmlCreateFileInputSource, [153](#)
 - rtXmlInitContext, [154](#)
 - rtXmlInitContextUsingKey, [154](#)
 - rtXmlInitCtxtAppInfo, [154](#)
 - rtXmlMatchBase64Str, [154](#)
 - rtXmlMatchDate, [155](#)
 - rtXmlMatchDateTime, [155](#)
 - rtXmlMatchGDay, [155](#)
 - rtXmlMatchGMonth, [156](#)

- rtXmlMatchGMonthDay, 156
- rtXmlMatchGYear, 156
- rtXmlMatchGYearMonth, 156
- rtXmlMatchHexStr, 157
- rtXmlMatchTime, 157
- rtXmlMemFreeAnyAttrs, 157
- rtXmlNewQName, 157
- rtXmlPrepareContext, 158
- rtXmlSetEncC14N, 158
- rtXmlSetEncDocHdr, 158
- rtXmlSetEncodingStr, 158
- rtXmlSetEncXSINamespace, 159
- rtXmlSetEncXSINilAttr, 159
- rtXmlSetFormatting, 159
- rtXmlSetIndent, 160
- rtXmlSetIndentChar, 160
- rtXmlSetNamespacesSet, 160
- rtXmlSetNoNSSchemaLocation, 160
- rtXmlSetNSPrefixLinks, 161
- rtXmlSetSchemaLocation, 161
- rtXmlSetSoapVersion, 161
- rtXmlSetWriteBOM, 161
- rtXmlSetXSITypeAttr, 162
- OSXMLContentHandler, 97
 - characters, 98
 - endElement, 98
 - startElement, 98
- OSXMLDecodeBuffer, 100
 - decodeXML, 102
 - init, 102
 - isA, 102
 - isWellFormed, 102
 - mbOwnStream, 103
 - OSXMLDecodeBuffer, 101
 - parseElementName, 102
 - parseElemQName, 103
 - setMaxErrors, 103
- OSXMLDecodeBuffer.h, 163
- OSXMLDefaultHandler, 104
 - characters, 104
 - endElement, 105
 - getState, 105
 - startElement, 105
- OSXMLDefaultHandlerIF, 107
- OSXMLEncodeBuffer, 108
 - addXMLHeader, 109
 - addXMLText, 109
 - getMsgLen, 109
 - init, 110
 - isA, 110
 - OSXMLEncodeBuffer, 109
 - setFragment, 110
 - write, 110
- OSXMLEncodeBuffer.h, 164
- OSXMLEncodeStream, 112
 - encodeAttr, 114
 - encodeText, 114
 - endElement, 114
 - getMsgPtr, 114
 - getStream, 115
 - init, 115
 - isA, 115
 - mbOwnStream, 116
 - mpCtxt, 116
 - mpStream, 116
 - OSXMLEncodeStream, 113
 - startDocument, 115
 - startElement, 115
 - termStartElement, 116
- OSXMLEncodeStream.h, 165
- OSXMLGroupDesc, 118
 - osrxml.h, 150
- OSXMLMessageBuffer, 119
 - getIndent, 120
 - getIndentChar, 120
 - getWriteBOM, 120
 - OSXMLMessageBuffer, 120
 - setAppInfo, 120
 - setFormatting, 121
 - setIndent, 121
 - setIndentChar, 121
 - setNamespace, 121
 - setWriteBOM, 122
- OSXMLMessageBuffer.h, 166
- OSXMLNamespaceClass, 123
 - OSXMLNamespaceClass, 123, 124
- OSXMLStrListHandler, 125
- OSXSDGlobalElement, 126
 - ~OSXSDGlobalElement, 128
 - decodeFrom, 128
 - encodeTo, 129
 - getCtxtPtr, 129
 - memAlloc, 129
 - memFreePtr, 129
 - mpContext, 131
 - OSXSDGlobalElement, 128
 - setDefaultNamespace, 129
 - setDiag, 130
 - setEncXSINamespace, 130
 - setMsgBuf, 130
 - setNamespace, 130
 - setNoNSSchemaLocation, 130
 - setSchemaLocation, 131
 - setXSIType, 131
 - validateFrom, 131
- parseElementName
 - OSXMLDecodeBuffer, 102

- parseElemQName
 - OSXMLDecodeBuffer, 103
- rtSaxCppAny.h, 167
- rtSaxCppAnyType.h, 168
- rtSaxCppSimpleType.h, 169
- rtSaxCppSoap.h, 170
- rtSaxCppStrList.h, 171
- rtSaxGetAttrValue
 - osrtxml.h, 150
- rtSaxGetElemID
 - osrtxml.h, 151
- rtSaxGetElemID8
 - osrtxml.h, 151
- rtSaxHasXMLNSAttrs
 - osrtxml.h, 151
- rtSaxIsEmptyBuffer
 - osrtxml.h, 152
- rtSaxSortAttrs
 - osrtxml.h, 152
- rtSaxStrListMatch
 - osrtxml.h, 152
- rtSaxStrListParse
 - osrtxml.h, 152
- rtXmlCmpBase64Str
 - osrtxml.h, 153
- rtXmlCmpHexStr
 - osrtxml.h, 153
- rtXmlCmpEncAnyAttr
 - rtXmlCmpEncFuncs.h, 172
- rtXmlCmpEncAnyTypeValue
 - rtXmlCmpEncFuncs.h, 172
- rtXmlCmpEncFuncs.h, 172
 - rtXmlCmpEncAnyAttr, 172
 - rtXmlCmpEncAnyTypeValue, 172
 - rtXmlCmpEncStartElement, 173
 - rtXmlEncAny, 173
 - rtXmlEncString, 174
- rtXmlCmpEncStartElement
 - rtXmlCmpEncFuncs.h, 173
- rtXmlCmpMsgBuf.h, 175
- rtXmlCmpNamespace.h, 176
- rtXmlCmpXSDElement.h, 177
- rtXmlCreateFileInputSource
 - osrtxml.h, 153
- rtXmlDec
 - rtXmlDecBase64Binary, 8
 - rtXmlDecBase64Str, 9
 - rtXmlDecBase64StrValue, 9
 - rtXmlDecBigInt, 9
 - rtXmlDecBool, 10
 - rtXmlDecDate, 10
 - rtXmlDecDateTime, 10
 - rtXmlDecDecimal, 11
 - rtXmlDecDouble, 11
 - rtXmlDecDynBase64Str, 11
 - rtXmlDecDynHexStr, 12
 - rtXmlDecDynUTF8Str, 12
 - rtXmlDecEmptyElement, 12
 - rtXmlDecGDay, 13
 - rtXmlDecGMonth, 13
 - rtXmlDecGMonthDay, 13
 - rtXmlDecGYear, 14
 - rtXmlDecGYearMonth, 14
 - rtXmlDecHexBinary, 14
 - rtXmlDecHexStr, 15
 - rtXmlDecInt, 15
 - rtXmlDecInt16, 16
 - rtXmlDecInt64, 16
 - rtXmlDecInt8, 16
 - rtXmlDecNSAttr, 17
 - rtXmlDecQName, 17
 - rtXmlDecTime, 18
 - rtXmlDecUInt, 18
 - rtXmlDecUInt16, 18
 - rtXmlDecUInt64, 19
 - rtXmlDecUInt8, 19
 - rtXmlDecUTF8Str, 19
 - rtXmlDecXmlStr, 20
 - rtXmlDecXSIAAttr, 20
 - rtXmlDecXSIAAttrs, 20
 - rtXmlParseElementName, 21
 - rtXmlParseElemQName, 21
- rtXmlDecBase64Binary
 - rtXmlDec, 8
- rtXmlDecBase64Str
 - rtXmlDec, 9
- rtXmlDecBase64StrValue
 - rtXmlDec, 9
- rtXmlDecBigInt
 - rtXmlDec, 9
- rtXmlDecBool
 - rtXmlDec, 10
- rtXmlDecDate
 - rtXmlDec, 10
- rtXmlDecDateTime
 - rtXmlDec, 10
- rtXmlDecDecimal
 - rtXmlDec, 11
- rtXmlDecDouble
 - rtXmlDec, 11
- rtXmlDecDynBase64Str
 - rtXmlDec, 11
- rtXmlDecDynHexStr
 - rtXmlDec, 12
- rtXmlDecDynUTF8Str
 - rtXmlDec, 12
- rtXmlDecEmptyElement

- rtXmlDec, [12](#)
- rtXmlDecGDay
 - rtXmlDec, [13](#)
- rtXmlDecGMonth
 - rtXmlDec, [13](#)
- rtXmlDecGMonthDay
 - rtXmlDec, [13](#)
- rtXmlDecGYear
 - rtXmlDec, [14](#)
- rtXmlDecGYearMonth
 - rtXmlDec, [14](#)
- rtXmlDecHexBinary
 - rtXmlDec, [14](#)
- rtXmlDecHexStr
 - rtXmlDec, [15](#)
- rtXmlDecInt
 - rtXmlDec, [15](#)
- rtXmlDecInt16
 - rtXmlDec, [16](#)
- rtXmlDecInt64
 - rtXmlDec, [16](#)
- rtXmlDecInt8
 - rtXmlDec, [16](#)
- rtXmlDecNSAttr
 - rtXmlDec, [17](#)
- rtXmlDecQName
 - rtXmlDec, [17](#)
- rtXmlDecTime
 - rtXmlDec, [18](#)
- rtXmlDecUInt
 - rtXmlDec, [18](#)
- rtXmlDecUInt16
 - rtXmlDec, [18](#)
- rtXmlDecUInt64
 - rtXmlDec, [19](#)
- rtXmlDecUInt8
 - rtXmlDec, [19](#)
- rtXmlDecUTF8Str
 - rtXmlDec, [19](#)
- rtXmlDecXmlStr
 - rtXmlDec, [20](#)
- rtXmlDecXSIAttr
 - rtXmlDec, [20](#)
- rtXmlDecXSIAttrs
 - rtXmlDec, [20](#)
- rtXmlEnc
 - rtXmlEncAny, [29](#)
 - rtXmlEncAnyAttr, [29](#)
 - rtXmlEncAnyTypeValue, [29](#)
 - rtXmlEncBase64Binary, [30](#)
 - rtXmlEncBase64BinaryAttr, [30](#)
 - rtXmlEncBase64StrValue, [30](#)
 - rtXmlEncBigInt, [31](#)
 - rtXmlEncBigIntAttr, [31](#)
 - rtXmlEncBigIntValue, [32](#)
 - rtXmlEncBinStrValue, [32](#)
 - rtXmlEncBitString, [32](#)
 - rtXmlEncBOM, [33](#)
 - rtXmlEncBool, [33](#)
 - rtXmlEncBoolAttr, [34](#)
 - rtXmlEncBoolValue, [34](#)
 - rtXmlEncComment, [34](#)
 - rtXmlEncDate, [35](#)
 - rtXmlEncDateTime, [35](#)
 - rtXmlEncDateTimeValue, [36](#)
 - rtXmlEncDateValue, [36](#)
 - rtXmlEncDecimal, [36](#)
 - rtXmlEncDecimalAttr, [37](#)
 - rtXmlEncDecimalValue, [37](#)
 - rtXmlEncDouble, [37](#)
 - rtXmlEncDoubleAttr, [38](#)
 - rtXmlEncDoubleNormalValue, [38](#)
 - rtXmlEncDoubleValue, [39](#)
 - rtXmlEncEmptyElement, [39](#)
 - rtXmlEncEndDocument, [39](#)
 - rtXmlEncEndElement, [40](#)
 - rtXmlEncEndSoapElems, [40](#)
 - rtXmlEncEndSoapEnv, [40](#)
 - rtXmlEncFloat, [41](#)
 - rtXmlEncFloatAttr, [41](#)
 - rtXmlEncGDay, [41](#)
 - rtXmlEncGDayValue, [42](#)
 - rtXmlEncGMonth, [42](#)
 - rtXmlEncGMonthDay, [42](#)
 - rtXmlEncGMonthDayValue, [43](#)
 - rtXmlEncGMonthValue, [43](#)
 - rtXmlEncGYear, [43](#)
 - rtXmlEncGYearMonth, [44](#)
 - rtXmlEncGYearMonthValue, [44](#)
 - rtXmlEncGYearValue, [44](#)
 - rtXmlEncHexBinary, [45](#)
 - rtXmlEncHexBinaryAttr, [45](#)
 - rtXmlEncHexStrValue, [46](#)
 - rtXmlEncIndent, [46](#)
 - rtXmlEncInt, [46](#)
 - rtXmlEncInt64, [47](#)
 - rtXmlEncInt64Attr, [47](#)
 - rtXmlEncInt64Value, [47](#)
 - rtXmlEncIntAttr, [48](#)
 - rtXmlEncIntPattern, [48](#)
 - rtXmlEncIntValue, [49](#)
 - rtXmlEncNamedBits, [49](#)
 - rtXmlEncNSAttrs, [49](#)
 - rtXmlEncReal10, [50](#)
 - rtXmlEncSoapArrayTypeAttr, [50](#)
 - rtXmlEncStartDocument, [51](#)
 - rtXmlEncStartElement, [51](#)
 - rtXmlEncStartSoapElems, [51](#)

- rtXmlEncStartSoapEnv, 52
- rtXmlEncString, 52
- rtXmlEncStringValue, 52
- rtXmlEncStringValue2, 53
- rtXmlEncTermStartElement, 53
- rtXmlEncTime, 53
- rtXmlEncTimeValue, 54
- rtXmlEncUInt, 54
- rtXmlEncUInt64, 55
- rtXmlEncUInt64Attr, 55
- rtXmlEncUInt64Value, 55
- rtXmlEncUIntAttr, 56
- rtXmlEncUIntValue, 56
- rtXmlEncUnicodeStr, 56
- rtXmlEncUTF8Attr, 57
- rtXmlEncUTF8Attr2, 57
- rtXmlEncUTF8Str, 58
- rtXmlEncXSIAattrs, 58
- rtXmlEncXSINilAttr, 58
- rtXmlEncXSITypeAttr, 59
- rtXmlEncXSITypeAttr2, 59
- rtXmlFreeInputSource, 59
- rtXmlGetEncBufLen, 28
- rtXmlGetEncBufPtr, 28
- rtXmlGetIndent, 60
- rtXmlGetIndentChar, 60
- rtXmlGetWriteBOM, 60
- rtXmlPrintNSAttrs, 60
- rtXmlSetEncBufPtr, 61
- rtXmlEncAny
 - rtXmlCppEncFuncs.h, 173
 - rtXmlEnc, 29
- rtXmlEncAnyAttr
 - rtXmlEnc, 29
- rtXmlEncAnyTypeValue
 - rtXmlEnc, 29
- rtXmlEncAttrC14N
 - rtXmlpDec, 68
- rtXmlEncBase64Binary
 - rtXmlEnc, 30
- rtXmlEncBase64BinaryAttr
 - rtXmlEnc, 30
- rtXmlEncBase64StrValue
 - rtXmlEnc, 30
- rtXmlEncBigInt
 - rtXmlEnc, 31
- rtXmlEncBigIntAttr
 - rtXmlEnc, 31
- rtXmlEncBigIntValue
 - rtXmlEnc, 32
- rtXmlEncBinStrValue
 - rtXmlEnc, 32
- rtXmlEncBitString
 - rtXmlEnc, 32
- rtXmlEncBOM
 - rtXmlEnc, 33
- rtXmlEncBool
 - rtXmlEnc, 33
- rtXmlEncBoolAttr
 - rtXmlEnc, 34
- rtXmlEncBoolValue
 - rtXmlEnc, 34
- rtXmlEncComment
 - rtXmlEnc, 34
- rtXmlEncDate
 - rtXmlEnc, 35
- rtXmlEncDateTime
 - rtXmlEnc, 35
- rtXmlEncDateTimeValue
 - rtXmlEnc, 36
- rtXmlEncDateValue
 - rtXmlEnc, 36
- rtXmlEncDecimal
 - rtXmlEnc, 36
- rtXmlEncDecimalAttr
 - rtXmlEnc, 37
- rtXmlEncDecimalValue
 - rtXmlEnc, 37
- rtXmlEncDouble
 - rtXmlEnc, 37
- rtXmlEncDoubleAttr
 - rtXmlEnc, 38
- rtXmlEncDoubleNormalValue
 - rtXmlEnc, 38
- rtXmlEncDoubleValue
 - rtXmlEnc, 39
- rtXmlEncEmptyElement
 - rtXmlEnc, 39
- rtXmlEncEndDocument
 - rtXmlEnc, 39
- rtXmlEncEndElement
 - rtXmlEnc, 40
- rtXmlEncEndSoapElems
 - rtXmlEnc, 40
- rtXmlEncEndSoapEnv
 - rtXmlEnc, 40
- rtXmlEncFloat
 - rtXmlEnc, 41
- rtXmlEncFloatAttr
 - rtXmlEnc, 41
- rtXmlEncGDay
 - rtXmlEnc, 41
- rtXmlEncGDayValue
 - rtXmlEnc, 42
- rtXmlEncGMonth
 - rtXmlEnc, 42
- rtXmlEncGMonthDay
 - rtXmlEnc, 42

rtXmlEncGMonthDayValue	rtXmlEnc, 43
rtXmlEncGMonthValue	rtXmlEnc, 43
rtXmlEncGYear	rtXmlEnc, 43
rtXmlEncGYearMonth	rtXmlEnc, 44
rtXmlEncGYearMonthValue	rtXmlEnc, 44
rtXmlEncGYearValue	rtXmlEnc, 44
rtXmlEncHexBinary	rtXmlEnc, 45
rtXmlEncHexBinaryAttr	rtXmlEnc, 45
rtXmlEncHexStrValue	rtXmlEnc, 46
rtXmlEncIndent	rtXmlEnc, 46
rtXmlEncInt	rtXmlEnc, 46
rtXmlEncInt64	rtXmlEnc, 47
rtXmlEncInt64Attr	rtXmlEnc, 47
rtXmlEncInt64Value	rtXmlEnc, 47
rtXmlEncIntAttr	rtXmlEnc, 48
rtXmlEncIntPattern	rtXmlEnc, 48
rtXmlEncIntValue	rtXmlEnc, 49
rtXmlEncNamedBits	rtXmlEnc, 49
rtXmlEncNSAttrs	rtXmlEnc, 49
rtXmlEncReal10	rtXmlEnc, 50
rtXmlEncSoapArrayTypeAttr	rtXmlEnc, 50
rtXmlEncStartDocument	rtXmlEnc, 51
rtXmlEncStartElement	rtXmlEnc, 51
rtXmlEncStartSoapElems	rtXmlEnc, 51
rtXmlEncStartSoapEnv	rtXmlEnc, 52
rtXmlEncString	rtXmlCcppEncFuncs.h, 174 rtXmlEnc, 52
rtXmlEncStringValue	rtXmlEnc, 52
rtXmlEncStringValue2	rtXmlEnc, 53
rtXmlEncTermStartElement	rtXmlEnc, 53
rtXmlEncTime	rtXmlEnc, 53
rtXmlEncTimeValue	rtXmlEnc, 54
rtXmlEncUInt	rtXmlEnc, 54
rtXmlEncUInt64	rtXmlEnc, 55
rtXmlEncUInt64Attr	rtXmlEnc, 55
rtXmlEncUInt64Value	rtXmlEnc, 55
rtXmlEncUIntAttr	rtXmlEnc, 56
rtXmlEncUIntValue	rtXmlEnc, 56
rtXmlEncUnicodeStr	rtXmlEnc, 56
rtXmlEncUTF8Attr	rtXmlEnc, 57
rtXmlEncUTF8Attr2	rtXmlEnc, 57
rtXmlEncUTF8Str	rtXmlEnc, 58
rtXmlEncXSIAttrs	rtXmlEnc, 58
rtXmlEncXSINilAttr	rtXmlEnc, 58
rtXmlEncXSITypeAttr	rtXmlEnc, 59
rtXmlEncXSITypeAttr2	rtXmlEnc, 59
rtXmlFreeInputSource	rtXmlEnc, 59
rtXmlGetEncBufLen	rtXmlEnc, 28
rtXmlGetEncBufPtr	rtXmlEnc, 28
rtXmlGetIndent	rtXmlEnc, 60
rtXmlGetIndentChar	rtXmlEnc, 60
rtXmlGetWriteBOM	rtXmlEnc, 60
rtXmlInitContext	osrtxml.h, 154
rtXmlInitContextUsingKey	osrtxml.h, 154
rtXmlInitCtxtAppInfo	

- osrtxml.h, 154
- rtXmlMatchBase64Str
 - osrtxml.h, 154
- rtXmlMatchDate
 - osrtxml.h, 155
- rtXmlMatchDateTime
 - osrtxml.h, 155
- rtXmlMatchGDay
 - osrtxml.h, 155
- rtXmlMatchGMonth
 - osrtxml.h, 156
- rtXmlMatchGMonthDay
 - osrtxml.h, 156
- rtXmlMatchGYear
 - osrtxml.h, 156
- rtXmlMatchGYearMonth
 - osrtxml.h, 156
- rtXmlMatchHexStr
 - osrtxml.h, 157
- rtXmlMatchTime
 - osrtxml.h, 157
- rtXmlMemFreeAnyAttrs
 - osrtxml.h, 157
- rtXmlNewQName
 - osrtxml.h, 157
- rtXmlParseElementName
 - rtXmlDec, 21
- rtXmlParseElemQName
 - rtXmlDec, 21
- rtXmlpCountListItems
 - rtXmlpDec, 68
- rtXmlpCppDecFuncs.h, 178
- rtXmlpCreateReader
 - rtXmlpDec, 68
- rtXmlpDec
 - rtXmlEncAttrC14N, 68
 - rtXmlpCountListItems, 68
 - rtXmlpCreateReader, 68
 - rtXmlpDecAny, 69
 - rtXmlpDecAny2, 69
 - rtXmlpDecAnyAttrStr, 69
 - rtXmlpDecAnyElem, 70
 - rtXmlpDecBase64Str, 70
 - rtXmlpDecBigInt, 70
 - rtXmlpDecBitString, 71
 - rtXmlpDecBool, 71
 - rtXmlpDecDate, 72
 - rtXmlpDecDateTime, 72
 - rtXmlpDecDecimal, 72
 - rtXmlpDecDouble, 73
 - rtXmlpDecDoubleExt, 73
 - rtXmlpDecDynBase64Str, 73
 - rtXmlpDecDynBitString, 74
 - rtXmlpDecDynHexStr, 74
 - rtXmlpDecDynUnicodeStr, 75
 - rtXmlpDecDynUTF8Str, 75
 - rtXmlpDecGDay, 75
 - rtXmlpDecGMonth, 76
 - rtXmlpDecGMonthDay, 76
 - rtXmlpDecGYear, 76
 - rtXmlpDecGYearMonth, 77
 - rtXmlpDecHexStr, 77
 - rtXmlpDecInt, 78
 - rtXmlpDecInt16, 78
 - rtXmlpDecInt64, 78
 - rtXmlpDecInt8, 79
 - rtXmlpDecNamedBits, 79
 - rtXmlpDecStrList, 79
 - rtXmlpDecTime, 80
 - rtXmlpDecUInt, 80
 - rtXmlpDecUInt16, 80
 - rtXmlpDecUInt64, 81
 - rtXmlpDecUInt8, 81
 - rtXmlpDecUTF8Str, 81
 - rtXmlpDecXmlStr, 82
 - rtXmlpDecXmlStrList, 82
 - rtXmlpDecXSIAAttr, 83
 - rtXmlpDecXSIAAttrs, 83
 - rtXmlpDecXSITypeAttr, 83
 - rtXmlpForceDecodeAsGroup, 84
 - rtXmlpGetAttributeCount, 84
 - rtXmlpGetAttributeID, 84
 - rtXmlpGetCurrentLevel, 85
 - rtXmlpGetNextAllElemID, 85
 - rtXmlpGetNextAllElemID16, 85
 - rtXmlpGetNextAllElemID32, 86
 - rtXmlpGetNextElem, 86
 - rtXmlpGetNextElemID, 87
 - rtXmlpGetNextSeqElemID, 87
 - rtXmlpGetNextSeqElemID2, 88
 - rtXmlpGetNextSeqElemIDExt, 88
 - rtXmlpGetReader, 89
 - rtXmlpGetXmInsAttrs, 89
 - rtXmlpGetXSITypeAttr, 89
 - rtXmlpGetXSITypeIndex, 90
 - rtXmlpHasAttributes, 90
 - rtXmlpHideAttributes, 90
 - rtXmlpIsDecodeAsGroup, 91
 - rtXmlpIsEmptyElement, 91
 - rtXmlpIsLastEventDone, 91
 - rtXmlpIsUTF8Encoding, 91
 - rtXmlpListHasItem, 92
 - rtXmlpLookupXSITypeIndex, 92
 - rtXmlpMarkLastEventActive, 92
 - rtXmlpMarkPos, 93
 - rtXmlpMatchEndTag, 93
 - rtXmlpMatchStartTag, 93
 - rtXmlpNeedDecodeAttributes, 94

- rtXmlpReadBytes, 94
- rtXmlpResetMarkedPos, 94
- rtXmlpRewindToMarkedPos, 94
- rtXmlpSelectAttribute, 95
- rtXmlpSetListMode, 95
- rtXmlpSetMixedContentMode, 95
- rtXmlpSetNamespaceTable, 95
- rtXmlpSetWhiteSpaceMode, 96
- rtXmlpDecAny
 - rtXmlpDec, 69
- rtXmlpDecAny2
 - rtXmlpDec, 69
- rtXmlpDecAnyAttrStr
 - rtXmlpDec, 69
- rtXmlpDecAnyElem
 - rtXmlpDec, 70
- rtXmlpDecBase64Str
 - rtXmlpDec, 70
- rtXmlpDecBigInt
 - rtXmlpDec, 70
- rtXmlpDecBitString
 - rtXmlpDec, 71
- rtXmlpDecBool
 - rtXmlpDec, 71
- rtXmlpDecDate
 - rtXmlpDec, 72
- rtXmlpDecDateTime
 - rtXmlpDec, 72
- rtXmlpDecDecimal
 - rtXmlpDec, 72
- rtXmlpDecDouble
 - rtXmlpDec, 73
- rtXmlpDecDoubleExt
 - rtXmlpDec, 73
- rtXmlpDecDynBase64Str
 - rtXmlpDec, 73
- rtXmlpDecDynBitString
 - rtXmlpDec, 74
- rtXmlpDecDynHexStr
 - rtXmlpDec, 74
- rtXmlpDecDynUnicodeStr
 - rtXmlpDec, 75
- rtXmlpDecDynUTF8Str
 - rtXmlpDec, 75
- rtXmlpDecGDay
 - rtXmlpDec, 75
- rtXmlpDecGMonth
 - rtXmlpDec, 76
- rtXmlpDecGMonthDay
 - rtXmlpDec, 76
- rtXmlpDecGYear
 - rtXmlpDec, 76
- rtXmlpDecGYearMonth
 - rtXmlpDec, 77
- rtXmlpDecHexStr
 - rtXmlpDec, 77
- rtXmlpDecInt
 - rtXmlpDec, 78
- rtXmlpDecInt16
 - rtXmlpDec, 78
- rtXmlpDecInt64
 - rtXmlpDec, 78
- rtXmlpDecInt8
 - rtXmlpDec, 79
- rtXmlpDecNamedBits
 - rtXmlpDec, 79
- rtXmlpDecStrList
 - rtXmlpDec, 79
- rtXmlpDecTime
 - rtXmlpDec, 80
- rtXmlpDecUInt
 - rtXmlpDec, 80
- rtXmlpDecUInt16
 - rtXmlpDec, 80
- rtXmlpDecUInt64
 - rtXmlpDec, 81
- rtXmlpDecUInt8
 - rtXmlpDec, 81
- rtXmlpDecUTF8Str
 - rtXmlpDec, 81
- rtXmlpDecXmlStr
 - rtXmlpDec, 82
- rtXmlpDecXmlStrList
 - rtXmlpDec, 82
- rtXmlpDecXSIAAttr
 - rtXmlpDec, 83
- rtXmlpDecXSIAAttrs
 - rtXmlpDec, 83
- rtXmlpDecXSITypeAttr
 - rtXmlpDec, 83
- rtXmlpForceDecodeAsGroup
 - rtXmlpDec, 84
- rtXmlpGetAttributeCount
 - rtXmlpDec, 84
- rtXmlpGetAttributeID
 - rtXmlpDec, 84
- rtXmlpGetCurrentLevel
 - rtXmlpDec, 85
- rtXmlpGetNextAllElemID
 - rtXmlpDec, 85
- rtXmlpGetNextAllElemID16
 - rtXmlpDec, 85
- rtXmlpGetNextAllElemID32
 - rtXmlpDec, 86
- rtXmlpGetNextElem
 - rtXmlpDec, 86
- rtXmlpGetNextElemID
 - rtXmlpDec, 87

- rtXmlpGetNextSeqElemID
rtXmlpDec, [87](#)
- rtXmlpGetNextSeqElemID2
rtXmlpDec, [88](#)
- rtXmlpGetNextSeqElemIDExt
rtXmlpDec, [88](#)
- rtXmlpGetReader
rtXmlpDec, [89](#)
- rtXmlpGetXmInsAttrs
rtXmlpDec, [89](#)
- rtXmlpGetXSITypeAttr
rtXmlpDec, [89](#)
- rtXmlpGetXSITypeIndex
rtXmlpDec, [90](#)
- rtXmlpHasAttributes
rtXmlpDec, [90](#)
- rtXmlpHideAttributes
rtXmlpDec, [90](#)
- rtXmlpIsDecodeAsGroup
rtXmlpDec, [91](#)
- rtXmlpIsEmptyElement
rtXmlpDec, [91](#)
- rtXmlpIsLastEventDone
rtXmlpDec, [91](#)
- rtXmlpIsUTF8Encoding
rtXmlpDec, [91](#)
- rtXmlpListHasItem
rtXmlpDec, [92](#)
- rtXmlpLookupXSITypeIndex
rtXmlpDec, [92](#)
- rtXmlpMarkLastEventActive
rtXmlpDec, [92](#)
- rtXmlpMarkPos
rtXmlpDec, [93](#)
- rtXmlpMatchEndTag
rtXmlpDec, [93](#)
- rtXmlpMatchStartTag
rtXmlpDec, [93](#)
- rtXmlpNeedDecodeAttributes
rtXmlpDec, [94](#)
- rtXmlpReadBytes
rtXmlpDec, [94](#)
- rtXmlPrepareContext
osrxml.h, [158](#)
- rtXmlpResetMarkedPos
rtXmlpDec, [94](#)
- rtXmlpRewindToMarkedPos
rtXmlpDec, [94](#)
- rtXmlPrintNSAttrs
rtXmlEnc, [60](#)
- rtXmlpSelectAttribute
rtXmlpDec, [95](#)
- rtXmlpSetListMode
rtXmlpDec, [95](#)
- rtXmlpSetMixedContentMode
rtXmlpDec, [95](#)
- rtXmlpSetNamespaceTable
rtXmlpDec, [95](#)
- rtXmlpSetWhiteSpaceMode
rtXmlpDec, [96](#)
- rtXmlSetEncBufPtr
rtXmlEnc, [61](#)
- rtXmlSetEncC14N
osrxml.h, [158](#)
- rtXmlSetEncDocHdr
osrxml.h, [158](#)
- rtXmlSetEncodingStr
osrxml.h, [158](#)
- rtXmlSetEncXSINamespace
osrxml.h, [159](#)
- rtXmlSetEncXSINilAttr
osrxml.h, [159](#)
- rtXmlSetFormatting
osrxml.h, [159](#)
- rtXmlSetIndent
osrxml.h, [160](#)
- rtXmlSetIndentChar
osrxml.h, [160](#)
- rtXmlSetNamespacesSet
osrxml.h, [160](#)
- rtXmlSetNoNSSchemaLocation
osrxml.h, [160](#)
- rtXmlSetNSPrefixLinks
osrxml.h, [161](#)
- rtXmlSetSchemaLocation
osrxml.h, [161](#)
- rtXmlSetSoapVersion
osrxml.h, [161](#)
- rtXmlSetWriteBOM
osrxml.h, [161](#)
- rtXmlSetXSITypeAttr
osrxml.h, [162](#)
- rtXmlUtil
rtXmlWriteToFile, [62](#)
- rtXmlWriteToFile
rtXmlUtil, [62](#)
- setAppInfo
OSXMLMessageBuffer, [120](#)
- setDefaultNamespace
OSXSDGlobalElement, [129](#)
- setDiag
OSXSDGlobalElement, [130](#)
- setEncXSINamespace
OSXSDGlobalElement, [130](#)
- setFormatting
OSXMLMessageBuffer, [121](#)
- setFragment

- OSXMLEncodeBuffer, [110](#)
- setIndent
 - OSXMLMessageBuffer, [121](#)
- setIndentChar
 - OSXMLMessageBuffer, [121](#)
- setMaxErrors
 - OSXMLDecodeBuffer, [103](#)
- setMsgBuf
 - OSXSDGlobalElement, [130](#)
- setNamespace
 - OSXMLMessageBuffer, [121](#)
 - OSXSDGlobalElement, [130](#)
- setNoNSSchemaLocation
 - OSXSDGlobalElement, [130](#)
- setSchemaLocation
 - OSXSDGlobalElement, [131](#)
- setWriteBOM
 - OSXMLMessageBuffer, [122](#)
- setXSIType
 - OSXSDGlobalElement, [131](#)
- startDocument
 - OSXMLEncodeStream, [115](#)
- startElement
 - OSXMLContentHandler, [98](#)
 - OSXMLDefaultHandler, [105](#)
 - OSXMLEncodeStream, [115](#)
- termStartElement
 - OSXMLEncodeStream, [116](#)
- validateFrom
 - OSXSDGlobalElement, [131](#)
- write
 - OSXMLEncodeBuffer, [110](#)
- XML decode functions., [6](#)
- XML encode functions., [22](#)
- XML pull-parser decode functions., [63](#)
- XML utility functions., [62](#)