XBinder

XML Schema Compiler
Version 2.9
C XML Runtime
Reference Manual

Objective Systems, Inc. — November 2023

Author's Contact Information

Comments, suggestions, and inquiries regarding XBinder may be submitted via electronic mail to info@obj-sys.com.

Contents

Chapter 1

C XML Runtime Library Functions

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

The categories of functions provided are as follows:

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

Chapter 2

C DOM Runtime Library Functions

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

The categories of functions provided are as follows:

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

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

Module Documentation

6.1 XML decode functions.

Functions

- int rtXmlDecBase64Binary (OSRTMEMBUF *pMemBuf, const OSUTF8CHAR *inpdata, OSSIZE length)

  This function decodes the contents of a Base64-encoded binary data type into a memory buffer.
- int rtXmlDecBase64Str (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSINT32 bufsize)

  This function decodes a contents of a Base64-encode binary string into a static memory structure.
- int rtXmlDecBase64Str64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnocts, OSSIZE bufsize)

  This function is identical to rtXmlDecBase64Str except that is supports a 64-bit integer length on 64-bit systems.
- int rtXmlDecBase64StrValue (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSSIZE bufSize, OS↩ SIZE srcDataLen)

  This function decodes a contents of a Base64-encode binary string into the specified octet array.
- int rtXmlDecBase64StrValue64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnocts, OSSIZE bufSize, OS↩ SIZE srcDataLen)

  This function decodes is identical to rtXmlDecBase64StrValue except that it supports a 64-bit integer length on 64-bit systems.
- int rtXmlDecBigInt (OSCTXT *pctxt, const OSUTF8CHAR **ppvalue)

  This function will decode a variable of the XSD integer type.
- int rtXmlDecBool (OSCTXT *pctxt, OSBOOL *pvalue)

  This function decodes a variable of the boolean type.
- int rtXmlDecDate (OSCTXT *pctxt, OSXSDDateTime *pvalue)

  This function decodes a variable of the XSD 'date' type.
- int rtXmlDecTime (OSCTXT *pctxt, OSXSDDateTime *pvalue)

  This function decodes a variable of the XSD 'time' type.
- int rtXmlDecDateTime (OSCTXT *pctxt, OSXSDDateTime *pvalue)

  This function decodes a variable of the XSD 'dateTime' type.
- int rtXmlDecDecimal (OSCTXT *pctxt, OSREAL *pvalue)

  This function decodes the contents of a decimal data type.
- int rtXmlDecDouble (OSCTXT *pctxt, OSREAL *pvalue)

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

- int rtXmlDecDynBase64Str (OSCTXT *pctxt, OSDynOctStr *pvalue)

  This function decodes a contents of a Base64-encode binary string.
- int rtXmlDecDynBase64Str64 (OSCTXT *pctxt, OSDynOctStr64 *pvalue)

  This function is identical to rtXmlDecDynBase64Str except that it supports a 64-bit integer length on 64-bit systems.
- int rtXmlDecDynHexStr (OSCTXT *pctxt, OSDynOctStr *pvalue)

  This function decodes a contents of a hexBinary string.
- int rtXmlDecDynHexStr64 (OSCTXT *pctxt, OSDynOctStr64 *pvalue)

  This function is identical to rtXmlDecDynHexStr except that it supports a 64-bit integer length on 64-bit systems.
- int rtXmlDecEmptyElement (OSCTXT *pctxt)

  This function is used to enforce a requirement that an element be empty.
- int rtXmlDecUTF8Str (OSCTXT *pctxt, OSUTF8CHAR *outdata, OSSIZE max_len)

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

  This function decodes the contents of a UTF-8 string data type.
- int rtXmlDecHexBinary (OSRTMEMBUF *pMemBuf, const OSUTF8CHAR *inpdata, OSSIZE length)

  This function decodes the contents of a hex-encoded binary data type into a memory buffer.
- int rtXmlDecHexStr (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSINT32 bufsize)

  This function decodes the contents of a hexBinary string into a static memory structure.
- int rtXmlDecHexStr64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnocts, OSSIZE bufsize)

  This function is identical to rtXmlDecHexStr except that it supports a 64-bit integer length on 64-bit systems.
- int rtXmlDecGYear (OSCTXT *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 namespac attribute (xmlns).
- const OSUTF8CHAR * rtXmlDecQName (OSCTXT *pctxt, const OSUTF8CHAR *qname, const OSUTF8CHAR **prefix)

  This function decodes an XML qualified name string (QName) type.
- int rtXmlDecXSIAttr (OSCTXT *pctxt, const OSUTF8CHAR *attrName, const OSUTF8CHAR *attrValue)

  This function decodes XML schema instance (XSI) attribute.
- int rtXmlDecXSIAttrs (OSCTXT *pctxt, const OSUTF8CHAR *const *attrs, const char *typeName)

  This function decodes XML schema instance (XSI) attributes.
- int rtXmlDecXmlStr (OSCTXT *pctxt, OSXMLSTRING *outdata)

  This function decodes the contents of an XML string data type.
- int rtXmlParseElementName (OSCTXT *pctxt, OSUTF8CHAR **ppName)

  This function parses the initial tag from an XML message.
- int rtXmlParseElemQName (OSCTXT *pctxt, OSXMLQName *pQName)

  This function parses the initial tag from an XML message.

6.1.1 Detailed Description

6.1.2 Function Documentation

6.1.2.1 rtXmlDecBase64Binary()

```
int rtXmlDecBase64Binary (
             OSRTMEMBUF * pMemBuf,
             const OSUTF8CHAR * inpdata,
             OSSIZE length )
```

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

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

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

Parameters

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.

6.1.2.2 rtXmlDecBase64Str()

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

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

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

Parameters

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.

6.1.2.3 rtXmlDecBase64Str64()

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

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

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.

6.1.2.4 rtXmlDecBase64StrValue()

```
int rtXmlDecBase64StrValue (
             OSCTXT * pctxt,
             OSOCTET * pvalue,
             OSUINT32 * pnocts,
             OSSIZE bufSize,
             OSSIZE srcDataLen )
```

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

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

Parameters

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.

6.1.2.5 rtXmlDecBase64StrValue64()

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

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

Parameters

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.

6.1.2.6 rtXmlDecBigInt()

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

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

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

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

Parameters

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.

6.1.2.7 rtXmlDecBool()

```
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.

6.1.2.8 rtXmlDecDate()

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

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

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

Parameters

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.

6.1.2.9 rtXmlDecDateTime()

```
int rtXmlDecDateTime (
             OSCTXT * pctxt,
             OSXSDDateTime * pvalue )
```

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

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

Parameters

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.

6.1.2.10 rtXmlDecDecimal()

```
int rtXmlDecDecimal (
             OSCTXT * pctxt,
             OSREAL * pvalue )
```

This function decodes the contents of a decimal data type.

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

Parameters

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.

6.1.2.11 rtXmlDecDouble()

```
int rtXmlDecDouble (
             OSCTXT * pctxt,
             OSREAL * pvalue )
```

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

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

Parameters

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.

6.1.2.12 rtXmlDecDynBase64Str()

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

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

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

Parameters

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.

6.1.2.13 rtXmlDecDynBase64Str64()

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

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

Parameters

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.

6.1.2.14 rtXmlDecDynHexStr()

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

This function decodes a contents of a hexBinary string.

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

Parameters

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.

6.1.2.15 rtXmlDecDynHexStr64()

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

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

Parameters

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.

6.1.2.16 rtXmlDecDynUTF8Str()

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

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

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

Parameters

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.

6.1.2.17 rtXmlDecEmptyElement()

```
int rtXmlDecEmptyElement (
             OSCTXT * pctxt )
```

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

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

Parameters

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.

6.1.2.18 rtXmlDecGDay()

```
int rtXmlDecGDay (
             OSCTXT * pctxt,
             OSXSDDateTime * pvalue )
```

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

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

Parameters

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.

6.1.2.19 rtXmlDecGMonth()

```
int rtXmlDecGMonth (
             OSCTXT * pctxt,
             OSXSDDateTime * pvalue )
```

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

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

Parameters

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.

6.1.2.20 rtXmlDecGMonthDay()

```
int rtXmlDecGMonthDay (
             OSCTXT * pctxt,
             OSXSDDateTime * pvalue )
```

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

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

Parameters

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.

6.1.2.21 rtXmlDecGYear()

```
int rtXmlDecGYear (
             OSCTXT * pctxt,
             OSXSDDateTime * pvalue )
```

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

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

Parameters

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.

6.1.2.22 rtXmlDecGYearMonth()

```
int rtXmlDecGYearMonth (
             OSCTXT * pctxt,
             OSXSDDateTime * pvalue )
```

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

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

Parameters

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.

6.1.2.23 rtXmlDecHexBinary()

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

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

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

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

Parameters

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.

6.1.2.24 rtXmlDecHexStr()

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

```
            OSOCTET * pvalue,
            OSUINT32 * pnocts,
            OSINT32 bufsize )
```

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

Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.
pnocts	A pointer to an integer value to receive the decoded number of octets.
bufsize	The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

Returns

Completion status of operation:

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

6.1.2.25 rtXmlDecHexStr64()

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

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

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.

6.1.2.26 rtXmlDecInt()

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

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

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

Parameters

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.

6.1.2.27 rtXmlDecInt16()

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

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

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

Parameters

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.

6.1.2.28 rtXmlDecInt64()

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

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

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

Parameters

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.

6.1.2.29 rtXmlDecInt8()

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

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

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

Parameters

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.

6.1.2.30 rtXmlDecNSAttr()

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

This function decodes an XML namespac 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.

6.1.2.31 rtXmlDecQName()

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

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

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

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

[NS 6] Prefix ::= NCName

[NS 7] LocalPart ::= NCName

Parameters

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.

6.1.2.32 rtXmlDecTime()

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

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

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

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

Parameters

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.

6.1.2.33 rtXmlDecUInt()

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

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

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

Parameters

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.

6.1.2.34 rtXmlDecUInt16()

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

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

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

Parameters

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.

6.1.2.35 rtXmlDecUInt64()

```
int rtXmlDecUInt64 (
             OSCTXT * pctxt,
             OSUINT64 * pvalue )
```

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

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

Parameters

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.

6.1.2.36 rtXmlDecUInt8()

```
int rtXmlDecUInt8 (
             OSCTXT * pctxt,
             OSUINT8 * pvalue )
```

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

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

Parameters

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.

6.1.2.37 rtXmlDecUTF8Str()

```
int rtXmlDecUTF8Str (
             OSCTXT * pctxt,
             OSUTF8CHAR * outdata,
             OSSIZE max_len )
```

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

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

Parameters

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.

6.1.2.38 rtXmlDecXmlStr()

```
int rtXmlDecXmlStr (
             OSCTXT * pctxt,
             OSXMLSTRING * outdata )
```

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

This type contains a pointer to a UTF-8 characer 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.

6.1.2.39 rtXmlDecXSIAttr()

```
int rtXmlDecXSIAttr (
             OSCTXT * pctxt,
             const OSUTF8CHAR * attrName,
             const OSUTF8CHAR * attrValue )
```

This function decodes XML schema instance (XSI) attribute.

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

Parameters

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.

6.1.2.40 rtXmlDecXSIAttrs()

```
int rtXmlDecXSIAttrs (
             OSCTXT * pctxt,
```

```
            const OSUTF8CHAR *const * attrs,
            const char * typeName )
```

This function decodes XML schema instance (XSI) attributes.

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

Parameters

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.

6.1.2.41 rtXmlParseElementName()

```
int rtXmlParseElementName (
            OSCTXT * pctxt,
            OSUTF8CHAR ** ppName )
```

This function parses the initial tag from an XML message.

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

Parameters

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.

6.1.2.42 rtXmlParseElemQName()

```
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.

Macros

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

  This macro returns the start address of the encoded XML message.
- #define rtXmlGetEncBufLen(pctxt) (pctxt)->buffer.byteIndex

  This macro returns the length of the encoded XML message.

Functions

- int rtXmlEncAny (OSCTXT *pctxt, OSXMLSTRING *pvalue, const OSUTF8CHAR *elemName, OSXML↩ Namespace *pNS)

  This function encodes a variable of the XSD any type.
- int rtXmlEncAnyTypeValue (OSCTXT *pctxt, const OSUTF8CHAR *pvalue)

  This function encodes a variable of the XSD anyType type.
- int rtXmlEncAnyAttr (OSCTXT *pctxt, OSRTDList *pAnyAttrList)

  This function encodes a list of OSAnyAttr attributes in which the name and value are given as a UTF-8 string.
- int rtXmlEncBase64Binary (OSCTXT *pctxt, OSSIZE nocts, const OSOCTET *value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

  This function encodes a variable of the XSD base64Binary type.
- int rtXmlEncBase64BinaryAttr (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

  This function encodes a variable of the XSD base64Binary type as an attribute.
- int rtXmlEncBase64StrValue (OSCTXT *pctxt, OSSIZE nocts, const OSOCTET *value)

  This function encodes a variable of the XSD base64Binary type.
- int rtXmlEncBigInt (OSCTXT *pctxt, const OSUTF8CHAR *value, const OSUTF8CHAR *elemName, OSXML↩ Namespace *pNS)

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

  This function encodes an XSD integer attribute value.
- int rtXmlEncBigIntValue (OSCTXT *pctxt, const OSUTF8CHAR *value)

  This function encodes an XSD integer attribute value.
- int rtXmlEncBitString (OSCTXT *pctxt, OSSIZE nbits, const OSOCTET *value, const OSUTF8CHAR *elem↩ Name, OSXMLNamespace *pNS)

  This function encodes a variable of the ASN.1 BIT STRING type.
- int rtXmlEncBitStringExt (OSCTXT *pctxt, OSSIZE nbits, const OSOCTET *value, OSSIZE dataSize, const O↩ SOCTET *extValue, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

  This function encodes a variable of the ASN.1 BIT STRING type.
- int rtXmlEncBinStrValue (OSCTXT *pctxt, OSSIZE nbits, const OSOCTET *data)

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

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

  This function encodes a variable of the XSD boolean type.

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

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

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

  This function encodes an XML comment.
- int rtXmlEncDate (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, OSX↩ MLNamespace *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, OSX↩ MLNamespace *pNS)

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

  This function encodes a variable of the XSD 'time' type as an string.
- int rtXmlEncDateTime (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, O↩ SXMLNamespace *pNS)

  This function encodes a numeric date/time value into an XML string representation.
- int rtXmlEncDateTimeValue (OSCTXT *pctxt, const OSXSDDateTime *pvalue)

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

  This function encodes a variable of the XSD decimal type.
- int rtXmlEncDecimalAttr (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *attrName, OSSIZE attrName↩ Len, const OSDecimalFmt *pFmtSpec)

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

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

  This function encodes a variable of the XSD double type.
- int rtXmlEncDoubleAttr (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *attrName, OSSIZE attrName↩ Len, 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 default↩ Precision)

  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, O↩ SRTDList *pNSAttrs, OSBOOL terminate)

  This function encodes an enpty element tag value (<elemName/>).
- int rtXmlEncEndDocument (OSCTXT *pctxt)

  This function adds trailor 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 *p↩NS, const OSDoubleFmt *pFmtSpec)

  This function encodes a variable of the XSD float type.
- int rtXmlEncFloatAttr (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen, const OSDoubleFmt *pFmtSpec)

  This function encodes a variable of the XSD float type as an attribute.
- int rtXmlEncGYear (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, OS↩XMLNamespace *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, O↩SXMLNamespace *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, OSX↩MLNamespace *pNS)

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

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

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

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

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

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

  This function encodes a variable of the XSD hexBinary type.
- int rtXmlEncHexBinaryAttr (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

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

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

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

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

This function encodes a variable of the XSD integer type.

- int rtXmlEncIntAttr (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

  This function encodes a variable of the XSD integer type as an attribute (name="value").
- int rtXmlEncIntPattern (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS, const OSUTF8CHAR *pattern)

  This function encodes a variable of the XSD integer type using a pattern to specify the format of the integer value.
- int rtXmlEncInt64 (OSCTXT *pctxt, OSINT64 value, const OSUTF8CHAR *elemName, OSXMLNamespace *p↩NS)

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

  This function encodes a variable of the XSD integer type.
- int rtXmlEncInt64Attr (OSCTXT *pctxt, OSINT64 value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

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

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

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

  This function prints a list of namespace attributes.
- int rtXmlEncReal10 (OSCTXT *pctxt, const OSUTF8CHAR *pvalue, const OSUTF8CHAR *elemName, OSX↩MLNamespace *pNS)

  This function encodes a variable of the ASN.1 REAL base 10 type.
- int rtXmlEncSoapArrayTypeAttr (OSCTXT *pctxt, const OSUTF8CHAR *name, const OSUTF8CHAR *value, OSSIZE itemCount)

  This function encodes the special SOAP encoding attrType attribute which specifies the number and type of elements in a SOAP array.
- int rtXmlEncStartDocument (OSCTXT *pctxt)

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

  This function encodes the Unicode byte order mark header at the start of the document.
- int rtXmlEncStartElement (OSCTXT *pctxt, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS, OS↩RTDList *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, OSXML↩Namespace *pNS)

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

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

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

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

This function encodes a Unicode string value.

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

  This function encodes an attribute in which the name and value are given as a null-terminated UTF-8 strings.
- int rtXmlEncUTF8Attr2 (OSCTXT *pctxt, const OSUTF8CHAR *name, OSSIZE nameLen, const OSUTF8CHAR *value, OSSIZE valueLen)

  This function encodes an attribute in which the name and value are given as a UTF-8 strings with lengths.
- int rtXmlEncUTF8Str (OSCTXT *pctxt, const OSUTF8CHAR *value, const OSUTF8CHAR *elemName, OSX↩ MLNamespace *pNS)

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

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

  This function encodes a variable of the XSD unsigned integer type.
- int rtXmlEncUIntAttr (OSCTXT *pctxt, OSUINT32 value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

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

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

  This function encodes a variable of the XSD integer type.
- int rtXmlEncUInt64Attr (OSCTXT *pctxt, OSUINT64 value, const OSUTF8CHAR *attrName, OSSIZE attrName↩ Len)

  This function encodes a variable of the XSD integer type as an attribute (name="value").
- int rtXmlEncXSIAttrs (OSCTXT *pctxt, OSBOOL needXSI)

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

  This function encodes an XML schema instance (XSI) type attribute value (xsi:type="value").
- int rtXmlEncXSITypeAttr2 (OSCTXT *pctxt, const OSUTF8CHAR *typeNsUri, const OSUTF8CHAR *typeName)

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

  This function encodes an XML nil attribute (xsi:nil="true").
- int rtXmlFreeInputSource (OSCTXT *pctxt)

  This function closes an input source that was previously created with one of the create input source functions such as 'rtXmlCreateFileInputSource'.
- int rtXmlSetEncBufPtr (OSCTXT *pctxt, OSOCTET *bufaddr, OSSIZE bufsiz)

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

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

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

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

6.2.1 Detailed Description

6.2.2 Macro Definition Documentation

6.2.2.1 rtXmlGetEncBufLen

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

This macro returns the length of the encoded XML message.

Parameters

pctxt	Pointer to a context structure.

Definition at line 2657 of file osrtxml.h.

6.2.2.2 rtXmlGetEncBufPtr

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

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

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

Parameters

pctxt	Pointer to a context structure.

Definition at line 2650 of file osrtxml.h.

6.2.3 Function Documentation

6.2.3.1 rtXmlEncAny()

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

This function encodes a variable of the XSD any type.

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

Parameters

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.

6.2.3.2 rtXmlEncAnyAttr()

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

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

Parameters

pctxt	Pointer to context block structure.
pAnyAttrList	List of attributes.

Returns

Completion status of operation:

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

6.2.3.3 rtXmlEncAnyTypeValue()

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

This function encodes a variable of the XSD anyType type.

This is considered to be a fully-wrapped element of any type, possibly containing attributes. (for example: * <my↩ Type>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.

6.2.3.4 rtXmlEncBase64Binary()

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

This function encodes a variable of the XSD base64Binary type.

Parameters

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.

6.2.3.5 rtXmlEncBase64BinaryAttr()

```
int rtXmlEncBase64BinaryAttr (
             OSCTXT * pctxt,
```

```
            OSUINT32 nocts,
            const OSOCTET * value,
            const OSUTF8CHAR * attrName,
            OSSIZE attrNameLen )
```

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

Parameters

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.

6.2.3.6 rtXmlEncBase64StrValue()

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

This function encodes a variable of the XSD base64Binary type.

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

Parameters

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.

6.2.3.7 rtXmlEncBigInt()

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

This function encodes a variable of the XSD integer type.

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

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

6.2.3.8 rtXmlEncBigIntAttr()

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

This function encodes an XSD integer attribute value.

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

Parameters

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.

6.2.3.9 rtXmlEncBigIntValue()

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

This function encodes an XSD integer attribute value.

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

Parameters

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.

6.2.3.10 rtXmlEncBinStrValue()

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

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

Parameters

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.

6.2.3.11 rtXmlEncBitString()

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

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

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

See also

rtXmlEncNamedBits).

Parameters

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.

6.2.3.12 rtXmlEncBitStringExt()

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

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

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

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

Parameters

pctxt	Pointer to context block structure.
nbits	Number of bits in the bit string.
value	Value to be encoded.
dataSize	Size of data member.
extValue	Value of extdata 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.

6.2.3.13 rtXmlEncBOM()

```
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.

6.2.3.14 rtXmlEncBool()

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

This function encodes a variable of the XSD boolean type.

Parameters

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.

6.2.3.15 rtXmlEncBoolAttr()

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

This function encodes an XSD boolean attribute value.

Parameters

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

Returns

Completion status of operation:

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

6.2.3.16 rtXmlEncBoolValue()

```
int rtXmlEncBoolValue (
             OSCTXT * pctxt,
             OSBOOL value )
```

This function encodes a variable of the XSD boolean type.

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

Parameters

pctxt	Pointer to context block structure.
value	Boolean value to be encoded.

Returns

Completion status of operation:

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

6.2.3.17 rtXmlEncCanonicalSort()

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

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

Parameters

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

6.2.3.18 rtXmlEncComment()

```
int rtXmlEncComment (
             OSCTXT * pctxt,
             const OSUTF8CHAR * comment )
```

This function encodes an XML comment.

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

Parameters

pctxt	Pointer to context block structure.
comment	The comment text.

Returns

Completion status of operation:

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

6.2.3.19 rtXmlEncDate()

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

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

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

Parameters

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.

6.2.3.20 rtXmlEncDateTime()

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

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

Parameters

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.

6.2.3.21 rtXmlEncDateTimeValue()

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

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

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

Parameters

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.

6.2.3.22 rtXmlEncDateValue()

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

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

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

Parameters

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.

6.2.3.23 rtXmlEncDecimal()

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

This function encodes a variable of the XSD decimal type.

Parameters

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.

6.2.3.24 rtXmlEncDecimalAttr()

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

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

Parameters

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.

6.2.3.25 rtXmlEncDecimalValue()

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

This function encodes a value of the XSD decimal type.

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

Parameters

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.

6.2.3.26 rtXmlEncDouble()

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

This function encodes a variable of the XSD double type.

Parameters

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.

6.2.3.27 rtXmlEncDoubleAttr()

```
int rtXmlEncDoubleAttr (
             OSCTXT * pctxt,
             OSREAL value,
             const OSUTF8CHAR * attrName,
             OSSIZE attrNameLen,
             const OSDoubleFmt * pFmtSpec )
```

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

Parameters

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.

6.2.3.28 rtXmlEncDoubleNormalValue()

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

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

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

Parameters

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.

6.2.3.29 rtXmlEncDoubleValue()

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

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

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

Parameters

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.

6.2.3.30 rtXmlEncEmptyElement()

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

This function encodes an enpty 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.

6.2.3.31 rtXmlEncEndDocument()

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

This function adds trailor 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.

6.2.3.32 rtXmlEncEndElement()

```
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.

6.2.3.33 rtXmlEncEndSoapElems()

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

This function encodes SOAP end element tags.

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

Parameters

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.

6.2.3.34 rtXmlEncEndSoapEnv()

```
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.

6.2.3.35 rtXmlEncFloat()

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

This function encodes a variable of the XSD float type.

Parameters

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.

6.2.3.36 rtXmlEncFloatAttr()

```
int rtXmlEncFloatAttr (
             OSCTXT * pctxt,
```

```
            OSREAL value,
            const OSUTF8CHAR * attrName,
            OSSIZE attrNameLen,
            const OSDoubleFmt * pFmtSpec )
```

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

Parameters

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.

6.2.3.37 rtXmlEncGDay()

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

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

Parameters

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.

6.2.3.38 rtXmlEncGDayValue()

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

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

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

Parameters

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.

6.2.3.39 rtXmlEncGMonth()

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

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

Parameters

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.

6.2.3.40 rtXmlEncGMonthDay()

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

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

Parameters

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.

6.2.3.41 rtXmlEncGMonthDayValue()

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

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

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

Parameters

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.

6.2.3.42 rtXmlEncGMonthValue()

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

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

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

Parameters

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.

6.2.3.43 rtXmlEncGYear()

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

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

Parameters

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.

6.2.3.44 rtXmlEncGYearMonth()

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

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

Parameters

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.

6.2.3.45 rtXmlEncGYearMonthValue()

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

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

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

Parameters

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.

6.2.3.46 rtXmlEncGYearValue()

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

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

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

Parameters

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.

6.2.3.47 rtXmlEncHexBinary()

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

This function encodes a variable of the XSD hexBinary type.

Parameters

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.

6.2.3.48 rtXmlEncHexBinaryAttr()

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

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

Parameters

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.

6.2.3.49 rtXmlEncHexStrValue()

```
int rtXmlEncHexStrValue (
             OSCTXT * pctxt,
             OSSIZE nocts,
             const OSOCTET * data )
```

This function encodes a variable of the XSD hexBinary type.

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

Parameters

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.

6.2.3.50 rtXmlEncIndent()

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

This function adds indentation whitespace to the output stream.

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

Parameters

pctxt	Pointer to context block structure.

Returns

Completion status of operation:

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

6.2.3.51 rtXmlEncInt()

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

This function encodes a variable of the XSD integer type.

Parameters

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.

6.2.3.52 rtXmlEncInt64()

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

This function encodes a variable of the XSD integer type.

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

Parameters

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.

6.2.3.53 rtXmlEncInt64Attr()

```
int rtXmlEncInt64Attr (
             OSCTXT * pctxt,
             OSINT64 value,
             const OSUTF8CHAR * attrName,
             OSSIZE attrNameLen )
```

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

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

Parameters

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.

6.2.3.54 rtXmlEncInt64Value()

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

This function encodes a variable of the XSD integer type.

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

Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.

Returns

Completion status of operation:

- 0 = success,

- negative return value is error.

6.2.3.55 rtXmlEncIntAttr()

```
int rtXmlEncIntAttr (
             OSCTXT * pctxt,
             OSINT32 value,
             const OSUTF8CHAR * attrName,
             OSSIZE attrNameLen )
```

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

Parameters

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.

6.2.3.56 rtXmlEncIntPattern()

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

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

Parameters

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.

6.2.3.57 rtXmlEncIntValue()

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

This function encodes a variable of the XSD integer type.

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

Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.

Returns

Completion status of operation:

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

6.2.3.58 rtXmlEncNamedBits()

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

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

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

6.2.3.59 rtXmlEncNSAttrs()

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

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

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

Parameters

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.

6.2.3.60 rtXmlEncReal10()

```
int rtXmlEncReal10 (
             OSCTXT * pctxt,
```

```
            const OSUTF8CHAR * pvalue,
            const OSUTF8CHAR * elemName,
            OSXMLNamespace * pNS )
```

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

Parameters

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.

6.2.3.61 rtXmlEncSoapArrayTypeAttr()

```
int rtXmlEncSoapArrayTypeAttr (
            OSCTXT * pctxt,
            const OSUTF8CHAR * name,
            const OSUTF8CHAR * value,
            OSSIZE itemCount )
```

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

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

Parameters

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.

6.2.3.62 rtXmlEncStartDocument()

```
int rtXmlEncStartDocument (
             OSCTXT * pctxt )
```

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

This is normally the following:

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

Parameters

pctxt	Pointer to context block structure.

Returns

Completion status of operation:

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

6.2.3.63 rtXmlEncStartElement()

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

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

Parameters

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.

6.2.3.64 rtXmlEncStartSoapElems()

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

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

This includes all of the standard SOAP namespace attributes.

Parameters

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.

6.2.3.65 rtXmlEncStartSoapEnv()

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

This function encodes a SOAP envelope start element tag.

This includes all of the standard SOAP namespace attributes.

Parameters

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.

6.2.3.66 rtXmlEncString()

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

This function encodes a variable of the XSD string type.

Parameters

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.

6.2.3.67 rtXmlEncStringValue()

```
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.

6.2.3.68 rtXmlEncStringValue2()

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

This function encodes a variable of the XSD string type.

Parameters

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

Returns

Completion status of operation:

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

6.2.3.69 rtXmlEncTermStartElement()

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

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

It will also add XSI attributes to the element.

Parameters

pctxt	Pointer to context block structure.

Returns

Completion status of operation:

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

6.2.3.70 rtXmlEncTime()

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

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

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

Parameters

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.

6.2.3.71 rtXmlEncTimeValue()

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

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

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

Parameters

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.

6.2.3.72 rtXmlEncUInt()

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

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

Parameters

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.

6.2.3.73 rtXmlEncUInt64()

```
int rtXmlEncUInt64 (
             OSCTXT * pctxt,
             OSUINT64 value,
```

```
            const OSUTF8CHAR * elemName,
            OSXMLNamespace * pNS )
```

This function encodes a variable of the XSD integer type.

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

Parameters

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.

6.2.3.74 rtXmlEncUInt64Attr()

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

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

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

Parameters

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.

6.2.3.75 rtXmlEncUInt64Value()

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

This function encodes a variable of the XSD integer type.

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

Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.

Returns

Completion status of operation:

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

6.2.3.76 rtXmlEncUIntAttr()

```
int rtXmlEncUIntAttr (
             OSCTXT * pctxt,
             OSUINT32 value,
             const OSUTF8CHAR * attrName,
             OSSIZE attrNameLen )
```

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

Parameters

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.

6.2.3.77 rtXmlEncUIntValue()

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

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

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

Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.

Returns

Completion status of operation:

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

6.2.3.78 rtXmlEncUnicodeStr()

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

This function encodes a Unicode string value.

Parameters

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.

6.2.3.79 rtXmlEncUTF8Attr()

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

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

Parameters

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.

6.2.3.80 rtXmlEncUTF8Attr2()

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

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

Parameters

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.

6.2.3.81 rtXmlEncUTF8Str()

```
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.

6.2.3.82 rtXmlEncXSIAttrs()

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

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

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

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

Parameters

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.

6.2.3.83 rtXmlEncXSINilAttr()

```
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.

6.2.3.84 rtXmlEncXSITypeAttr()

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

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

Parameters

pctxt	Pointer to context block structure.
value	XSI type attribute value.

Returns

Completion status of operation:

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

6.2.3.85 rtXmlEncXSITypeAttr2()

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

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

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

Parameters

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.

6.2.3.86 rtXmlFreeInputSource()

```
int rtXmlFreeInputSource (
             OSCTXT * pctxt )
```

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

Parameters

pctxt	Pointer to context block structure.

Returns

Completion status of operation:

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

6.2.3.87 rtXmlGetIndent()

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

This function returns current XML output indent value.

Parameters

pctxt	Pointer to OSCTXT structure

Returns

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

6.2.3.88 rtXmlGetIndentChar()

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

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

Parameters

pctxt	Pointer to OSCTXT structure

Returns

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

6.2.3.89 rtXmlGetWriteBOM()

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

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

Parameters

pctxt	Pointer to OSCTXT structure.

Returns

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

6.2.3.90 rtXmlPrintNSAttrs()

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

This function prints a list of namespace attributes.

Parameters

name	Name to print.
data	List of namespace attributes.

Returns

Completion status of operation:

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

6.2.3.91 rtXmlSetEncBufPtr()

```
int rtXmlSetEncBufPtr (
             OSCTXT * pctxt,
             OSOCTET * bufaddr,
             OSSIZE bufsiz )
```

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

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

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

Parameters

pctxt	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
bufaddr	A pointer to a memory buffer to use to encode a message. The buffer should be declared as an array of unsigned characters (OCTETs). This parameter can be set to NULL to specify dynamic encoding (i.e., the encode functions will dynamically allocate a buffer to hold the encoded message).
bufsiz	The length of the memory buffer in bytes. Should be set to zero if NULL was specified for bufaddr (i.e. dynamic encoding was selected).

6.3 XML utility functions.

Functions

- int rtXmlWriteToFile (OSCTXT *pctxt, const char *filename)

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

6.3.1 Detailed Description

6.3.2 Function Documentation

6.3.2.1 rtXmlWriteToFile()

```
int rtXmlWriteToFile (
             OSCTXT * pctxt,
             const char * filename )
```

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

Parameters

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, OSSIZE attrIndex)

  This function decodes an any attribute string.
- int rtXmlpDecAnyElem (OSCTXT *pctxt, const OSUTF8CHAR **pvalue)

  This function decodes an arbitrary XML section of code as defined by the XSD any type (xsd:any).
- int rtXmlpDecBase64Str (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSSIZE bufsize)

  This function decodes a contents of a Base64-encode binary string into a static memory structure.
- int rtXmlpDecBase64Str64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnocts, OSSIZE bufsize)

  This function is identical to rtXmlpDecBase64Str except that it supports 64-bit integer lengths on 64-bit systems.
- int rtXmlpDecBigInt (OSCTXT *pctxt, const OSUTF8CHAR **pvalue)

  This function will decode a variable of the XSD integer type.
- int rtXmlpDecBitString (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnbits, OSUINT32 bufsize)

  This function decodes a bit string value.
- int rtXmlpDecBitString64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnbits, OSSIZE bufsize)

  This function is identical to rtXmlpDecBitString except that it supports lengths up to 64-bits in size on 64-bit machines.
- int rtXmlpDecBitStringExt (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnbits, OSOCTET **ppextdata, OSUINT32 bufsize)

  This function decodes a bit string value.
- int rtXmlpDecBitStringExt64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnbits, OSOCTET **ppextdata, OSSIZE bufsize)

  This function is identical to rtXmlpDecBitStringExt except that it supports lengths up to 64-bits in size on 64-bit machines.
- int rtXmlpDecBool (OSCTXT *pctxt, OSBOOL *pvalue)

  This function decodes a variable of the boolean type.
- int rtXmlpDecDate (OSCTXT *pctxt, OSXSDDateTime *pvalue)

  This function decodes a variable of the XSD 'date' type.
- int rtXmlpDecDateTime (OSCTXT *pctxt, OSXSDDateTime *pvalue)

  This function decodes a variable of the XSD 'dateTime' type.
- int rtXmlpDecDecimal (OSCTXT *pctxt, OSREAL *pvalue, int totalDigits, int fractionDigits)

  This function decodes the contents of a decimal data type.
- int rtXmlpDecDouble (OSCTXT *pctxt, OSREAL *pvalue)

  This function decodes the contents of a float or double data type.
- int rtXmlpDecDoubleExt (OSCTXT *pctxt, OSUINT8 flags, OSREAL *pvalue)

  This function decodes the contents of a float or double data type.
- int rtXmlpDecDynBase64Str (OSCTXT *pctxt, OSDynOctStr *pvalue)

  This function decodes a contents of a Base64-encode binary string.
- int rtXmlpDecDynBase64Str64 (OSCTXT *pctxt, OSDynOctStr64 *pvalue)

  This function is identical to rtXmlpDecDynBase64Str except that it supports 64-bit integer lengths on 64-bit systems.
- int rtXmlpDecDynBitString (OSCTXT *pctxt, OSDynOctStr *pvalue)

  This function decodes a bit string value.
- int rtXmlpDecDynHexStr (OSCTXT *pctxt, OSDynOctStr *pvalue)

This function decodes a contents of a hexBinary string.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  This function decodes the contents of a named bit field.
- int rtXmlpDecStrList (OSCTXT *pctxt, OSRTDList *plist)

  This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.
- int rtXmlpDecTime (OSCTXT *pctxt, OSXSDDateTime *pvalue)

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

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

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

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

- int rtXmlpDecUInt64 (OSCTXT *pctxt, OSUINT64 *pvalue)

  This function decodes the contents of an unsigned 64-bit integer data type.
- int rtXmlpDecXmlStr (OSCTXT *pctxt, OSXMLSTRING *outdata)

  This function decodes the contents of an XML string data type.
- int rtXmlpDecXmlStrList (OSCTXT *pctxt, OSRTDList *plist)

  This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.
- int rtXmlpDecXSIAttr (OSCTXT *pctxt, const OSXMLNameFragments *attrName)

  This function decodes XSI (XML Schema Instance) attributes that may be present in any arbitrary XML element within a document.
- int rtXmlpDecXSITypeAttr (OSCTXT *pctxt, const OSXMLNameFragments *attrName, const OSUTF8CHAR **ppAttrValue)

  This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).
- int rtXmlpGetAttributeID (const OSXMLStrFragment *attrName, OSINT16 nsidx, OSSIZE nAttr, const OSXML↩ AttrDescr attrNames[ ], OSUINT32 attrPresent[ ])

  This function finds an attribute in the descriptor table.
- int rtXmlpGetNextElem (OSCTXT *pctxt, OSXMLElemDescr *pElem, OSINT32 level)

  This function parse the next element start tag.
- int rtXmlpGetNextElemID (OSCTXT *pctxt, const OSXMLElemIDRec *tab, OSSIZE nrows, OSINT32 level, O↩ SBOOL continueParse)

  This function parses the next start tag and finds the index of the element name in the descriptor table.
- int 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, OSSIZE 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 *p↩ Group, 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 *p↩ Group, 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 group↩ Mode)

  This is an ASN.1 extension-supporting version of rtXmlpGetNextSeqElemID.
- int rtXmlpGetNextAllElemID (OSCTXT *pctxt, const OSXMLElemIDRec *tab, OSSIZE nrows, const OSUINT8 *pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)

  This function parses the next start tag and finds index of element name in descriptor table.
- int rtXmlpGetNextAllElemID16 (OSCTXT *pctxt, const OSXMLElemIDRec *tab, OSSIZE nrows, const OSUINT16 *pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)

  This function parses the next start tag and finds index of element name in descriptor table.
- int rtXmlpGetNextAllElemID32 (OSCTXT *pctxt, const OSXMLElemIDRec *tab, OSSIZE nrows, const OSUINT32 *pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)

  This function parses the next start tag and finds index of element name in descriptor table.
- void rtXmlpSetNamespaceTable (OSCTXT *pctxt, const OSUTF8CHAR *namespaceTable[ ], OSSIZE nm↩ Namespaces)

  Sets user namespace table.
- int rtXmlpCreateReader (OSCTXT *pctxt)

  Creates pull parser reader structure within the context.
- void rtXmlpHideAttributes (OSCTXT *pctxt)

  Disable access to attributes.
- OSBOOL rtXmlpNeedDecodeAttributes (OSCTXT *pctxt)

  This function checks if attributes were previously decoded.
- void rtXmlpMarkPos (OSCTXT *pctxt)

  Save current decode position.
- void rtXmlpRewindToMarkedPos (OSCTXT *pctxt)

  Rewind to saved decode position.
- void rtXmlpResetMarkedPos (OSCTXT *pctxt)

  Reset saved decode position.
- int rtXmlpGetXSITypeAttr (OSCTXT *pctxt, const OSUTF8CHAR **ppAttrValue, OSINT16 *nsidx, OSSIZE *p↩ LocalOffs)

  This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).
- int rtXmlpGetXmlnsAttrs (OSCTXT *pctxt, OSRTDList *pNSAttrs)

  This function decodes namespace attributes from start tag and adds them to the given list.
- int rtXmlpDecXSIAttrs (OSCTXT *pctxt)

  This function decodes XSI (XML Schema Instance) that may be present in any arbitrary XML element within a document.
- OSBOOL 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[ ], OSSIZE typetabsiz)

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

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

  This function find index of XSI (XML Schema Instance) type in descriptor table.
- void rtXmlpForceDecodeAsGroup (OSCTXT *pctxt)

  Disable skipping of unknown elements in optional sequence tail.
- OSBOOL 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, OSSIZE nbytes)

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

6.4.1 Detailed Description

6.4.2 Function Documentation

6.4.2.1 rtXmlEncAttrC14N()

```
int rtXmlEncAttrC14N (
             OSCTXT * pctxt )
```

This function used only in C14 mode.

It provide atributes sorting.

Parameters

pctxt	Pointer to context block structure.

Returns

Completion status of operation:

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

6.4.2.2 rtXmlpCountListItems()

```
void rtXmlpCountListItems (
             OSCTXT * 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.

6.4.2.3 rtXmlpCreateReader()

```
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.

6.4.2.4 rtXmlpDecAny()

```
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.

6.4.2.5 rtXmlpDecAny2()

```
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.

6.4.2.6 rtXmlpDecAnyAttrStr()

```
int rtXmlpDecAnyAttrStr (
             OSCTXT * pctxt,
             const OSUTF8CHAR ** ppAttrStr,
             OSSIZE attrIndex )
```

This function decodes an any attribute string.

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

Parameters

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.

6.4.2.7 rtXmlpDecAnyElem()

```
int rtXmlpDecAnyElem (
             OSCTXT * pctxt,
             const OSUTF8CHAR ** pvalue )
```

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

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

Parameters

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.

6.4.2.8 rtXmlpDecBase64Str()

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

```
            OSOCTET * pvalue,
            OSUINT32 * pnocts,
            OSSIZE bufsize )
```

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

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

Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.
pnocts	A pointer to an integer value to receive the decoded number of octets.
bufsize	The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

Returns

Completion status of operation:

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

6.4.2.9 rtXmlpDecBase64Str64()

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

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

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.

6.4.2.10 rtXmlpDecBigInt()

```
int rtXmlpDecBigInt (
             OSCTXT * pctxt,
             const OSUTF8CHAR ** pvalue )
```

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

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

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

Parameters

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.

6.4.2.11 rtXmlpDecBitString()

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

This function decodes a bit string value.

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

Parameters

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.

6.4.2.12 rtXmlpDecBitString64()

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

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

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

Parameters

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.

See also

rtXmlpDecBitString

6.4.2.13 rtXmlpDecBitStringExt()

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

This function decodes a bit string value.

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

Parameters

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.
ppextdata	Pointer to byte array containing extension data.
bufsize	Size of buffer passed in pvalue argument.

Returns

Completion status of operation:

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

6.4.2.14 rtXmlpDecBitStringExt64()

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

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

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

Parameters

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.
ppextdata	Pointer to byte array containing extension data.
bufsize	Size of buffer passed in pvalue argument.

Returns

Completion status of operation:

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

See also

rtXmlpDecBitStringExt

6.4.2.15 rtXmlpDecBool()

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

This function decodes a variable of the boolean type.

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

Parameters

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.

6.4.2.16 rtXmlpDecDate()

```
int rtXmlpDecDate (
             OSCTXT * pctxt,
             OSXSDDateTime * pvalue )
```

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

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

Parameters

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.

6.4.2.17 rtXmlpDecDateTime()

```
int rtXmlpDecDateTime (
             OSCTXT * pctxt,
             OSXSDDateTime * pvalue )
```

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

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

Parameters

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.

6.4.2.18 rtXmlpDecDecimal()

```
int rtXmlpDecDecimal (
             OSCTXT * pctxt,
             OSREAL * pvalue,
             int totalDigits,
             int fractionDigits )
```

This function decodes the contents of a decimal data type.

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

Parameters

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.

6.4.2.19 rtXmlpDecDouble()

```
int rtXmlpDecDouble (
             OSCTXT * pctxt,
             OSREAL * pvalue )
```

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

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

Parameters

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.

6.4.2.20 rtXmlpDecDoubleExt()

```
int rtXmlpDecDoubleExt (
            OSCTXT * pctxt,
            OSUINT8 flags,
            OSREAL * pvalue )
```

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

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

Parameters

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.

6.4.2.21 rtXmlpDecDynBase64Str()

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

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

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

Parameters

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.

6.4.2.22 rtXmlpDecDynBase64Str64()

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

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

Parameters

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.

6.4.2.23 rtXmlpDecDynBitString()

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

This function decodes a bit string value.

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

Parameters

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.

6.4.2.24 rtXmlpDecDynHexStr()

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

This function decodes a contents of a hexBinary string.

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

Parameters

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.

6.4.2.25 rtXmlpDecDynHexStr64()

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

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

Parameters

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.

6.4.2.26 rtXmlpDecDynUnicodeStr()

```
int rtXmlpDecDynUnicodeStr (
             OSCTXT * pctxt,
             const OSUNICHAR ** ppdata,
             OSSIZE * pnchars )
```

This function decodes a Unicode string data type.

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

Parameters

pctxt	Pointer to context block structure.
ppdata	Pointer to Unicode character string. A Unicode character string is represented as an array of unsigned 16-bit integers in C. Memory is allocated for the string using the run-time memory manager.
pnchars	Pointer to integer variables to receive the number of characters in the string.

Returns

Completion status of operation:

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

6.4.2.27 rtXmlpDecDynUTF8Str()

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

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

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

Parameters

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.

6.4.2.28 rtXmlpDecGDay()

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

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

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

Parameters

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.

6.4.2.29 rtXmlpDecGMonth()

```
int rtXmlpDecGMonth (
             OSCTXT * pctxt,
             OSXSDDateTime * pvalue )
```

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

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

Parameters

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.

6.4.2.30 rtXmlpDecGMonthDay()

```
int rtXmlpDecGMonthDay (
             OSCTXT * pctxt,
             OSXSDDateTime * pvalue )
```

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

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

Parameters

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.

6.4.2.31 rtXmlpDecGYear()

```
int rtXmlpDecGYear (
             OSCTXT * pctxt,
             OSXSDDateTime * pvalue )
```

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

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

Parameters

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.

6.4.2.32 rtXmlpDecGYearMonth()

```
int rtXmlpDecGYearMonth (
             OSCTXT * pctxt,
             OSXSDDateTime * pvalue )
```

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

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

Parameters

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.

6.4.2.33 rtXmlpDecHexStr()

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

```
            OSOCTET * pvalue,
            OSUINT32 * pnocts,
            OSSIZE bufsize )
```

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

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

Parameters

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.

6.4.2.34 rtXmlpDecHexStr64()

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

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

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.

See also

rtXmlpDecHexStr

6.4.2.35 rtXmlpDecInt()

```
int rtXmlpDecInt (
             OSCTXT * pctxt,
             OSINT32 * pvalue )
```

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

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

Parameters

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.

6.4.2.36 rtXmlpDecInt16()

```
int rtXmlpDecInt16 (
             OSCTXT * pctxt,
             OSINT16 * pvalue )
```

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

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

Parameters

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.

6.4.2.37 rtXmlpDecInt64()

```
int rtXmlpDecInt64 (
             OSCTXT * pctxt,
             OSINT64 * pvalue )
```

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

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

Parameters

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.

6.4.2.38 rtXmlpDecInt8()

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

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

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

Parameters

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.

6.4.2.39 rtXmlpDecNamedBits()

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

This function decodes the contents of a named bit field.

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

Parameters

pctxt	Pointer to context block structure.
pBitMap	Pointer to bit map structure that defined token to bit mappings.
pvalue	Pointer to buffer to recieve 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.

6.4.2.40 rtXmlpDecNamedBits64()

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

```
            const OSBitMapItem * pBitMap,
            OSOCTET * pvalue,
            OSSIZE * pnbits,
            OSSIZE bufsize )
```

This function decodes the contents of a named bit field.

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

Parameters

pctxt	Pointer to context block structure.
pBitMap	Pointer to bit map structure that defined token to bit mappings.
pvalue	Pointer to buffer to recieve 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.

6.4.2.41 rtXmlpDecStrList()

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

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

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

Parameters

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.

6.4.2.42 rtXmlpDecTime()

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

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

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

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

Parameters

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.

6.4.2.43 rtXmlpDecUInt()

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

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

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

Parameters

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.

6.4.2.44 rtXmlpDecUInt16()

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

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

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

Parameters

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.

6.4.2.45 rtXmlpDecUInt64()

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

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

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

Parameters

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.

6.4.2.46 rtXmlpDecUInt8()

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

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

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

Parameters

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.

6.4.2.47 rtXmlpDecUTF8Str()

```
int rtXmlpDecUTF8Str (
             OSCTXT * pctxt,
             OSUTF8CHAR * out,
             OSSIZE max_len )
```

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

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

Parameters

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.

6.4.2.48 rtXmlpDecXmlStr()

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

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

This type contains a pointer to a UTF-8 characer 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.

6.4.2.49 rtXmlpDecXmlStrList()

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

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

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

Parameters

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.

6.4.2.50 rtXmlpDecXSIAttr()

```
int rtXmlpDecXSIAttr (
             OSCTXT * pctxt,
             const OSXMLNameFragments * attrName )
```

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

Parameters

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.

6.4.2.51 rtXmlpDecXSIAttrs()

```
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.

6.4.2.52 rtXmlpDecXSITypeAttr()

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

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

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

Parameters

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.

6.4.2.53 rtXmlpForceDecodeAsGroup()

```
void rtXmlpForceDecodeAsGroup (
             OSCTXT * pctxt )
```

Disable skipping of unknown elements in optional sequence tail.

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

Parameters

pctxt	Pointer to context block structure.

6.4.2.54 rtXmlpGetAttributeCount()

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

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

Parameters

pctxt	Pointer to context block structure.

Returns

Completion status of operation:

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

6.4.2.55 rtXmlpGetAttributeID()

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

This function finds an attribute in the descriptor table.

Parameters

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.

Parameters

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.

6.4.2.56 rtXmlpGetCurrentLevel()

```
OSINT32 rtXmlpGetCurrentLevel (
             OSCTXT * pctxt )
```

This function returns current nesting level.

Parameters

pctxt	Pointer to context block structure.

Returns

Current nesting level.

6.4.2.57 rtXmlpGetNextAllElemID()

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

```
            OSSIZE nOrder,
            OSSIZE maxOrder,
            int anyID )
```

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

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

Parameters

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.

6.4.2.58 rtXmlpGetNextAllElemID16()

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

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

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

Parameters

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.

6.4.2.59 rtXmlpGetNextAllElemID32()

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

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

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

Parameters

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.

6.4.2.60 rtXmlpGetNextElem()

```
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.

6.4.2.61 rtXmlpGetNextElemID()

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

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

If this reaches the closing tag for the current level, it returns XML_OK_EOB. If this encounters an unexpected element and !continueParse, it returns RTERR_UNEXPELEM (without logging it).

Parameters

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.

6.4.2.62 rtXmlpGetNextSeqElemID()

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

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

It is used to decode sequences in strict mode.

Handling of unexpected elements:

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

This function is equivalent to: rtXmlpGetNextSeqElemID2(pctxt, tab, pGroup, curID, lastMandatoryID, groupMode, F↩ ALSE);

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 setted to TRUE when decoding groups or base types.

Returns

Completion status of operation:

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

6.4.2.63 rtXmlpGetNextSeqElemID2()

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

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

It is used to decode sequences in strict mode.

Parameters

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 setted 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.

6.4.2.64 rtXmlpGetNextSeqElemIDExt()

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

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

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

Parameters

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 setted to TRUE when decoding groups or base types.

Returns

Completion status of operation:

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

6.4.2.65 rtXmlpGetReader()

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

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

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

Parameters

pctxt	Pointer to context block structure.

Returns

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

6.4.2.66 rtXmlpGetXmlnsAttrs()

```
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.

6.4.2.67 rtXmlpGetXSITypeAttr()

```
int rtXmlpGetXSITypeAttr (
             OSCTXT * pctxt,
             const OSUTF8CHAR ** ppAttrValue,
             OSINT16 * nsidx,
             OSSIZE * pLocalOffs )
```

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

Parameters

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.

6.4.2.68 rtXmlpGetXSITypeIndex()

```
int rtXmlpGetXSITypeIndex (
             OSCTXT * pctxt,
```

```
const OSXMLItemDescr typetab[],
OSSIZE typetabsiz )
```

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

Parameters

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.

6.4.2.69 rtXmlpHasAttributes()

```
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.

6.4.2.70 rtXmlpHideAttributes()

```
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.

6.4.2.71 rtXmlpIsDecodeAsGroup()

```
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.

6.4.2.72 rtXmlpIsEmptyElement()

```
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.

6.4.2.73 rtXmlpIsLastEventDone()

```
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.

6.4.2.74 rtXmlpIsUTF8Encoding()

```
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.

6.4.2.75 rtXmlpListHasItem()

```
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.

6.4.2.76 rtXmlpLookupXSITypeIndex()

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

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

Parameters

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.

6.4.2.77 rtXmlpMarkLastEventActive()

```
int rtXmlpMarkLastEventActive (
             OSCTXT * pctxt )
```

This function marks current tag as unprocessed.

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

Parameters

pctxt	Pointer to context block structure.

Returns

Completion status of operation:

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

6.4.2.78 rtXmlpMarkPos()

```
void rtXmlpMarkPos (
             OSCTXT * pctxt )
```

Save current decode position.

Parameters

pctxt	Pointer to context block structure.

6.4.2.79 rtXmlpMatchEndTag()

```
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.

6.4.2.80 rtXmlpMatchStartTag()

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

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

Parameters

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.

6.4.2.81 rtXmlpNeedDecodeAttributes()

```
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.

6.4.2.82 rtXmlpReadBytes()

```
int rtXmlpReadBytes (
             OSCTXT * pctxt,
             OSOCTET * pbuf,
             OSSIZE nbytes )
```

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

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

Parameters

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.

6.4.2.83 rtXmlpResetMarkedPos()

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

Reset saved decode position.

Parameters

pctxt	Pointer to context block structure.

6.4.2.84 rtXmlpRewindToMarkedPos()

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

Rewind to saved decode position.

Parameters

pctxt	Pointer to context block structure.

6.4.2.85 rtXmlpSelectAttribute()

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

This function selects attribute to decode.

Parameters

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.

6.4.2.86 rtXmlpSetListMode()

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

Sets list mode.

Attribute value or element content is decoded by tokens.

Parameters

pctxt	Pointer to context block structure.

6.4.2.87 rtXmlpSetMixedContentMode()

```
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.

6.4.2.88 rtXmlpSetNamespaceTable()

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

Sets user namespace table.

Parameters

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

6.4.2.89 rtXmlpSetWhiteSpaceMode()

```
void rtXmlpSetWhiteSpaceMode (
```

```
OSCTXT * pctxt,
OSXMLWhiteSpaceMode whiteSpaceMode )
```

Sets the whitespace treatment mode.

This mode affects the content and attribute values reading. For example, if OSXMLWSM_COLLAPSE mode is set then all spaces in returned data will be already collapsed.

Parameters

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

Returns

Previously set whitespace mode.

6.5 XML run-time error status codes.

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

Macros

- #define XML_OK_EOB 0x7fffffff

  End of block marker.
- #define XML_OK_FRAG XML_OK_EOB

  Maintained for backward compatibility.
- #define XML_E_BASE -200

  Error base.
- #define XML_E_GENERR (XML_E_BASE)

  General error.
- #define XML_E_INVSYMBOL (XML_E_BASE-1)

  An invalid XML symbol (character) was detected at the given point in the parse stream.
- #define XML_E_TAGMISMATCH (XML_E_BASE-2)

  Start/end tag mismatch.
- #define XML_E_DUPLATTR (XML_E_BASE-3)

  Duplicate attribute found.
- #define XML_E_BADCHARREF (XML_E_BASE-4)

  Bad character reference found.
- #define XML_E_INVMODE (XML_E_BASE-5)

  Invalid mode.
- #define XML_E_UNEXPEOF (XML_E_BASE-6)

  Unexpected end of file (document).
- #define XML_E_NOMATCH (XML_E_BASE-7)

  Current tag is not matched to specified one.
- #define XML_E_ELEMMISRQ (XML_E_BASE-8)

  Missing required element.
- #define XML_E_ELEMSMISRQ (XML_E_BASE-9)

  Missing required elements.
- #define XML_E_TOOFEWELEMS (XML_E_BASE-10)

  The number of elements in a repeating collection was less than the number of elements specified in the XSD minOccurs facet for this type or element.
- #define XML_E_UNEXPSTARTTAG (XML_E_BASE-11)

  Unexpected start tag.
- #define XML_E_UNEXPENDTAG (XML_E_BASE-12)

  Unexpected end tag.
- #define XML_E_IDNOTFOU (XML_E_BASE-13)

  Expected identifier not found.
- #define XML_E_INVTYPEINFO (XML_E_BASE-14)

  Unknown xsi:type.
- #define XML_E_NSURINOTFOU (XML_E_BASE-15)

  Namespace URI not defined for given prefix.
- #define XML_E_KEYNOTFOU (XML_E_BASE-16)

Keyref constraint has some key that not present in refered constraint.

- #define XML_E_DUPLKEY (XML_E_BASE-17)

  Key or unique constraint has duplicated key.
- #define XML_E_FLDABSENT (XML_E_BASE-18)

  Some key has no full set of fields.
- #define XML_E_DUPLFLD (XML_E_BASE-19)

  Some key has more than one value for field.
- #define XML_E_NOTEMPTY (XML_E_BASE-20)

  An element was not empty when expected.

6.5.1 Detailed Description

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

In many cases, additional information and parameters for the different errors are stored in the context structure at the time the error in raised. This additional information can be output using the `rtxErrPrint` or `rtxErrLogUsingCB` run-time functions.

6.5.2 Macro Definition Documentation

6.5.2.1 XML_E_BASE

```
#define XML_E_BASE -200
```

Error base.

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

Definition at line 60 of file rtXmlErrCodes.h.

6.5.2.2 XML_E_ELEMMISRQ

```
#define XML_E_ELEMMISRQ (XML_E_BASE-8)
```

Missing required element.

This status code is returned by the decoder when the decoder knows exactly which element is absent.

Definition at line 121 of file rtXmlErrCodes.h.

6.5.2.3 XML_E_ELEMSMISRQ

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

Missing required elements.

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

Definition at line 128 of file rtXmlErrCodes.h.

6.5.2.4 XML_E_FLDABSENT

```
#define XML_E_FLDABSENT (XML_E_BASE-18)
```

Some key has no full set of fields.

It is not valid for key constraint.

Definition at line 199 of file rtXmlErrCodes.h.

6.5.2.5 XML_E_NOMATCH

```
#define XML_E_NOMATCH (XML_E_BASE-7)
```

Current tag is not matched to specified one.

Informational code.

Definition at line 115 of file rtXmlErrCodes.h.

6.5.2.6 XML_E_NSURINOTFOU

```
#define XML_E_NSURINOTFOU (XML_E_BASE-15)
```

Namespace URI not defined for given prefix.

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

Definition at line 166 of file rtXmlErrCodes.h.

6.5.2.7 XML_E_TAGMISMATCH

```
#define XML_E_TAGMISMATCH (XML_E_BASE-2)
```

Start/end tag mismatch.

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

Definition at line 90 of file rtXmlErrCodes.h.

6.5.2.8 XML_OK_EOB

```
#define XML_OK_EOB 0x7fffffff
```

End of block marker.

Definition at line 51 of file rtXmlErrCodes.h.

6.5.2.9 XML_OK_FRAG

```
#define XML_OK_FRAG XML_OK_EOB
```

Maintained for backward compatibility.

Definition at line 54 of file rtXmlErrCodes.h.

6.6 DOM API functions.

Typedefs

- typedef int OSRTDOMError

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

Functions

- EXTERNDOM void domParserInit ()

  Inits parser.
- EXTERNDOM void domParserShutdown ()

  Shutdowns parser.
- EXTERNDOM OSRTDOMError domParseFile (const char *fileSpec, OSRTDOMDocPtr *pXmlDoc)

  Parse the file into a DOM document.
- EXTERNDOM OSRTDOMError domGetRootElement (OSRTDOMDocPtr xmlDoc, OSRTDOMNodePtr *pNode)

  Returns root node for the document.
- EXTERNDOM OSRTDOMError domGetDoc (OSRTDOMNodePtr node, OSRTDOMDocPtr *pXmlDoc)

  Returns the document for the given element.
- EXTERNDOM void domFreeDoc (OSRTDOMDocPtr xmlDoc)

  Frees document.
- EXTERNDOM OSRTDOMError domGetNext (const OSRTDOMNodePtr curNode, OSRTDOMNodePtr *pNext↩ Node)

  Returns next element node (down sibling).
- EXTERNDOM OSRTDOMError domGetChild (const OSRTDOMNodePtr curNode, OSRTDOMNodePtr *p↩ ChildNode)

  Returns first (top) child node.
- EXTERNDOM OSRTDOMError domGetElementName (const OSRTDOMNodePtr curNode, const OSUTF8C↩ HAR **ppName, const OSUTF8CHAR **ppNsPrefix, const OSUTF8CHAR **ppNsUri)

  Returns node's name and prefix (if any).
- EXTERNDOM int domGetNodeContent (OSRTDOMContCtxtPtr *ppCtxt, const OSRTDOMNodePtr curNode, const OSUTF8CHAR **ppValue, size_t *pValueLen, OSBOOL *pCdataProcessed)

  Gets the node's content.
- EXTERNDOM OSRTDOMError domGetNodeFirstAttribute (OSRTDOMAttrCtxtPtr *ppCtxt, const OSRTDOM↩ NodePtr curNode, OSRTDOMAttrPtr *pTopAttrNode)

  Returns the pointer to the first (top) attribute of the node.
- EXTERNDOM int domGetNodeAttributesNum (const OSRTDOMNodePtr curNode)

  Returns number of attributes in the node.
- EXTERNDOM OSRTDOMError domGetNextAttr (OSRTDOMAttrCtxtPtr *ppCtxt, const OSRTDOMAttrPtr cur↩ AttrNode, OSRTDOMAttrPtr *pAttrNode)

  Returns the next attribute.
- EXTERNDOM OSRTDOMError domGetAttrData (const OSRTDOMAttrPtr curAttrNode, const OSUTF8CHAR **ppAttrName, const OSUTF8CHAR **ppAttrValue, const OSUTF8CHAR **ppAttrPrefix, const OSUTF8CHAR **ppAttrUri)

  Returns the attribute's name and value.
- EXTERNDOM OSRTDOMError domSaveDoc (OSRTDOMDocPtr xmlDoc, const char *filename)

  Saves DOM document to a file.

- EXTERNDOM OSRTDOMError domCreateDocument (OSCTXT *pctxt, OSRTDOMDocPtr *pXmlDoc, const O↩ SUTF8CHAR *pNodeName, OSXMLNamespace *pNS, OSRTDList *pNSAttrs)

  Creates a new document and a root element.
- EXTERNDOM OSRTDOMError domCreateChild (OSCTXT *pctxt, OSRTDOMNodePtr parentNode, const OS↩ UTF8CHAR *pNodeName, OSXMLNamespace *pNS, OSRTDList *pNSAttrs, OSRTDOMNodePtr *pNewNode)

  Creates new child node.
- EXTERNDOM OSRTDOMError domAddAttribute (OSCTXT *pctxt, OSRTDOMNodePtr node, const OSUTF8↩ CHAR *pAttrName, const OSUTF8CHAR *pAttrValue, OSXMLNamespace *pNS, OSRTDList *pNSAttrs)

  Creates new attribute and adds it to the node.
- EXTERNDOM OSRTDOMError domAddContent (OSRTDOMNodePtr node, const OSUTF8CHAR *pContent, size_t contentLen)

  Adds content (text) to the node.
- EXTERNDOM OSRTDOMError domAddCdata (OSRTDOMNodePtr node, const OSUTF8CHAR *pContent, size_t contentLen)

  Adds CDATA section to the node.

6.6.1 Detailed Description

6.6.2 Function Documentation

6.6.2.1 domAddAttribute()

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

Creates new attribute and adds it to the node.

Parameters

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

Returns

0 - success, otherwise error code.

6.6.2.2 domAddCdata()

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

Adds CDATA section to the node.

Parameters

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

Returns

0 - success, otherwise error code.

6.6.2.3 domAddContent()

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

Adds content (text) to the node.

Parameters

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

Returns

0 - success, otherwise error code.

6.6.2.4 domCreateChild()

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

Creates new child node.

Parameters

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

Returns

0 - success, otherwise error code.

6.6.2.5 domCreateDocument()

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

Creates a new document and a root element.

Parameters

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

Returns

0 - success, otherwise error code.

6.6.2.6 domFreeDoc()

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

Frees document.

Parameters

xmlDoc	Pointer to the DOM document context

6.6.2.7 domGetAttrData()

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

Returns the attribute's name and value.

Parameters

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

Returns

0 - success, otherwise error code.

6.6.2.8 domGetChild()

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

Returns first (top) child node.

Parameters

curNode	Pointer to current node,
pChildNode	Pointer to pointer to receive the child node.

Returns

0 - success, otherwise error code.

6.6.2.9 domGetDoc()

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

Returns the document for the given element.

Parameters

node	Pointer to a node.
pXmlDoc	Pointer to a pointer to the DOM document context

Returns

0 - success, otherwise error code.

6.6.2.10 domGetElementName()

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

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

Parameters

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

Returns

0 - success, otherwise error code.

6.6.2.11 domGetNext()

```
EXTERNDOM OSRTDOMError domGetNext (
             const OSRTDOMNodePtr curNode,
             OSRTDOMNodePtr * pNextNode )
```

Returns next element node (down sibling).

Parameters

curNode	Pointer to current node,
pNextNode	Pointer to pointer to receive the next node.

Returns

0 - success, otherwise error code.

6.6.2.12 domGetNextAttr()

```
EXTERNDOM OSRTDOMError domGetNextAttr (
             OSRTDOMAttrCtxtPtr * ppCtxt,
             const OSRTDOMAttrPtr curAttrNode,
             OSRTDOMAttrPtr * pAttrNode )
```

Returns the next attribute.

Parameters

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

Returns

0 - success, otherwise error code.

6.6.2.13 domGetNodeAttributesNum()

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

Returns number of attributes in the node.

Parameters

curNode	Pointer to current node,

Returns

Number of attributes in the node.

6.6.2.14 domGetNodeContent()

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

Gets the node's content.

Parameters

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

Returns

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

6.6.2.15 domGetNodeFirstAttribute()

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

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

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

Parameters

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

Returns

0 - success, otherwise error code.

6.6.2.16 domGetRootElement()

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

Returns root node for the document.

Parameters

xmlDoc	Pointer to the DOM document context
pNode	Pointer to pointer to receive the root node.

Returns

0 - success, otherwise error code.

6.6.2.17 domParseFile()

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

Parse the file into a DOM document.

Parameters

fileSpec	File name
pXmlDoc	Pointer to a pointer to newly created DOM document.

Returns

0 - success, otherwise error code.

6.6.2.18 domSaveDoc()

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

Saves DOM document to a file.

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

Parameters

xmlDoc	Pointer to the DOM document context.
filename	Pointer to file name.

Returns

0 - success, otherwise error code.

6.7 DOM runtime encode/decode functions.

Functions

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

  Adds attribute to the node.
- EXTERNDOM int rtDomAddNode (OSCTXT *pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr parent↩ Node, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS, OSRTDList *pNSAttrs)

  Adds child node to the parent's node.
- EXTERNDOM int rtDomAddNSAttrs (OSCTXT *pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr root↩ Node, OSRTDList *pNSAttrs)

  Adds namespace attributes to the root node.
- EXTERNDOM int rtDomEncXSIAttrs (OSCTXT *pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr node, OSBOOL needXSI)

  Adds XSI attributes to the node.
- EXTERNDOM int rtDomDecodeDoc (OSRTDOMDocPtr domDoc, struct OSSAXHandlerBase *pSaxBase)

  This function starts decoding of the DOM document.
- EXTERNDOM int rtDomEncStringValue (OSCTXT *pctxt, const OSUTF8CHAR *pvalue)

  This function encodes a variable of the XSD string type.
- EXTERNDOM int rtDomEncString (OSCTXT *pctxt, OSXMLSTRING *pvalue)

  This function encodes a variable of the XSD string type.
- EXTERNDOM int rtDomEncAny (OSCTXT *pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr parentNode, const OSXMLSTRING *pXmlData, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS, OSRTDList *pNSAttrs)

  This function encodes a variable of the XSD any type.
- EXTERNDOM int rtDomEncAnyAttr (OSCTXT *pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr node, OSRTDList *pAnyAttrList)

  This function encodes a list of OSAnyAttr attributes in which the name and value are given as a UTF-8 string.
- EXTERNDOM int rtDomSetNode (OSCTXT *pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr curNode)

  Adds content or CDATA section to the node.
- EXTERNDOM int rtDomAddSubTree (OSCTXT *pctxt, OSRTDOMDocPtr xmlDoc, OSRTDOMNodePtr node, const OSUTF8CHAR *pXmlData, size_t dataLen)

  This function adds the subtree to the specified node.

6.7.1 Detailed Description

6.7.2 Function Documentation

6.7.2.1 rtDomAddAttr()

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

Adds attribute to the node.

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

Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
domDoc	A pointer to a DOM-document.
node	A pointer to a node where attribute to be added.
attrName	A pointer to attribute's name.
pNS	XML namespace information (prefix and URI).
pNSAttrs	List of namespace attributes to be added to element.

Returns

Completion status of operation:

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

6.7.2.2 rtDomAddNode()

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

Adds child node to the parent's node.

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

Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
domDoc	A pointer to a DOM-document.
parentNode	A pointer to a parent node.
elemName	A pointer to element's name.
pNS	XML namespace information (prefix and URI).
pNSAttrs	List of namespace attributes to be added to element.

Returns

Completion status of operation:

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

6.7.2.3 rtDomAddNSAttrs()

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

Adds namespace attributes to the root node.

Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
domDoc	A pointer to a DOM-document.
rootNode	A pointer to a root node.
pNSAttrs	A pointer to a list of namespace attrs.

Returns

Completion status of operation:

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

6.7.2.4 rtDomAddSubTree()

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

This function adds the subtree to the specified node.

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

Parameters

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

Returns

Completion status of operation:

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

6.7.2.5 rtDomDecodeDoc()

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

This function starts decoding of the DOM document.

Parameters

domDoc	A pointer to a DOM-document.
pSaxBase	A pointer to SAX handler's base.

Returns

Completion status of operation:

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

6.7.2.6 rtDomEncAny()

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

This function encodes a variable of the XSD any type.

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

Parameters

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

Returns

Completion status of operation:

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

6.7.2.7 rtDomEncAnyAttr()

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

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

Parameters

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

Returns

Completion status of operation:

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

6.7.2.8 rtDomEncString()

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

This function encodes a variable of the XSD string type.

Parameters

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

Returns

Completion status of operation:

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

6.7.2.9 rtDomEncStringValue()

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

This function encodes a variable of the XSD string type.

Parameters

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

Returns

Completion status of operation:

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

6.7.2.10 rtDomEncXSIAttrs()

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

Adds XSI attributes to the node.

Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
domDoc	A pointer to a DOM-document.
node	A pointer to a node.
needXSI	Determines whether XSI namespace declaration needed.

Returns

Completion status of operation:

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

6.7.2.11 rtDomSetNode()

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

```
OSRTDOMDocPtr domDoc,
OSRTDOMNodePtr curNode )
```

Adds content or CDATA section to the node.

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

Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
domDoc	A pointer to a DOM-document.
curNode	A pointer to a current node.

Returns

Completion status of operation:

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

Chapter 7

Class Documentation

7.1 OSXMLGroupDesc Struct Reference

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

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

7.1.1 Detailed Description

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

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

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

Definition at line 140 of file osrtxml.h.

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

- osrtxml.h

7.2 OSXMLStringListParser Class Reference

Class enabling parsing of an XML Schema list into strings.

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

Public Member Functions

- OSXMLStringListParser (OSCTXT *pctxt)

  Create a parser on the given context.
- int next (OSRTXMLString &value)

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

7.2.1 Detailed Description

Class enabling parsing of an XML Schema list into strings.

Definition at line 47 of file rtXmlpCppDecFuncs.h.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 OSXMLStringListParser()

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

Create a parser on the given context.

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

Parameters

pctxt	Holds state information and provides access to the input that is to be parsed.

7.2.3 Member Function Documentation

7.2.3.1 next()

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

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

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

Parameters

value	Receives the next string value or empty string if there isn't one.

Returns

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

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

- rtXmlpCppDecFuncs.h

Chapter 8

File Documentation

8.1 osrtdom.h File Reference

DOM low-level C encode/decode functions.

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

Functions

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

  Adds attribute to the node.
- EXTERNDOM int rtDomAddNode (OSCTXT *pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr parent↩ Node, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS, OSRTDList *pNSAttrs)

  Adds child node to the parent's node.
- EXTERNDOM int rtDomAddNSAttrs (OSCTXT *pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr root↩ Node, OSRTDList *pNSAttrs)

  Adds namespace attributes to the root node.
- EXTERNDOM int rtDomEncXSIAttrs (OSCTXT *pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr node, OSBOOL needXSI)

  Adds XSI attributes to the node.
- EXTERNDOM int rtDomDecodeDoc (OSRTDOMDocPtr domDoc, struct OSSAXHandlerBase *pSaxBase)

  This function starts decoding of the DOM document.
- EXTERNDOM int rtDomEncStringValue (OSCTXT *pctxt, const OSUTF8CHAR *pvalue)

  This function encodes a variable of the XSD string type.
- EXTERNDOM int rtDomEncString (OSCTXT *pctxt, OSXMLSTRING *pvalue)

  This function encodes a variable of the XSD string type.

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

  This function encodes a variable of the XSD any type.
- EXTERNDOM int rtDomEncAnyAttr (OSCTXT *pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr node, OSRTDList *pAnyAttrList)

  This function encodes a list of OSAnyAttr attributes in which the name and value are given as a UTF-8 string.
- EXTERNDOM int rtDomSetNode (OSCTXT *pctxt, OSRTDOMDocPtr domDoc, OSRTDOMNodePtr curNode)

  Adds content or CDATA section to the node.
- EXTERNDOM int rtDomAddSubTree (OSCTXT *pctxt, OSRTDOMDocPtr xmlDoc, OSRTDOMNodePtr node, const OSUTF8CHAR *pXmlData, size_t dataLen)

  This function adds the subtree to the specified node.

8.1.1 Detailed Description

DOM low-level C encode/decode functions.

8.2 osrtxml.h File Reference

XML low-level C encode/decode functions.

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

Classes

- struct OSXMLGroupDesc

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

Macros

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

  This macro returns the start address of the encoded XML message.
- #define rtXmlGetEncBufLen(pctxt) (pctxt)->buffer.byteIndex

  This macro returns the length of the encoded XML message.

Typedefs

- typedef struct OSXMLGroupDesc OSXMLGroupDesc

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

Enumerations

- enum OSXMLWhiteSpaceMode

  Whitespace treatment options.

Functions

- int rtXmlInitContext (OSCTXT *pctxt)

  This function initializes a context variable for XML encoding or decoding.
- int rtXmlInitContextUsingKey (OSCTXT *pctxt, const OSOCTET *key, OSSIZE keylen)

  This function initializes a context using a run-time key.
- int rtXmlInitCtxtAppInfo (OSCTXT *pctxt)

  This function initializes the XML application info section of the given context.
- int rtXmlCreateFileInputSource (OSCTXT *pctxt, const char *filepath)

  This function creates an XML document file input source.
- void rtXmlMemFreeAnyAttrs (OSCTXT *pctxt, OSRTDList *pAnyAttrList)

  This function frees a list of anyAttribute that is a member of OSXSDAnyType structure.
- int rtXmlDecBase64Binary (OSRTMEMBUF *pMemBuf, const OSUTF8CHAR *inpdata, OSSIZE length)

  This function decodes the contents of a Base64-encoded binary data type into a memory buffer.
- int rtXmlDecBase64Str (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSINT32 bufsize)

  This function decodes a contents of a Base64-encode binary string into a static memory structure.
- int rtXmlDecBase64Str64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnocts, OSSIZE bufsize)

  This function is identical to rtXmlDecBase64Str except that is supports a 64-bit integer length on 64-bit systems.
- int rtXmlDecBase64StrValue (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSSIZE bufSize, OS↩ SIZE srcDataLen)

  This function decodes a contents of a Base64-encode binary string into the specified octet array.
- int rtXmlDecBase64StrValue64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnocts, OSSIZE bufSize, OS↩ SIZE srcDataLen)

  This function decodes is identical to rtXmlDecBase64StrValue except that it supports a 64-bit integer length on 64-bit systems.
- int rtXmlDecBigInt (OSCTXT *pctxt, const OSUTF8CHAR **ppvalue)

  This function will decode a variable of the XSD integer type.
- int rtXmlDecBool (OSCTXT *pctxt, OSBOOL *pvalue)

  This function decodes a variable of the boolean type.
- int rtXmlDecDate (OSCTXT *pctxt, OSXSDDateTime *pvalue)

  This function decodes a variable of the XSD 'date' type.
- int rtXmlDecTime (OSCTXT *pctxt, OSXSDDateTime *pvalue)

  This function decodes a variable of the XSD 'time' type.
- int rtXmlDecDateTime (OSCTXT *pctxt, OSXSDDateTime *pvalue)

  This function decodes a variable of the XSD 'dateTime' type.
- int rtXmlDecDecimal (OSCTXT *pctxt, OSREAL *pvalue)

This function decodes the contents of a decimal data type.

- int rtXmlDecDouble (OSCTXT *pctxt, OSREAL *pvalue)

  This function decodes the contents of a float or double data type.
- int rtXmlDecDynBase64Str (OSCTXT *pctxt, OSDynOctStr *pvalue)

  This function decodes a contents of a Base64-encode binary string.
- int rtXmlDecDynBase64Str64 (OSCTXT *pctxt, OSDynOctStr64 *pvalue)

  This function is identical to rtXmlDecDynBase64Str except that it supports a 64-bit integer length on 64-bit systems.
- int rtXmlDecDynHexStr (OSCTXT *pctxt, OSDynOctStr *pvalue)

  This function decodes a contents of a hexBinary string.
- int rtXmlDecDynHexStr64 (OSCTXT *pctxt, OSDynOctStr64 *pvalue)

  This function is identical to rtXmlDecDynHexStr except that it supports a 64-bit integer length on 64-bit systems.
- int rtXmlDecEmptyElement (OSCTXT *pctxt)

  This function is used to enforce a requirement that an element be empty.
- int rtXmlDecUTF8Str (OSCTXT *pctxt, OSUTF8CHAR *outdata, OSSIZE max_len)

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

  This function decodes the contents of a UTF-8 string data type.
- int rtXmlDecHexBinary (OSRTMEMBUF *pMemBuf, const OSUTF8CHAR *inpdata, OSSIZE length)

  This function decodes the contents of a hex-encoded binary data type into a memory buffer.
- int rtXmlDecHexStr (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSINT32 bufsize)

  This function decodes the contents of a hexBinary string into a static memory structure.
- int rtXmlDecHexStr64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnocts, OSSIZE bufsize)

  This function is identical to rtXmlDecHexStr except that it supports a 64-bit integer length on 64-bit systems.
- int rtXmlDecGYear (OSCTXT *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, OSR↩ TDList *pNSAttrs, const OSUTF8CHAR *nsTable[ ], OSUINT32 nsTableRowCount)

  This function decodes an XML namespac attribute (xmlns).
- const OSUTF8CHAR * rtXmlDecQName (OSCTXT *pctxt, const OSUTF8CHAR *qname, const OSUTF8CHAR **prefix)

  This function decodes an XML qualified name string (QName) type.
- int rtXmlDecXSIAttr (OSCTXT *pctxt, const OSUTF8CHAR *attrName, const OSUTF8CHAR *attrValue)

  This function decodes XML schema instance (XSI) attribute.
- int rtXmlDecXSIAttrs (OSCTXT *pctxt, const OSUTF8CHAR *const *attrs, const char *typeName)

  This function decodes XML schema instance (XSI) attributes.
- int rtXmlDecXmlStr (OSCTXT *pctxt, OSXMLSTRING *outdata)

  This function decodes the contents of an XML string data type.
- int rtXmlParseElementName (OSCTXT *pctxt, OSUTF8CHAR **ppName)

  This function parses the initial tag from an XML message.
- int rtXmlParseElemQName (OSCTXT *pctxt, OSXMLQName *pQName)

  This function parses the initial tag from an XML message.
- int rtXmlEncAny (OSCTXT *pctxt, OSXMLSTRING *pvalue, const OSUTF8CHAR *elemName, OSXML↩ Namespace *pNS)

  This function encodes a variable of the XSD any type.
- int rtXmlEncAnyTypeValue (OSCTXT *pctxt, const OSUTF8CHAR *pvalue)

  This function encodes a variable of the XSD anyType type.
- int rtXmlEncAnyAttr (OSCTXT *pctxt, OSRTDList *pAnyAttrList)

  This function encodes a list of OSAnyAttr attributes in which the name and value are given as a UTF-8 string.
- int rtXmlEncBase64Binary (OSCTXT *pctxt, OSSIZE nocts, const OSOCTET *value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

  This function encodes a variable of the XSD base64Binary type.
- int rtXmlEncBase64BinaryAttr (OSCTXT *pctxt, OSUINT32 nocts, const OSOCTET *value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

  This function encodes a variable of the XSD base64Binary type as an attribute.
- int rtXmlEncBase64StrValue (OSCTXT *pctxt, OSSIZE nocts, const OSOCTET *value)

  This function encodes a variable of the XSD base64Binary type.
- int rtXmlEncBigInt (OSCTXT *pctxt, const OSUTF8CHAR *value, const OSUTF8CHAR *elemName, OSXML↩ Namespace *pNS)

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

  This function encodes an XSD integer attribute value.
- int rtXmlEncBigIntValue (OSCTXT *pctxt, const OSUTF8CHAR *value)

  This function encodes an XSD integer attribute value.
- int rtXmlEncBitString (OSCTXT *pctxt, OSSIZE nbits, const OSOCTET *value, const OSUTF8CHAR *elem↩ Name, OSXMLNamespace *pNS)

  This function encodes a variable of the ASN.1 BIT STRING type.
- int rtXmlEncBitStringExt (OSCTXT *pctxt, OSSIZE nbits, const OSOCTET *value, OSSIZE dataSize, const O↩ SOCTET *extValue, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS)

  This function encodes a variable of the ASN.1 BIT STRING type.
- int rtXmlEncBinStrValue (OSCTXT *pctxt, OSSIZE nbits, const OSOCTET *data)

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

- int rtXmlEncBool (OSCTXT *pctxt, OSBOOL value, const OSUTF8CHAR *elemName, OSXMLNamespace *p↩ NS)

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

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

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

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

  This function encodes an XML comment.
- int rtXmlEncDate (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, OSX↩ MLNamespace *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, OSX↩ MLNamespace *pNS)

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

  This function encodes a variable of the XSD 'time' type as an string.
- int rtXmlEncDateTime (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, O↩ SXMLNamespace *pNS)

  This function encodes a numeric date/time value into an XML string representation.
- int rtXmlEncDateTimeValue (OSCTXT *pctxt, const OSXSDDateTime *pvalue)

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

  This function encodes a variable of the XSD decimal type.
- int rtXmlEncDecimalAttr (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *attrName, OSSIZE attrName↩ Len, const OSDecimalFmt *pFmtSpec)

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

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

  This function encodes a variable of the XSD double type.
- int rtXmlEncDoubleAttr (OSCTXT *pctxt, OSREAL value, const OSUTF8CHAR *attrName, OSSIZE attrName↩ Len, 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 default↩ Precision)

  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, O↩ SRTDList *pNSAttrs, OSBOOL terminate)

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

- int rtXmlEncEndDocument (OSCTXT *pctxt)

  This function adds trailor 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 *p↩NS, const OSDoubleFmt *pFmtSpec)

  This function encodes a variable of the XSD float type.

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

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

- int rtXmlEncGYear (OSCTXT *pctxt, const OSXSDDateTime *pvalue, const OSUTF8CHAR *elemName, OS↩XMLNamespace *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, O↩SXMLNamespace *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, OSX↩MLNamespace *pNS)

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

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

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

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

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

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

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

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

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

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

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

- int rtXmlEncHexBinary (OSCTXT *pctxt, OSSIZE nocts, const OSOCTET *value, const OSUTF8CHAR *elem↩Name, OSXMLNamespace *pNS)

  This function encodes a variable of the XSD hexBinary type.

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

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

- int rtXmlEncHexStrValue (OSCTXT *pctxt, OSSIZE nocts, const OSOCTET *data)

  This function encodes a variable of the XSD hexBinary type.

- int rtXmlEncIndent (OSCTXT *pctxt)

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

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

  This function encodes a variable of the XSD integer type.
- int rtXmlEncIntAttr (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

  This function encodes a variable of the XSD integer type as an attribute (name="value").
- int rtXmlEncIntPattern (OSCTXT *pctxt, OSINT32 value, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS, const OSUTF8CHAR *pattern)

  This function encodes a variable of the XSD integer type using a pattern to specify the format of the integer value.
- int rtXmlEncInt64 (OSCTXT *pctxt, OSINT64 value, const OSUTF8CHAR *elemName, OSXMLNamespace *p↩NS)

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

  This function encodes a variable of the XSD integer type.
- int rtXmlEncInt64Attr (OSCTXT *pctxt, OSINT64 value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

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

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

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

  This function prints a list of namespace attributes.
- int rtXmlEncReal10 (OSCTXT *pctxt, const OSUTF8CHAR *pvalue, const OSUTF8CHAR *elemName, OSX↩MLNamespace *pNS)

  This function encodes a variable of the ASN.1 REAL base 10 type.
- int rtXmlEncSoapArrayTypeAttr (OSCTXT *pctxt, const OSUTF8CHAR *name, const OSUTF8CHAR *value, OSSIZE itemCount)

  This function encodes the special SOAP encoding attrType attribute which specifies the number and type of elements in a SOAP array.
- int rtXmlEncStartDocument (OSCTXT *pctxt)

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

  This function encodes the Unicode byte order mark header at the start of the document.
- int rtXmlEncStartElement (OSCTXT *pctxt, const OSUTF8CHAR *elemName, OSXMLNamespace *pNS, OS↩RTDList *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, OSXML↩Namespace *pNS)

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

  This function encodes a variable of the XSD string type.

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

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

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

  This function encodes a Unicode string value.
- int rtXmlEncUTF8Attr (OSCTXT *pctxt, const OSUTF8CHAR *name, const OSUTF8CHAR *value)

  This function encodes an attribute in which the name and value are given as a null-terminated UTF-8 strings.
- int rtXmlEncUTF8Attr2 (OSCTXT *pctxt, const OSUTF8CHAR *name, OSSIZE nameLen, const OSUTF8CHAR *value, OSSIZE valueLen)

  This function encodes an attribute in which the name and value are given as a UTF-8 strings with lengths.
- int rtXmlEncUTF8Str (OSCTXT *pctxt, const OSUTF8CHAR *value, const OSUTF8CHAR *elemName, OSX↩ MLNamespace *pNS)

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

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

  This function encodes a variable of the XSD unsigned integer type.
- int rtXmlEncUIntAttr (OSCTXT *pctxt, OSUINT32 value, const OSUTF8CHAR *attrName, OSSIZE attrNameLen)

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

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

  This function encodes a variable of the XSD integer type.
- int rtXmlEncUInt64Attr (OSCTXT *pctxt, OSUINT64 value, const OSUTF8CHAR *attrName, OSSIZE attrName↩ Len)

  This function encodes a variable of the XSD integer type as an attribute (name="value").
- int rtXmlEncXSIAttrs (OSCTXT *pctxt, OSBOOL needXSI)

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

  This function encodes an XML schema instance (XSI) type attribute value (xsi:type="value").
- int rtXmlEncXSITypeAttr2 (OSCTXT *pctxt, const OSUTF8CHAR *typeNsUri, const OSUTF8CHAR *typeName)

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

  This function encodes an XML nil attribute (xsi:nil="true").
- int rtXmlFreeInputSource (OSCTXT *pctxt)

  This function closes an input source that was previously created with one of the create input source functions such as 'rtXmlCreateFileInputSource'.
- int rtXmlSetEncBufPtr (OSCTXT *pctxt, OSOCTET *bufaddr, OSSIZE bufsiz)

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

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

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

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

- int rtXmlPrepareContext (OSCTXT *pctxt)

  This function prepares the context for another encode by setting the state back to OSXMLINIT and moving the buffer's cursor back to the beginning of the buffer.
- int rtXmlSetEncC14N (OSCTXT *pctxt, OSBOOL value)

  This function sets the option to encode in C14N mode.
- int rtXmlSetEncXSINamespace (OSCTXT *pctxt, OSBOOL value)

  This function sets a flag in the context that indicates the XSI namespace declaration (xmlns:xsi) should be added to the encoded XML instance.
- int rtXmlSetEncXSINilAttr (OSCTXT *pctxt, OSBOOL value)

  This function sets a flag in the context that indicates the XSI attribute declaration (xmlns:xsi) should be added to the encoded XML instance.
- int rtXmlSetEncDocHdr (OSCTXT *pctxt, OSBOOL value)

  This function sets the option to add the XML document header (i.e.
- int rtXmlSetEncodingStr (OSCTXT *pctxt, const OSUTF8CHAR *encodingStr)

  This function sets the XML output encoding to the given value.
- int rtXmlSetFormatting (OSCTXT *pctxt, OSBOOL doFormatting)

  This function sets XML output formatting to the given value.
- int rtXmlSetIndent (OSCTXT *pctxt, OSUINT8 indent)

  This function sets XML output indent to the given value.
- int rtXmlSetIndentChar (OSCTXT *pctxt, char indentChar)

  This function sets XML output indent character to the given value.
- void rtXmlSetNamespacesSet (OSCTXT *pctxt, OSBOOL value)

  This function sets the context 'namespaces are set' flag.
- int rtXmlSetNSPrefixLinks (OSCTXT *pctxt, OSRTDList *pNSAttrs)

  This function sets namespace prefix/URI links in the namspace prefix stack in the context structure.
- int rtXmlSetSchemaLocation (OSCTXT *pctxt, const OSUTF8CHAR *schemaLocation)

  This function sets the XML Schema Instance (xsi) schema location attribute to be added to an encoded document.
- int rtXmlSetNoNSSchemaLocation (OSCTXT *pctxt, const OSUTF8CHAR *schemaLocation)

  This function sets the XML Schema Instance (xsi) no namespace schema location attribute to be added to an encoded document.
- void rtXmlSetSoapVersion (OSCTXT *pctxt, OSUINT8 version)

  This function sets the SOAP version number.
- int rtXmlSetXSITypeAttr (OSCTXT *pctxt, const OSUTF8CHAR *xsiType)

  This function sets the XML Schema Instance (xsi) type attribute value.
- int rtXmlSetWriteBOM (OSCTXT *pctxt, OSBOOL write)

  This function sets whether the Unicode byte order mark is encoded.
- int rtXmlMatchHexStr (OSCTXT *pctxt, OSSIZE minLength, OSSIZE maxLength)

  This function tests the context buffer for containing a correct hexadecimal string.
- int rtXmlMatchBase64Str (OSCTXT *pctxt, OSSIZE minLength, OSSIZE maxLength)

  This function tests the context buffer for containing a correct base64 string.
- int rtXmlMatchDate (OSCTXT *pctxt)

  This function tests the context buffer for containing a correct date string.
- int rtXmlMatchTime (OSCTXT *pctxt)

  This function tests the context buffer for containing a correct time string.
- int rtXmlMatchDateTime (OSCTXT *pctxt)

  This function tests the context buffer for containing a correct dateTime string.
- int rtXmlMatchGYear (OSCTXT *pctxt)

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

- int rtXmlMatchGYearMonth (OSCTXT *pctxt)

  This function tests the context buffer for containing a correct gYearMonth string.

- int rtXmlMatchGMonth (OSCTXT *pctxt)

  This function tests the context buffer for containing a correct gMonth string.

- int rtXmlMatchGMonthDay (OSCTXT *pctxt)

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

- int rtXmlMatchGDay (OSCTXT *pctxt)

  This function tests the context buffer for containing a correct gDay string.

- OSUTF8CHAR * rtXmlNewQName (OSCTXT *pctxt, const OSUTF8CHAR *localName, const OSUTF8CHAR *prefix)

  This function creates a new QName given the localName and prefix parts.

- OSBOOL rtXmlCmpBase64Str (OSUINT32 nocts1, const OSOCTET *data1, const OSUTF8CHAR *data2)

  This function compares an array of octets to a base64 string.

- OSBOOL rtXmlCmpHexStr (OSUINT32 nocts1, const OSOCTET *data1, const OSUTF8CHAR *data2)

  This function compares an array of octets to a hex string.

- const OSUTF8CHAR * rtSaxGetAttrValue (const OSUTF8CHAR *attrName, const OSUTF8CHAR *const *attrs)

  This function looks up an attribute in the attribute array returned by SAX to the startElement function.

- OSINT16 rtSaxGetElemID (OSINT16 *pState, OSINT16 prevElemIdx, const OSUTF8CHAR *localName, OSINT32 nsidx, const OSSAXElemTableRec idtab[ ], const OSINT16 *fstab, OSINT16 fstabRows, OSINT16 fstabCols)

  This function looks up a sequence element name in the given element info array.

- OSINT16 rtSaxGetElemID8 (OSINT16 *pState, OSINT16 prevElemIdx, const OSUTF8CHAR *localName, OSINT32 nsidx, const OSSAXElemTableRec idtab[ ], const OSINT8 *fstab, OSINT16 fstabRows, OSINT16 fstabCols)

  This function is a space optimized version of `rtSaxGetElemID`.

- OSBOOL rtSaxHasXMLNSAttrs (const OSUTF8CHAR *const *attrs)

  This function checks if the given attribute list contains one or more XML namespace attributes (xmlns).

- OSBOOL rtSaxIsEmptyBuffer (OSCTXT *pctxt)

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

- int rtSaxStrListParse (OSCTXT *pctxt, OSRTMEMBUF *pMemBuf, OSRTDList *pvalue)

  This function parses the list of strings.

- int rtSaxSortAttrs (OSCTXT *pctxt, const OSUTF8CHAR *const *attrs, OSUINT16 **order)

  This function sorts a SAX attribute list in ascending order based on attribute name.

- int rtSaxStrListMatch (OSCTXT *pctxt)

  This function mathes the list of strings.

- int rtXmlWriteToFile (OSCTXT *pctxt, const char *filename)

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

- int rtXmlpDecAny (OSCTXT *pctxt, const OSUTF8CHAR **pvalue)

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

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

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

- int rtXmlpDecAnyAttrStr (OSCTXT *pctxt, const OSUTF8CHAR **ppAttrStr, OSSIZE attrIndex)

  This function decodes an any attribute string.

- int rtXmlpDecAnyElem (OSCTXT *pctxt, const OSUTF8CHAR **pvalue)

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

- int rtXmlpDecBase64Str (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnocts, OSSIZE bufsize)

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

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

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

- int rtXmlpDecBigInt (OSCTXT *pctxt, const OSUTF8CHAR **pvalue)

  This function will decode a variable of the XSD integer type.
- int rtXmlpDecBitString (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnbits, OSUINT32 bufsize)

  This function decodes a bit string value.
- int rtXmlpDecBitString64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnbits, OSSIZE bufsize)

  This function is identical to rtXmlpDecBitString except that it supports lengths up to 64-bits in size on 64-bit machines.
- int rtXmlpDecBitStringExt (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 *pnbits, OSOCTET **ppextdata, OSUINT32 bufsize)

  This function decodes a bit string value.
- int rtXmlpDecBitStringExt64 (OSCTXT *pctxt, OSOCTET *pvalue, OSSIZE *pnbits, OSOCTET **ppextdata, OSSIZE bufsize)

  This function is identical to rtXmlpDecBitStringExt except that it supports lengths up to 64-bits in size on 64-bit machines.
- int rtXmlpDecBool (OSCTXT *pctxt, OSBOOL *pvalue)

  This function decodes a variable of the boolean type.
- int rtXmlpDecDate (OSCTXT *pctxt, OSXSDDateTime *pvalue)

  This function decodes a variable of the XSD 'date' type.
- int rtXmlpDecDateTime (OSCTXT *pctxt, OSXSDDateTime *pvalue)

  This function decodes a variable of the XSD 'dateTime' type.
- int rtXmlpDecDecimal (OSCTXT *pctxt, OSREAL *pvalue, int totalDigits, int fractionDigits)

  This function decodes the contents of a decimal data type.
- int rtXmlpDecDouble (OSCTXT *pctxt, OSREAL *pvalue)

  This function decodes the contents of a float or double data type.
- int rtXmlpDecDoubleExt (OSCTXT *pctxt, OSUINT8 flags, OSREAL *pvalue)

  This function decodes the contents of a float or double data type.
- int rtXmlpDecDynBase64Str (OSCTXT *pctxt, OSDynOctStr *pvalue)

  This function decodes a contents of a Base64-encode binary string.
- int rtXmlpDecDynBase64Str64 (OSCTXT *pctxt, OSDynOctStr64 *pvalue)

  This function is identical to rtXmlpDecDynBase64Str except that it supports 64-bit integer lengths on 64-bit systems.
- int rtXmlpDecDynBitString (OSCTXT *pctxt, OSDynOctStr *pvalue)

  This function decodes a bit string value.
- int rtXmlpDecDynHexStr (OSCTXT *pctxt, OSDynOctStr *pvalue)

  This function decodes a contents of a hexBinary string.
- int rtXmlpDecDynHexStr64 (OSCTXT *pctxt, OSDynOctStr64 *pvalue)

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

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

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

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

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

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

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

- int rtXmlpDecGYear (OSCTXT *pctxt, OSXSDDateTime *pvalue)

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

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

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

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

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

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

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

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

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

  This function decodes the contents of a named bit field.
- int rtXmlpDecStrList (OSCTXT *pctxt, OSRTDList *plist)

  This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.
- int rtXmlpDecTime (OSCTXT *pctxt, OSXSDDateTime *pvalue)

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

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

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

  This function decodes the contents of an unsigned 16-bit integer data type.
- int rtXmlpDecUInt64 (OSCTXT *pctxt, OSUINT64 *pvalue)

  This function decodes the contents of an unsigned 64-bit integer data type.
- int rtXmlpDecXmlStr (OSCTXT *pctxt, OSXMLSTRING *outdata)

  This function decodes the contents of an XML string data type.
- int rtXmlpDecXmlStrList (OSCTXT *pctxt, OSRTDList *plist)

  This function decodes a list of space-separated tokens and returns each token as a separate item on the given list.
- int rtXmlpDecXSIAttr (OSCTXT *pctxt, const OSXMLNameFragments *attrName)

  This function decodes XSI (XML Schema Instance) attributes that may be present in any arbitrary XML element within a document.
- int rtXmlpDecXSITypeAttr (OSCTXT *pctxt, const OSXMLNameFragments *attrName, const OSUTF8CHAR **ppAttrValue)

  This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).
- int rtXmlpGetAttributeID (const OSXMLStrFragment *attrName, OSINT16 nsidx, OSSIZE nAttr, const OSXML↩ AttrDescr attrNames[ ], OSUINT32 attrPresent[ ])

  This function finds an attribute in the descriptor table.
- int rtXmlpGetNextElem (OSCTXT *pctxt, OSXMLElemDescr *pElem, OSINT32 level)

This function parse the next element start tag.

- int rtXmlpGetNextElemID (OSCTXT *pctxt, const OSXMLElemIDRec *tab, OSSIZE nrows, OSINT32 level, O↩ SBOOL continueParse)

  This function parses the next start tag and finds the index of the element name in the descriptor table.
- int 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, OSSIZE 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 *p↩ Group, 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 *p↩ Group, 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 group↩ Mode)

  This is an ASN.1 extension-supporting version of rtXmlpGetNextSeqElemID.
- int rtXmlpGetNextAllElemID (OSCTXT *pctxt, const OSXMLElemIDRec *tab, OSSIZE nrows, const OSUINT8 *pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)

  This function parses the next start tag and finds index of element name in descriptor table.
- int rtXmlpGetNextAllElemID16 (OSCTXT *pctxt, const OSXMLElemIDRec *tab, OSSIZE nrows, const OSUINT16 *pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)

  This function parses the next start tag and finds index of element name in descriptor table.
- int rtXmlpGetNextAllElemID32 (OSCTXT *pctxt, const OSXMLElemIDRec *tab, OSSIZE nrows, const OSUINT32 *pOrder, OSSIZE nOrder, OSSIZE maxOrder, int anyID)

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

- void rtXmlpSetNamespaceTable (OSCTXT *pctxt, const OSUTF8CHAR *namespaceTable[], OSSIZE nm↩ Namespaces)

  Sets user namespace table.
- int rtXmlpCreateReader (OSCTXT *pctxt)

  Creates pull parser reader structure within the context.
- void rtXmlpHideAttributes (OSCTXT *pctxt)

  Disable access to attributes.
- OSBOOL rtXmlpNeedDecodeAttributes (OSCTXT *pctxt)

  This function checks if attributes were previously decoded.
- void rtXmlpMarkPos (OSCTXT *pctxt)

  Save current decode position.
- void rtXmlpRewindToMarkedPos (OSCTXT *pctxt)

  Rewind to saved decode position.
- void rtXmlpResetMarkedPos (OSCTXT *pctxt)

  Reset saved decode position.
- int rtXmlpGetXSITypeAttr (OSCTXT *pctxt, const OSUTF8CHAR **ppAttrValue, OSINT16 *nsidx, OSSIZE *p↩ LocalOffs)

  This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type).
- int rtXmlpGetXmlnsAttrs (OSCTXT *pctxt, OSRTDList *pNSAttrs)

  This function decodes namespace attributes from start tag and adds them to the given list.
- int rtXmlpDecXSIAttrs (OSCTXT *pctxt)

  This function decodes XSI (XML Schema Instance) that may be present in any arbitrary XML element within a document.
- OSBOOL 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[], OSSIZE typetabsiz)

  This function decodes the contents of an XSI (XML Schema Instance) type attribute (xsi:type) and find type index in descriptor table.
- int rtXmlpLookupXSITypeIndex (OSCTXT *pctxt, const OSUTF8CHAR *pXsiType, OSINT16 xsiTypeIdx, const OSXMLItemDescr typetab[], OSSIZE typetabsiz)

  This function find index of XSI (XML Schema Instance) type in descriptor table.
- void rtXmlpForceDecodeAsGroup (OSCTXT *pctxt)

  Disable skipping of unknown elements in optional sequence tail.
- OSBOOL 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, OSSIZE nbytes)

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

8.2.1 Detailed Description

XML low-level C encode/decode functions.

8.2.2 Typedef Documentation

8.2.2.1 OSXMLGroupDesc

```
typedef struct OSXMLGroupDesc OSXMLGroupDesc
```

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

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

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

8.2.3 Function Documentation

8.2.3.1 rtSaxGetAttrValue()

```
const OSUTF8CHAR* rtSaxGetAttrValue (
            const OSUTF8CHAR * attrName,
            const OSUTF8CHAR *const * attrs )
```

This function looks up an attribute in the attribute array returned by SAX to the startElement function.

Parameters

attrName	Name of the attribute to find.
attrs	Attribute array returned in SAX startElement function. This is an array of character strings containing name1, value1, name2, value2, ... List is terminated by a null name.

Returns

Pointer to character string containing attribute value or NULL if attrName not found.

8.2.3.2 rtSaxGetElemID()

```
OSINT16 rtSaxGetElemID (
            OSINT16 * pState,
```

```
            OSINT16 prevElemIdx,
            const OSUTF8CHAR * localName,
            OSINT32 nsidx,
            const OSSAXElemTableRec idtab[],
            const OSINT16 * fstab,
            OSINT16 fstabRows,
            OSINT16 fstabCols )
```

This function looks up a sequence element name in the given element info array.

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

Parameters

pState	The pointer to state variable to be changed.
prevElemIdx	Previous index of element. The search will be started from this element for better performance.
localName	Local name of XML element
nsidx	Namespace index
idtab	Element ID table
fstab	Finite state table
fstabRows	Number of rows in `fstab`.
fstabCols	Number of columns in `fstab`.

8.2.3.3 rtSaxGetElemID8()

```
OSINT16 rtSaxGetElemID8 (
            OSINT16 * pState,
            OSINT16 prevElemIdx,
            const OSUTF8CHAR * localName,
            OSINT32 nsidx,
            const OSSAXElemTableRec idtab[],
            const OSINT8 * fstab,
            OSINT16 fstabRows,
            OSINT16 fstabCols )
```

This function is a space optimized version of `rtSaxGetElemID`.

It operates with array of 8-bit integers (OSINT8) instead of 32-bit integers (int).

Parameters

pState	The pointer to state variable to be changed.
prevElemIdx	Previous index of element. The search will be started from this element + 1 for better performance.
localName	Local name of XML element
nsidx	Namespace index
idtab	Element ID table
fstab	Finite state table (array of 8-bit integers)
fstabRows	Number of rows in `fstab`.
fstabCols	Number of columns in `fstab`.

8.2.3.4 rtSaxHasXMLNSAttrs()

```
OSBOOL rtSaxHasXMLNSAttrs (
            const OSUTF8CHAR *const * attrs )
```

This function checks if the given attribute list contains one or more XML namespace attributes (xmlns).

Parameters

attrs	Attribute list in form passed by parser into SAX startElement function.

Returns

TRUE, if xmlns attribute found in list.

8.2.3.5 rtSaxIsEmptyBuffer()

```
OSBOOL rtSaxIsEmptyBuffer (
            OSCTXT * pctxt )
```

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

Parameters

pctxt	Pointer to OSCTXT structure

Returns

TRUE, if the buffer contains empty string.

8.2.3.6 rtSaxSortAttrs()

```
int rtSaxSortAttrs (
            OSCTXT * pctxt,
            const OSUTF8CHAR *const * attrs,
            OSUINT16 ** order )
```

This function sorts a SAX attribute list in ascending order based on attribute name.

It currently only supports unqualified attributes.

Parameters

pctxt	Pointer to OSCTXT structure.
attrs	Standard SAX attribute list. Entry i is attribute name and i+1 is value. List is terminated by a null name.
order	Order array containing the order of sorted attributes. This array is allocated using rtxMemAlloc, it can be freed using rtxMemFreePtr or will be freed when the context is freed. The list holds indicies to name items in the attribute list that is passed in.

Returns

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

8.2.3.7 rtSaxStrListMatch()

```
int rtSaxStrListMatch (
             OSCTXT * pctxt )
```

This function mathes the list of strings.

It is used for matching NMTOKENS, IDREFS, NMENTITIES.

Parameters

pctxt	Pointer to OSCTXT structure

Returns

0 - if success, negative value is error.

8.2.3.8 rtSaxStrListParse()

```
int rtSaxStrListParse (
             OSCTXT * pctxt,
             OSRTMEMBUF * pMemBuf,
             OSRTDList * pvalue )
```

This function parses the list of strings.

It is used for parsing NMTOKENS, IDREFS, NMENTITIES.

Parameters

pctxt	Pointer to OSCTXT structure. Can be NULL, if pMemBuf is not NULL.
pMemBuf	Pointer to memory buffer structure. Can be NULL, if pctxt is not NULL.
pvalue	Doubly-linked list for parsed strings.

Returns

0 - if success, negative value is error.

8.2.3.9 rtXmlCmpBase64Str()

```
OSBOOL rtXmlCmpBase64Str (
             OSUINT32 nocts1,
             const OSOCTET * data1,
             const OSUTF8CHAR * data2 )
```

This function compares an array of octets to a base64 string.

Parameters

nocts1	Number of octets in data1.
data1	Pointer to array of OSOCTET.
data2	Pointer to null-terminated array of OSUTF8CHAR.

Returns

TRUE if data2 is a base64 string representation of data1, false otherwise.

8.2.3.10 rtXmlCmpHexStr()

```
OSBOOL rtXmlCmpHexStr (
             OSUINT32 nocts1,
             const OSOCTET * data1,
             const OSUTF8CHAR * data2 )
```

This function compares an array of octets to a hex string.

Parameters

nocts1	Number of octets in data1.
data1	Pointer to array of OSOCTET.
data2	Pointer to null-terminated array of OSUTF8CHAR.

Returns

TRUE if data2 is a hex string representation of data1, false otherwise.

8.2.3.11 rtXmlCreateFileInputSource()

```
int rtXmlCreateFileInputSource (
            OSCTXT * pctxt,
            const char * filepath )
```

This function creates an XML document file input source.

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

Parameters

pctxt	Pointer to context block structure.
filepath	Full pathname of XML document file to open.

Returns

Completion status of operation:

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

8.2.3.12 rtXmlInitContext()

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

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

Parameters

pctxt	Pointer to OSCTXT structure

8.2.3.13 rtXmlInitContextUsingKey()

```
int rtXmlInitContextUsingKey (
            OSCTXT * pctxt,
```

```
            const OSOCTET * key,
            OSSIZE keylen )
```

This function initializes a context using a run-time key.

This form is required for evaluation and limited distribution software. The compiler will generate a macro for rtXmlInit↩ Context in the rtkey.h file that will invoke this function with the generated run-time key.

Parameters

pctxt	The pointer to the context structure variable to be initialized.
key	Key data generated by ASN1C compiler.
keylen	Key data field length.

Returns

Completion status of operation:

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

8.2.3.14 rtXmlInitCtxtAppInfo()

```
int rtXmlInitCtxtAppInfo (
            OSCTXT * pctxt )
```

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

Parameters

pctxt	Pointer to OSCTXT structure

8.2.3.15 rtXmlMatchBase64Str()

```
int rtXmlMatchBase64Str (
            OSCTXT * pctxt,
            OSSIZE minLength,
            OSSIZE maxLength )
```

This function tests the context buffer for containing a correct base64 string.

It does not decode the value.

Parameters

pctxt	Pointer to OSCTXT structure
minLength	A minimal length of expected string.
maxLength	A maximal length of expected string.

Returns

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

8.2.3.16 rtXmlMatchDate()

```
int rtXmlMatchDate (
             OSCTXT * pctxt )
```

This function tests the context buffer for containing a correct date string.

It does not decode the value.

Parameters

pctxt	Pointer to OSCTXT structure

Returns

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

8.2.3.17 rtXmlMatchDateTime()

```
int rtXmlMatchDateTime (
             OSCTXT * pctxt )
```

This function tests the context buffer for containing a correct dateTime string.

It does not decode the value.

Parameters

pctxt	Pointer to OSCTXT structure

Returns

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

8.2.3.18 rtXmlMatchGDay()

```
int rtXmlMatchGDay (
             OSCTXT * pctxt )
```

This function tests the context buffer for containing a correct gDay string.

It does not decode the value.

Parameters

pctxt	Pointer to OSCTXT structure

Returns

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

8.2.3.19 rtXmlMatchGMonth()

```
int rtXmlMatchGMonth (
             OSCTXT * pctxt )
```

This function tests the context buffer for containing a correct gMonth string.

It does not decode the value.

Parameters

pctxt	Pointer to OSCTXT structure

Returns

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

8.2.3.20 rtXmlMatchGMonthDay()

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

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

It does not decode the value.

Parameters

pctxt	Pointer to OSCTXT structure

Returns

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

8.2.3.21 rtXmlMatchGYear()

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

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

It does not decode the value.

Parameters

pctxt	Pointer to OSCTXT structure

Returns

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

8.2.3.22 rtXmlMatchGYearMonth()

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

This function tests the context buffer for containing a correct gYearMonth string.

It does not decode the value.

Parameters

pctxt	Pointer to OSCTXT structure

Returns

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

8.2.3.23 rtXmlMatchHexStr()

```
int rtXmlMatchHexStr (
             OSCTXT * pctxt,
             OSSIZE minLength,
             OSSIZE maxLength )
```

This function tests the context buffer for containing a correct hexadecimal string.

It does not decode the value.

Parameters

pctxt	Pointer to OSCTXT structure
minLength	A minimal length of expected string.
maxLength	A maximal length of expected string.

Returns

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

8.2.3.24 rtXmlMatchTime()

```
int rtXmlMatchTime (
             OSCTXT * pctxt )
```

This function tests the context buffer for containing a correct time string.

It does not decode the value.

Parameters

pctxt	Pointer to OSCTXT structure

Returns

If the string in the context buffer is a correct one, function returns zero. Error code (negative) will be returned otherwise. Note, error record will NOT be added in the error's list of the context.

8.2.3.25 rtXmlMemFreeAnyAttrs()

```
void rtXmlMemFreeAnyAttrs (
             OSCTXT * pctxt,
             OSRTDList * pAnyAttrList )
```

This function frees a list of anyAttribute that is a member of OSXSDAnyType structure.

Parameters

pctxt	Pointer to context block structure.
pAnyAttrList	Pointer to list of anyAttribute that is to be freed.

8.2.3.26 rtXmlNewQName()

```
OSUTF8CHAR* rtXmlNewQName (
             OSCTXT * pctxt,
             const OSUTF8CHAR * localName,
             const OSUTF8CHAR * prefix )
```

This function creates a new QName given the localName and prefix parts.

Parameters

pctxt	Pointer to a context structure.
localName	Element local name.
prefix	Namespace prefix.

Returns

QName value. Memory for the value will have been allocated by rtxMemAlloc and thus must be freed using one of the rtxMemFree functions. The value will be NULL if no dynamic memory was available.

8.2.3.27 rtXmlPrepareContext()

```
int rtXmlPrepareContext (
             OSCTXT * pctxt )
```

This function prepares the context for another encode by setting the state back to OSXMLINIT and moving the buffer's cursor back to the beginning of the buffer.

Parameters

pctxt	Pointer to OSCTXT structure

Returns

0 if OK, negative status code if error.

8.2.3.28 rtXmlSetEncC14N()

```
int rtXmlSetEncC14N (
             OSCTXT * pctxt,
             OSBOOL value )
```

This function sets the option to encode in C14N mode.

Parameters

pctxt	Pointer to OSCTXT structure
value	Boolean value: true = C14N mode enabled.

Returns

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

8.2.3.29 rtXmlSetEncDocHdr()

```
int rtXmlSetEncDocHdr (
             OSCTXT * pctxt,
             OSBOOL value )
```

This function sets the option to add the XML document header (i.e.

<?xml version="1.0" encoding="UTF-8"?>) to the XML output stream.

Parameters

pctxt	Pointer to OSCTXT structure
value	Boolean value: true = add document header

Returns

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

8.2.3.30 rtXmlSetEncodingStr()

```
int rtXmlSetEncodingStr (
             OSCTXT * pctxt,
             const OSUTF8CHAR * encodingStr )
```

This function sets the XML output encoding to the given value.

Currently, UTF-8/UTF-16/ISO-8859-1 encodings are supported.

Parameters

pctxt	Pointer to OSCTXT structure
encodingStr	XML output encoding format

Returns

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

8.2.3.31 rtXmlSetEncXSINamespace()

```
int rtXmlSetEncXSINamespace (
             OSCTXT * pctxt,
             OSBOOL value )
```

This function sets a flag in the context that indicates the XSI namespace declaration (xmlns:xsi) should be added to the encoded XML instance.

Parameters

pctxt	Pointer to OSCTXT structure
value	Boolean value: true = encode XSI namespace attribute.

Returns

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

8.2.3.32 rtXmlSetEncXSINilAttr()

```
int rtXmlSetEncXSINilAttr (
             OSCTXT * pctxt,
             OSBOOL value )
```

This function sets a flag in the context that indicates the XSI attribute declaration (xmlns:xsi) should be added to the encoded XML instance.

Parameters

pctxt	Pointer to OSCTXT structure
value	Boolean value: true = encode xsi:nil attribute.

Returns

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

8.2.3.33 rtXmlSetFormatting()

```
int rtXmlSetFormatting (
             OSCTXT * pctxt,
             OSBOOL doFormatting )
```

This function sets XML output formatting to the given value.

If TRUE (the default), the XML document is formatted with indentation and newlines. If FALSE, all whitespace between elements is suppressed. Turning formatting off can provide more compressed documents and also a more canonical representation which is important for security applications. Also the function 'rtXmlSetIndent' might be used to set the exact size of identation.

Parameters

pctxt	Pointer to OSCTXT structure
doFormatting	Boolean value indicating if formatting is to be done

Returns

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

8.2.3.34 rtXmlSetIndent()

```
int rtXmlSetIndent (
             OSCTXT * pctxt,
             OSUINT8 indent )
```

This function sets XML output indent to the given value.

Parameters

pctxt	Pointer to OSCTXT structure
indent	Number of spaces per indent. Default is 3.

Returns

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

8.2.3.35 rtXmlSetIndentChar()

```
int rtXmlSetIndentChar (
             OSCTXT * pctxt,
             char indentChar )
```

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

Parameters

pctxt	Pointer to OSCTXT structure
indentChar	Indent character. Default is space.

Returns

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

8.2.3.36 rtXmlSetNamespacesSet()

```
void rtXmlSetNamespacesSet (
             OSCTXT * pctxt,
             OSBOOL value )
```

This function sets the context 'namespaces are set' flag.

This indicates that namespace declarations have been set either by the decoder or externally by the end user. It is used by the encoder to know not to set the default namespaces specified in the schema before starting encoding.

Parameters

pctxt	Pointer to OSCTXT structure.
value	Boolean value to which flag is to be set.

8.2.3.37 rtXmlSetNoNSSchemaLocation()

```
int rtXmlSetNoNSSchemaLocation (
             OSCTXT * pctxt,
             const OSUTF8CHAR * schemaLocation )
```

This function sets the XML Schema Instance (xsi) no namespace schema location attribute to be added to an encoded document.

This attribute is optional: if not set, no xsi:noNamespaceSchemaLocation attribute will be added.

Parameters

pctxt	Pointer to OSCTXT structure
schemaLocation	Schema location attribute value

Returns

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

8.2.3.38 rtXmlSetNSPrefixLinks()

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

This function sets namespace prefix/URI links in the namspace prefix stack in the context structure.

Parameters

pctxt	Pointer to OSCTXT structure.
pNSAttrs	List of namespace attributes.

Returns

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

8.2.3.39 rtXmlSetSchemaLocation()

```
int rtXmlSetSchemaLocation (
             OSCTXT * pctxt,
             const OSUTF8CHAR * schemaLocation )
```

This function sets the XML Schema Instance (xsi) schema location attribute to be added to an encoded document.

This attribute is optional: if not set, no xsi:schemaLocation attribute will be added.

Parameters

pctxt	Pointer to OSCTXT structure
schemaLocation	Schema location attribute value

Returns

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

8.2.3.40 rtXmlSetSoapVersion()

```
void rtXmlSetSoapVersion (
             OSCTXT * pctxt,
             OSUINT8 version )
```

This function sets the SOAP version number.

Parameters

pctxt	Pointer to OSCTXT structure
version	SOAP version number as 2 digit integer (for example, 11 is SOAP version 1.1, 12 is version 1.2, etc.)

8.2.3.41 rtXmlSetWriteBOM()

```
int rtXmlSetWriteBOM (
             OSCTXT * pctxt,
             OSBOOL write )
```

This function sets whether the Unicode byte order mark is encoded.

Parameters

pctxt	Pointer to OSCTXT structure
write	TRUE to encode BOM, FALSE to not encode BOM.

Returns

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

8.2.3.42 rtXmlSetXSITypeAttr()

```
int rtXmlSetXSITypeAttr (
             OSCTXT * pctxt,
             const OSUTF8CHAR * xsiType )
```

This function sets the XML Schema Instance (xsi) type attribute value.

This will cause an xsi:type attribute to be added to the top level element in an encoded XML instance.

Parameters

pctxt	Pointer to OSCTXT structure
xsiType	xsi:type attribute value

Returns

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

8.3 rtXmlErrCodes.h File Reference

List of numeric status codes that can be returned by ASN1C run-time functions and generated code.

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

Macros

- #define XML_OK_EOB 0x7fffffff

  End of block marker.
- #define XML_OK_FRAG XML_OK_EOB

  Maintained for backward compatibility.
- #define XML_E_BASE -200

  Error base.
- #define XML_E_GENERR (XML_E_BASE)

  General error.
- #define XML_E_INVSYMBOL (XML_E_BASE-1)

  An invalid XML symbol (character) was detected at the given point in the parse stream.
- #define XML_E_TAGMISMATCH (XML_E_BASE-2)

  Start/end tag mismatch.
- #define XML_E_DUPLATTR (XML_E_BASE-3)

  Duplicate attribute found.
- #define XML_E_BADCHARREF (XML_E_BASE-4)

  Bad character reference found.
- #define XML_E_INVMODE (XML_E_BASE-5)

  Invalid mode.
- #define XML_E_UNEXPEOF (XML_E_BASE-6)

  Unexpected end of file (document).
- #define XML_E_NOMATCH (XML_E_BASE-7)

  Current tag is not matched to specified one.
- #define XML_E_ELEMMISRQ (XML_E_BASE-8)

  Missing required element.
- #define XML_E_ELEMSMISRQ (XML_E_BASE-9)

  Missing required elements.
- #define XML_E_TOOFEWELEMS (XML_E_BASE-10)

  The number of elements in a repeating collection was less than the number of elements specified in the XSD minOccurs facet for this type or element.
- #define XML_E_UNEXPSTARTTAG (XML_E_BASE-11)

  Unexpected start tag.
- #define XML_E_UNEXPENDTAG (XML_E_BASE-12)

  Unexpected end tag.
- #define XML_E_IDNOTFOU (XML_E_BASE-13)

  Expected identifier not found.
- #define XML_E_INVTYPEINFO (XML_E_BASE-14)

  Unknown xsi:type.
- #define XML_E_NSURINOTFOU (XML_E_BASE-15)

  Namespace URI not defined for given prefix.
- #define XML_E_KEYNOTFOU (XML_E_BASE-16)

  Keyref constraint has some key that not present in refered constraint.
- #define XML_E_DUPLKEY (XML_E_BASE-17)

  Key or unique constraint has duplicated key.
- #define XML_E_FLDABSENT (XML_E_BASE-18)

  Some key has no full set of fields.
- #define XML_E_DUPLFLD (XML_E_BASE-19)

  Some key has more than one value for field.
- #define XML_E_NOTEMPTY (XML_E_BASE-20)

  An element was not empty when expected.

8.3.1 Detailed Description

List of numeric status codes that can be returned by ASN1C run-time functions and generated code.

8.4 rtXmlExternDefs.h File Reference

XML external definitions macro.

8.4.1 Detailed Description

XML external definitions macro.

This is used for Windows to properly declare function scope within DLL's.

8.5 rtXmlKeyArray.h File Reference

Implementation of a dynamic pointer sorted array.

```
#include "rtxsrc/rtxContext.h"
#include "rtxmlsrc/rtXmlExternDefs.h"
```

Functions

- int rtXmlKeyArrayInit (OSCTXT *pctxt, OSXSDKeyArray *pArray, OSUINT32 nmFields, OSBOOL key, const char *name)

  Initialize the array by allocating memory for it.
- void rtXmlKeyArraySetString (OSXSDKeyArray *pArray, const OSUTF8CHAR *pValue, OSUINT32 fldNum)

  Set the given field's value to the given string value.
- void rtXmlKeyArraySetInt (OSXSDKeyArray *pArray, OSINT32 value, OSUINT32 fldNum)

  Set the given field's value to the given integer value.
- void rtXmlKeyArraySetUInt (OSXSDKeyArray *pArray, OSUINT32 value, OSUINT32 fldNum)

  Set the given field's value to the given unsigned integer value.
- void rtXmlKeyArraySetDecimal (OSXSDKeyArray *pArray, const OSREAL *value, OSUINT32 fldNum)

  Set the given field's value to the given decimal value.
- int rtXmlKeyArrayAdd (OSCTXT *pctxt, OSXSDKeyArray *pArray)

  Once all the fields for a key are set, invoke this method to add the key.
- int rtXmlKeyArrayContains (OSCTXT *pctxt, OSXSDKeyArray *pArray)

  Once all the fields for a key are set, this method is used to check whether the key is already present in the array of keys.

8.5.1 Detailed Description

Implementation of a dynamic pointer sorted array.

8.5.2 Function Documentation

8.5.2.1 rtXmlKeyArrayAdd()

```
int rtXmlKeyArrayAdd (
             OSCTXT * pctxt,
             OSXSDKeyArray * pArray )
```

Once all the fields for a key are set, invoke this method to add the key.

It will report an error if any of the following are true:

- a field was skipped and the constraint is a key constraint
- a field was set more than once This method adds the "new" key (pointed to by pArray) into the correct place within the array of keys (and their fields), also pointed to by pArray.

8.5.2.2 rtXmlKeyArrayContains()

```
int rtXmlKeyArrayContains (
             OSCTXT * pctxt,
             OSXSDKeyArray * pArray )
```

Once all the fields for a key are set, this method is used to check whether the key is already present in the array of keys.

Thus, to check a key ref, you set all the fields, then invoke this method. You do NOT invoke rtXmlKeyArrayAdd.

Returns

0 if key is found as expected

8.5.2.3 rtXmlKeyArrayInit()

```
int rtXmlKeyArrayInit (
             OSCTXT * pctxt,
             OSXSDKeyArray * pArray,
             OSUINT32 nmFields,
             OSBOOL key,
             const char * name )
```

Initialize the array by allocating memory for it.

Parameters

pctxt	
pArray	The array whose data should be initialized.
nmFields	The number of fields in the key whose data will be held in pArray->data.
key	TRUE if the data is for a key; FALSE if for a unique identity constraint.
name	of the identity constraint

8.5.2.4 rtXmlKeyArraySetDecimal()

```
void rtXmlKeyArraySetDecimal (
             OSXSDKeyArray * pArray,
             const OSREAL * value,
             OSUINT32 fldNum )
```

Set the given field's value to the given decimal value.

see comment on set* methods above.

8.5.2.5 rtXmlKeyArraySetInt()

```
void rtXmlKeyArraySetInt (
             OSXSDKeyArray * pArray,
             OSINT32 value,
             OSUINT32 fldNum )
```

Set the given field's value to the given integer value.

see comment on set* methods above.

8.5.2.6 rtXmlKeyArraySetString()

```
void rtXmlKeyArraySetString (
             OSXSDKeyArray * pArray,
             const OSUTF8CHAR * pValue,
             OSUINT32 fldNum )
```

Set the given field's value to the given string value.

see comment on set* methods above.

8.5.2.7 rtXmlKeyArraySetUInt()

```
void rtXmlKeyArraySetUInt (
             OSXSDKeyArray * pArray,
             OSUINT32 value,
             OSUINT32 fldNum )
```

Set the given field's value to the given unsigned integer value.

see comment on set∗ methods above.

8.6 rtXmlNamespace.h File Reference

XML namespace handling structures and function definitions.

```
#include "rtxsrc/rtxContext.h"
#include "rtxsrc/rtxDynPtrArray.h"
#include "rtxsrc/rtxXmlQName.h"
#include "rtxmlsrc/rtXmlExternDefs.h"
```

Macros

- #define RTXMLNSSETQNAME(qname, pNS)

  This macro populates the given QName structure with information from the given namespace structure (namespace URI and prefix).

Functions

- OSXMLNamespace ∗ rtXmlNSAddNamespace (OSCTXT ∗pctxt, OSRTDList ∗pNSAttrs, const OSUTF8CHAR ∗prefix, const OSUTF8CHAR ∗uri)

  This function adds a namespace to the context namespace list.
- OSBOOL rtXmlNSEqual (OSXMLNamespace ∗pNS1, OSXMLNamespace ∗pNS2)

  This function checks if two namespace records are equal.
- void rtXmlNSFreeAttrList (OSCTXT ∗pctxt, OSRTDList ∗pNSAttrs)

  This function frees dynamic memory used to hold namespace attribute values.
- const OSUTF8CHAR ∗ rtXmlNSGetPrefix (OSCTXT ∗pctxt, const OSUTF8CHAR ∗uri)

  This function gets a namespace prefix assigned to the given URI.
- const OSUTF8CHAR ∗ rtXmlNSGetPrefixUsingIndex (OSCTXT ∗pctxt, const OSUTF8CHAR ∗uri, OSUINT32 idx)

  This function gets a namespace prefix assigned to the given URI using the given index to select a specific prefix from the URI/prefix map.
- OSUINT32 rtXmlNSGetPrefixCount (OSCTXT ∗pctxt, const OSUTF8CHAR ∗uri)

  This function returns the total number of prefixes currently assigned to the given URI.
- int rtXmlNSGetPrefixIndex (OSCTXT ∗pctxt, const OSUTF8CHAR ∗uri, const OSUTF8CHAR ∗prefix, OSUINT32 ∗pcount)

This function gets the index of a given prefix in the internal list of prefixes maintained for a given URI in teh namespace stack.

- const OSUTF8CHAR * rtXmlNSGetQName (OSCTXT *pctxt, OSUTF8CHAR *buf, size_t bufsiz, const OSUTF8CHAR *uri, const OSUTF8CHAR *localName)

  This function creates a QName in the given fixed-site buffer.

- EXTXMLMETHOD const OSUTF8CHAR * rtXmlNSGetAttrPrefix (OSCTXT *pctxt, const OSUTF8CHAR *namespaceURI, OSRTDList *pNSAttrs)

  This function returns a namespace prefix for use with attributes in the given namespace.

- const OSUTF8CHAR * rtXmlNSGetAttrQName (OSCTXT *pctxt, OSUTF8CHAR *buf, size_t bufsiz, OSXMLNamespace *pNS, const OSUTF8CHAR *localName, OSRTDList *pNSAttrs)

  This function creates a QName for a qualified attribute.

- OSXMLNamespace * rtXmlNSLookupURI (OSCTXT *pctxt, const OSUTF8CHAR *uri)

  This function looks up a namespace in the context namespace stack using URI as the key value.

- OSXMLNamespace * rtXmlNSLookupURIInList (OSRTDList *pNSAttrs, const OSUTF8CHAR *uri)

  This function looks up a namespace in the given list using URI as the key value.

- const OSUTF8CHAR * rtXmlNSLookupPrefixForURI (OSCTXT *pctxt, const OSUTF8CHAR *uri)

  This function looks up a namespace in the context namespace stack using URI as the key value and returns a non-empty prefix, if one has been defined.

- const OSUTF8CHAR * rtXmlNSLookupPrefix (OSCTXT *pctxt, const OSUTF8CHAR *prefix)

  This function looks up a namespace in the context namespace list using the prefix as the key value.

- const OSUTF8CHAR * rtXmlNSLookupPrefixFrag (OSCTXT *pctxt, const OSUTF8CHAR *prefix, size_t prefixLen)

  This function looks up a namespace in the context namespace list using the prefix as the key value.

- void rtXmlNSRemoveAll (OSCTXT *pctxt)

  This function removes all namespaces from the context namespace list and frees the dynamic memory used to hold the names.

- OSXMLNamespace * rtXmlNSSetNamespace (OSCTXT *pctxt, OSRTDList *pNSAttrs, const OSUTF8CHAR *prefix, const OSUTF8CHAR *uri, OSBOOL override)

  This function sets a namespace in the context namespace list.

- const OSUTF8CHAR * rtXmlNSNewPrefix (OSCTXT *pctxt, const OSUTF8CHAR *uri, OSRTDList *pNSAttrs)

  This function returns the next unused prefix of the form "nsX" where X is a number.

- int rtXmlNSAddPrefixLink (OSCTXT *pctxt, const OSUTF8CHAR *prefix, const OSUTF8CHAR *uri, const OSUTF8CHAR *nsTable[], OSUINT32 nsTableRowCount)

  Add a prefix link at the current stack level.

- int rtXmlNSFreeAllPrefixLinks (OSCTXT *pctxt, OSXMLNSPfxLinkStackNode *pStackNode)

  Free all prefixs links in the given namespace stack entry.

- int rtXmlNSFreePrefixLink (OSCTXT *pctxt, OSXMLNSPfxLink *plink)

  Free all data within a given namespace prefix link structure.

- int rtXmlNSGetIndex (OSCTXT *pctxt, const OSUTF8CHAR *prefix)

  Get namespace index for a given namespace prefix based on current namespace stack in context.

- int rtXmlNSPush (OSCTXT *pctxt)

  Push new namespace prefix mapping level onto stack.

- int rtXmlNSPop (OSCTXT *pctxt)

  Remove top namespace prefix mapping level from stack.

- void rtXmlNSSetURITable (OSCTXT *pctxt, const OSUTF8CHAR *data[], OSUINT32 nrows)

  Set namespace URI table in context.

8.6.1 Detailed Description

XML namespace handling structures and function definitions.

8.6.2 Macro Definition Documentation

8.6.2.1 RTXMLNSSETQNAME

```
#define RTXMLNSSETQNAME(
              qname,
              pNS )
```

Value:

```
if (0 != pNS) { qname.nsPrefix = pNS->prefix; qname.nsURI = pNS->uri; } \
else { qname.nsPrefix = qname.nsURI = 0; }
```

This macro populates the given QName structure with information from the given namespace structure (namespace URI and prefix).

Parameters

qname	Reference to QName structure to be populated. Pointers to items in pNS are assigned directly to fields in qname. No copies of data are made.
pNS	Pointer to namespace structure.

Definition at line 330 of file rtXmlNamespace.h.

8.6.3 Function Documentation

8.6.3.1 rtXmlNSAddNamespace()

```
OSXMLNamespace* rtXmlNSAddNamespace (
             OSCTXT * pctxt,
             OSRTDList * pNSAttrs,
             const OSUTF8CHAR * prefix,
             const OSUTF8CHAR * uri )
```

This function adds a namespace to the context namespace list.

Parameters

pctxt	Pointer to OSCTXT structure
pNSAttrs	Namespace attribute list to which namespace info is to be added.
prefix	Namespace prefix to be added
uri	Namespace URI to be added

Returns

Pointer to namespace structure or NULL if not added.

8.6.3.2 rtXmlNSEqual()

```
OSBOOL rtXmlNSEqual (
             OSXMLNamespace * pNS1,
             OSXMLNamespace * pNS2 )
```

This function checks if two namespace records are equal.

This does a a deep compare in that it will first check if the pointers are equal and then it will check if the contents are equal (same prefix and URI).

Parameters

pNS1	Pointer to first namespace records to check.
pNS2	Pointer to second record.

Returns

True if records are equal, false otherwise.

8.6.3.3 rtXmlNSFreeAttrList()

```
void rtXmlNSFreeAttrList (
             OSCTXT * pctxt,
             OSRTDList * pNSAttrs )
```

This function frees dynamic memory used to hold namespace attribute values.

Parameters

pctxt	Pointer to OSCTXT structure
pNSAttrs	Pointer to namespace attribute list to be freed.

8.6.3.4 rtXmlNSGetAttrPrefix()

```
EXTXMLMETHOD const OSUTF8CHAR* rtXmlNSGetAttrPrefix (
             OSCTXT * pctxt,
             const OSUTF8CHAR * namespaceURI,
             OSRTDList * pNSAttrs )
```

This function returns a namespace prefix for use with attributes in the given namespace.

If a prefix is not found for the namespace, a new namespace entry is created with a generated prefix.

Parameters

pctxt	Pointer to OSCTXT structure
namespaceURI	Pointer to namespace URI
pNSAttrs	List of namespace records. If null, the namespace list in the context will be used.

Returns

The prefix.

8.6.3.5 rtXmlNSGetAttrQName()

```
const OSUTF8CHAR* rtXmlNSGetAttrQName (
             OSCTXT * pctxt,
             OSUTF8CHAR * buf,
             size_t bufsiz,
             OSXMLNamespace * pNS,
             const OSUTF8CHAR * localName,
             OSRTDList * pNSAttrs )
```

This function creates a QName for a qualified attribute.

If a prefix is not found for the name, a new namespace entry is created with a generated prefix.

Parameters

pctxt	Pointer to OSCTXT structure
buf	Buffer into which qname will be written.
bufsiz	Size of the buffer.
pNS	Pointer to namespace URI and prefix structure.
localName	Local name of the item.
pNSAttrs	List of namespace records. If null, the namespace list in the context will be used.

Returns

Pointer to QName buffer (buf).

8.6.3.6 rtXmlNSGetPrefix()

```
const OSUTF8CHAR* rtXmlNSGetPrefix (
             OSCTXT * pctxt,
             const OSUTF8CHAR * uri )
```

This function gets a namespace prefix assigned to the given URI.

This gives preference to empty prefixes.

Parameters

pctxt	Pointer to OSCTXT structure
uri	Namespace URI to be searched for

Returns

Pointer to namespace prefix string. If a NULL or empty prefix was assigned to the URI, an empty sring is returned. Otherwise, any assigned prefix is returned. If no prefix was assigned, null is returned.

8.6.3.7 rtXmlNSGetPrefixCount()

```
OSUINT32 rtXmlNSGetPrefixCount (
             OSCTXT * pctxt,
             const OSUTF8CHAR * uri )
```

This function returns the total number of prefixes currently assigned to the given URI.

Parameters

pctxt	Pointer to OSCTXT structure
uri	Namespace URI to be searched for

Returns

Count of prefixes assigned to the URI.

8.6.3.8 rtXmlNSGetPrefixIndex()

```
int rtXmlNSGetPrefixIndex (
             OSCTXT * pctxt,
             const OSUTF8CHAR * uri,
             const OSUTF8CHAR * prefix,
             OSUINT32 * pcount )
```

This function gets the index of a given prefix in the internal list of prefixes maintained for a given URI in teh namespace stack.

It also may return the total number of prefixes currently assigned to the URI.

Parameters

pctxt	Pointer to OSCTXT structure
uri	Namespace URI to be searched for
prefix	Namespace prefix for which to get index.
pcount	Optional pointer to an integer count variable. If provided, the total number of prefixes currently assigned to the URI will be returned.

Returns

Index to namespace prefix or -1 if the prefix is not assigned to the given URI.

8.6.3.9 rtXmlNSGetPrefixUsingIndex()

```
const OSUTF8CHAR* rtXmlNSGetPrefixUsingIndex (
             OSCTXT * pctxt,
             const OSUTF8CHAR * uri,
             OSUINT32 idx )
```

This function gets a namespace prefix assigned to the given URI using the given index to select a specific prefix from the URI/prefix map.

Parameters

pctxt	Pointer to OSCTXT structure
uri	Namespace URI to be searched for
idx	Index to prefix in map. This only has meaning when multiple prefixes have been assigned to the given URI.

Returns

Pointer to namespace prefix string

8.6.3.10 rtXmlNSGetQName()

```
const OSUTF8CHAR* rtXmlNSGetQName (
             OSCTXT * pctxt,
             OSUTF8CHAR * buf,
             size_t bufsiz,
             const OSUTF8CHAR * uri,
             const OSUTF8CHAR * localName )
```

This function creates a QName in the given fixed-site buffer.

If the name will not fit in the buffer, it is truncated.

Parameters

pctxt	Pointer to OSCTXT structure
buf	Buffer into which qname will be written.
bufsiz	Size of the buffer.
uri	Namespace URI.
localName	Local name of the item.

Returns

Pointer to QName buffer (buf).

8.6.3.11 rtXmlNSLookupPrefix()

```
const OSUTF8CHAR* rtXmlNSLookupPrefix (
             OSCTXT * pctxt,
             const OSUTF8CHAR * prefix )
```

This function looks up a namespace in the context namespace list using the prefix as the key value.

Parameters

pctxt	Pointer to OSCTXT structure
prefix	Namespace Prefix to be found.

Returns

Pointer to namespace URI or NULL if not found.

8.6.3.12 rtXmlNSLookupPrefixForURI()

```
const OSUTF8CHAR* rtXmlNSLookupPrefixForURI (
             OSCTXT * pctxt,
             const OSUTF8CHAR * uri )
```

This function looks up a namespace in the context namespace stack using URI as the key value and returns a non-empty prefix, if one has been defined.

Parameters

pctxt	Pointer to OSCTXT structure
uri	Namespace URI to be found.

Returns

Pointer to non-empty prefix. NULL if URI is not found or URI has no associated non-empty prefix.

8.6.3.13 rtXmlNSLookupPrefixFrag()

```
const OSUTF8CHAR* rtXmlNSLookupPrefixFrag (
             OSCTXT * pctxt,
             const OSUTF8CHAR * prefix,
             size_t prefixLen )
```

This function looks up a namespace in the context namespace list using the prefix as the key value.

Parameters

pctxt	Pointer to OSCTXT structure
prefix	Namespace Prefix to be found.
prefixLen	Namespace Prefix length.

Returns

Pointer to namespace URI or NULL if not found.

8.6.3.14 rtXmlNSLookupURI()

```
OSXMLNamespace* rtXmlNSLookupURI (
            OSCTXT * pctxt,
            const OSUTF8CHAR * uri )
```

This function looks up a namespace in the context namespace stack using URI as the key value.

Parameters

pctxt	Pointer to OSCTXT structure
uri	Namespace URI to be found.

Returns

Pointer to namespace structure or NULL if not found.

8.6.3.15 rtXmlNSLookupURIInList()

```
OSXMLNamespace* rtXmlNSLookupURIInList (
            OSRTDList * pNSAttrs,
            const OSUTF8CHAR * uri )
```

This function looks up a namespace in the given list using URI as the key value.

Parameters

pNSAttrs	List of namespace records.
uri	Namespace URI to be found.

Returns

Pointer to namespace structure or NULL if not found.

8.6.3.16 rtXmlNSNewPrefix()

```
const OSUTF8CHAR* rtXmlNSNewPrefix (
             OSCTXT * pctxt,
             const OSUTF8CHAR * uri,
             OSRTDList * pNSAttrs )
```

This function returns the next unused prefix of the form "nsX" where X is a number.

The new namespace declaration is added to the list provided or the context list if a NULL pointer is passed for pNSAttrs.

Parameters

pctxt	Pointer to OSCTXT structure
uri	Namespace URI. Must not be NULL or empty string.
pNSAttrs	Pointer to list of namespace attributes. If null, the namespace list in the context will be used.

Returns

New namespace prefix.

8.6.3.17 rtXmlNSRemoveAll()

```
void rtXmlNSRemoveAll (
             OSCTXT * pctxt )
```

This function removes all namespaces from the context namespace list and frees the dynamic memory used to hold the names.

Parameters

pctxt	Pointer to OSCTXT structure

8.6.3.18 rtXmlNSSetNamespace()

```
OSXMLNamespace* rtXmlNSSetNamespace (
             OSCTXT * pctxt,
             OSRTDList * pNSAttrs,
             const OSUTF8CHAR * prefix,
             const OSUTF8CHAR * uri,
             OSBOOL override )
```

This function sets a namespace in the context namespace list.

If the given namespace URI does not exist in the list, the namespace is added. If the URI is found, the action depends on the value of the override flag. If true, the value of the namespace prefix will be changed to the given prefix. If false, the existing namespace specification is not altered.

Parameters

pctxt	Pointer to OSCTXT structure
pNSAttrs	Namespace attribute list to which namespace info is to be added.
prefix	Namespace prefix
uri	Namespace URI
override	Should existing definition be changed?

Returns

Pointer to namespace structure or NULL if not set.

8.7 rtXmlpCppDecFuncs.h File Reference

XML low-level C++ decode functions.

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

Classes

- class OSXMLStringListParser

  Class enabling parsing of an XML Schema list into strings.

8.7.1 Detailed Description

XML low-level C++ decode functions.

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

Index