

# **XBinder**

---

XML Schema Compiler  
Version 1.4  
C EXI Runtime  
Reference Manual



The software described in this document is furnished under a license agreement and may be used only in accordance with the terms of this agreement.

### **Copyright Notice**

Copyright ©1997-2008 Objective Systems, Inc. All rights reserved.

This document may be distributed in any form, electronic or otherwise, provided that it is distributed in its entirety and that the copyright and this notice are included.

### **Author's Contact Information**

Comments, suggestions, and inquiries regarding XBinder may be submitted via electronic mail to [info@obj-sys.com](mailto:info@obj-sys.com).



# Contents

<b>1</b>	<b>XBinder Main Page</b>	<b>1</b>
<b>2</b>	<b>XBinder Module Index</b>	<b>2</b>
2.1	XBinder Modules . . . . .	2
<b>3</b>	<b>XBinder Class Index</b>	<b>3</b>
3.1	XBinder Class List . . . . .	3
<b>4</b>	<b>XBinder File Index</b>	<b>4</b>
4.1	XBinder File List . . . . .	4
<b>5</b>	<b>XBinder Module Documentation</b>	<b>5</b>
5.1	EXI runtime library functions. . . . .	5
5.2	EXI to XML serialization functions. . . . .	15
5.3	EXI decode structures and functions. . . . .	18
5.4	EXI encode structures and functions. . . . .	39
5.5	EXI event code definitions and functions. . . . .	49
5.6	XML to EXI serialization functions. . . . .	56
<b>6</b>	<b>XBinder Class Documentation</b>	<b>59</b>
6.1	_OSEXIEvent Struct Reference . . . . .	59
6.2	EXI2SAXAttribute Struct Reference . . . . .	60
6.3	EXI2SAXReader Struct Reference . . . . .	61
6.4	OSEXIAtmState Struct Reference . . . . .	62
6.5	OSEXIAutomaton Struct Reference . . . . .	63
6.6	OSEXICtxInfo Struct Reference . . . . .	65
6.7	OSEXIDecStringTable Struct Reference . . . . .	67
6.8	OSEXIDecStringTables Struct Reference . . . . .	68
6.9	OSEXIEncStringTable Struct Reference . . . . .	70
6.10	OSEXIEncStringTables Struct Reference . . . . .	71

6.11	OSEXIEventCode Struct Reference	73
6.12	OSEXIEventCodeGroup Struct Reference	74
6.13	OSEXIHeader Struct Reference	75
6.14	OSEXIStateEvent Struct Reference	76
6.15	OSEXIStateTableRecord Struct Reference	77
6.16	XML2EXISAXUserData Struct Reference	78
<b>7</b>	<b>XBinder File Documentation</b>	<b>79</b>
7.1	osrtexi.h File Reference	79
7.2	rtEXI2SAX.h File Reference	81
7.3	rtEXI2XML.h File Reference	82
7.4	rtEXIAutomaton.h File Reference	83
7.5	rtEXIDecAutomaton.h File Reference	85
7.6	rtEXIDecBitTrace.h File Reference	86
7.7	rtEXIDecoder.h File Reference	87
7.8	rtEXIDecStringTable.h File Reference	90
7.9	rtEXIDecStringTables.h File Reference	91
7.10	rtEXIEncAutomaton.h File Reference	93
7.11	rtEXIEncoder.h File Reference	94
7.12	rtEXIEncStringTable.h File Reference	106
7.13	rtEXIEncStringTables.h File Reference	107
7.14	rtEXIEvent.h File Reference	109
7.15	rtEXIEventCode.h File Reference	114
7.16	rtEXIEventCodeGroup.h File Reference	116
7.17	rtEXIExternDefs.h File Reference	118
7.18	rtXML2EXI.h File Reference	119
7.19	rtXML2EXISAX.h File Reference	120

# Chapter 1

## XBinder Main Page

### C EXI Runtime Library Functions

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

The categories of functions provided are as follows:

- Functions to manage the EXI context.
- Functions to encode XML to EXI.
- Functions to encode EXI to XML.
- Functions to manage string tables.
- Functions to run and manage finite-state automata.
- EXI encoder and decoder implementations.

# Chapter 2

## XBinder Module Index

### 2.1 XBinder Modules

Here is a list of all modules:

EXI runtime library functions. . . . .	5
EXI to XML serialization functions. . . . .	15
EXI decode structures and functions. . . . .	18
EXI encode structures and functions. . . . .	39
EXI event code definitions and functions. . . . .	49
XML to EXI serialization functions. . . . .	56

# Chapter 3

## XBinder Class Index

### 3.1 XBinder Class List

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

<a href="#">_OSEXIEvent</a> . . . . .	59
<a href="#">EXI2SAXAttribute</a> . . . . .	60
<a href="#">EXI2SAXReader</a> . . . . .	61
<a href="#">OSEXIAtmState</a> (This structure defines state information from an automaton that must be preserved at each stack level ) . . . . .	62
<a href="#">OSEXIAutomaton</a> (This structure defines a finite state automata for EXI grammars ) . . . . .	63
<a href="#">OSEXICtxtInfo</a> . . . . .	65
<a href="#">OSEXIDecStringTable</a> (This structure defines the structure of the various string table partitions used by the decoder ) . . . . .	67
<a href="#">OSEXIDecStringTables</a> (This structure defines the complete set of string table partitions used by the decoder ) . . . . .	68
<a href="#">OSEXIEncStringTable</a> (This structure defines the structure of the various string table partitions used by the encoder ) . . . . .	70
<a href="#">OSEXIEncStringTables</a> (This structure defines the complete set of string table partitions used by the encoder ) . . . . .	71
<a href="#">OSEXIEventCode</a> (A structure representing a production's event code ) . . . . .	73
<a href="#">OSEXIEventCodeGroup</a> (An EventCodeGroup is a group of related event codes ) . . . . .	74
<a href="#">OSEXIHeader</a> . . . . .	75
<a href="#">OSEXIStateEvent</a> (This structure holds state/event information ) . . . . .	76
<a href="#">OSEXIStateTableRecord</a> . . . . .	77
<a href="#">XML2EXISAXUserData</a> . . . . .	78

## Chapter 4

# XBinder File Index

### 4.1 XBinder File List

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

<a href="#">osrtexi.h</a> (EXI low-level C context and structure definitions ) . . . . .	79
<a href="#">rtEXI2SAX.h</a> (EXI SAX parser interface ) . . . . .	81
<a href="#">rtEXI2XML.h</a> (Functions for converting EXI encoded data into XML format ) . . . . .	82
<a href="#">rtEXIAutomaton.h</a> (EXI automaton structure and functions ) . . . . .	83
<a href="#">rtEXIDecAutomaton.h</a> (EXI decoding automaton structure and functions ) . . . . .	85
<a href="#">rtEXIDecBitTrace.h</a> (Functions for logging bit trace diagnostic events that occur during the decoding of an EXI-encoded document ) . . . . .	86
<a href="#">rtEXIDecoder.h</a> (Interface for EXI low-level decoders ) . . . . .	87
<b>rtEXIDecStateTable.h</b> . . . . .	<b>??</b>
<a href="#">rtEXIDecStringTable.h</a> (EXI decoder string table structure and functions ) . . . . .	90
<a href="#">rtEXIDecStringTables.h</a> (EXI decoder string table structure and functions ) . . . . .	91
<a href="#">rtEXIEncAutomaton.h</a> (EXI encoding automaton structure and functions ) . . . . .	93
<a href="#">rtEXIEncoder.h</a> (EXI low-level C encode functions ) . . . . .	94
<a href="#">rtEXIEncStringTable.h</a> (EXI encoder string table structure and functions ) . . . . .	106
<a href="#">rtEXIEncStringTables.h</a> (EXI encoder string table structure and functions ) . . . . .	107
<a href="#">rtEXIEvent.h</a> (EXI event definitions and functions ) . . . . .	109
<a href="#">rtEXIEventCode.h</a> (EXI event code definitions and functions ) . . . . .	114
<a href="#">rtEXIEventCodeGroup.h</a> (EXI event code definitions and functions ) . . . . .	116
<a href="#">rtEXIExternDefs.h</a> (EXI external function declaration macros for building Windows DLL's ) . . . . .	118
<a href="#">rtXML2EXI.h</a> (Functions for serializing XML data into EXI format ) . . . . .	119
<a href="#">rtXML2EXISAX.h</a> (SAX callback functions for serializing XML data into EXI format ) . . . . .	120

# Chapter 5

## XBinder Module Documentation

### 5.1 EXI runtime library functions.

#### Files

- file [osrtexi.h](#)

*EXI low-level C context and structure definitions.*

#### Classes

- struct [OSEXICtxtInfo](#)
- struct [OSEXIHeader](#)
- struct [OSEXIStateEvent](#)

*This structure holds state/event information.*

- struct [OSEXIAutomaton](#)

*This structure defines a finite state automata for EXI grammars.*

- struct [OSEXIAtmState](#)

*This structure defines state information from an automaton that must be preserved at each stack level.*

#### Defines

- #define [OSEXINULLINDEX](#) OSUINT32\_MAX
- #define [OSEXI\\_PRESERVE\\_DTD](#) 0x80000000
- #define [OSEXI\\_PRESERVE\\_PIS](#) 0x40000000
- #define [OSEXI\\_PRESERVE\\_COMMENTS](#) 0x20000000
- #define [OSEXI\\_PRESERVE\\_PREFIXES](#) 0x10000000
- #define [OSEXI\\_PRESERVE\\_LEXICAL](#) 0x08000000
- #define [OSEXI\\_FRAGMENT](#) 0x04000000
- #define [OSEXI\\_COMPRESSED](#) 0x02000000
- #define [OSEXI\\_ALIGNED](#) 0x01000000
- #define [OSEXI\\_PRECOMPRESSED](#) 0x00800000

- #define `OSEXI_SCHEMA` 0x00400000
- #define `OSEXI_NOWHITESPACE` 0x00200000
- #define `EXICTXT`(pctx) ((`OSEXICtxtInfo*`)(pctx) → `pEXIInfo`)
- #define `rtEXISetBufPtr`(pctx, bufaddr, bufsiz) `rtxCtxtSetBufPtr` (pctx, bufaddr, bufsiz)  
*This function is used to set the internal buffer within the run-time library context.*
- #define `rtEXIGetMsgPtr`(pctx) (pctx) → `buffer.data`  
*This macro returns the start address of the encoded EXI message.*
- #define `rtEXIGetMsgLen`(pctx) (pctx) → `buffer.byteIndex`  
*This macro returns the length of the encoded XML message.*
- #define `rtEXISetOption`(pctx, option) `EXICTXT`(pctx) → `options |= option;`  
*This macro is used to set a bit flag in the EXI options bit mask.*
- #define `rtEXIClearOption`(pctx, option) `EXICTXT`(pctx) → `options &= ~option;`  
*This macro is used to clear a bit flag in the EXI options bit mask.*
- #define `rtEXITestOption`(pctx, option) ((`EXICTXT`(pctx) → `options & option`) != 0)  
*This macro tests if the given option is set in the EXI context.*

## Typedefs

- typedef OSINT16 `OSEXIState`
- typedef int(\*) `OSEXIAtmAddTransFunc` (`OSCTXT *pctx`, `OSEXIAutomaton *pAutomaton`, `OSEXIState` fromState, `OSEXIState` toState, const `OSEXIEvent *pEvent`, const `OSEXIEventCode *pEventCode`)  
*Add transition function definition.*

## Enumerations

- enum `OSEXIAtmType`

## Functions

- EXTERNEXI int `rtEXIInitContext` (`OSCTXT *pctx`)  
*This function initializes a context variable for EXI encoding or decoding.*
- EXTERNEXI int `rtEXIInitCtxtAppInfo` (`OSCTXT *pctx`)  
*This function initializes the EXI application info section of the given context.*
- EXTERNEXI void `rtEXIEnableBitFieldTrace` (`OSCTXT *pctx`)  
*This function turns on bit field tracing and initializes the bit field trace list.*
- EXTERNEXI void `rtEXIAutomatonInit` (`OSCTXT *pctx`, `OSEXIAutomaton *pAutomaton`, const `OSXMLFullQName *pElemName`, `OSEXIState` numStates)  
*This function initializes the automaton to its default state.*

- EXTERNEXI `OSEXIAutomaton * rtEXINewAutomaton (OSCTXT *pctx, const OSXMLFullQName *pElemName, OSEXISState numStates)`  
*This function allocates memory for a new automaton structure and initializes the structure.*
- EXTERNEXI `OSEXIAutomaton * rtEXIAutomatonCopy (OSCTXT *pctx, OSEXIAutomaton *pAutomaton)`  
*This function copies an automaton structure.*
- EXTERNEXI void `rtEXIAutomatonFreeMem (OSCTXT *pctx, OSEXIAutomaton *pAutomaton, OSBOOL dynamic)`  
*This function frees all memory within an Automaton structure.*
- EXTERNEXI `OSEXIAutomaton * rtEXIAutomatonAddTransition (OSCTXT *pctx, OSEXIAutomaton *pAutomaton, OSEXISState fromState, OSEXISState toState, const OSEXIEventCode *pEventCode)`  
*This function adds a transition between two states.*
- EXTERNEXI int `rtEXIAtmAddUndeclaredItems (OSCTXT *pctx, OSEXIAutomaton *pAutomaton, OSEXISState fromState, OSEXISState toState, OSEXIEventCode *pEventCode, size_t numDeclAttrs, const OSXMLFullQName *declAttrs, OSEXIAtmAddTransFunc addTransFunc)`  
*This function adds all undeclared items (end element, start tag, and content) to an element automaton in schema-informed mode.*
- EXTERNEXI int `rtEXIAtmAddUndeclaredStartTagItems (OSCTXT *pctx, OSEXIAutomaton *pAutomaton, OSEXISState fromState, OSEXISState toState, OSEXIEventCode *pEventCode, size_t numDeclAttrs, const OSXMLFullQName *declAttrs, OSEXIAtmAddTransFunc addTransFunc)`  
*This function adds undeclared start tag items to an element automaton in schema-informed mode.*
- EXTERNEXI int `rtEXIAtmAddUndeclaredContentItems (OSCTXT *pctx, OSEXIAutomaton *pAutomaton, OSEXISState fromState, OSEXISState toState, OSEXIEventCode *pEventCode, OSEXIAtmAddTransFunc addTransFunc)`  
*This function adds undeclared content items to an element automaton in schema-informed mode.*
- EXTERNEXI `OSEXIAutomaton * rtEXIAutomatonInitCopy (OSCTXT *pctx, OSEXIAutomaton *pDestAtm, OSEXIAutomaton *pSrcAtm)`  
*This function initializes an automaton structure using the data from an existing automaton.*
- EXTERNEXI int `rtEXIAutomatonPush (OSCTXT *pctx, OSEXIAutomaton *pAutomaton)`  
*This function pushes an automaton onto the context stack.*
- EXTERNEXI `OSEXIAutomaton * rtEXIAutomatonPop (OSCTXT *pctx)`  
*This function pops an automaton from the context stack.*
- EXTERNEXI `OSEXIEventCodeGroup * rtEXIAtmGetCurrentEventCodeGroup (OSEXIAutomaton *pAutomaton)`  
*This function returns a pointer to the event code group corresponding to the current state.*
- EXTERNEXI `OSEXIAutomaton * rtEXIGetDocAutomaton (OSCTXT *pctx, size_t numGblElems, const OSXMLFullQName *gblElems, OSEXIAtmAddTransFunc addTransFunc)`  
*This functions returns an automaton that accepts the built-in document grammar.*

- EXTERNEXI `OSEXIAutomaton * rtEXIGetElemAutomaton (OSCTXT *pctxt, OSXMLFullQName *pqname, int(*addTransFunc)(OSCTXT *pctxt, OSEXIAutomaton *pAutomaton, OSEXIState fromState, OSEXIState toState, const OSEXIEvent *pEvent, const OSEXIEventCode *pEventCode))`

*This function returns an automaton that accepts the built-in element grammar.*

## 5.1.1 Define Documentation

### 5.1.1.1 #define rtEXIClearOption(pctxt, option) EXICTXT(pctxt) → options &= ~option;

This macro is used to clear a bit flag in the EXI options bit mask.

#### 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.

*option* Option bit mask value as defined earlier in this header file (for example, OSEXI\_PRESERVE\_DTD).

Definition at line 253 of file osrtexi.h.

### 5.1.1.2 #define rtEXIGetMsgLen(pctxt) (pctxt) → buffer.byteIndex

This macro returns the length of the encoded XML message.

#### Parameters:

*pctxt* Pointer to a context structure.

Definition at line 226 of file osrtexi.h.

### 5.1.1.3 #define rtEXIGetMsgPtr(pctxt) (pctxt) → buffer.data

This macro returns the start address of the encoded EXI 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 219 of file osrtexi.h.

### 5.1.1.4 #define rtEXISetBufPtr(pctxt, bufaddr, bufsiz) rtxCtxtSetBufPtr (pctxt, bufaddr, bufsiz)

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

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

#### 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).

Definition at line 208 of file osrtexi.h.

#### 5.1.1.5 **#define rtEXISetOption(pctx, option) EXICTXT(pctx) → options |= option;**

This macro is used to set a bit flag in the EXI options bit mask.

##### **Parameters:**

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

*option* Option bit mask value as defined earlier in this header file (for example, OSEXI\_PRESERVE\_DTD).

Definition at line 239 of file osrtexi.h.

#### 5.1.1.6 **#define rtEXITestOption(pctx, option) ((EXICTXT(pctx) → options & option) != 0)**

This macro tests if the given option is set in the EXI context.

##### **Parameters:**

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

*option* Option bit mask value as defined earlier in this header file (for example, OSEXI\_PRESERVE\_DTD).

##### **Returns:**

True if set, false if not.

Definition at line 269 of file osrtexi.h.

## 5.1.2 Function Documentation

### 5.1.2.1 **EXTERNEXI int rtEXIAtmAddUndeclaredContentItems (OSCTXT \* pctx, OSEXIAutomaton \* pAutomaton, OSEXIState fromState, OSEXIState toState, OSEXIEventCode \* pEventCode, OSEXIAtmAddTransFunc addTransFunc)**

This function adds undeclared content items to an element automaton in schema-informed mode.

##### **Parameters:**

*pctx* Pointer to context block structure.

*pAutomaton* Pointer to automaton structure.

*fromState* The origin state for this transition.

*toState* The destination state for this transition.

*pEventCode* Pointer to event code structure that contains the current state of assigned events in the grammar.

**Returns:**

Status of operation: 0 if successful; otherwise, a negative return code.

**5.1.2.2 EXTERNEXI int rtEXIAtmAddUndeclaredItems (OSCTXT \* *pctxt*, OSEXIAutomaton \* *pAutomaton*, OSEXISState *fromState*, OSEXISState *toState*, OSEXIEventCode \* *pEventCode*, size\_t *numDeclAttrs*, const OSXMLFullQName \* *declAttrs*, OSEXIAtmAddTransFunc *addTransFunc*)**

This function adds all undeclared items (end element, start tag, and content) to an element automaton in schema-informed mode.

**Parameters:**

*pctxt* Pointer to context block structure.

*pAutomaton* Pointer to automaton structure.

*fromState* The origin state for this transition.

*toState* The destination state for this transition.

*pEventCode* Pointer to event code structure that contains the current state of assigned events in the grammar.

*numDeclAttrs* Number of items in *declAttrs* array.

*declAttrs* Array of declared attribute QNames. These are attributes that were declared in the schema but which are not valid within the current context.

**Returns:**

Status of operation: 0 if successful; otherwise, a negative return code.

**5.1.2.3 EXTERNEXI int rtEXIAtmAddUndeclaredStartTagItems (OSCTXT \* *pctxt*, OSEXIAutomaton \* *pAutomaton*, OSEXISState *fromState*, OSEXISState *toState*, OSEXIEventCode \* *pEventCode*, size\_t *numDeclAttrs*, const OSXMLFullQName \* *declAttrs*, OSEXIAtmAddTransFunc *addTransFunc*)**

This function adds undeclared start tag items to an element automaton in schema-informed mode.

**Parameters:**

*pctxt* Pointer to context block structure.

*pAutomaton* Pointer to automaton structure.

*fromState* The origin state for this transition.

*toState* The destination state for this transition.

*pEventCode* Pointer to event code structure that contains the current state of assigned events in the grammar.

*numDeclAttrs* Number of items in *declAttrs* array.

*declAttrs* Array of declared attribute QNames. These are attributes that were declared in the schema but which are not valid within the current context.

**Returns:**

Status of operation: 0 if successful; otherwise, a negative return code.

**5.1.2.4 EXTERNEXI OSEXIEventCodeGroup\* rtEXIAtmGetCurrentEventCodeGroup (OSEXIAutomaton \* pAutomaton)**

This function returns a pointer to the event code group corresponding to the current state.

**Parameters:**

*pAutomaton* Pointer to automaton structure.

**Returns:**

Pointer to event group group.

**5.1.2.5 EXTERNEXI OSEXIAutomaton\* rtEXIAutomatonAddTransition (OSCTXT \* pctxt, OSEXIAutomaton \* pAutomaton, OSEXISate fromState, OSEXISate toState, const OSEXIEventCode \* pEventCode)**

This function adds a transition between two states.

The transition is defined by a pair of states, and event and event code.

**Parameters:**

*pctxt* Pointer to context block structure.

*pAutomaton* Pointer to automaton structure.

*fromState* The origin state for this transition.

*toState* The destination state for this transition.

*pEventCode* The event code returned after the transition.

**5.1.2.6 EXTERNEXI OSEXIAutomaton\* rtEXIAutomatonCopy (OSCTXT \* pctxt, OSEXIAutomaton \* pAutomaton)**

This function copies an automaton structure.

If the structure is closed, a shallow copy is done; otherwise, a deep copy.

**Parameters:**

*pctxt* Pointer to context block structure.

*pAutomaton* Pointer to automaton structure to copy.

**Returns:**

Copied automaton structure.

**5.1.2.7 EXTERNEXI void rtEXIAutomatonFreeMem (OSCTXT \* pctxt, OSEXIAutomaton \* pAutomaton, OSBOOL dynamic)**

This function frees all memory within an Automaton structure.

**Parameters:**

*pctxt* Pointer to a context structure.

*pAutomaton* Pointer to object in which memory will be freed.

*dynamic* Boolean indicating if pAutomaton is dynamic. If true, the memory for pAutomaton is freed.

**5.1.2.8 EXTERNEXI void rtEXIAutomatonInit (OSCTXT \* *pctxt*, OSEXIAutomaton \* *pAutomaton*, const OSXMLFullQName \* *pElemName*, OSEXISState *numStates*)**

This function initializes the automaton to its default state.

**Parameters:**

*pctxt* Pointer to context block structure.

*pAutomaton* Pointer to automaton structure to initialize.

*pElemName* Pointer to element QName.

*numStates* Number of states (if known) or zero.

**5.1.2.9 EXTERNEXI OSEXIAutomaton\* rtEXIAutomatonInitCopy (OSCTXT \* *pctxt*, OSEXIAutomaton \* *pDestAtm*, OSEXIAutomaton \* *pSrcAtm*)**

This function initializes an automaton structure using the data from an existing automaton.

**Parameters:**

*pctxt* Pointer to context block structure.

*pDestAtm* Pointer to destination (target) automaton structure.

*pSrcAtm* Pointer to source automaton structure.

**Returns:**

Copied (destination) automaton structure.

**5.1.2.10 EXTERNEXI OSEXIAutomaton\* rtEXIAutomatonPop (OSCTXT \* *pctxt*)**

This function pops an automaton from the context stack.

**Parameters:**

*pctxt* Pointer to context block structure.

**Returns:**

Last automaton pushed on stack or NULL if stack empty or error.

**5.1.2.11 EXTERNEXI int rtEXIAutomatonPush (OSCTXT \* *pctxt*, OSEXIAutomaton \* *pAutomaton*)**

This function pushes an automaton onto the context stack.

**Parameters:**

*pctxt* Pointer to context block structure.

*pAutomaton* Automaton to be pushed onto stack.

**Returns:**

Status of operation: 0 = success, negative = error.

**5.1.2.12 EXTERNEXI void rtEXIEnableBitFieldTrace (OSCTXT \* *pctxt*)**

This function turns on bit field tracing and initializes the bit field trace list.

**Parameters:**

*pctxt* Pointer to OSCTXT structure

**5.1.2.13 EXTERNEXI OSEXIAutomaton\* rtEXIGetDocAutomaton (OSCTXT \* *pctxt*, size\_t *numGblElems*, const OSXMLFullQName \* *gblElems*, OSEXIAtmAddTransFunc *addTransFunc*)**

This functions returns an automaton that accepts the built-in document grammar.

This grammar is not extensible, so all the event codes created are immutable.

**Parameters:**

*pctxt* Pointer to context block structure.

*numGblElems* Number of global elements.

*gblElems* Array of global element objects.

*addTransFunc* Pointer to function to add a transition.

**Returns:**

Pointer to document automaton.

**5.1.2.14 EXTERNEXI OSEXIAutomaton\* rtEXIGetElemAutomaton (OSCTXT \* *pctxt*, OSXMLFullQName \* *pqname*, int(\*) (OSCTXT \* *pctxt*, OSEXIAutomaton \* *pAutomaton*, OSEXIState fromState, OSEXIState toState, const OSEXIEvent \* *pEvent*, const OSEXIEventCode \* *pEventCode*) *addTransFunc*)**

This function returns an automaton that accepts the built-in element grammar.

The `elementName` is associated with the automaton.

**Parameters:**

*pctxt* Pointer to context block structure.

*pqname* Pointer to element QName.

*addTransFunc* Pointer to function to add a transition.

**Returns:**

Pointer to element automaton.

#### 5.1.2.15 EXTERNEXI int rtEXIInitContext (OSCTXT \* *pctxt*)

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

##### Parameters:

*pctxt* Pointer to OSCTXT structure

##### Returns:

Completion status of operation:

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

#### 5.1.2.16 EXTERNEXI int rtEXIInitCtxtAppInfo (OSCTXT \* *pctxt*)

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

##### Parameters:

*pctxt* Pointer to OSCTXT structure

##### Returns:

Completion status of operation:

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

#### 5.1.2.17 EXTERNEXI OSEXIAutomaton\* rtEXINewAutomaton (OSCTXT \* *pctxt*, const OSXMLFullQName \* *pElemName*, OSEXIState *numStates*)

This function allocates memory for a new automaton structure and initializes the structure.

##### Parameters:

*pctxt* Pointer to context block structure.

*pElemName* Pointer to element QName.

*numStates* Number of states (if known) or zero.

##### Returns:

Allocated automaton structure or NULL if no dynamic memory available.

## 5.2 EXI to XML serialization functions.

### Classes

- struct [EXI2SAXReader](#)
- struct [EXI2SAXAttribute](#)

### Defines

- #define [rtEXIDecBitFieldStart](#)(pctxt)
- #define [rtEXIDecBitFieldEnd](#)(pctxt, nameSuffix)
- #define [rtEXIDecBitFieldEnd2](#)(pctxt, name, value)
- #define [rtEXIDecBitFieldEndAndStartNew](#)(pctxt, nameSuffix)

### Typedefs

- typedef int(\*) [StartElementHandler](#) (void \*userData, const OSUTF8CHAR \*localname, const OSUTF8CHAR \*qname, const OSUTF8CHAR \*const \*attrs)  
*SAX start element handler callback function definition.*
- typedef int(\*) [EndElementHandler](#) (void \*userData, const OSUTF8CHAR \*localname, const OSUTF8CHAR \*qname)  
*SAX end element handler callback function definition.*
- typedef int(\*) [CharacterDataHandler](#) (void \*userData, const OSUTF8CHAR \*value, int len)  
*SAX characters handler callback function definition.*

### Functions

- EXTERNEXI [EXI2SAXReader](#) \* [rtEXI2SAXCreateReader](#) (OSCTXT \*pctxt, void \*pUserData, [StartElementHandler](#) startElemFunc, [EndElementHandler](#) endElemFunc, [CharacterDataHandler](#) charactersFunc)  
*This function is used to create a reader structure for use by the main SAX parser function.*
- EXTERNEXI int [rtEXI2SAXParse](#) ([EXI2SAXReader](#) \*pReader)  
*This is the main SAX parser function.*
- EXTERNEXI int [rtExi2XmlStream](#) (OSCTXT \*pctxt, OSCTXT \*poutctxt)  
*This function transcodes an EXI encoded message to an XML output stream.*

### 5.2.1 Typedef Documentation

#### 5.2.1.1 typedef int(\*) [CharacterDataHandler](#)(void \*userData, const OSUTF8CHAR \*value, int len)

SAX characters handler callback function definition.

#### Parameters:

*userData* User-defined data structure.

*value* Character data. String is not null-terminated.

*len* Number of characters in character string.

**Returns:**

Status of operation.

Definition at line 76 of file rtEXI2SAX.h.

**5.2.1.2** `typedef int(*) EndElementHandler(void *userData, const OSUTF8CHAR *localname, const OSUTF8CHAR *qname)`

SAX end element handler callback function definition.

**Parameters:**

*userData* User-defined data structure.

*localname* Local name of element.

*qname* Qualified name of element.

**Returns:**

Status of operation.

Definition at line 63 of file rtEXI2SAX.h.

**5.2.1.3** `typedef int(*) StartElementHandler(void *userData, const OSUTF8CHAR *localname, const OSUTF8CHAR *qname, const OSUTF8CHAR *const *attrs)`

SAX start element handler callback function definition.

**Parameters:**

*userData* User-defined data structure.

*localname* Local name of element.

*qname* Qualified name of element.

*attrs* Array of attribute name/value pairs, terminated by 0;

**Returns:**

Status of operation.

Definition at line 51 of file rtEXI2SAX.h.

## 5.2.2 Function Documentation

**5.2.2.1** `EXTERNEXI EXI2SAXReader* rtEXI2SAXCreateReader (OSCTXT *pctxt, void *pUserData, StartElementHandler startElemFunc, EndElementHandler endElemFunc, CharacterDataHandler charactersFunc)`

This function is used to create a reader structure for use by the main SAX parser function.

**Parameters:**

*pctxt* Pointer to a context structure.

*pUserData* Pointer to a user defined data structure that will be passed to the SAX callback functions.

*startElemFunc* Start element callback function.

*endElemFunc* End element callback function.

*charactersFunc* Characters callback function.

**Returns:**

Initialized SAX reader structure.

**5.2.2.2 EXTERNEXI int rtEXI2SAXParse (EXI2SAXReader \* pReader)**

This is the main SAX parser function.

It converts decoded EXI events to SAX callback function invocations.

**Parameters:**

*pReader* Pointer to EXI-to-SAX reader structure.

**Returns:**

Status of parse operation. Zero if success, negative status code if error.

**5.2.2.3 EXTERNEXI int rtExi2XmlStream (OSCTXT \* pctxt, OSCTXT \* poutctxt)**

This function transcodes an EXI encoded message to an XML output stream.

The input context structure describes an encoded EXI message as either an input stream or in memory in its internal buffer. The output context is assumed to contain an initialized XML output stream set up with one of the `rtxStream*CreateWriter` functions (for example, `rtxStreamFileCreateWriter` to write to a file).

**Parameters:**

*pctxt* Pointer to input context structure. This is assumed to contain an input stream or memory buffer containing an EXI encoded message.

*poutctxt* Pointer to an output context assumed to contain an output stream create with an `rtxStreamCreateWriter` function.

**Returns:**

Completion status of operation:

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

## 5.3 EXI decode structures and functions.

### Classes

- struct [OSEXIDecStringTable](#)  
*This structure defines the structure of the various string table partitions used by the decoder.*
- struct [OSEXIDecStringTables](#)  
*This structure defines the complete set of string table partitions used by the decoder.*

### Functions

- EXTERNEXI int [rtEXIDecAtmAddTransition](#) (OSCTXT \*pctx, [OSEXIAutomaton](#) \*pAutomaton, [OSEXIS-  
tate](#) fromState, [OSEXIS-  
tate](#) toState, const [OSEXIEvent](#) \*pEvent, const [OSEXIEventCode](#) \*pEventCode)  
*This function adds a transition between two states.*
- EXTERNEXI [OSEXIEvent](#) \* [rtEXIDecAutomatonAdvance](#) (OSCTXT \*pctx, [OSEXIAutomaton](#) \*pAutomaton, const [OSEXIEventCode](#) \*pEventCode, OSBOOL dynamicItems)  
*This function advances the decoder automaton based on event code, adding new transitions if dynamicItems is set to true and either SE(\*) or AT(\*) are matched instead of SE(qname) or AT(qname), respectively.*
- EXTERNEXI [OSEXIAutomaton](#) \* [rtEXIDecGetDocAutomaton](#) (OSCTXT \*pctx, size\_t numGblElems, const OSXMLFullQName \*gblElems)  
*This functions returns an automaton that accepts the built-in document grammar.*
- EXTERNEXI [OSEXIAutomaton](#) \* [rtEXIDecGetElemAutomaton](#) (OSCTXT \*pctx, OSXMLFullQName \*pqname)  
*This function returns an automaton that accepts the built-in element grammar.*
- EXTERNEXI int [rtEXIDecAttribute](#) (OSCTXT \*pctx, OSXMLFullQName \*pqname, const OSUTF8CHAR \*\*ppvalue)  
*Decodes the attribute at the current position in the decode stream.*
- EXTERNEXI int [rtEXIDecBoolValue](#) (OSCTXT \*pctx, OSBOOL \*pvalue)  
*This function decodes a boolean value.*
- EXTERNEXI int [rtEXIDec\\_CH\\_String\\_EE](#) (OSCTXT \*pctx, const OSXMLFullQName \*pqname, const OSUTF8CHAR \*\*ppvalue)  
*This function decodes a CH event (assumed to be a one part code = 0 in a 1 bit field) followed by string content followed by an EE event (assumed to be a one part code = 0 in a 1 bit field).*
- EXTERNEXI int [rtEXIDecDate](#) (OSCTXT \*pctx, OSNumDateTime \*pvalue)  
*This function decodes a date value into a structured variable.*
- EXTERNEXI int [rtEXIDecDateString](#) (OSCTXT \*pctx, const OSUTF8CHAR \*\*ppvalue)  
*This function decodes a date value into a string.*
- EXTERNEXI int [rtEXIDecDocumentType](#) (OSCTXT \*pctx, const OSUTF8CHAR \*\*ppName, const OSUTF8CHAR \*\*ppPublic, const OSUTF8CHAR \*\*ppSystem, const OSUTF8CHAR \*\*ppText)

*This function decodes an XML document type declaration (DTD).*

- EXTERNEXI int [rtEXIDecEventCodePart1](#) (OSCTXT \*pctxt, OSINT32 \*ppart, OSUINT32 nbits, OSUINT32 maxval)

*This function decodes part 1 of an event code.*

- EXTERNEXI OSBOOL [rtEXIDecHasNext](#) (OSCTXT \*pctxt)

*This functions checks for additional events in the decode stream.*

- EXTERNEXI int [rtEXIDecIntValue](#) (OSCTXT \*pctxt, OSINT32 \*pvalue)

*This function decodes a signed integer value.*

- EXTERNEXI int [rtEXIDecLocalName](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*ppname)

*Returns the local name associated with the current event.*

- EXTERNEXI int [rtEXIDecNamespaceURI](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*ppNSURI)

*Returns the namespace associated with the current event.*

- EXTERNEXI int [rtEXIDecNBitUIntValue](#) (OSCTXT \*pctxt, OSUINT32 \*pvalue, OSUINT32 nbits)

*This function decodes an unsigned integer value from a bit field of the given width.*

- EXTERNEXI int [rtEXIDecNextEventType](#) (OSCTXT \*pctxt, [OSEXIEventType](#) \*pEventType)

*Returns the next [OSEXIEventType](#) read by this decoder.*

- EXTERNEXI int [rtEXIDecoderInit](#) (OSCTXT \*pctxt)

*This function initializes the decoder.*

- EXTERNEXI int [rtEXIDecPrefix](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*\*ppPrefix)

*Returns the namespace associated with the current event.*

- EXTERNEXI int [rtEXIDecProcessingInstruction](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*ppTarget, const OSUTF8CHAR \*\*ppData)

*This function decodes an XML processing instruction (PI).*

- EXTERNEXI int [rtEXIDecQName](#) (OSCTXT \*pctxt, OSXMLFullQName \*pqname)

*Returns the qname associated with the current event.*

- EXTERNEXI int [rtEXIDecReset](#) (OSCTXT \*pctxt)

*Resets the decoder for decoding a new message instance.*

- EXTERNEXI int [rtEXIDecString](#) (OSCTXT \*pctxt, const OSXMLFullQName \*pqname, const OSUTF8CHAR \*\*ppvalue)

*Returns the value associated with the current event.*

- EXTERNEXI int [rtEXIDecStringCHEvent](#) (OSCTXT \*pctxt, const OSXMLFullQName \*pqname, const OSUTF8CHAR \*\*ppvalue)

*This function decodes a CH event (assumed to be a one-bit field with code 0) followed by string content.*

- EXTERNEXI int [rtEXIDecStringToCharArray](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*target, size\_t start, size\_t length)

Similar to `rtEXIDecString` but the characters are copied into a fixed-size character array.

- EXTERNEXI int `rtEXIDecStringLength` (OSCTXT \*pctxt)  
*Returns the length of the string returned by `rtEXIDecString`.*
- EXTERNEXI int `rtEXIDecUIntValue` (OSCTXT \*pctxt, OSUINT32 \*pvalue)  
*This function decodes an unsigned integer value.*
- EXTERNEXI int `rtEXIDecUTF8Str` (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)  
*This function decodes a UTF-8 string value.*
- EXTERNEXI int `rtEXIDecUTF8Chars` (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue, OSUINT32 nchars)  
*This function reads the given number of characters from the decode stream and creates a UTF-8 string.*
- EXTERNEXI void `rtEXIDecStringTableInit` (OSCTXT \*pctxt, OSEXIDecStringTable \*pstrtab, size\_t capacity)  
*This function initializes the given string table structure.*
- EXTERNEXI OSEXIDecStringTable \* `rtEXIDecNewStringTable` (OSCTXT \*pctxt, size\_t capacity)  
*This function allocates and initializes a new string table structure.*
- EXTERNEXI void `rtEXIDecStringTableClear` (OSCTXT \*pctxt, OSEXIDecStringTable \*pstrtab)  
*This function clears all strings out of the existing table.*
- EXTERNEXI OSUINT32 `rtEXIDecStringTableAdd` (OSCTXT \*pctxt, OSEXIDecStringTable \*pstrtab, const OSUTF8CHAR \*str)  
*This function adds a string to the given string table.*
- EXTERNEXI const OSUTF8CHAR \* `rtEXIDecStringTableGetString` (OSEXIDecStringTable \*pstrtab, OSUINT32 index)  
*This function gets the string at the given index (i.e.*
- EXTERNEXI void `rtEXIDecStrTabsInit` (OSCTXT \*pctxt, OSEXIDecStringTables \*pstrtabs)  
*This function initializes all EXI string table partitions.*
- EXTERNEXI void `rtEXIDecStrTabsClear` (OSCTXT \*pctxt, OSEXIDecStringTables \*pstrtabs)  
*This function clears all EXI string table partitions.*
- EXTERNEXI OSUINT32 `rtEXIDecStrTabsAddURI` (OSCTXT \*pctxt, OSEXIDecStringTables \*pstrtabs, const OSUTF8CHAR \*uri)  
*This function will add a URI to the URI string table partition.*
- EXTERNEXI const OSUTF8CHAR \* `rtEXIDecStrTabsGetURI` (OSEXIDecStringTables \*pstrtabs, OSUINT32 index)  
*This function will get the compact identifier of the given URI from the URI string table partition.*
- EXTERNEXI OSUINT32 `rtEXIDecStrTabsGetURITableSize` (OSEXIDecStringTables \*pstrtabs)  
*This function returns the current number of entries in the URI string table partition.*
- EXTERNEXI OSUINT32 `rtEXIDecStrTabsAddPrefix` (OSCTXT \*pctxt, OSEXIDecStringTables \*pstrtabs, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*prefix)

*This function adds the given prefix to the prefix table partition identified by the given URI.*

- EXTERNEXI const OSUTF8CHAR \* [rtEXIDecStrTabsGetPrefix](#) (OSEXIDecStringTables \*pstrtabs, const OSUTF8CHAR \*uri, OSUINT32 index)

*This function will get the compact identifier of the given prefix from the prefix string table partition identified by the given URI.*

- EXTERNEXI OSUINT32 [rtEXIDecStrTabsGetPrefixTableSize](#) (OSEXIDecStringTables \*pstrtabs, const OSUTF8CHAR \*uri)

*This function returns the current number of entries in the prefix string table partition identified by the given URI.*

- EXTERNEXI OSUINT32 [rtEXIDecStrTabsAddLocalName](#) (OSCTXT \*pctxt, OSEXIDecStringTables \*pstrtabs, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*name)

*This function adds the given local name to the local name table partition identified by the given URI.*

- EXTERNEXI const OSUTF8CHAR \* [rtEXIDecStrTabsGetLocalName](#) (OSEXIDecStringTables \*pstrtabs, const OSUTF8CHAR \*uri, OSUINT32 index)

*This function will get the compact identifier of the given localName from the localName string table partition identified by the given URI.*

- EXTERNEXI OSUINT32 [rtEXIDecStrTabsGetLocalNameTableSize](#) (OSEXIDecStringTables \*pstrtabs, const OSUTF8CHAR \*uri)

*This function returns the current number of entries in the localName string table partition identified by the given URI.*

- EXTERNEXI OSUINT32 [rtEXIDecStrTabsAddLocalValue](#) (OSCTXT \*pctxt, OSEXIDecStringTables \*pstrtabs, const OSXMLFullQName \*qname, const OSUTF8CHAR \*value)

*This function adds the given local value to the local value table partition identified by the given QName.*

- EXTERNEXI const OSUTF8CHAR \* [rtEXIDecStrTabsGetLocalValue](#) (OSEXIDecStringTables \*pstrtabs, const OSXMLFullQName \*qname, OSUINT32 index)

*This function will get the compact identifier of the given local value from the local value string table partition identified by the given QName.*

- EXTERNEXI OSUINT32 [rtEXIDecStrTabsGetLocalValueTableSize](#) (OSEXIDecStringTables \*pstrtabs, const OSXMLFullQName \*qname)

*This function returns the current number of entries in the local value string table partition identified by the given QName.*

- EXTERNEXI OSUINT32 [rtEXIDecStrTabsAddGlobalValue](#) (OSCTXT \*pctxt, OSEXIDecStringTables \*pstrtabs, const OSUTF8CHAR \*value)

*This function will add a string value to the global value string table partition.*

- EXTERNEXI const OSUTF8CHAR \* [rtEXIDecStrTabsGetGlobalValue](#) (OSEXIDecStringTables \*pstrtabs, OSUINT32 index)

*This function will get the compact identifier of the given string value from the global value string table partition.*

- EXTERNEXI OSUINT32 [rtEXIDecStrTabsGetGlobalValueTableSize](#) (OSEXIDecStringTables \*pstrtabs)

*This function returns the current number of entries in the global value string table partition.*

## 5.3.1 Function Documentation

### 5.3.1.1 EXTERNEXI int rtEXIDec\_CH\_String\_EE (OSCTXT \* *pctxt*, const OSXMLFullQName \* *pqname*, const OSUTF8CHAR \*\* *ppvalue*)

This function decodes a CH event (assumed to be a one part code = 0 in a 1 bit field) followed by string content followed by an EE event (assumed to be a one part code = 0 in a 1 bit field).

#### Parameters:

*pctxt* Pointer to a context structure.

*pqname* QName the value is associated with. This should be the QName that was decoded from the stream prior to calling this function.

*ppvalue* Pointer to variable to receive decoded data. Memory for the decoded name is allocated with rtxMemAlloc. It must be freed with rtxMemFreePtr or it will be freed when all memory in the context is freed (rtxMemFreeAll or rtxFreeContext).

#### Returns:

Status of the operation:

- 0 if success
- a negative status code if failure

### 5.3.1.2 EXTERNEXI int rtEXIDecAtmAddTransition (OSCTXT \* *pctxt*, OSEXIAutomaton \* *pAutomaton*, OSEXISState *fromState*, OSEXISState *toState*, const OSEXIEvent \* *pEvent*, const OSEXIEventCode \* *pEventCode*)

This function adds a transition between two states.

The transition is defined by a pair of states, and event and event code.

#### Parameters:

*pctxt* Pointer to context block structure.

*pAutomaton* Pointer to automaton structure.

*fromState* The origin state for this transition.

*toState* The destination state for this transition.

*pEvent* The event on which this transition occurs.

*pEventCode* The event code returned after the transition.

#### Returns:

Zero if operation was successful; a negative status code otherwise.

### 5.3.1.3 EXTERNEXI int rtEXIDecAttribute (OSCTXT \* *pctxt*, OSXMLFullQName \* *pqname*, const OSUTF8CHAR \*\* *ppvalue*)

Decodes the attribute at the current position in the decode stream.

This function is called when the application receives an AT event to get the attribute data.

**Parameters:**

*pctxt* Pointer to a context structure.

*pqname* Pointer to variable to receive decoded attribute QName. Memory for the decoded name components is allocated with `rtxMemAlloc`. It must be freed with `rtxMemFreePtr` or it will be freed when all memory in the context is freed (`rtxMemFreeAll` or `rtxFreeContext`).

*ppvalue* Pointer to variable to receive decoded data. Memory for the decoded name is allocated with `rtxMemAlloc`. It must be freed with `rtxMemFreePtr` or it will be freed when all memory in the context is freed (`rtxMemFreeAll` or `rtxFreeContext`).

**Returns:**

Status of the operation:

- 0 if success
- a negative status code if failure

#### 5.3.1.4 EXTERNEXI **OSEXIEvent\*** **rtEXIDecAutomatonAdvance** (**OSCTXT \*** *pctxt*, **OSEXIAutomaton \*** *pAutomaton*, **const OSEXIEventCode \*** *pEventCode*, **OSBOOL** *dynamicItems*)

This function advances the decoder automaton based on event code, adding new transitions if `dynamicItems` is set to `true` and either `SE(*)` or `AT(*)` are matched instead of `SE(qname)` or `AT(qname)`, respectively.

Dynamic transitions for CH are also added according to the spec.

**Parameters:**

*pctxt* Pointer to context block structure.

*pAutomaton* Pointer to automaton structure.

*pEventCode* Event code on which to advance the automaton.

*dynamicItems* Flag to add dynamic transitions to the automaton.

**Returns:**

The event code returned as a result of this transition.

#### 5.3.1.5 EXTERNEXI **int** **rtEXIDecBoolValue** (**OSCTXT \*** *pctxt*, **OSBOOL \*** *pvalue*)

This function decodes a boolean value.

**Parameters:**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to boolean to receive decoded value.

**Returns:**

Completion status of operation:

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

### 5.3.1.6 EXTERNEXI int rtEXIDecDate (OSCTXT \* *pctxt*, OSNumDateTime \* *pvalue*)

This function decodes a date value into a structured variable.

#### Parameters:

*pctxt* Pointer to context block structure.

*pvalue* Pointer to structured variable to receive decoded data.

#### Returns:

Completion status of operation:

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

### 5.3.1.7 EXTERNEXI int rtEXIDecDateString (OSCTXT \* *pctxt*, const OSUTF8CHAR \*\* *ppvalue*)

This function decodes a date value into a string.

#### Parameters:

*pctxt* Pointer to context block structure.

*ppvalue* Pointer to variable to receive decoded data. Memory for the decoded name is allocated with `rtxMemAlloc`. It must be freed with `rtxMemFreePtr` or it will be freed when all memory in the context is freed (`rtxMemFreeAll` or `rtxFreeContext`).

#### Returns:

Completion status of operation:

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

### 5.3.1.8 EXTERNEXI int rtEXIDecDocumentType (OSCTXT \* *pctxt*, const OSUTF8CHAR \*\* *ppName*, const OSUTF8CHAR \*\* *ppPublic*, const OSUTF8CHAR \*\* *ppSystem*, const OSUTF8CHAR \*\* *ppText*)

This function decodes an XML document type declaration (DTD).

#### Parameters:

*pctxt* Pointer to a context structure.

*ppName* Pointer to variable to receive decoded DTD name. Memory for the decoded name is allocated with `rtxMemAlloc`. It must be freed with `rtxMemFreePtr` or it will be freed when all memory in the context is freed (`rtxMemFreeAll` or `rtxFreeContext`).

*ppPublic* Pointer to variable to receive decoded DT public data. Memory for the decoded name is allocated with `rtxMemAlloc`. It must be freed with `rtxMemFreePtr` or it will be freed when all memory in the context is freed (`rtxMemFreeAll` or `rtxFreeContext`).

*ppSystem* Pointer to variable to receive decoded DT system data. Memory for the decoded name is allocated with `rtxMemAlloc`. It must be freed with `rtxMemFreePtr` or it will be freed when all memory in the context is freed (`rtxMemFreeAll` or `rtxFreeContext`).

*ppText* Pointer to variable to receive decoded DT text. Memory for the decoded name is allocated with `rtxMemAlloc`. It must be freed with `rtxMemFreePtr` or it will be freed when all memory in the context is freed (`rtxMemFreeAll` or `rtxFreeContext`).

**Returns:**

Status of the operation:

- 0 if success
- a negative status code if failure

**5.3.1.9 EXTERNEXI int rtEXIDecEventCodePart1 (OSCTXT \* *pctxt*, OSINT32 \* *ppart*, OSUINT32 *nbits*, OSUINT32 *maxval*)**

This function decodes part 1 of an event code.

It will decode and skip undeclared items until a schema-valid event code is received.

**Parameters:**

*pctxt* Pointer to a context structure.

*ppart* Pointer to event code part to receive decoded value.

*nbits* Number of bits in the part.

*maxval* Maximum allowed value for this code. If the decoded value is equal to this value, it indicates that undeclared content was encountered.

**Returns:**

Status of the operation:

- 0 if success
- a negative status code if failure

**5.3.1.10 EXTERNEXI OSEXIAutomaton\* rtEXIDecGetDocAutomaton (OSCTXT \* *pctxt*, size\_t *numGblElems*, const OSXMLFullQName \* *gblElems*)**

This functions returns an automaton that accepts the built-in document grammar.

This grammar is not extensible, so all the event codes created are immutable.

**Parameters:**

*pctxt* Pointer to context block structure.

**Returns:**

Pointer to document automaton.

**5.3.1.11 EXTERNEXI OSEXIAutomaton\* rtEXIDecGetElemAutomaton (OSCTXT \* *pctxt*, OSXMLFullQName \* *pqname*)**

This function returns an automaton that accepts the built-in element grammar.

The `elementName` is associated with the automaton.

**Parameters:**

*pctxt* Pointer to context block structure.

*pqname* Pointer to element QName.

**Returns:**

Pointer to element automaton.

**5.3.1.12 EXTERNEXI OSBOOL rtEXIDecHasNext (OSCTXT \* *pctxt*)**

This functions checks for additional events in the decode stream.

**Parameters:**

*pctxt* Pointer to a context structure.

**Returns:**

True if there are more decoding events or false otherwise.

**5.3.1.13 EXTERNEXI int rtEXIDecIntValue (OSCTXT \* *pctxt*, OSINT32 \* *pvalue*)**

This function decodes a signed integer value.

**Parameters:**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to integer to receive decoded value.

**Returns:**

Completion status of operation:

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

**5.3.1.14 EXTERNEXI int rtEXIDecLocalName (OSCTXT \* *pctxt*, const OSUTF8CHAR \*\* *ppname*)**

Returns the local name associated with the current event.

For event type SE and AT, it returns the local name of the element and attribute, respectively. For all other event types, the value returned by this method is undefined.

**Parameters:**

*pctxt* Pointer to a context structure.

*ppname* Pointer to variable to receive decoded data. Memory for the decoded name is allocated with `rtxMemAlloc`. It must be freed with `rtxMemFreePtr` or it will be freed when all memory in the context is freed (`rtxMemFreeAll` or `rtxFreeContext`).

**Returns:**

Status of the operation:

- 0 if success
- a negative status code if failure

### 5.3.1.15 EXTERNEXI int rtEXIDecNamespaceURI (OSCTXT \* *pctxt*, const OSUTF8CHAR \*\* *ppNSURI*)

Returns the namespace associated with the current event.

For event type SE and AT, it returns the namespace URI of the element and attribute, respectively. For NS, it returns the URI bound to the prefix in the declaration. For all other event types, the value returned by this function is undefined.

#### Parameters:

*pctxt* Pointer to a context structure.

*ppNSURI* Pointer to variable to receive decoded data. Memory for the decoded name is allocated with `rtxMemAlloc`. It must be freed with `rtxMemFreePtr` or it will be freed when all memory in the context is freed (`rtxMemFreeAll` or `rtxFreeContext`).

#### Returns:

Status of the operation:

- 0 if success
- a negative status code if failure

### 5.3.1.16 EXTERNEXI int rtEXIDecNBitUIntValue (OSCTXT \* *pctxt*, OSUINT32 \* *pvalue*, OSUINT32 *nbits*)

This function decodes an unsigned integer value from a bit field of the given width.

#### Parameters:

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned integer to receive decoded value.

*nbits* Size of bit field.

#### Returns:

Completion status of operation:

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

### 5.3.1.17 EXTERNEXI OSEXIDecStringTable\* rtEXIDecNewStringTable (OSCTXT \* *pctxt*, size\_t *capacity*)

This function allocates and initializes a new string table structure.

#### Parameters:

*pctxt* Pointer to context block structure.

*capacity* Capacity of the array list or zero to use default.

#### Returns:

Allocated string table structure.

### 5.3.1.18 EXTERNEXI int rtEXIDecNextEventType (OSCTXT \* *pctxt*, OSEXIEventType \* *pEventType*)

Returns the next OSEXIEventType read by this decoder.

Attributes and namespaces are reported as separate events.

#### Parameters:

*pctxt* Pointer to a context structure.

*pEventType* Pointer to variable to receive decoded data.

#### Returns:

Status of the operation:

- 0 if success
- a negative status code if failure

### 5.3.1.19 EXTERNEXI int rtEXIDecoderInit (OSCTXT \* *pctxt*)

This function initializes the decoder.

It must be called before any other decode functions when decoding a document or fragment.

#### Parameters:

*pctxt* Pointer to a context structure.

#### Returns:

Status of the operation:

- 0 if success
- a negative status code if failure

### 5.3.1.20 EXTERNEXI int rtEXIDecPrefix (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *uri*, const OSUTF8CHAR \*\* *ppPrefix*)

Returns the namespace associated with the current event.

For event type SE and AT, it returns the namespace URI of the element and attribute, respectively. For NS, it returns the URI bound to the prefix in the declaration. For all other event types, the value returned by this function is undefined.

#### Parameters:

*pctxt* Pointer to a context structure.

*uri* Namespace URI that prefix is associated with.

*ppPrefix* Pointer to variable to receive decoded data. Memory for the decoded name is allocated with rtxMemAlloc. It must be freed with rtxMemFreePtr or it will be freed when all memory in the context is freed (rtxMemFreeAll or rtxFreeContext).

#### Returns:

Status of the operation:

- 0 if success
- a negative status code if failure

**5.3.1.21 EXTERNEXI int rtEXIDecProcessingInstruction (OSCTXT \* *pctxt*, const OSUTF8CHAR \*\* *ppTarget*, const OSUTF8CHAR \*\* *ppData*)**

This function decodes an XML processing instruction (PI).

**Parameters:**

*pctxt* Pointer to a context structure.

*ppTarget* Pointer to variable to receive decoded PI target. Memory for the decoded name is allocated with `rtxMemAlloc`. It must be freed with `rtxMemFreePtr` or it will be freed when all memory in the context is freed (`rtxMemFreeAll` or `rtxFreeContext`).

*ppData* Pointer to variable to receive decoded PI data. Memory for the decoded name is allocated with `rtxMemAlloc`. It must be freed with `rtxMemFreePtr` or it will be freed when all memory in the context is freed (`rtxMemFreeAll` or `rtxFreeContext`).

**Returns:**

Status of the operation:

- 0 if success
- a negative status code if failure

**5.3.1.22 EXTERNEXI int rtEXIDecQName (OSCTXT \* *pctxt*, OSXMLFullQName \* *pqname*)**

Returns the QName associated with the current event.

For SE and AT it returns the name of the element and attribute, respectively. For all other event types, the value returned by this method is undefined.

**Parameters:**

*pctxt* Pointer to a context structure.

*pqname* Pointer to variable to receive decoded data. Memory for the decoded name components is allocated with `rtxMemAlloc`. It must be freed with `rtxMemFreePtr` or it will be freed when all memory in the context is freed (`rtxMemFreeAll` or `rtxFreeContext`).

**Returns:**

Status of the operation:

- 0 if success
- a negative status code if failure

**5.3.1.23 EXTERNEXI int rtEXIDecReset (OSCTXT \* *pctxt*)**

Resets the decoder for decoding a new message instance.

The decoder's state after calling this method is identical to that of a newly created instance.

**Parameters:**

*pctxt* Pointer to a context structure.

**Returns:**

Status of the operation:

- 0 if success
- a negative status code if failure

### 5.3.1.24 EXTERNEXI int rtEXIDecString (OSCTXT \* *pctxt*, const OSXMLFullQName \* *pqname*, const OSUTF8CHAR \*\* *ppvalue*)

Returns the value associated with the current event.

For character (CH) and comment (CM), it returns the corresponding characters. For attribute (AT) it returns the attribute value. For entity reference (ER), it returns the entity's name. For all other event types, the value returned by this method is undefined.

**Parameters:**

*pctxt* Pointer to a context structure.

*pqname* QName the value is associated with. This should be the QName that was decoded from the stream prior to calling this function.

*ppvalue* Pointer to variable to receive decoded data. Memory for the decoded name is allocated with `rtxMemAlloc`. It must be freed with `rtxMemFreePtr` or it will be freed when all memory in the context is freed (`rtxMemFreeAll` or `rtxFreeContext`).

**Returns:**

Status of the operation:

- 0 if success
- a negative status code if failure

### 5.3.1.25 EXTERNEXI int rtEXIDecStringCHEvent (OSCTXT \* *pctxt*, const OSXMLFullQName \* *pqname*, const OSUTF8CHAR \*\* *ppvalue*)

This function decodes a CH event (assumed to be a one-bit field with code 0) followed by string content.

**Parameters:**

*pctxt* Pointer to a context structure.

*pqname* QName the value is associated with. This should be the QName that was decoded from the stream prior to calling this function.

*ppvalue* Pointer to variable to receive decoded data. Memory for the decoded name is allocated with `rtxMemAlloc`. It must be freed with `rtxMemFreePtr` or it will be freed when all memory in the context is freed (`rtxMemFreeAll` or `rtxFreeContext`).

**Returns:**

Status of the operation:

- 0 if success
- a negative status code if failure

### 5.3.1.26 EXTERNEXI int rtEXIDecStringLength (OSCTXT \* *pctxt*)

Returns the length of the string returned by `rtEXIDecString`.

#### Parameters:

*pctxt* Pointer to a context structure.

#### Returns:

The length of the string or a negative status code if decoding failed.

### 5.3.1.27 EXTERNEXI OSUINT32 rtEXIDecStringTableAdd (OSCTXT \* *pctxt*, OSEXIDecStringTable \* *pstrtab*, const OSUTF8CHAR \* *str*)

This function adds a string to the given string table.

The string is not added if it already exists in the table. Its compact identifier is returned in either case.

#### Parameters:

*pctxt* Pointer to context block structure.

*pstrtab* Pointer to string table structure.

*str* Pointer to string to be added to table. Memory for the string is assumed to have been allocated by the user using `rtxMemAlloc`. It will be freed when the table is cleared.

#### Returns:

Index of string in table. Note that this is an unsigned value and therefore a negative value cannot be used to signal an error condition as in other functions. In this case, the user must check the status value within the error information in the context to determine if an error occurred.

### 5.3.1.28 EXTERNEXI void rtEXIDecStringTableClear (OSCTXT \* *pctxt*, OSEXIDecStringTable \* *pstrtab*)

This function clears all strings out of the existing table.

Memory for all strings is freed using `rtxMemFreePtr`.

#### Parameters:

*pctxt* Pointer to context block structure.

*pstrtab* Pointer to string table structure.

### 5.3.1.29 EXTERNEXI const OSUTF8CHAR\* rtEXIDecStringTableGetString (OSEXIDecStringTable \* *pstrtab*, OSUINT32 *index*)

This function gets the string at the given index (i.e. compact identifier) in the given string table.

#### Parameters:

*pstrtab* Pointer to string table structure.

*index* Index to string in table.

**Returns:**

Pointer to string in table.

**5.3.1.30 EXTERNEXI void rtEXIDecStringTableInit (OSCTXT \* *pctxt*, OSEXIDecStringTable \* *pstrtab*, size\_t *capacity*)**

This function initializes the given string table structure.

**Parameters:**

*pctxt* Pointer to context block structure.

*pstrtab* Pointer to string table structure.

*capacity* Capacity of the array list or zero to use default.

**5.3.1.31 EXTERNEXI int rtEXIDecStringToCharArray (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *target*, size\_t *start*, size\_t *length*)**

Similar to `rtEXIDecString` but the characters are copied into a fixed-size character array.

**Parameters:**

*pctxt* Pointer to a context structure.

*target* Destination array in which to copy the chars.

*start* Start index in target where the chars are copied.

*length* Maximum number of chars to copy.

**Returns:**

The actual number of chars copied into target or a negative error code if decoding failed.

**5.3.1.32 EXTERNEXI OSUINT32 rtEXIDecStrTabsAddGlobalValue (OSCTXT \* *pctxt*, OSEXIDecStringTables \* *pstrtabs*, const OSUTF8CHAR \* *value*)**

This function will add a string value to the global value string table partition.

**Parameters:**

*pctxt* Pointer to context block structure.

*pstrtabs* Pointer to full string table set structure.

*value* Value to be added.

**Returns:**

Index (compact identifier) of the added value.

**5.3.1.33 EXTERNEXI OSUINT32 rtEXIDecStrTabsAddLocalName (OSCTXT \* *pctxt*, OSEXIDecStringTables \* *pstrtabs*, const OSUTF8CHAR \* *uri*, const OSUTF8CHAR \* *name*)**

This function adds the given local name to the local name table partition identified by the given URI.

**Parameters:**

- pctxt* Pointer to context block structure.
- pstrtabs* Pointer to full string table set structure.
- uri* URI identifier of local name table partition.
- name* Name to be added.

**Returns:**

Index (compact identifier) of the added value.

**5.3.1.34 EXTERNEXI OSUINT32 rtEXIDecStrTabsAddLocalValue (OSCTXT \* *pctxt*, OSEXIDecStringTables \* *pstrtabs*, const OSXMLFullQName \* *qname*, const OSUTF8CHAR \* *value*)**

This function adds the given local value to the local value table partition identified by the given QName.

**Parameters:**

- pctxt* Pointer to context block structure.
- pstrtabs* Pointer to full string table set structure.
- qname* QName identifier of local value table partition.
- value* Value to be added.

**Returns:**

Index (compact identifier) of the added value.

**5.3.1.35 EXTERNEXI OSUINT32 rtEXIDecStrTabsAddPrefix (OSCTXT \* *pctxt*, OSEXIDecStringTables \* *pstrtabs*, const OSUTF8CHAR \* *uri*, const OSUTF8CHAR \* *prefix*)**

This function adds the given prefix to the prefix table partition identified by the given URI.

**Parameters:**

- pctxt* Pointer to context block structure.
- pstrtabs* Pointer to full string table set structure.
- uri* URI identifier of prefix table partition.
- prefix* Prefix to be added.

**Returns:**

Index (compact identifier) of the added value.

**5.3.1.36 EXTERNEXI OSUINT32 rtEXIDecStrTabsAddURI (OSCTXT \* *pctxt*, OSEXIDecStringTables \* *pstrtabs*, const OSUTF8CHAR \* *uri*)**

This function will add a URI to the URI string table partition.

**Parameters:**

*pctxt* Pointer to context block structure.  
*pstrtabs* Pointer to full string table set structure.  
*uri* URI to be added.

**Returns:**

Index (compact identifier) of the added URI.

**5.3.1.37 EXTERNEXI void rtEXIDecStrTabsClear (OSCTXT \* *pctxt*, OSEXIDecStringTables \* *pstrtabs*)**

This function clears all EXI string table partitions.

**Parameters:**

*pctxt* Pointer to context block structure.  
*pstrtabs* Pointer to string table set structure.

**5.3.1.38 EXTERNEXI const OSUTF8CHAR\* rtEXIDecStrTabsGetGlobalValue (OSEXIDecStringTables \* *pstrtabs*, OSUINT32 *index*)**

This function will get the compact identifier of the given string value from the global value string table partition.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.  
*index* Global value table compact identifier.

**Returns:**

Index (compact identifier) or the null index identifier (OSEXINULLINDEX) if not found.

**5.3.1.39 EXTERNEXI OSUINT32 rtEXIDecStrTabsGetGlobalValueTableSize (OSEXIDecStringTables \* *pstrtabs*)**

This function returns the current number of entries in the global value string table partition.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.

**Returns:**

Current number of entries in the partition.

**5.3.1.40 EXTERNEXI const OSUTF8CHAR\* rtEXIDecStrTabsGetLocalName (OSEXIDecStringTables \* pstrtabs, const OSUTF8CHAR \* uri, OSUINT32 index)**

This function will get the compact identifier of the given localName from the localName string table partition identified by the given URI.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.

*uri* URI identifier of localName table partition.

*index* Name table compact identifier.

**Returns:**

Index (compact identifier) of the value.

**5.3.1.41 EXTERNEXI OSUINT32 rtEXIDecStrTabsGetLocalNameTableSize (OSEXIDecStringTables \* pstrtabs, const OSUTF8CHAR \* uri)**

This function returns the current number of entries in the localName string table partition identified by the given URI.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.

*uri* URI identifier of localName table partition.

**Returns:**

Current number of entries in the partition.

**5.3.1.42 EXTERNEXI const OSUTF8CHAR\* rtEXIDecStrTabsGetLocalValue (OSEXIDecStringTables \* pstrtabs, const OSXMLFullQName \* qname, OSUINT32 index)**

This function will get the compact identifier of the given local value from the local value string table partition identified by the given QName.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.

*qname* Identifier of table partition.

*index* Local value table compact identifier.

**Returns:**

Index (compact identifier) of the value.

**5.3.1.43 EXTERNEXI OSUINT32 rtEXIDecStrTabsGetLocalValueTableSize (OSEXIDecStringTables \* pstrtabs, const OSXMLFullQName \* qname)**

This function returns the current number of entries in the local value string table partition identified by the given QName.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.  
*qname* Identifier of table partition.

**Returns:**

Current number of entries in the partition.

**5.3.1.44 EXTERNEXI const OSUTF8CHAR\* rtEXIDecStrTabsGetPrefix (OSEXIDecStringTables \* pstrtabs, const OSUTF8CHAR \* uri, OSUINT32 index)**

This function will get the compact identifier of the given prefix from the prefix string table partition identified by the given URI.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.  
*uri* URI identifier of prefix table partition.  
*index* Prefix table compact identifier.

**Returns:**

Index (compact identifier) of the value.

**5.3.1.45 EXTERNEXI OSUINT32 rtEXIDecStrTabsGetPrefixTableSize (OSEXIDecStringTables \* pstrtabs, const OSUTF8CHAR \* uri)**

This function returns the current number of entries in the prefix string table partition identified by the given URI.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.  
*uri* URI identifier of prefix table partition.

**Returns:**

Current number of entries in the partition.

**5.3.1.46 EXTERNEXI const OSUTF8CHAR\* rtEXIDecStrTabsGetURI (OSEXIDecStringTables \* pstrtabs, OSUINT32 index)**

This function will get the compact identifier of the given URI from the URI string table partition.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.  
*index* URI table compact identifier.

**Returns:**

Index (compact identifier) or the null index identifier (OSEXINULLINDEX) if not found.

#### 5.3.1.47 EXTERNEXI OSUINT32 rtEXIDecStrTabsGetURITableSize (OSEXIDecStringTables \* pstrtabs)

This function returns the current number of entries in the URI string table partition.

##### Parameters:

*pstrtabs* Pointer to full string table set structure.

##### Returns:

Current number of entries in the partition.

#### 5.3.1.48 EXTERNEXI void rtEXIDecStrTabsInit (OSCTXT \* pctxt, OSEXIDecStringTables \* pstrtabs)

This function initializes all EXI string table partitions.

##### Parameters:

*pctxt* Pointer to context block structure.

*pstrtabs* Pointer to string table set structure.

#### 5.3.1.49 EXTERNEXI int rtEXIDecUIntValue (OSCTXT \* pctxt, OSUINT32 \* pvalue)

This function decodes an unsigned integer value.

##### Parameters:

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned integer to receive decoded value.

##### Returns:

Completion status of operation:

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

#### 5.3.1.50 EXTERNEXI int rtEXIDecUTF8Chars (OSCTXT \* pctxt, OSUTF8CHAR \*\* ppvalue, OSUINT32 nchars)

This function reads the given number of characters from the decode stream and creates a UTF-8 string.

##### Parameters:

*pctxt* Pointer to context block structure.

*ppvalue* Pointer to character string pointer to receive decoded string value. Memory for the string is allocated using `rtxMemAlloc`. It is freed using `rtxMemFreePtr` or when the context is freed.

*nchars* Number of characters to read from stream.

##### Returns:

Completion status of operation:

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

### 5.3.1.51 EXTERNEXI int rtEXIDecUTF8Str (OSCTXT \* *pctxt*, OSUTF8CHAR \*\* *ppvalue*)

This function decodes a UTF-8 string value.

#### Parameters:

*pctxt* Pointer to context block structure.

*ppvalue* Pointer to character string pointer to receive decoded string value. Memory for the string is allocated using `rtxMemAlloc`. It is freed using `rtxMemFreePtr` or when the context is freed.

#### Returns:

Completion status of operation:

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

## 5.4 EXI encode structures and functions.

### Classes

- struct [OSEXIEncStringTable](#)  
*This structure defines the structure of the various string table partitions used by the encoder.*
- struct [OSEXIEncStringTables](#)  
*This structure defines the complete set of string table partitions used by the encoder.*

### Functions

- EXTERNEXI int [rtEXIEncAtmAddTransition](#) (OSCTXT \*pctxt, [OSEXIAutomaton](#) \*pAutomaton, [OSEXIS-  
tate](#) fromState, [OSEXIS-  
tate](#) toState, const [OSEXIEvent](#) \*pEvent, const [OSEXIEventCode](#) \*pEventCode)  
*This function adds a transition between two states.*
- EXTERNEXI [OSEXIEventCode](#) \* [rtEXIEncAutomatonAdvance](#) (OSCTXT \*pctxt, [OSEXIAutomaton](#) \*p-  
Automaton, const [OSEXIEvent](#) \*pEvent, OSBOOL dynamicItems)  
*This function advances the encoder automaton based on event, adding new transitions if dynamicItems is set to  
true and either SE(\*) or AT(\*) are matched instead of SE(qname) or AT(qname), respectively.*
- EXTERNEXI [OSEXIAutomaton](#) \* [rtEXIEncGetDocAutomaton](#) (OSCTXT \*pctxt, size\_t numGblElems, const  
OSXMLFullQName \*gblElems)  
*This functions returns an automaton that accepts the built-in document grammar.*
- EXTERNEXI [OSEXIAutomaton](#) \* [rtEXIEncGetElemAutomaton](#) (OSCTXT \*pctxt, OSXMLFullQName  
\*pqname)  
*This function returns an automaton that accepts the built-in element grammar.*
- EXTERNEXI void [rtEXIEncStringTableInit](#) (OSCTXT \*pctxt, [OSEXIEncStringTable](#) \*pstrtab, size\_t capac-  
ity)  
*This function initializes the given string table structure.*
- EXTERNEXI [OSEXIEncStringTable](#) \* [rtEXIEncNewStringTable](#) (OSCTXT \*pctxt, size\_t capacity)  
*This function allocates and initializes a new string table structure.*
- EXTERNEXI void [rtEXIEncStringTableClear](#) (OSCTXT \*pctxt, [OSEXIEncStringTable](#) \*pstrtab)  
*This function clears all strings out of the existing table.*
- EXTERNEXI OSUINT32 [rtEXIEncStringTableAdd](#) (OSCTXT \*pctxt, [OSEXIEncStringTable](#) \*pstrtab, const  
OSUTF8CHAR \*str)  
*This function adds a string to the given string table.*
- EXTERNEXI OSUINT32 [rtEXIEncStringTableGetIndex](#) ([OSEXIEncStringTable](#) \*pstrtab, const OS-  
UTF8CHAR \*str)  
*This function gets the index (i.e.*
- EXTERNEXI void [rtEXIEncStrTabsInit](#) (OSCTXT \*pctxt, [OSEXIEncStringTables](#) \*pstrtabs)  
*This function initializes all EXI string table partitions.*

- EXTERNEXI void [rtEXIEncStrTabsClear](#) (OSCTXT \*pctxt, [OSEXIEncStringTables](#) \*pstrtabs)  
*This function clears all EXI string table partitions.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsAddURI](#) (OSCTXT \*pctxt, [OSEXIEncStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri)  
*This function will add a URI to the URI string table partition.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetURIID](#) ([OSEXIEncStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri)  
*This function will get the compact identifier of the given URI from the URI string table partition.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetURITableSize](#) ([OSEXIEncStringTables](#) \*pstrtabs)  
*This function returns the current number of entries in the URI string table partition.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsAddPrefix](#) (OSCTXT \*pctxt, [OSEXIEncStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*prefix)  
*This function adds the given prefix to the prefix table partition identified by the given URI.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetPrefixID](#) ([OSEXIEncStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*prefix)  
*This function will get the compact identifier of the given prefix from the prefix string table partition identified by the given URI.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetPrefixTableSize](#) ([OSEXIEncStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri)  
*This function returns the current number of entries in the prefix string table partition identified by the given URI.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsAddLocalName](#) (OSCTXT \*pctxt, [OSEXIEncStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*name)  
*This function adds the given local name to the local name table partition identified by the given URI.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetLocalNameID](#) ([OSEXIEncStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*name)  
*This function will get the compact identifier of the given localName from the localName string table partition identified by the given URI.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetLocalNameTableSize](#) ([OSEXIEncStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri)  
*This function returns the current number of entries in the localName string table partition identified by the given URI.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsAddLocalValue](#) (OSCTXT \*pctxt, [OSEXIEncStringTables](#) \*pstrtabs, const OSXMLFullQName \*qname, const OSUTF8CHAR \*value)  
*This function adds the given local value to the local value table partition identified by the given QName.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetLocalValueID](#) ([OSEXIEncStringTables](#) \*pstrtabs, const OSXMLFullQName \*qname, const OSUTF8CHAR \*value)  
*This function will get the compact identifier of the given local value from the local value string table partition identified by the given QName.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetLocalValueTableSize](#) ([OSEXIEncStringTables](#) \*pstrtabs, const OSXMLFullQName \*qname)

*This function returns the current number of entries in the local value string table partition identified by the given QName.*

- EXTERNEXI OSUINT32 **rtEXIEncStrTabsAddGlobalValue** (OSCTXT \*pctxt, OSEXIEncStringTables \*pstrtabs, const OSUTF8CHAR \*value)

*This function will add a string value to the global value string table partition.*

- EXTERNEXI OSUINT32 **rtEXIEncStrTabsGetGlobalValueID** (OSEXIEncStringTables \*pstrtabs, const OSUTF8CHAR \*value)

*This function will get the compact identifier of the given string value from the global value string table partition.*

- EXTERNEXI OSUINT32 **rtEXIEncStrTabsGetGlobalValueTableSize** (OSEXIEncStringTables \*pstrtabs)

*This function returns the current number of entries in the global value string table partition.*

## 5.4.1 Function Documentation

### 5.4.1.1 EXTERNEXI int rtEXIEncAtmAddTransition (OSCTXT \*pctxt, OSEXIAutomaton \*pAutomaton, OSEXIState fromState, OSEXIState toState, const OSEXIEvent \*pEvent, const OSEXIEventCode \*pEventCode)

This function adds a transition between two states.

The transition is defined by a pair of states, and event and event code.

#### Parameters:

*pctxt* Pointer to context block structure.

*pAutomaton* Pointer to automaton structure.

*fromState* The origin state for this transition.

*toState* The destination state for this transition.

*pEvent* The event on which this transition occurs.

*pEventCode* The event code returned after the transition.

#### Returns:

Zero if operation was successful; a negative status code otherwise.

### 5.4.1.2 EXTERNEXI OSEXIEventCode\* rtEXIEncAutomatonAdvance (OSCTXT \*pctxt, OSEXIAutomaton \*pAutomaton, const OSEXIEvent \*pEvent, OSBOOL dynamicItems)

This function advances the encoder automaton based on event, adding new transitions if `dynamicItems` is set to `true` and either `SE(*)` or `AT(*)` are matched instead of `SE(qname)` or `AT(qname)`, respectively.

Dynamic transitions for CH are also added according to the spec.

#### Parameters:

*pctxt* Pointer to context block structure.

*pAutomaton* Pointer to automaton structure.

*pEvent* Event on which to advance the automaton.

*dynamicItems* Flag to add dynamic transitions to the automaton.

**Returns:**

The event code returned as a result of this transition.

**5.4.1.3 EXTERNEXI OSEXIAutomaton\* rtEXIEncGetDocAutomaton (OSCTXT \* *pctxt*, size\_t *numGblElems*, const OSXMLFullQName \* *gblElems*)**

This functions returns an automaton that accepts the built-in document grammar.

This grammar is not extensible, so all the event codes created are immutable.

**Parameters:**

*pctxt* Pointer to context block structure.

*numGblElems* Number of global elements.

*gblElems* Array of global element objects.

**Returns:**

Pointer to document automaton.

**5.4.1.4 EXTERNEXI OSEXIAutomaton\* rtEXIEncGetElemAutomaton (OSCTXT \* *pctxt*, OSXMLFullQName \* *pqname*)**

This function returns an automaton that accepts the built-in element grammar.

The `elementName` is associated with the automaton.

**Parameters:**

*pctxt* Pointer to context block structure.

*pqname* Pointer to element QName.

**Returns:**

Pointer to element automaton.

**5.4.1.5 EXTERNEXI OSEXIEncStringTable\* rtEXIEncNewStringTable (OSCTXT \* *pctxt*, size\_t *capacity*)**

This function allocates and initializes a new string table structure.

**Parameters:**

*pctxt* Pointer to context block structure.

*capacity* Capacity of the hash map or zero to use default.

**Returns:**

Allocated string table structure.

#### 5.4.1.6 EXTERNEXI OSUINT32 rtEXIEncStringTableAdd (OSCTXT \* *pctxt*, OSEXIEncStringTable \* *pstrtab*, const OSUTF8CHAR \* *str*)

This function adds a string to the given string table.

The string is not added if it already exists in the table. Its compact identifier is returned in either case.

##### Parameters:

*pctxt* Pointer to context block structure.

*pstrtab* Pointer to string table structure.

*str* Pointer to string to be added to table. A copy of the string is not made; the pointer is simply added to the table.

##### Returns:

Index of string in table. Note that this is an unsigned value and therefore a negative value cannot be used to signal an error condition as in other functions. In this case, the user must check the status value within the error information in the context to determine if an error occurred.

#### 5.4.1.7 EXTERNEXI void rtEXIEncStringTableClear (OSCTXT \* *pctxt*, OSEXIEncStringTable \* *pstrtab*)

This function clears all strings out of the existing table.

##### Parameters:

*pctxt* Pointer to context block structure.

*pstrtab* Pointer to string table structure.

#### 5.4.1.8 EXTERNEXI OSUINT32 rtEXIEncStringTableGetIndex (OSEXIEncStringTable \* *pstrtab*, const OSUTF8CHAR \* *str*)

This function gets the index (i.e. compact identifier) of a string in the given string table.

##### Parameters:

*pstrtab* Pointer to string table structure.

*str* Pointer to string to be added to table. A copy of the string is not made; the pointer is simply added to the table.

##### Returns:

Index of string in table. Note that this is an unsigned value. If the string is not found or an error occurs, the unsigned integer max value is returned (OSUINT32\_MAX).

#### 5.4.1.9 EXTERNEXI void rtEXIEncStringTableInit (OSCTXT \* *pctxt*, OSEXIEncStringTable \* *pstrtab*, size\_t *capacity*)

This function initializes the given string table structure.

##### Parameters:

*pctxt* Pointer to context block structure.

*pstrtab* Pointer to string table structure.

*capacity* Capacity of the hash map or zero to use default.

**5.4.1.10 EXTERNEXI OSUINT32 rtEXIEncStrTabsAddGlobalValue (OSCTXT \* *pctxt*, OSEXIEncStringTables \* *pstrtabs*, const OSUTF8CHAR \* *value*)**

This function will add a string value to the global value string table partition.

**Parameters:**

- pctxt* Pointer to context block structure.
- pstrtabs* Pointer to full string table set structure.
- value* Value to be added.

**Returns:**

Index (compact identifier) of the added value.

**5.4.1.11 EXTERNEXI OSUINT32 rtEXIEncStrTabsAddLocalName (OSCTXT \* *pctxt*, OSEXIEncStringTables \* *pstrtabs*, const OSUTF8CHAR \* *uri*, const OSUTF8CHAR \* *name*)**

This function adds the given local name to the local name table partition identified by the given URI.

**Parameters:**

- pctxt* Pointer to context block structure.
- pstrtabs* Pointer to full string table set structure.
- uri* URI identifier of local name table partition.
- name* Name to be added.

**Returns:**

Index (compact identifier) of the added value.

**5.4.1.12 EXTERNEXI OSUINT32 rtEXIEncStrTabsAddLocalValue (OSCTXT \* *pctxt*, OSEXIEncStringTables \* *pstrtabs*, const OSXMLFullQName \* *qname*, const OSUTF8CHAR \* *value*)**

This function adds the given local value to the local value table partition identified by the given QName.

**Parameters:**

- pctxt* Pointer to context block structure.
- pstrtabs* Pointer to full string table set structure.
- qname* QName identifier of local value table partition.
- value* Value to be added.

**Returns:**

Index (compact identifier) of the added value.

**5.4.1.13 EXTERNEXI OSUINT32 rtEXIEncStrTabsAddPrefix (OSCTXT \* *pctxt*, OSEXIEncStringTables \* *pstrtabs*, const OSUTF8CHAR \* *uri*, const OSUTF8CHAR \* *prefix*)**

This function adds the given prefix to the prefix table partition identified by the given URI.

**Parameters:**

- pctxt* Pointer to context block structure.
- pstrtabs* Pointer to full string table set structure.
- uri* URI identifier of prefix table partition.
- prefix* Prefix to be added.

**Returns:**

Index (compact identifier) of the added value.

**5.4.1.14 EXTERNEXI OSUINT32 rtEXIEncStrTabsAddURI (OSCTXT \* *pctxt*, OSEXIEncStringTables \* *pstrtabs*, const OSUTF8CHAR \* *uri*)**

This function will add a URI to the URI string table partition.

**Parameters:**

- pctxt* Pointer to context block structure.
- pstrtabs* Pointer to full string table set structure.
- uri* URI to be added.

**Returns:**

Index (compact identifier) of the added URI.

**5.4.1.15 EXTERNEXI void rtEXIEncStrTabsClear (OSCTXT \* *pctxt*, OSEXIEncStringTables \* *pstrtabs*)**

This function clears all EXI string table partitions.

**Parameters:**

- pctxt* Pointer to context block structure.
- pstrtabs* Pointer to string table set structure.

**5.4.1.16 EXTERNEXI OSUINT32 rtEXIEncStrTabsGetGlobalValueID (OSEXIEncStringTables \* *pstrtabs*, const OSUTF8CHAR \* *value*)**

This function will get the compact identifier of the given string value from the global value string table partition.

**Parameters:**

- pstrtabs* Pointer to full string table set structure.
- value* Value to be looked up.

**Returns:**

Index (compact identifier) or the null index identifier (OSEXINULLINDEX) if not found.

**5.4.1.17 EXTERNEXI OSUINT32 rtEXIEncStrTabsGetGlobalValueTableSize (OSEXIEncStringTables \* pstrtabs)**

This function returns the current number of entries in the global value string table partition.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.

**Returns:**

Current number of entries in the partition.

**5.4.1.18 EXTERNEXI OSUINT32 rtEXIEncStrTabsGetLocalNameID (OSEXIEncStringTables \* pstrtabs, const OSUTF8CHAR \* uri, const OSUTF8CHAR \* name)**

This function will get the compact identifier of the given localName from the localName string table partition identified by the given URI.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.

*uri* URI identifier of localName table partition.

*name* Name to be looked up.

**Returns:**

Index (compact identifier) of the value.

**5.4.1.19 EXTERNEXI OSUINT32 rtEXIEncStrTabsGetLocalNameTableSize (OSEXIEncStringTables \* pstrtabs, const OSUTF8CHAR \* uri)**

This function returns the current number of entries in the localName string table partition identified by the given URI.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.

*uri* URI identifier of localName table partition.

**Returns:**

Current number of entries in the partition.

**5.4.1.20 EXTERNEXI OSUINT32 rtEXIEncStrTabsGetLocalValueID (OSEXIEncStringTables \* pstrtabs, const OSXMLFullQName \* qname, const OSUTF8CHAR \* value)**

This function will get the compact identifier of the given local value from the local value string table partition identified by the given QName.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.

*qname* Identifier of table partition.

*value* Value to be looked up.

**Returns:**

Index (compact identifier) of the value.

**5.4.1.21 EXTERNEXI OSUINT32 rtEXIEncStrTabsGetLocalValueTableSize (OSEXIEncStringTables \* pstrtabs, const OSXMLFullQName \* qname)**

This function returns the current number of entries in the local value string table partition identified by the given QName.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.

*qname* Identifier of table partition.

**Returns:**

Current number of entries in the partition.

**5.4.1.22 EXTERNEXI OSUINT32 rtEXIEncStrTabsGetPrefixID (OSEXIEncStringTables \* pstrtabs, const OSUTF8CHAR \* uri, const OSUTF8CHAR \* prefix)**

This function will get the compact identifier of the given prefix from the prefix string table partition identified by the given URI.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.

*uri* URI identifier of prefix table partition.

*prefix* Prefix to be looked up.

**Returns:**

Index (compact identifier) of the value.

**5.4.1.23 EXTERNEXI OSUINT32 rtEXIEncStrTabsGetPrefixTableSize (OSEXIEncStringTables \* pstrtabs, const OSUTF8CHAR \* uri)**

This function returns the current number of entries in the prefix string table partition identified by the given URI.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.

*uri* URI identifier of prefix table partition.

**Returns:**

Current number of entries in the partition.

**5.4.1.24 EXTERNEXI OSUINT32 rtEXIEncStrTabsGetURIID (OSEXIEncStringTables \* pstrtabs, const OSUTF8CHAR \* uri)**

This function will get the compact identifier of the given URI from the URI string table partition.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.

*uri* URI to be looked up.

**Returns:**

Index (compact identifier) or the null index identifier (OSEXINULLINDEX) if not found.

**5.4.1.25 EXTERNEXI OSUINT32 rtEXIEncStrTabsGetURITableSize (OSEXIEncStringTables \* pstrtabs)**

This function returns the current number of entries in the URI string table partition.

**Parameters:**

*pstrtabs* Pointer to full string table set structure.

**Returns:**

Current number of entries in the partition.

**5.4.1.26 EXTERNEXI void rtEXIEncStrTabsInit (OSCTXT \* ptxt, OSEXIEncStringTables \* pstrtabs)**

This function initializes all EXI string table partitions.

**Parameters:**

*ptxt* Pointer to context block structure.

*pstrtabs* Pointer to string table set structure.

## 5.5 EXI event code definitions and functions.

### Classes

- struct [OSEXIEventCode](#)  
*A structure representing a production's event code.*
- struct [OSEXIEventCodeGroup](#)  
*An EventCodeGroup is a group of related event codes.*

### Defines

- #define [rtEXISetEventCode1](#)(pEventCode, part1) rtEXISetEventCode3(pEventCode, part1, OSINT32\_MIN, OSINT32\_MIN)  
*This macro sets a one-part event code.*
- #define [rtEXISetEventCode2](#)(pEventCode, part1, part2) rtEXISetEventCode3(pEventCode, part1, part2, OSINT32\_MIN)  
*This macro sets a two-part event code.*
- #define [rtEXIEventCodeGroupHasPart1](#)(pecgrp, part1) rtxDynBitSetTestBit(&pecgrp → part1Set,part1)  
*This macro returns true if there is an event code in this group whose length is 1 and whose first part is equal to part1.*
- #define [rtEXIEventCodeGroupHasPart2](#)(pecgrp, part2) rtxDynBitSetTestBit(&pecgrp → part2Set,part2)  
*This macro returns true if there is an event code in this group whose length is 2 and whose first part is equal to part2.*

### Functions

- EXTERNEXI int [rtEXIEventCodeCompare](#) (const [OSEXIEventCode](#) \*pec1, const [OSEXIEventCode](#) \*pec2)  
*This function compares two event codes.*
- EXTERNEXI [OSEXIEventCode](#) \* [rtEXIEventCodeCopy](#) (OSCTXT \*pctxt, const [OSEXIEventCode](#) \*pec)  
*This function does a deep-copy of an event code structure.*
- EXTERNEXI OSBOOL [rtEXIEventCodesEqual](#) (const [OSEXIEventCode](#) \*pec1, const [OSEXIEventCode](#) \*pec2)  
*This function compares two event codes for equality.*
- EXTERNEXI char \* [rtEXIEventCodeToString](#) (OSCTXT \*pctxt, const [OSEXIEventCode](#) \*pec)  
*This function returns a string representation of the given event code in dot notation (part1.part2.part3).*
- EXTERNEXI OSUINT32 [rtEXIEventCodeLength](#) (const [OSEXIEventCode](#) \*pec)  
*This function returns the length of the given event code.*
- EXTERNEXI [OSEXIEventCode](#) \* [rtEXINewEventCode1](#) (OSCTXT \*pctxt, OSINT32 part1)  
*This function allocates and initializes a one-part event code.*

- EXTERNEXI `OSEXIEventCode * rtEXINewEventCode2 (OSCTXT *pctxt, OSINT32 part1, OSINT32 part2)`  
*This function allocates and initializes a two-part event code.*
- EXTERNEXI `OSEXIEventCode * rtEXINewEventCode3 (OSCTXT *pctxt, OSINT32 part1, OSINT32 part2, OSINT32 part3)`  
*This function allocates and initializes a three-part event code.*
- EXTERNEXI `void rtEXISetEventCode3 (OSEXIEventCode *pEventCode, OSINT32 part1, OSINT32 part2, OSINT32 part3)`  
*This function sets a three-part event code.*
- EXTERNEXI `void rtEXIEventCodePrint (const OSEXIEventCode *pEventCode)`  
*This function prints information on the given event code to stdout.*
- EXTERNEXI `void rtEXIEventCodeGroupInit (OSCTXT *pctxt, OSEXIEventCodeGroup *pecgrp)`  
*This function initializes an event code group structure.*
- EXTERNEXI `OSEXIEventCodeGroup * rtEXINewEventCodeGroup (OSCTXT *pctxt)`  
*This function allocates and initializes a new event code group structure.*
- EXTERNEXI `int rtEXIEventCodeGroupAdd (OSEXIEventCodeGroup *pecgrp, const OSEXIEventCode *pec)`  
*This function adds an event code to an event code group and updates the maximum part variables.*
- EXTERNEXI `OSEXIEventCodeGroup * rtEXIEventCodeGroupCopy (const OSEXIEventCodeGroup *pecgrp)`  
*This function performs a deep copy of the given event group.*
- EXTERNEXI `void rtEXIEventCodeGroupFreeMem (OSEXIEventCodeGroup *pecgrp)`  
*This function frees all memory associated with an event group.*
- EXTERNEXI `int rtEXIEventCodeGroupGetBitsPart1 (OSEXIEventCodeGroup *pecgrp)`  
*This function returns the number of bits necessary to encode the max part1 value in this group.*
- EXTERNEXI `int rtEXIEventCodeGroupGetBitsPart2 (OSEXIEventCodeGroup *pecgrp)`  
*This function returns the number of bits necessary to encode the max part2 value in this group.*
- EXTERNEXI `int rtEXIEventCodeGroupGetBitsPart3 (OSEXIEventCodeGroup *pecgrp)`  
*This function returns the number of bits necessary to encode the max part3 value in this group.*
- EXTERNEXI `void rtEXIEventCodeGroupIncrPart1 (OSCTXT *pctxt, OSEXIEventCodeGroup *pecgrp)`  
*This function increments part1 in all event codes in this group, as well as the max part1 value.*

## 5.5.1 Define Documentation

### 5.5.1.1 `#define rtEXISetEventCode1(pEventCode, part1) rtEXISetEventCode3(pEventCode, part1, OSINT32_MIN, OSINT32_MIN)`

This macro sets a one-part event code.

**Parameters:**

*pEventCode* Pointer to event code structure.

*part1* Part 1 of the event code.

Definition at line 172 of file rtEXIEventCode.h.

**5.5.1.2 #define rtEXISetEventCode2(pEventCode, part1, part2) rtEXISetEventCode3(pEventCode, part1, part2, OSINT32\_MIN)**

This macro sets a two-part event code.

**Parameters:**

*pEventCode* Pointer to event code structure.

*part1* Part 1 of the event code.

*part2* Part 2 of the event code.

Definition at line 182 of file rtEXIEventCode.h.

## 5.5.2 Function Documentation

**5.5.2.1 EXTERNEXI int rtEXIEventCodeCompare (const OSEXIEventCode \* pec1, const OSEXIEventCode \* pec2)**

This function compares two event codes.

Event codes are first compared based on length and then partwise.

**Parameters:**

*pec1* Pointer to event code to compare.

*pec2* Pointer to event code to compare.

**Returns:**

Comparison result:  $< 0 = (ec1 < ec2)$ ,  $0 = (ec1 == ec2)$ ,  $> 0 = (ec1 > ec2)$

**5.5.2.2 EXTERNEXI OSEXIEventCode\* rtEXIEventCodeCopy (OSCTXT \* pctxt, const OSEXIEventCode \* pec)**

This function does a deep-copy of an event code structure.

**Parameters:**

*pctxt* Pointer to context block structure.

*pec* Pointer to event code to copy.

**Returns:**

Copied structure. Memory for the new structure is allocated with the rtxMemAlloc run-time functions and thus must be freed with an rtxMemFree\* function. If memory allocation fails, NULL is returned.

**5.5.2.3 EXTERNEXI int rtEXIEventCodeGroupAdd (OSEXIEventCodeGroup \* pecgrp, const OSEXIEventCode \* pec)**

This function adds an event code to an event code group and updates the maximum part variables.

**Parameters:**

*pecgrp* Pointer to event code group structure.

*pec* Pointer to event code to add.

**Returns:**

Status of operation: 0 if success, negative status code if failure.

**5.5.2.4 EXTERNEXI OSEXIEventCodeGroup\* rtEXIEventCodeGroupCopy (const OSEXIEventCodeGroup \* pecgrp)**

This function performs a deep copy of the given event group.

This includes copying the group list as well as all its members.

**Parameters:**

*pecgrp* Pointer to event code group structure.

**Returns:**

Copied group structure. Memory for the new structure is allocated with the rtxMemAlloc run-time functions and thus must be freed with an rtxMemFree\* function. If memory allocation fails, NULL is returned.

**5.5.2.5 EXTERNEXI void rtEXIEventCodeGroupFreeMem (OSEXIEventCodeGroup \* pecgrp)**

This function frees all memory associated with an event group.

**Parameters:**

*pecgrp* Pointer to event code group structure.

**5.5.2.6 EXTERNEXI int rtEXIEventCodeGroupGetBitsPart1 (OSEXIEventCodeGroup \* pecgrp)**

This function returns the number of bits necessary to encode the max part1 value in this group.

**Parameters:**

*pecgrp* Pointer to event code group structure.

**Returns:**

Number of bits.

### 5.5.2.7 EXTERNEXI int rtEXIEventCodeGroupGetBitsPart2 (OSEXIEventCodeGroup \* pecgrp)

This function returns the number of bits necessary to encode the max part2 value in this group.

#### Parameters:

*pecgrp* Pointer to event code group structure.

#### Returns:

Number of bits.

### 5.5.2.8 EXTERNEXI int rtEXIEventCodeGroupGetBitsPart3 (OSEXIEventCodeGroup \* pecgrp)

This function returns the number of bits necessary to encode the max part3 value in this group.

#### Parameters:

*pecgrp* Pointer to event code group structure.

#### Returns:

Number of bits.

### 5.5.2.9 EXTERNEXI void rtEXIEventCodeGroupIncrPart1 (OSCTXT \* ptxt, OSEXIEventCodeGroup \* pecgrp)

This function increments part1 in all event codes in this group, as well as the max part1 value.

#### Parameters:

*ptxt* Pointer to a context block structure.

*pecgrp* Pointer to event code group structure.

### 5.5.2.10 EXTERNEXI void rtEXIEventCodeGroupInit (OSCTXT \* ptxt, OSEXIEventCodeGroup \* pecgrp)

This function initializes an event code group structure.

#### Parameters:

*ptxt* Pointer to a context block structure.

*pecgrp* Pointer to event code group structure.

### 5.5.2.11 EXTERNEXI OSUINT32 rtEXIEventCodeLength (const OSEXIEventCode \* pec)

This function returns the length of the given event code.

#### Parameters:

*pec* Pointer to event code.

#### Returns:

Length of event code (0 - 3)

#### 5.5.2.12 EXTERNEXI void rtEXIEventCodePrint (const OSEXIEventCode \* pEventCode)

This function prints information on the given event code to stdout.

##### Parameters:

*pEventCode* Pointer to event code structure.

#### 5.5.2.13 EXTERNEXI OSBOOL rtEXIEventCodesEqual (const OSEXIEventCode \* pec1, const OSEXIEventCode \* pec2)

This function compares two event codes for equality.

A deep compare is done (i.e. will return TRUE if either the pointers are equal or all parts inside are equal).

##### Parameters:

*pec1* Pointer to event code to compare.

*pec2* Pointer to event code to compare.

##### Returns:

Boolean comparison result.

#### 5.5.2.14 EXTERNEXI char\* rtEXIEventCodeToString (OSCTXT \* pctxt, const OSEXIEventCode \* pec)

This function returns a string representation of the given event code in dot notation (part1.part2.part3).

Memory is allocated for the string using the rtxMemAlloc run-time function and must be freed using one the rtxMemFree functions.

##### Parameters:

*pctxt* Pointer to context block structure.

*pec* Pointer to event code.

##### Returns:

Pointer to string version of code or NULL if no dynamic memory is available.

#### 5.5.2.15 EXTERNEXI OSEXIEventCode\* rtEXINewEventCode1 (OSCTXT \* pctxt, OSINT32 part1)

This function allocates and initializes a one-part event code.

##### Parameters:

*pctxt* Pointer to context block structure.

*part1* Part 1 of the event code.

##### Returns:

Pointer to allocated event code structure. NULL if no dynamic memory available.

**5.5.2.16 EXTERNEXI OSEXIEventCode\* rtEXINewEventCode2 (OSCTXT \* *pctxt*, OSINT32 *part1*, OSINT32 *part2*)**

This function allocates and initializes a two-part event code.

**Parameters:**

*pctxt* Pointer to context block structure.

*part1* Part 1 of the event code.

*part2* Part 2 of the event code.

**Returns:**

Pointer to allocated event code structure. NULL if no dynamic memory available.

**5.5.2.17 EXTERNEXI OSEXIEventCode\* rtEXINewEventCode3 (OSCTXT \* *pctxt*, OSINT32 *part1*, OSINT32 *part2*, OSINT32 *part3*)**

This function allocates and initializes a three-part event code.

**Parameters:**

*pctxt* Pointer to context block structure.

*part1* Part 1 of the event code.

*part2* Part 2 of the event code.

*part3* Part 3 of the event code.

**Returns:**

Pointer to allocated event code structure. NULL if no dynamic memory available.

**5.5.2.18 EXTERNEXI OSEXIEventCodeGroup\* rtEXINewEventCodeGroup (OSCTXT \* *pctxt*)**

This function allocates and initializes a new event code group structure.

**Parameters:**

*pctxt* Pointer to a context block structure.

**Returns:**

Pointer to new event code group structure or NULL if no dynamic memory available.

**5.5.2.19 EXTERNEXI void rtEXISetEventCode3 (OSEXIEventCode \* *pEventCode*, OSINT32 *part1*, OSINT32 *part2*, OSINT32 *part3*)**

This function sets a three-part event code.

**Parameters:**

*pEventCode* Pointer to event code structure.

*part1* Part 1 of the event code.

*part2* Part 2 of the event code.

*part3* Part 3 of the event code.

## 5.6 XML to EXI serialization functions.

### Classes

- struct [XML2EXISAXUserData](#)

### Functions

- EXTERNEXI int [rtXmlMem2ExiMem](#) (OSCTXT \*pctxt, OSOCTET \*outbuf, size\_t outbufsiz)  
*This function serializes XML data to EXI data and stores the result in the given memory buffer.*
- EXTERNEXI int [rtXmlFile2ExiStream](#) (const char \*xmlFileName, OSCTXT \*pExiCtxt)  
*This function serializes XML data from a file to EXI data and outputs the result to the output stream defined within the given output context.*
- EXTERNEXI int [rtXmlMem2ExiStream](#) (OSCTXT \*pctxt, OSCTXT \*pExiCtxt)  
*This function serializes XML data to EXI data and outputs the result to the output stream defined within the given output context.*
- EXTERNEXI int [xml2ExiStartElement](#) (void \*userData, const OSUTF8CHAR \*localname, const OSUTF8CHAR \*qname, const OSUTF8CHAR \*const \*attrs)  
*SAX start element handler callback function.*
- EXTERNEXI int [xml2ExiCharacters](#) (void \*userData, const OSUTF8CHAR \*chars, int length)  
*SAX characters handler callback function definition.*
- EXTERNEXI int [xml2ExiEndElement](#) (void \*userData, const OSUTF8CHAR \*localname, const OSUTF8CHAR \*qname)  
*SAX end element handler callback function definition.*

### 5.6.1 Function Documentation

#### 5.6.1.1 EXTERNEXI int [rtXmlFile2ExiStream](#) (const char \* *xmlFileName*, OSCTXT \* *pExiCtxt*)

This function serializes XML data from a file to EXI data and outputs the result to the output stream defined within the given output context.

The input context structure (pctxt) is assumed to contain an encoded XML message in its internal buffer. This will be the case after encoding an XML instance.

#### Parameters:

*xmlFileName* Full path to a file containing an XML document.

*pExiCtxt* Pointer to an output context assumed to contain an output stream descriptor created with an `rtxStream-CreateWriter` function.

#### Returns:

Completion status of operation:

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

### 5.6.1.2 EXTERNEXI int rtXmlMem2ExiMem (OSCTXT \* *pctxt*, OSOCTET \* *outbuf*, size\_t *outbufsiz*)

This function serializes XML data to EXI data and stores the result in the given memory buffer.

The context structure is assumed to contain an encoded XML message in its internal buffer. This will be the case after encoding an XML instance.

#### Parameters:

*pctxt* Pointer to a context structure. This is assumed to hold an encoded XML document or fragment.

*outbuf* Pointer to output buffer to receive EXI data.

*outbufsiz* Size of static output buffer.

#### Returns:

Completion status of operation:

- positive value indicates success and is encoded message length.
- negative return value is error.

### 5.6.1.3 EXTERNEXI int rtXmlMem2ExiStream (OSCTXT \* *pctxt*, OSCTXT \* *pExiCtxt*)

This function serializes XML data to EXI data and outputs the result to the output stream defined within the given output context.

The input context structure (*pctxt*) is assumed to contain an encoded XML message in its internal buffer. This will be the case after encoding an XML instance.

#### Parameters:

*pctxt* Pointer to a context structure. This is assumed to hold an encoded XML document or fragment.

*pExiCtxt* Pointer to an output context assumed to contain an output stream descriptor created with an `rtxStream-CreateWriter` function.

#### Returns:

Completion status of operation:

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

### 5.6.1.4 EXTERNEXI int xml2ExiCharacters (void \* *userData*, const OSUTF8CHAR \* *chars*, int *length*)

SAX characters handler callback function definition.

#### Parameters:

*userData* User-defined data structure. In this case, the [XML2EXISAXUserData](#) defined above is used.

*chars* Character data. String is not null-terminated.

*length* Number of characters in character string.

#### Returns:

Status of operation.

**5.6.1.5 EXTERNEXI int xml2ExiEndElement (void \* *userData*, const OSUTF8CHAR \* *localname*, const OSUTF8CHAR \* *qname*)**

SAX end element handler callback function definition.

**Parameters:**

*userData* User-defined data structure. In this case, the [XML2EXISAXUserData](#) defined above is used.

*localname* Local name of element.

*qname* Qualified name of element.

**Returns:**

Status of operation.

**5.6.1.6 EXTERNEXI int xml2ExiStartElement (void \* *userData*, const OSUTF8CHAR \* *localname*, const OSUTF8CHAR \* *qname*, const OSUTF8CHAR \*const \* *attrs*)**

SAX start element handler callback function.

**Parameters:**

*userData* User-defined data structure. In this case, the [XML2EXISAXUserData](#) defined above is used.

*localname* Local name of element.

*qname* Qualified name of element.

*attrs* Array of attribute name/value pairs, terminated by 0;

**Returns:**

Status of operation.

## Chapter 6

# XBinder Class Documentation

### 6.1 `_OSEXIEvent` Struct Reference

#### Public Attributes

- [OSEXIEventType](#) type  
*This event's type.*
- `OSXMLFullQName * qname`  
*Possibly null qname associated with this event.*

#### 6.1.1 Detailed Description

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

#### 6.1.2 Member Data Documentation

##### 6.1.2.1 `OSXMLFullQName* _OSEXIEvent::qname`

Possibly null qname associated with this event.

E.g., for `SE("foo")` the type is SE and the qname "foo".

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

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

- [rtEXIEvent.h](#)

## 6.2 EXI2SAXAttribute Struct Reference

### Public Attributes

- OSXMLFullQName \* [pqname](#)
- OSUTF8CHAR \* [value](#)

### 6.2.1 Detailed Description

Definition at line 87 of file [rtEXI2SAX.h](#).

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

- [rtEXI2SAX.h](#)

## 6.3 EXI2SAXReader Struct Reference

### Public Attributes

- OSCTXT \* [pctxt](#)
- void \* [userData](#)
- OSRTStack [elemQNameStack](#)
- [StartElementHandler](#) [pStartElementFunc](#)
- [EndElementHandler](#) [pEndElementFunc](#)
- [CharacterDataHandler](#) [pCharactersFunc](#)

### 6.3.1 Detailed Description

Definition at line 78 of file [rtEXI2SAX.h](#).

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

- [rtEXI2SAX.h](#)

## 6.4 OSEXIAtmState Struct Reference

This structure defines state information from an automaton that must be preserved at each stack level.

```
#include <rtEXIAutomaton.h>
```

### Public Attributes

- [OSEXIState currentState](#)  
*The automaton's current state.*

### 6.4.1 Detailed Description

This structure defines state information from an automaton that must be preserved at each stack level.

Definition at line 138 of file `rtEXIAutomaton.h`.

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

- [rtEXIAutomaton.h](#)

## 6.5 OSEXIAutomaton Struct Reference

This structure defines a finite state automata for EXI grammars.

```
#include <rtEXIAutomaton.h>
```

### Public Attributes

- [OSEXIState numberOfStates](#)  
*The number of states in this automaton.*
- [OSEXIState currentState](#)  
*The automaton's current state.*
- [OSEXIState acceptingState](#)  
*The automaton's accepting state.*
- `const OSXMLFullQName * elementName`  
*Name of the element's grammar this automaton accepts, or `null` if unspecified.*
- `OSRTArrayList eventCodeGroups`  
*List of event code groups indexed by state.*
- `OSBOOL isClosed`  
*Indicates if this automaton can be extended by adding additional states or transitions.*
- `OSBOOL matchedBaseEvent`  
*Flag indicating if the last call to `rtEXIDecAutomatonAdvance` matched a base event instead.*
- `OSRTArrayList eventStates`  
*Mapping defining transitions between (from, event) into (to, eventcode) pairs.*
- `OSEXIEvent * pDynEvent`  
*Event of the form `SE(null)` or `AT(null)` used in the last dynamic transition added to this automaton in `rtEXIDecAutomatonAdvance`.*
- `OSUINT32 valueChannel`  
*Value channel index.*
- `OSEXIStateEvent * pAttrEventState`  
*Attribute value state.*

### 6.5.1 Detailed Description

This structure defines a finite state automata for EXI grammars.

Definition at line 61 of file `rtEXIAutomaton.h`.

## 6.5.2 Member Data Documentation

### 6.5.2.1 OSRTArrayList [OSEXIAutomaton::eventCodeGroups](#)

List of event code groups indexed by state.

Event code groups are used to calculate the number of bits needed to encode an event code.

Definition at line 88 of file `rtEXIAutomaton.h`.

### 6.5.2.2 OSBOOL [OSEXIAutomaton::isClosed](#)

Indicates if this automaton can be extended by adding additional states or transitions.

An automaton that has been *closed* cannot be further extended.

Definition at line 95 of file `rtEXIAutomaton.h`.

### 6.5.2.3 OSBOOL [OSEXIAutomaton::matchedBaseEvent](#)

Flag indicating if the last call to `rtEXIDecAutomatonAdvance` matched a base event instead.

The base event of `SE(qname)` is `SE(*)`; the base event of `AT(qname)` is `AT(*)`.

Definition at line 102 of file `rtEXIAutomaton.h`.

### 6.5.2.4 OSRTArrayList [OSEXIAutomaton::eventStates](#)

Mapping defining transitions between (from, event) into (to, eventcode) pairs.

These records are maintained in a flat array list of all possible combinations. New relations can be added as knowledge is learned in an EXI encoding.

Definition at line 110 of file `rtEXIAutomaton.h`.

### 6.5.2.5 [OSEXIEvent\\*](#) [OSEXIAutomaton::pDynEvent](#)

Event of the form `SE(null)` or `AT(null)` used in the last dynamic transition added to this automaton in `rtEXIDecAutomatonAdvance`.

This event is returned by calling `rtEXIGetDynamicEvent`. The decoder is responsible for filling in the `qname` in this event, as it isn't available when the dynamic production is added.

Definition at line 120 of file `rtEXIAutomaton.h`.

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

- [rtEXIAutomaton.h](#)

## 6.6 OSEXICtxtInfo Struct Reference

### Public Attributes

- OSFreeCtxtAppInfoPtr [pFreeFunc](#)
- OSResetCtxtAppInfoPtr [pResetFunc](#)
- OSUINT32 [options](#)
- OSEXIAutomaton \* [pCurrAtm](#)
- OSEXIAutomaton \* [pEncDocAtm](#)
- OSEXIAutomaton \* [pEncSTypeAtm](#)
- OSEXIAutomaton \* [pDecDocAtm](#)
- OSRTStack [atmStack](#)
- OSRTStack [dynAtmStack](#)
- OSEXIEncStringTables [encStrTabs](#)
- OSRTHashMap [automataMap](#)
- OSEXIEvent \* [pDecState](#)  
*Internal state represented using an event.*
- OSXMLFullQName \* [pDecQName](#)  
*QName of last element or attribute read.*
- const OSUTF8CHAR \* [pDecValue](#)  
*Value of last value of an AT, CH, CM or ER.*
- const OSUTF8CHAR \* [pDecPrefix](#)  
*Last prefix read in NS event.*
- const OSUTF8CHAR \* [pDecNamespaceURI](#)  
*Last namespace URI in NS event.*
- OSEXIDecStringTables [decStrTabs](#)
- OSRTDiagBitFieldList [bitFieldList](#)
- OSUINT32 [blockSize](#)  
*Compression variables.*
- OSUINT32 [valueCnt](#)
- OSUINT32 [firstChannel](#)
- OSUINT32 [nmChannels](#)
- OSUINT32 \* [valuesInChannel](#)
- OSUINT32 [allocatedValueCnts](#)
- OSRTHashMap [attrChannelMap](#)
- size\_t [lastValuePos](#)
- size\_t [firstValuePos](#)
- OSUINT32 [curAttrChannel](#)
- OSRTBuffer [savedBuffer](#)
- OSRTFLAGS [savedFlags](#)
- OSOCTET \* [shadowBuff](#)
- size\_t [shadowBuffSize](#)
- size\_t [shadowBuffEnd](#)

## 6.6.1 Detailed Description

Definition at line 76 of file osrtexi.h.

## 6.6.2 Member Data Documentation

### 6.6.2.1 [OSEXIEvent\\*](#) [OSEXICtxtInfo::pDecState](#)

Internal state represented using an event.

NULL is reserved to indicate the initial state.

Definition at line 96 of file osrtexi.h.

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

- [osrtexi.h](#)

## 6.7 OSEXIDecStringTable Struct Reference

This structure defines the structure of the various string table partitions used by the decoder.

```
#include <rtEXIDecStringTable.h>
```

### Public Attributes

- OSRTArrayList [records](#)  
*An expandable array relating string compact identifier (index) values to strings.*

### 6.7.1 Detailed Description

This structure defines the structure of the various string table partitions used by the decoder.

In this case, an array list can be used for record storage because the compact identifiers are simply indexes into the array. A different structure using a has map is used by the encoder.

Definition at line 49 of file `rtEXIDecStringTable.h`.

### 6.7.2 Member Data Documentation

#### 6.7.2.1 OSRTArrayList [OSEXIDecStringTable::records](#)

An expandable array relating string compact identifier (index) values to strings.

The array contains pointers to UTF8 strings (const OSUTF8CHAR\*).

Note that a count variable is not maintained in this structure as it is for the encode string table. That is because count can be obtained from the OSRTArrayList structure (`records.size`).

Definition at line 59 of file `rtEXIDecStringTable.h`.

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

- [rtEXIDecStringTable.h](#)

## 6.8 OSEXIDecStringTables Struct Reference

This structure defines the complete set of string table partitions used by the decoder.

```
#include <rtEXIDecStringTables.h>
```

### Public Attributes

- [OSEXIDecStringTable uriTable](#)  
*The URI table.*
- [OSRTHashMap prefixTables](#)  
*The prefix table set.*
- [OSRTHashMap localNameTables](#)  
*The local name table set.*
- [OSRTHashMap localValueTables](#)  
*The local value table set.*
- [OSEXIDecStringTable globalValueTable](#)  
*The global value table.*

### 6.8.1 Detailed Description

This structure defines the complete set of string table partitions used by the decoder.

A different structure is used by the decoder.

Definition at line 46 of file `rtEXIDecStringTables.h`.

### 6.8.2 Member Data Documentation

#### 6.8.2.1 [OSEXIDecStringTable OSEXIDecStringTables::uriTable](#)

The URI table.

This holds URI content items. There is one URI table for a given EXI application.

Definition at line 51 of file `rtEXIDecStringTables.h`.

#### 6.8.2.2 [OSRTHashMap OSEXIDecStringTables::prefixTables](#)

The prefix table set.

There is a prefix table (referred to in the spec as a 'table partition') for each namespace URI. These are represented using a `HashMap` with URI as key and `StringTable` as value.

Definition at line 59 of file `rtEXIDecStringTables.h`.

### 6.8.2.3 OSRTHashMap [OSEXIDecStringTables::localNameTables](#)

The local name table set.

There is a local name table (referred to in the spec as a 'table partition') for each namespace URI. These are represented using a HashMap with URI as key and StringTable as value.

Definition at line 67 of file [rtEXIDecStringTables.h](#).

### 6.8.2.4 OSRTHashMap [OSEXIDecStringTables::localValueTables](#)

The local value table set.

This set of tables holds Value content items. They are partitioned based on QName of their associated attribute or element definitions.

Definition at line 74 of file [rtEXIDecStringTables.h](#).

### 6.8.2.5 [OSEXIDecStringTable](#) [OSEXIDecStringTables::globalValueTable](#)

The global value table.

This holds Value content items. There is one global value table for a given EXI application.

Definition at line 80 of file [rtEXIDecStringTables.h](#).

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

- [rtEXIDecStringTables.h](#)

## 6.9 OSEXIEncStringTable Struct Reference

This structure defines the structure of the various string table partitions used by the encoder.

```
#include <rtEXIEncStringTable.h>
```

### Public Attributes

- OSUINT32 [count](#)  
*The count of the number of items in the table.*
- OSRTHashMapStr2UInt [records](#)  
*A hash map relating string to compact identifier (index) values.*

### 6.9.1 Detailed Description

This structure defines the structure of the various string table partitions used by the encoder.

A different structure is used by the decoder.

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

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

- [rtEXIEncStringTable.h](#)

## 6.10 OSEXIEncStringTables Struct Reference

This structure defines the complete set of string table partitions used by the encoder.

```
#include <rtEXIEncStringTables.h>
```

### Public Attributes

- [OSEXIEncStringTable uriTable](#)  
*The URI table.*
- [OSRTHashMap prefixTables](#)  
*The prefix table set.*
- [OSRTHashMap localNameTables](#)  
*The local name table set.*
- [OSRTHashMap localValueTables](#)  
*The local value table set.*
- [OSEXIEncStringTable globalValueTable](#)  
*The global value table.*

### 6.10.1 Detailed Description

This structure defines the complete set of string table partitions used by the encoder.

A different structure is used by the decoder.

Definition at line 46 of file `rtEXIEncStringTables.h`.

### 6.10.2 Member Data Documentation

#### 6.10.2.1 [OSEXIEncStringTable OSEXIEncStringTables::uriTable](#)

The URI table.

This holds URI content items. There is one URI table for a given EXI application.

Definition at line 51 of file `rtEXIEncStringTables.h`.

#### 6.10.2.2 [OSRTHashMap OSEXIEncStringTables::prefixTables](#)

The prefix table set.

There is a prefix table (referred to in the spec as a 'table partition') for each namespace URI. These are represented using a `HashMap` with URI as key and `StringTable` as value.

Definition at line 59 of file `rtEXIEncStringTables.h`.

### 6.10.2.3 OSRTHashMap [OSEXIEncStringTables::localNameTables](#)

The local name table set.

There is a local name table (referred to in the spec as a 'table partition') for each namespace URI. These are represented using a HashMap with URI as key and StringTable as value.

Definition at line 67 of file [rtEXIEncStringTables.h](#).

### 6.10.2.4 OSRTHashMap [OSEXIEncStringTables::localValueTables](#)

The local value table set.

This set of tables holds Value content items. They are partitioned based on QName of their associated attribute or element definitions.

Definition at line 74 of file [rtEXIEncStringTables.h](#).

### 6.10.2.5 [OSEXIEncStringTable](#) [OSEXIEncStringTables::globalValueTable](#)

The global value table.

This holds Value content items. There is one global value table for a given EXI application.

Definition at line 80 of file [rtEXIEncStringTables.h](#).

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

- [rtEXIEncStringTables.h](#)

## 6.11 OSEXIEventCode Struct Reference

A structure representing a production's event code.

```
#include <rtEXIEventCode.h>
```

### Public Attributes

- OSINT32 [part1](#)
- OSINT32 [part2](#)
- OSINT32 [part3](#)

### 6.11.1 Detailed Description

A structure representing a production's event code.

An event code has at least one part and at most three parts. Each part is represented by an integer number. A part that is absent is represented by the minimum integer value.

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

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

- [rtEXIEventCode.h](#)

## 6.12 OSEXIEventCodeGroup Struct Reference

An EventCodeGroup is a group of related event codes.

```
#include <rtEXIEventCodeGroup.h>
```

### Public Attributes

- OSRTSList [group](#)
- OSINT32 [maxPart1](#)
- OSINT32 [maxPart2](#)
- OSINT32 [maxPart3](#)
- OSRTDynBitSet [part1Set](#)
- OSRTDynBitSet [part2Set](#)

### 6.12.1 Detailed Description

An EventCodeGroup is a group of related event codes.

Event codes are related if they all correspond to grammar productions with the same LHS non-terminal.

Definition at line 47 of file rtEXIEventCodeGroup.h.

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

- [rtEXIEventCodeGroup.h](#)

## 6.13 OSEXIHeader Struct Reference

### Public Attributes

- OSBOOL [mbOptionsPresent](#)
- OSBOOL [mbPreview](#)
- OSUINT32 [version](#)

### 6.13.1 Detailed Description

Definition at line 147 of file [osrtexi.h](#).

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

- [osrtexi.h](#)

## 6.14 OSEXIStateEvent Struct Reference

This structure holds state/event information.

```
#include <rtEXIAutomaton.h>
```

### Public Attributes

- [OSEXIState fromState](#)
- [OSEXIState toState](#)
- const [OSEXIEvent \\* pEvent](#)
- [OSEXIEventCode \\* pEventCode](#)
- OSUINT32 [valueChannel](#)

### 6.14.1 Detailed Description

This structure holds state/event information.

Definition at line 50 of file [rtEXIAutomaton.h](#).

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

- [rtEXIAutomaton.h](#)

## 6.15 OSEXIStateTableRecord Struct Reference

### Public Attributes

- OSUINT16 [eventId](#)
- OSUINT16 [stateId](#)
- OSINT16 [nextStateIdx](#)
- const OSUTF8CHAR \* [localName](#)
- [OSEXIEventType](#) [eventType](#)
- OSUINT8 [eventCode](#)
- OSUINT8 [eventNBits](#)
- OSUINT8 [maxEvtCode](#)

### 6.15.1 Detailed Description

Definition at line 37 of file `rtEXIDecStateTable.h`.

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

- `rtEXIDecStateTable.h`

## 6.16 XML2EXISAXUserData Struct Reference

### Public Attributes

- OSCTXT \* [pExiEncCtxt](#)
- OSCTXT \* [pXmlDecCtxt](#)

### 6.16.1 Detailed Description

Definition at line 40 of file [rtXML2EXISAX.h](#).

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

- [rtXML2EXISAX.h](#)

# Chapter 7

## XBinder File Documentation

### 7.1 osrtexi.h File Reference

EXI low-level C context and structure definitions.

```
#include "rtxsrc/rtxContext.h"
#include "rtxsrc/rtxStack.h"
#include "rtexisrc/rtEXIEncAutomaton.h"
#include "rtexisrc/rtEXIEncStringTables.h"
#include "rtexisrc/rtEXIDecStringTables.h"
#include "rtxsrc/rtxDiagBitTrace.h"
```

#### Classes

- struct [OSEXICtxtInfo](#)
- struct [OSEXIHeader](#)

#### Defines

- #define [OSEXINULLINDEX](#) OSUINT32\_MAX
- #define [OSEXI\\_PRESERVE\\_DTD](#) 0x80000000
- #define [OSEXI\\_PRESERVE\\_PIS](#) 0x40000000
- #define [OSEXI\\_PRESERVE\\_COMMENTS](#) 0x20000000
- #define [OSEXI\\_PRESERVE\\_PREFIXES](#) 0x10000000
- #define [OSEXI\\_PRESERVE\\_LEXICAL](#) 0x08000000
- #define [OSEXI\\_FRAGMENT](#) 0x04000000
- #define [OSEXI\\_COMPRESSED](#) 0x02000000
- #define [OSEXI\\_ALIGNED](#) 0x01000000
- #define [OSEXI\\_PRECOMPRESSED](#) 0x00800000
- #define [OSEXI\\_SCHEMA](#) 0x00400000
- #define [OSEXI\\_NOWHITESPACE](#) 0x00200000
- #define [EXICTXT](#)(pctx) (([OSEXICtxtInfo\\*](#))((pctx) → [pEXIInfo](#)))
- #define [rtEXISetBufPtr](#)(pctx, bufaddr, bufsiz) rtxCtxtSetBufPtr (pctx, bufaddr, bufsiz)

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

- **#define `rtEXIGetMsgPtr`**(pctx) (pctx) → `buffer.data`  
*This macro returns the start address of the encoded EXI message.*
- **#define `rtEXIGetMsgLen`**(pctx) (pctx) → `buffer.byteIndex`  
*This macro returns the length of the encoded XML message.*
- **#define `rtEXISetOption`**(pctx, option) `EXICTXT(pctx) → options |= option;`  
*This macro is used to set a bit flag in the EXI options bit mask.*
- **#define `rtEXIClearOption`**(pctx, option) `EXICTXT(pctx) → options &= ~option;`  
*This macro is used to clear a bit flag in the EXI options bit mask.*
- **#define `rtEXITestOption`**(pctx, option) `((EXICTXT(pctx) → options & option) != 0)`  
*This macro tests if the given option is set in the EXI context.*

## Functions

- EXTERNEXI int **`rtEXIInitContext`** (OSCTXT \*pctx)  
*This function initializes a context variable for EXI encoding or decoding.*
- EXTERNEXI int **`rtEXIInitCtxAppInfo`** (OSCTXT \*pctx)  
*This function initializes the EXI application info section of the given context.*
- EXTERNEXI void **`rtEXIEnableBitFieldTrace`** (OSCTXT \*pctx)  
*This function turns on bit field tracing and initializes the bit field trace list.*

### 7.1.1 Detailed Description

EXI low-level C context and structure definitions.

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

## 7.2 rtEXI2SAX.h File Reference

EXI SAX parser interface.

```
#include "rtxsrc/rtxStack.h"
#include "rtxsrc/rtxXmlQName.h"
#include "rtexisrc/rtEXIExternDefs.h"
```

### Classes

- struct [EXI2SAXReader](#)
- struct [EXI2SAXAttribute](#)

### Typedefs

- typedef int(\*) [StartElementHandler](#) (void \*userData, const OSUTF8CHAR \*localname, const OSUTF8CHAR \*qname, const OSUTF8CHAR \*const \*attrs)  
*SAX start element handler callback function definition.*
- typedef int(\*) [EndElementHandler](#) (void \*userData, const OSUTF8CHAR \*localname, const OSUTF8CHAR \*qname)  
*SAX end element handler callback function definition.*
- typedef int(\*) [CharacterDataHandler](#) (void \*userData, const OSUTF8CHAR \*value, int len)  
*SAX characters handler callback function definition.*

### Functions

- EXTERNEXI [EXI2SAXReader](#) \* [rtEXI2SAXCreateReader](#) (OSCTXT \*pctxt, void \*pUserData, [StartElementHandler](#) startElemFunc, [EndElementHandler](#) endElemFunc, [CharacterDataHandler](#) charactersFunc)  
*This function is used to create a reader structure for use by the main SAX parser function.*
- EXTERNEXI int [rtEXI2SAXParse](#) ([EXI2SAXReader](#) \*pReader)  
*This is the main SAX parser function.*

### 7.2.1 Detailed Description

EXI SAX parser interface.

Definition in file [rtEXI2SAX.h](#).

## 7.3 rtEXI2XML.h File Reference

Functions for converting EXI encoded data into XML format.

```
#include "rtexisrc/osrtexi.h"
```

### Functions

- EXTERNEXI int [rtExi2XmlStream](#) (OSCTXT \*pctxt, OSCTXT \*poutctxt)  
*This function transcodes an EXI encoded message to an XML output stream.*

### 7.3.1 Detailed Description

Functions for converting EXI encoded data into XML format.

Definition in file [rtEXI2XML.h](#).

## 7.4 rtEXIAutomaton.h File Reference

EXI automaton structure and functions.

```
#include "rtexisrc/rtEXIEvent.h"
#include "rtexisrc/rtEXIEventCodeGroup.h"
#include "rtxmlsrc/osrtxml.h"
#include "rtxsrc/rtxArrayList.h"
```

### Classes

- struct [OSEXISStateEvent](#)  
*This structure holds state/event information.*
- struct [OSEXIAutomaton](#)  
*This structure defines a finite state automata for EXI grammars.*
- struct [OSEXIAtmState](#)  
*This structure defines state information from an automaton that must be preserved at each stack level.*

### Typedefs

- typedef OSINT16 [OSEXISState](#)
- typedef int(\*) [OSEXIAtmAddTransFunc](#) (OSCTXT \*pctxt, [OSEXIAutomaton](#) \*pAutomaton, [OSEXISState](#) fromState, [OSEXISState](#) toState, const [OSEXIEvent](#) \*pEvent, const [OSEXIEventCode](#) \*pEventCode)  
*Add transition function definition.*

### Enumerations

- enum [OSEXIAtmType](#)

### Functions

- EXTERNEXI void [rtEXIAutomatonInit](#) (OSCTXT \*pctxt, [OSEXIAutomaton](#) \*pAutomaton, const OSXMLFullQName \*pElemName, [OSEXISState](#) numStates)  
*This function initializes the automaton to its default state.*
- EXTERNEXI [OSEXIAutomaton](#) \* [rtEXINewAutomaton](#) (OSCTXT \*pctxt, const OSXMLFullQName \*pElemName, [OSEXISState](#) numStates)  
*This function allocates memory for a new automaton structure and initializes the structure.*
- EXTERNEXI [OSEXIAutomaton](#) \* [rtEXIAutomatonCopy](#) (OSCTXT \*pctxt, [OSEXIAutomaton](#) \*pAutomaton)  
*This function copies an automaton structure.*
- EXTERNEXI void [rtEXIAutomatonFreeMem](#) (OSCTXT \*pctxt, [OSEXIAutomaton](#) \*pAutomaton, OSBOOL dynamic)

*This function frees all memory within an Automaton structure.*

- EXTERNEXI [OSEXIAutomaton](#) \* [rtEXIAutomatonAddTransition](#) (OSCTXT \*pctx, [OSEXIAutomaton](#) \*pAutomaton, [OSEXISState](#) fromState, [OSEXISState](#) toState, const [OSEXIEventCode](#) \*pEventCode)

*This function adds a transition between two states.*

- EXTERNEXI int [rtEXIAtmAddUndeclaredItems](#) (OSCTXT \*pctx, [OSEXIAutomaton](#) \*pAutomaton, [OSEXISState](#) fromState, [OSEXISState](#) toState, [OSEXIEventCode](#) \*pEventCode, size\_t numDeclAttrs, const [OSXMLFullQName](#) \*declAttrs, [OSEXIAtmAddTransFunc](#) addTransFunc)

*This function adds all undeclared items (end element, start tag, and content) to an element automaton in schema-informed mode.*

- EXTERNEXI int [rtEXIAtmAddUndeclaredStartTagItems](#) (OSCTXT \*pctx, [OSEXIAutomaton](#) \*pAutomaton, [OSEXISState](#) fromState, [OSEXISState](#) toState, [OSEXIEventCode](#) \*pEventCode, size\_t numDeclAttrs, const [OSXMLFullQName](#) \*declAttrs, [OSEXIAtmAddTransFunc](#) addTransFunc)

*This function adds undeclared start tag items to an element automaton in schema-informed mode.*

- EXTERNEXI int [rtEXIAtmAddUndeclaredContentItems](#) (OSCTXT \*pctx, [OSEXIAutomaton](#) \*pAutomaton, [OSEXISState](#) fromState, [OSEXISState](#) toState, [OSEXIEventCode](#) \*pEventCode, [OSEXIAtmAddTransFunc](#) addTransFunc)

*This function adds undeclared content items to an element automaton in schema-informed mode.*

- EXTERNEXI [OSEXIAutomaton](#) \* [rtEXIAutomatonInitCopy](#) (OSCTXT \*pctx, [OSEXIAutomaton](#) \*pDestAtm, [OSEXIAutomaton](#) \*pSrcAtm)

*This function initializes an automaton structure using the data from an existing automaton.*

- EXTERNEXI int [rtEXIAutomatonPush](#) (OSCTXT \*pctx, [OSEXIAutomaton](#) \*pAutomaton)

*This function pushes an automaton onto the context stack.*

- EXTERNEXI [OSEXIAutomaton](#) \* [rtEXIAutomatonPop](#) (OSCTXT \*pctx)

*This function pops an automaton from the context stack.*

- EXTERNEXI [OSEXIEventCodeGroup](#) \* [rtEXIAtmGetCurrentEventCodeGroup](#) ([OSEXIAutomaton](#) \*pAutomaton)

*This function returns a pointer to the event code group corresponding to the current state.*

- EXTERNEXI [OSEXIAutomaton](#) \* [rtEXIGetDocAutomaton](#) (OSCTXT \*pctx, size\_t numGblElems, const [OSXMLFullQName](#) \*gblElems, [OSEXIAtmAddTransFunc](#) addTransFunc)

*This functions returns an automaton that accepts the built-in document grammar.*

- EXTERNEXI [OSEXIAutomaton](#) \* [rtEXIGetElemAutomaton](#) (OSCTXT \*pctx, [OSXMLFullQName](#) \*pqname, int(\*addTransFunc)(OSCTXT \*pctx, [OSEXIAutomaton](#) \*pAutomaton, [OSEXISState](#) fromState, [OSEXISState](#) toState, const [OSEXIEvent](#) \*pEvent, const [OSEXIEventCode](#) \*pEventCode))

*This function returns an automaton that accepts the built-in element grammar.*

## 7.4.1 Detailed Description

EXI automaton structure and functions.

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

## 7.5 rtEXIDecAutomaton.h File Reference

EXI decoding automaton structure and functions.

```
#include "rtexisrc/rtEXIAutomaton.h"
```

### Functions

- EXTERNEXI int [rtEXIDecAtmAddTransition](#) (OSCTXT \*pctx, [OSEXIAutomaton](#) \*pAutomaton, [OSEXIS-  
tate](#) fromState, [OSEXIS-  
tate](#) toState, const [OSEXIEvent](#) \*pEvent, const [OSEXIEventCode](#) \*pEventCode)  
*This function adds a transition between two states.*
- EXTERNEXI [OSEXIEvent](#) \* [rtEXIDecAutomatonAdvance](#) (OSCTXT \*pctx, [OSEXIAutomaton](#) \*p-  
Automaton, const [OSEXIEventCode](#) \*pEventCode, OSBOOL dynamicItems)  
*This function advances the decoder automaton based on event code, adding new transitions if dynamicItems is set  
to true and either SE(\*) or AT(\*) are matched instead of SE(qname) or AT(qname), respectively.*
- EXTERNEXI [OSEXIAutomaton](#) \* [rtEXIDecGetDocAutomaton](#) (OSCTXT \*pctx, size\_t numGblElems, const  
OSXMLFullQName \*gblElems)  
*This functions returns an automaton that accepts the built-in document grammar.*
- EXTERNEXI [OSEXIAutomaton](#) \* [rtEXIDecGetElemAutomaton](#) (OSCTXT \*pctx, OSXMLFullQName  
\*pqname)  
*This function returns an automaton that accepts the built-in element grammar.*

### 7.5.1 Detailed Description

EXI decoding automaton structure and functions.

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

## 7.6 rtEXIDecBitTrace.h File Reference

Functions for logging bit trace diagnostic events that occur during the decoding of an EXI-encoded document.

```
#include "rtexisrc/osrtexi.h"
```

### Defines

- #define [rtEXIDecBitFieldStart](#)(pctxt)
- #define [rtEXIDecBitFieldEnd](#)(pctxt, nameSuffix)
- #define [rtEXIDecBitFieldEnd2](#)(pctxt, name, value)
- #define [rtEXIDecBitFieldEndAndStartNew](#)(pctxt, nameSuffix)

### 7.6.1 Detailed Description

Functions for logging bit trace diagnostic events that occur during the decoding of an EXI-encoded document.

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

## 7.7 rtEXIDecoder.h File Reference

Interface for EXI low-level decoders.

```
#include "rtexisrc/rtEXIDecStateTable.h"
#include "rtexisrc/rtEXIEvent.h"
#include "rtexisrc/rtEXIEventCodeGroup.h"
#include "rtxmlsrc/osrtxml.h"
#include "rtxsrc/rtxArrayList.h"
```

### Functions

- EXTERNEXI int [rtEXIDecAttribute](#) (OSCTXT \*pctxt, OSXMLFullQName \*pqname, const OSUTF8CHAR \*\*ppvalue)  
*Decodes the attribute at the current position in the decode stream.*
- EXTERNEXI int [rtEXIDecBoolValue](#) (OSCTXT \*pctxt, OSBOOL \*pvalue)  
*This function decodes a boolean value.*
- EXTERNEXI int [rtEXIDec\\_CH\\_String\\_EE](#) (OSCTXT \*pctxt, const OSXMLFullQName \*pqname, const OSUTF8CHAR \*\*ppvalue)  
*This function decodes a CH event (assumed to be a one part code = 0 in a 1 bit field) followed by string content followed by an EE event (assumed to be a one part code = 0 in a 1 bit field).*
- EXTERNEXI int [rtEXIDecDate](#) (OSCTXT \*pctxt, OSNumDateTime \*pvalue)  
*This function decodes a date value into a structured variable.*
- EXTERNEXI int [rtEXIDecDateString](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*ppvalue)  
*This function decodes a date value into a string.*
- EXTERNEXI int [rtEXIDecDocumentType](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*ppName, const OSUTF8CHAR \*\*ppPublic, const OSUTF8CHAR \*\*ppSystem, const OSUTF8CHAR \*\*ppText)  
*This function decodes an XML document type declaration (DTD).*
- EXTERNEXI int [rtEXIDecEventCodePart1](#) (OSCTXT \*pctxt, OSINT32 \*ppart, OSUINT32 nbits, OSUINT32 maxval)  
*This function decodes part 1 of an event code.*
- EXTERNEXI OSBOOL [rtEXIDecHasNext](#) (OSCTXT \*pctxt)  
*This functions checks for additional events in the decode stream.*
- EXTERNEXI int [rtEXIDecIntValue](#) (OSCTXT \*pctxt, OSINT32 \*pvalue)  
*This function decodes a signed integer value.*
- EXTERNEXI int [rtEXIDecLocalName](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*ppname)  
*Returns the local name associated with the current event.*
- EXTERNEXI int [rtEXIDecNamespaceURI](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*\*ppNSURI)  
*Returns the namespace associated with the current event.*

- EXTERNEXI int [rtEXIDecNBitUIntValue](#) (OSCTXT \*pctx, OSUINT32 \*pvalue, OSUINT32 nbits)  
*This function decodes an unsigned integer value from a bit field of the given width.*
- EXTERNEXI int [rtEXIDecNextEventType](#) (OSCTXT \*pctx, OSEXIEventType \*pEventType)  
*Returns the next OSEXIEventType read by this decoder.*
- EXTERNEXI int [rtEXIDecoderInit](#) (OSCTXT \*pctx)  
*This function initializes the decoder.*
- EXTERNEXI int [rtEXIDecPrefix](#) (OSCTXT \*pctx, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*\*ppPrefix)  
*Returns the namespace associated with the current event.*
- EXTERNEXI int [rtEXIDecProcessingInstruction](#) (OSCTXT \*pctx, const OSUTF8CHAR \*\*ppTarget, const OSUTF8CHAR \*\*ppData)  
*This function decodes an XML processing instruction (PI).*
- EXTERNEXI int [rtEXIDecQName](#) (OSCTXT \*pctx, OSXMLFullQName \*pqname)  
*Returns the qname associated with the current event.*
- EXTERNEXI int [rtEXIDecReset](#) (OSCTXT \*pctx)  
*Resets the decoder for decoding a new message instance.*
- EXTERNEXI int [rtEXIDecString](#) (OSCTXT \*pctx, const OSXMLFullQName \*pqname, const OSUTF8CHAR \*\*ppvalue)  
*Returns the value associated with the current event.*
- EXTERNEXI int [rtEXIDecStringCHEvent](#) (OSCTXT \*pctx, const OSXMLFullQName \*pqname, const OSUTF8CHAR \*\*ppvalue)  
*This function decodes a CH event (assumed to be a one-bit field with code 0) followed by string content.*
- EXTERNEXI int [rtEXIDecStringToCharArray](#) (OSCTXT \*pctx, const OSUTF8CHAR \*target, size\_t start, size\_t length)  
*Similar to rtEXIDecString but the characters are copied into a fixed-size character array.*
- EXTERNEXI int [rtEXIDecStringLength](#) (OSCTXT \*pctx)  
*Returns the length of the string returned by rtEXIDecString.*
- EXTERNEXI int [rtEXIDecUIntValue](#) (OSCTXT \*pctx, OSUINT32 \*pvalue)  
*This function decodes an unsigned integer value.*
- EXTERNEXI int [rtEXIDecUTF8Str](#) (OSCTXT \*pctx, OSUTF8CHAR \*\*ppvalue)  
*This function decodes a UTF-8 string value.*
- EXTERNEXI int [rtEXIDecUTF8Chars](#) (OSCTXT \*pctx, OSUTF8CHAR \*\*ppvalue, OSUINT32 nchars)  
*This function reads the given number of characters from the decode stream and creates a UTF-8 string.*

## 7.7.1 Detailed Description

Interface for EXI low-level decoders.

The interface is similar to StAX, but uses enumerations for the different event types and reports attributes and namespaces as individual events.

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

## 7.8 rtEXIDecStringTable.h File Reference

EXI decoder string table structure and functions.

```
#include "rtxsrc/rtxArrayList.h"
#include "rtxsrc/rtxContext.h"
#include "rtexisrc/rtEXIExternDefs.h"
```

### Classes

- struct [OSEXIDecStringTable](#)

*This structure defines the structure of the various string table partitions used by the decoder.*

### Functions

- EXTERNEXI void [rtEXIDecStringTableInit](#) (OSCTXT \*pctxt, [OSEXIDecStringTable](#) \*pstrtab, size\_t capacity)  
*This function initializes the given string table structure.*
- EXTERNEXI [OSEXIDecStringTable](#) \* [rtEXIDecNewStringTable](#) (OSCTXT \*pctxt, size\_t capacity)  
*This function allocates and initializes a new string table structure.*
- EXTERNEXI void [rtEXIDecStringTableClear](#) (OSCTXT \*pctxt, [OSEXIDecStringTable](#) \*pstrtab)  
*This function clears all strings out of the existing table.*
- EXTERNEXI OSUINT32 [rtEXIDecStringTableAdd](#) (OSCTXT \*pctxt, [OSEXIDecStringTable](#) \*pstrtab, const OSUTF8CHAR \*str)  
*This function adds a string to the given string table.*
- EXTERNEXI const OSUTF8CHAR \* [rtEXIDecStringTableGetString](#) ([OSEXIDecStringTable](#) \*pstrtab, OSUINT32 index)  
*This function gets the string at the given index (i.e.*

### 7.8.1 Detailed Description

EXI decoder string table structure and functions.

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

## 7.9 rtEXIDecStringTables.h File Reference

EXI decoder string table structure and functions.

```
#include "rtexisrc/rtEXIDecStringTable.h"  
#include "rtxsrc/rtxHashMap.h"  
#include "rtxmlsrc/osrtxml.h"
```

### Classes

- struct [OSEXIDecStringTables](#)  
*This structure defines the complete set of string table partitions used by the decoder.*

### Functions

- EXTERNEXI void [rtEXIDecStrTabsInit](#) (OSCTXT \*pctxt, [OSEXIDecStringTables](#) \*pstrtabs)  
*This function initializes all EXI string table partitions.*
- EXTERNEXI void [rtEXIDecStrTabsClear](#) (OSCTXT \*pctxt, [OSEXIDecStringTables](#) \*pstrtabs)  
*This function clears all EXI string table partitions.*
- EXTERNEXI OSUINT32 [rtEXIDecStrTabsAddURI](#) (OSCTXT \*pctxt, [OSEXIDecStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri)  
*This function will add a URI to the URI string table partition.*
- EXTERNEXI const OSUTF8CHAR \* [rtEXIDecStrTabsGetURI](#) ([OSEXIDecStringTables](#) \*pstrtabs, OSUINT32 index)  
*This function will get the compact identifier of the given URI from the URI string table partition.*
- EXTERNEXI OSUINT32 [rtEXIDecStrTabsGetURITableSize](#) ([OSEXIDecStringTables](#) \*pstrtabs)  
*This function returns the current number of entries in the URI string table partition.*
- EXTERNEXI OSUINT32 [rtEXIDecStrTabsAddPrefix](#) (OSCTXT \*pctxt, [OSEXIDecStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*prefix)  
*This function adds the given prefix to the prefix table partition identified by the given URI.*
- EXTERNEXI const OSUTF8CHAR \* [rtEXIDecStrTabsGetPrefix](#) ([OSEXIDecStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri, OSUINT32 index)  
*This function will get the compact identifier of the given prefix from the prefix string table partition identified by the given URI.*
- EXTERNEXI OSUINT32 [rtEXIDecStrTabsGetPrefixTableSize](#) ([OSEXIDecStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri)  
*This function returns the current number of entries in the prefix string table partition identified by the given URI.*
- EXTERNEXI OSUINT32 [rtEXIDecStrTabsAddLocalName](#) (OSCTXT \*pctxt, [OSEXIDecStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*name)  
*This function adds the given local name to the local name table partition identified by the given URI.*

- EXTERNEXI const OSUTF8CHAR \* [rtEXIDecStrTabsGetLocalName](#) (OSEXIDecStringTables \*pstrtabs, const OSUTF8CHAR \*uri, OSUINT32 index)  
*This function will get the compact identifier of the given localName from the localName string table partition identified by the given URI.*
- EXTERNEXI OSUINT32 [rtEXIDecStrTabsGetLocalNameTableSize](#) (OSEXIDecStringTables \*pstrtabs, const OSUTF8CHAR \*uri)  
*This function returns the current number of entries in the localName string table partition identified by the given URI.*
- EXTERNEXI OSUINT32 [rtEXIDecStrTabsAddLocalValue](#) (OSCTXT \*pctxt, OSEXIDecStringTables \*pstrtabs, const OSXMLFullQName \*qname, const OSUTF8CHAR \*value)  
*This function adds the given local value to the local value table partition identified by the given QName.*
- EXTERNEXI const OSUTF8CHAR \* [rtEXIDecStrTabsGetLocalValue](#) (OSEXIDecStringTables \*pstrtabs, const OSXMLFullQName \*qname, OSUINT32 index)  
*This function will get the compact identifier of the given local value from the local value string table partition identified by the given QName.*
- EXTERNEXI OSUINT32 [rtEXIDecStrTabsGetLocalValueTableSize](#) (OSEXIDecStringTables \*pstrtabs, const OSXMLFullQName \*qname)  
*This function returns the current number of entries in the local value string table partition identified by the given QName.*
- EXTERNEXI OSUINT32 [rtEXIDecStrTabsAddGlobalValue](#) (OSCTXT \*pctxt, OSEXIDecStringTables \*pstrtabs, const OSUTF8CHAR \*value)  
*This function will add a string value to the global value string table partition.*
- EXTERNEXI const OSUTF8CHAR \* [rtEXIDecStrTabsGetGlobalValue](#) (OSEXIDecStringTables \*pstrtabs, OSUINT32 index)  
*This function will get the compact identifier of the given string value from the global value string table partition.*
- EXTERNEXI OSUINT32 [rtEXIDecStrTabsGetGlobalValueTableSize](#) (OSEXIDecStringTables \*pstrtabs)  
*This function returns the current number of entries in the global value string table partition.*

## 7.9.1 Detailed Description

EXI decoder string table structure and functions.

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

## 7.10 rtEXIEncAutomaton.h File Reference

EXI encoding automaton structure and functions.

```
#include "rtexisrc/rtEXIAutomaton.h"
```

### Functions

- EXTERNEXI int [rtEXIEncAtmAddTransition](#) (OSCTXT \*pctx, [OSEXIAutomaton](#) \*pAutomaton, [OSEXIS-  
tate](#) fromState, [OSEXIS-  
tate](#) toState, const [OSEXIEvent](#) \*pEvent, const [OSEXIEventCode](#) \*pEventCode)  
*This function adds a transition between two states.*
- EXTERNEXI [OSEXIEventCode](#) \* [rtEXIEncAutomatonAdvance](#) (OSCTXT \*pctx, [OSEXIAutomaton](#) \*p-  
Automaton, const [OSEXIEvent](#) \*pEvent, OSBOOL dynamicItems)  
*This function advances the encoder automaton based on event, adding new transitions if dynamicItems is set to  
true and either SE(\*) or AT(\*) are matched instead of SE(qname) or AT(qname), respectively.*
- EXTERNEXI [OSEXIAutomaton](#) \* [rtEXIEncGetDocAutomaton](#) (OSCTXT \*pctx, size\_t numGblElems, const  
OSXMLFullQName \*gblElems)  
*This functions returns an automaton that accepts the built-in document grammar.*
- EXTERNEXI [OSEXIAutomaton](#) \* [rtEXIEncGetElemAutomaton](#) (OSCTXT \*pctx, OSXMLFullQName  
\*pqname)  
*This function returns an automaton that accepts the built-in element grammar.*

### 7.10.1 Detailed Description

EXI encoding automaton structure and functions.

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

## 7.11 rtEXIEncoder.h File Reference

EXI low-level C encode functions.

```
#include "rtexisrc/osrtexi.h"
```

### Defines

- #define [rtEXIEncEventCode1](#)(pctx, name, part1, nbits1) rtEXIEncEventCode (pctx, name, part1, -1, -1, nbits1, 0, 0)
- #define [rtEXIEncEventCode2](#)(pctx, name, part1, part2, nbits1, nbits2) rtEXIEncEventCode (pctx, name, part1, part2, -1, nbits1, nbits2, 0)
- #define [RTEXIENCCHEVENT](#)(pctx, encfunc, value, stat)

### Functions

- EXTERNEXI int [rtEXIEncAttribute](#) (OSCTXT \*pctx, const OSUTF8CHAR \*prefix, const OSUTF8CHAR \*namespaceURI, const OSUTF8CHAR \*localName, const OSUTF8CHAR \*value)  
*This function writes an attribute in the current element.*
- EXTERNEXI int [rtEXIEncBinary](#) (OSCTXT \*pctx, OSUINT32 nocts, const OSOCTET \*value)  
*This function encodes a binary (OCTET) string value.*
- EXTERNEXI int [rtEXIEncBoolValue](#) (OSCTXT \*pctx, OSBOOL value)  
*This function encodes a boolean value.*
- EXTERNEXI int [rtEXIEncComment](#) (OSCTXT \*pctx, const OSUTF8CHAR \*data)  
*This function encode an XML comment.*
- EXTERNEXI int [rtEXIEncCharacters](#) (OSCTXT \*pctx, const OSUTF8CHAR \*text)  
*This function encodes a string of characters.*
- EXTERNEXI int [rtEXIEncCharArray](#) (OSCTXT \*pctx, const OSUTF8CHAR \*text, OSUINT32 nbytes)  
*This function encodes a given number of bytes from a character array or string.*
- EXTERNEXI int [rtEXIEncDate](#) (OSCTXT \*pctx, const OSNumDateTime \*pvalue)  
*This function encodes a numeric date value.*
- EXTERNEXI int [rtEXIEncDateString](#) (OSCTXT \*pctx, const OSUTF8CHAR \*pvalue)  
*This function encodes a date string value.*
- EXTERNEXI int [rtEXIEncDTD](#) (OSCTXT \*pctx, const OSUTF8CHAR \*name, const OSUTF8CHAR \*publix, const OSUTF8CHAR \*system, const OSUTF8CHAR \*text)  
*This function encodes a DTD declaration.*
- EXTERNEXI int [rtEXIEncEndDocument](#) (OSCTXT \*pctx)  
*This function writes an end document event.*
- EXTERNEXI int [rtEXIEncEndElement](#) (OSCTXT \*pctx)  
*This function writes an end element event.*

- EXTERNEXI int [rtEXIEncEntityRef](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*name)  
*This function writes an XML entity reference.*
- EXTERNEXI int [rtEXIEncEventCode](#) (OSCTXT \*pctxt, const char \*name, OSINT32 part1, OSINT32 part2, OSINT32 part3, OSUINT32 nbits1, OSUINT32 nbits2, OSUINT32 nbits3)  
*This function writes an event code using the given bit field sizes.*
- EXTERNEXI int [rtEXIEncIntCHEvent](#) (OSCTXT \*pctxt, OSINT32 value)  
*This function encodes a variable of the XSD integer type as a CH content event.*
- EXTERNEXI int [rtEXIEncIntElem](#) (OSCTXT \*pctxt, OSINT32 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a complete XML element (<tag>content</tag>) in which the content is a signed integer.*
- EXTERNEXI int [rtEXIEncIntValue](#) (OSCTXT \*pctxt, OSINT32 value)  
*This function encodes a variable of the XSD integer type.*
- EXTERNEXI int [rtEXIEncNBitUIntValue](#) (OSCTXT \*pctxt, OSUINT32 value, OSUINT32 nbits)  
*This function encodes an unsigned integer value in a bit field of the given width.*
- EXTERNEXI int [rtEXIEncNamespace](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*prefix, const OSUTF8CHAR \*namespaceURI, OSBOOL indicator)  
*This function encodes a namespace declaration.*
- EXTERNEXI int [rtEXIEncProcessingInstruction](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*target, const OSUTF8CHAR \*data)  
*This function writes an XML processing instruction.*
- EXTERNEXI int [rtEXIEncStartDocument](#) (OSCTXT \*pctxt)  
*This function writes a start document event.*
- EXTERNEXI int [rtEXIEncStartElement](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function writes a start element event.*
- EXTERNEXI int [rtEXIEncString](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*value, const OSXMLFullQName \*pqname)  
*This function encodes a character string value.*
- EXTERNEXI int [rtEXIEncStringCHEvent](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*value, const OSXMLFullQName \*pqname)  
*This function encodes a character string value as a CH content event.*
- EXTERNEXI int [rtEXIEncStringElem](#) (OSCTXT \*pctxt, OSXMLSTRING \*pxmlstr, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a complete XML element (<tag>content</tag>) in which the content is a string.*
- EXTERNEXI int [rtEXIEncUIntCHEvent](#) (OSCTXT \*pctxt, OSUINT32 value)  
*This function encodes a variable of the XSD unsigned integer type as a CH content event.*

- EXTERNEXI int [rtEXIEncUIntElem](#) (OSCTXT \*pctxt, OSUINT32 value, const OSUTF8CHAR \*elemName, OSXMLNamespace \*pNS)  
*This function encodes a complete XML element (<tag>content</tag>) in which the content is an unsigned integer.*
- EXTERNEXI int [rtEXIEncUIntValue](#) (OSCTXT \*pctxt, OSUINT32 value)  
*This function encodes an unsigned integer value.*
- EXTERNEXI int [rtEXIEncUTF8Str](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*value, size\_t lengthIncr)  
*This function encodes a UTF-8 character string value.*

### 7.11.1 Detailed Description

EXI low-level C encode functions.

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

### 7.11.2 Function Documentation

#### 7.11.2.1 EXTERNEXI int [rtEXIEncAttribute](#) (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *prefix*, const OSUTF8CHAR \* *namespaceURI*, const OSUTF8CHAR \* *localName*, const OSUTF8CHAR \* *value*)

This function writes an attribute in the current element.

##### Parameters:

*pctxt* Pointer to OSCTXT structure

*prefix* The attribute's prefix or empty (null or "") for no prefix.

*namespaceURI* The attribute's namespace URI or empty for the default namespace.

*localName* The attribute's local name which must be a non-empty string.

*value* The attributes's value which must be non-null.

##### Returns:

Completion status of operation:

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

#### 7.11.2.2 EXTERNEXI int [rtEXIEncBinary](#) (OSCTXT \* *pctxt*, OSUINT32 *nocts*, const OSOCTET \* *value*)

This function encodes a binary (OCTET) string value.

##### Parameters:

*pctxt* Pointer to context block structure.

*nocts* Number of octets in value

*value* Pointer to string of octets to be encoded.

##### Returns:

Completion status of operation:

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

### 7.11.2.3 EXTERNEXI int rtEXIEncBoolValue (OSCTXT \* *pctxt*, OSBOOL *value*)

This function encodes a boolean value.

#### Parameters:

*pctxt* Pointer to context block structure.  
*value* OSBOOL value.

#### Returns:

Completion status of operation:

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

### 7.11.2.4 EXTERNEXI int rtEXIEncCharacters (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *text*)

This function encodes a string of characters.

#### Parameters:

*pctxt* Pointer to context block structure.  
*text* Pointer to text to be encoded.

#### Returns:

Completion status of operation:

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

### 7.11.2.5 EXTERNEXI int rtEXIEncCharArray (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *text*, OSUINT32 *nbytes*)

This function encodes a given number of bytes from a character array or string.

This function may be used from within a SAX characters handler when the string is not null-terminated.

#### Parameters:

*pctxt* Pointer to context block structure.  
*text* Pointer to text to be encoded.  
*nbytes* Number of bytes to encode.

#### Returns:

Completion status of operation:

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

#### 7.11.2.6 EXTERNEXI int rtEXIEncComment (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *data*)

This function encode an XML comment.

##### Parameters:

*pctxt* Pointer to context block structure.

*data* The comment to be encoded

##### Returns:

Completion status of operation:

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

#### 7.11.2.7 EXTERNEXI int rtEXIEncDate (OSCTXT \* *pctxt*, const OSNumDateTime \* *pvalue*)

This function encodes a numeric date value.

##### Parameters:

*pctxt* Pointer to context block structure.

*pvalue* Pointer to numeric date/time structure.

##### Returns:

Completion status of operation:

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

#### 7.11.2.8 EXTERNEXI int rtEXIEncDateString (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *pvalue*)

This function encodes a date string value.

##### Parameters:

*pctxt* Pointer to context block structure.

*pvalue* Date string to be encoded

- The format of date is CCYY-MM-DD
- The value of CCYY is from 0000-9999
- The value of MM is 01 - 12
- The value of DD is 01 - XX (where XX is the Days in MM month in CCYY year)

##### Returns:

Completion status of operation:

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

**7.11.2.9 EXTERNEXI int rtEXIEncDTD (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *name*, const OSUTF8CHAR \* *public*, const OSUTF8CHAR \* *system*, const OSUTF8CHAR \* *text*)**

This function encodes a DTD declaration.

**Parameters:**

- pctxt* Pointer to context block structure.
- name* The DTD's root element.
- public* The DTD's public identifier.
- system* The DTD's system identifier.
- text* The DTD's textual representation.

**Returns:**

Completion status of operation:

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

**7.11.2.10 EXTERNEXI int rtEXIEncEndDocument (OSCTXT \* *pctxt*)**

This function writes an end document event.

**Parameters:**

- pctxt* Pointer to context block structure.

**Returns:**

Completion status of operation:

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

**7.11.2.11 EXTERNEXI int rtEXIEncEndElement (OSCTXT \* *pctxt*)**

This function writes an end element event.

**Parameters:**

- pctxt* Pointer to context block structure.

**Returns:**

Completion status of operation:

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

#### 7.11.2.12 EXTERNEXI int rtEXIEncEntityRef (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *name*)

This function writes an XML entity reference.

##### Parameters:

*pctxt* Pointer to context block structure.

*name* The entity's name.

##### Returns:

Completion status of operation:

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

#### 7.11.2.13 EXTERNEXI int rtEXIEncEventCode (OSCTXT \* *pctxt*, const char \* *name*, OSINT32 *part1*, OSINT32 *part2*, OSINT32 *part3*, OSUINT32 *nbits1*, OSUINT32 *nbits2*, OSUINT32 *nbits3*)

This function writes an event code using the given bit field sizes.

##### Parameters:

*pctxt* Pointer to context block structure.

*name* Textual name of event (for example, "SE(\*)")

*part1* Part 1 of event code. -1 if not used.

*part2* Part 2 of event code. -1 if not used.

*part3* Part 3 of event code. -1 if not used.

*nbits1* Length of bit field for part 1.

*nbits2* Length of bit field for part 2.

*nbits3* Length of bit field for part 3.

##### Returns:

Completion status of operation:

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

#### 7.11.2.14 EXTERNEXI int rtEXIEncIntCHEvent (OSCTXT \* *pctxt*, OSINT32 *value*)

This function encodes a variable of the XSD integer type as a CH content event.

##### Parameters:

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

##### Returns:

Completion status of operation:

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

**7.11.2.15 EXTERNEXI int rtEXIEncIntElem (OSCTXT \* *pctxt*, OSINT32 *value*, const OSUTF8CHAR \* *elemName*, OSXMLNamespace \* *pNS*)**

This function encodes a complete XML element (<tag>content</tag>) in which the content is a signed integer.

**Parameters:**

- pctxt* Pointer to context block structure.
- value* Integer value to be encoded.
- elemName* Local name of element.
- pNS* Pointer to namespace structure containing prefix, and namespace URI.

**Returns:**

Completion status of operation:

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

**7.11.2.16 EXTERNEXI int rtEXIEncIntValue (OSCTXT \* *pctxt*, OSINT32 *value*)**

This function encodes a variable of the XSD integer type.

**Parameters:**

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

**Returns:**

Completion status of operation:

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

**7.11.2.17 EXTERNEXI int rtEXIEncNamespace (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *prefix*, const OSUTF8CHAR \* *namespaceURI*, OSBOOL *indicator*)**

This function encodes a namespace declaration.

If the "preserve prefixes" option is set to false, nothing is written.

**Parameters:**

- pctxt* Pointer to context block structure.
- prefix* The non-empty prefix being declared.
- namespaceURI* The namespace associated with the prefix.
- indicator* Indicator bit used to indicate this namespace prefix is associated with last encoded element QName.

**Returns:**

Completion status of operation:

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

### 7.11.2.18 EXTERNEXI int rtEXIEncNBitUIntValue (OSCTXT \* *pctxt*, OSUINT32 *value*, OSUINT32 *nbits*)

This function encodes an unsigned integer value in a bit field of the given width.

#### Parameters:

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

*nbits* Size of bit field.

#### Returns:

Completion status of operation:

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

### 7.11.2.19 EXTERNEXI int rtEXIEncProcessingInstruction (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *target*, const OSUTF8CHAR \* *data*)

This function writes an XML processing instruction.

#### Parameters:

*pctxt* Pointer to context block structure.

*target* The PI's target.

*data* The PI's data.

#### Returns:

Completion status of operation:

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

### 7.11.2.20 EXTERNEXI int rtEXIEncStartDocument (OSCTXT \* *pctxt*)

This function writes a start document event.

This method must be called before writing a document or a fragment.

#### Parameters:

*pctxt* Pointer to context block structure.

#### Returns:

Completion status of operation:

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

**7.11.2.21 EXTERNEXI int rtEXIEncStartElement (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *elemName*, OSXMLNamespace \* *pNS*)**

This function writes a start element event.

If this method is called before `rtEXIEncStartDocument`, then the encoder assumes a fragment is being written.

**Parameters:**

*pctxt* Pointer to context block structure.

*elemName* The non-empty element name.

*pNS* Pointer to namespace structure containing prefix, and namespace URI.

**Returns:**

Completion status of operation:

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

**7.11.2.22 EXTERNEXI int rtEXIEncString (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *value*, const OSXMLFullQName \* *pqname*)**

This function encodes a character string value.

It uses the string tables to determine if the actual string value should be encoded or not.

**Parameters:**

*pctxt* Pointer to context block structure.

*value* Pointer to string value to be encoded.

*pqname* Pointer to a QName structure containing localName, prefix, and namespace URI.

**Returns:**

Completion status of operation:

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

**7.11.2.23 EXTERNEXI int rtEXIEncStringCHEvent (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *value*, const OSXMLFullQName \* *pqname*)**

This function encodes a character string value as a CH content event.

**Parameters:**

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

**Returns:**

Completion status of operation:

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

**7.11.2.24 EXTERNEXI int rtEXIEncStringElem (OSCTXT \* *pctxt*, OSXMLSTRING \* *pxmlstr*, const OSUTF8CHAR \* *elemName*, OSXMLNamespace \* *pNS*)**

This function encodes a complete XML element (<tag>content</tag>) in which the content is a string.

**Parameters:**

- pctxt* Pointer to context block structure.
- pxmlstr* Pointer to string value to be encoded.
- elemName* Local name of element.
- pNS* Pointer to namespace structure containing prefix, and namespace URI.

**Returns:**

Completion status of operation:

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

**7.11.2.25 EXTERNEXI int rtEXIEncUIntCHEvent (OSCTXT \* *pctxt*, OSUINT32 *value*)**

This function encodes a variable of the XSD unsigned integer type as a CH content event.

**Parameters:**

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

**Returns:**

Completion status of operation:

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

**7.11.2.26 EXTERNEXI int rtEXIEncUIntElem (OSCTXT \* *pctxt*, OSUINT32 *value*, const OSUTF8CHAR \* *elemName*, OSXMLNamespace \* *pNS*)**

This function encodes a complete XML element (<tag>content</tag>) in which the content is an unsigned integer.

**Parameters:**

- pctxt* Pointer to context block structure.
- value* Integer value to be encoded.
- elemName* Local name of element.
- pNS* Pointer to namespace structure containing prefix, and namespace URI.

**Returns:**

Completion status of operation:

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

### 7.11.2.27 EXTERNEXI int rtEXIEncUIntValue (OSCTXT \* *pctxt*, OSUINT32 *value*)

This function encodes an unsigned integer value.

#### Parameters:

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

#### Returns:

Completion status of operation:

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

### 7.11.2.28 EXTERNEXI int rtEXIEncUTF8Str (OSCTXT \* *pctxt*, const OSUTF8CHAR \* *value*, size\_t *lengthIncr*)

This function encodes a UTF-8 character string value.

It does not use the string tables to determine if the actual string value should be encoded or not (i.e. it directly encodes the content).

#### Parameters:

*pctxt* Pointer to context block structure.

*value* Pointer to string value to be encoded.

*lengthIncr* Value to be added to string length before encoding. This value is normally zero; however, some string table operations require an increment of 1 or 2.

#### Returns:

Completion status of operation:

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

## 7.12 rtEXIEncStringTable.h File Reference

EXI encoder string table structure and functions.

```
#include "rtxsrc/rtxContext.h"
#include "rtxsrc/rtxHashMapStr2UInt.h"
#include "rtexisrc/rtEXIExternDefs.h"
#include "rtxsrc/rtxHashMapUndef.h"
```

### Classes

- struct [OSEXIEncStringTable](#)

*This structure defines the structure of the various string table partitions used by the encoder.*

### Functions

- EXTERNEXI void [rtEXIEncStringTableInit](#) (OSCTXT \*pctxt, [OSEXIEncStringTable](#) \*pstrtab, size\_t capacity)

*This function initializes the given string table structure.*

- EXTERNEXI [OSEXIEncStringTable](#) \* [rtEXIEncNewStringTable](#) (OSCTXT \*pctxt, size\_t capacity)

*This function allocates and initializes a new string table structure.*

- EXTERNEXI void [rtEXIEncStringTableClear](#) (OSCTXT \*pctxt, [OSEXIEncStringTable](#) \*pstrtab)

*This function clears all strings out of the existing table.*

- EXTERNEXI OSUINT32 [rtEXIEncStringTableAdd](#) (OSCTXT \*pctxt, [OSEXIEncStringTable](#) \*pstrtab, const OSUTF8CHAR \*str)

*This function adds a string to the given string table.*

- EXTERNEXI OSUINT32 [rtEXIEncStringTableGetIndex](#) ([OSEXIEncStringTable](#) \*pstrtab, const OSUTF8CHAR \*str)

*This function gets the index (i.e.*

### 7.12.1 Detailed Description

EXI encoder string table structure and functions.

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

## 7.13 rtEXIEncStringTables.h File Reference

EXI encoder string table structure and functions.

```
#include "rtexisrc/rtEXIEncStringTable.h"
#include "rtxsrc/rtxHashMap.h"
#include "rtxmlsrc/osrtxml.h"
```

### Classes

- struct [OSEXIEncStringTables](#)  
*This structure defines the complete set of string table partitions used by the encoder.*

### Functions

- EXTERNEXI void [rtEXIEncStrTabsInit](#) (OSCTXT \*pctxt, [OSEXIEncStringTables](#) \*pstrtabs)  
*This function initializes all EXI string table partitions.*
- EXTERNEXI void [rtEXIEncStrTabsClear](#) (OSCTXT \*pctxt, [OSEXIEncStringTables](#) \*pstrtabs)  
*This function clears all EXI string table partitions.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsAddURI](#) (OSCTXT \*pctxt, [OSEXIEncStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri)  
*This function will add a URI to the URI string table partition.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetURIID](#) ([OSEXIEncStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri)  
*This function will get the compact identifier of the given URI from the URI string table partition.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetURITableSize](#) ([OSEXIEncStringTables](#) \*pstrtabs)  
*This function returns the current number of entries in the URI string table partition.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsAddPrefix](#) (OSCTXT \*pctxt, [OSEXIEncStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*prefix)  
*This function adds the given prefix to the prefix table partition identified by the given URI.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetPrefixID](#) ([OSEXIEncStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*prefix)  
*This function will get the compact identifier of the given prefix from the prefix string table partition identified by the given URI.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetPrefixTableSize](#) ([OSEXIEncStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri)  
*This function returns the current number of entries in the prefix string table partition identified by the given URI.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsAddLocalName](#) (OSCTXT \*pctxt, [OSEXIEncStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri, const OSUTF8CHAR \*name)  
*This function adds the given local name to the local name table partition identified by the given URI.*

- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetLocalNameID](#) ([OSEXIEncStringTables](#) \*pstrtabs, const OS-UTF8CHAR \*uri, const OSUTF8CHAR \*name)  
*This function will get the compact identifier of the given localName from the localName string table partition identified by the given URI.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetLocalNameTableSize](#) ([OSEXIEncStringTables](#) \*pstrtabs, const OSUTF8CHAR \*uri)  
*This function returns the current number of entries in the localName string table partition identified by the given URI.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsAddLocalValue](#) (OSCTXT \*pctxt, [OSEXIEncStringTables](#) \*pstrtabs, const OSXMLFullQName \*qname, const OSUTF8CHAR \*value)  
*This function adds the given local value to the local value table partition identified by the given QName.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetLocalValueID](#) ([OSEXIEncStringTables](#) \*pstrtabs, const OSXMLFullQName \*qname, const OSUTF8CHAR \*value)  
*This function will get the compact identifier of the given local value from the local value string table partition identified by the given QName.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetLocalValueTableSize](#) ([OSEXIEncStringTables](#) \*pstrtabs, const OSXMLFullQName \*qname)  
*This function returns the current number of entries in the local value string table partition identified by the given QName.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsAddGlobalValue](#) (OSCTXT \*pctxt, [OSEXIEncStringTables](#) \*pstrtabs, const OSUTF8CHAR \*value)  
*This function will add a string value to the global value string table partition.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetGlobalValueID](#) ([OSEXIEncStringTables](#) \*pstrtabs, const OS-UTF8CHAR \*value)  
*This function will get the compact identifier of the given string value from the global value string table partition.*
- EXTERNEXI OSUINT32 [rtEXIEncStrTabsGetGlobalValueTableSize](#) ([OSEXIEncStringTables](#) \*pstrtabs)  
*This function returns the current number of entries in the global value string table partition.*

### 7.13.1 Detailed Description

EXI encoder string table structure and functions.

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

## 7.14 rtEXIEvent.h File Reference

EXI event definitions and functions.

```
#include "rtexisrc/rtEXIExternDefs.h"  
#include "rtxsrc/rtxXmlQName.h"
```

### Classes

- [struct \\_OSEXIEvent](#)

### Enumerations

- [enum OSEXIEventType](#)  
*A structure representing an EXI event.*

### Functions

- EXTERNEXI void [rtEXIEventInit](#) (OSCTXT \*pctxt, [OSEXIEvent](#) \*pEvent, [OSEXIEventType](#) type, const OSXMLFullQName \*qname)  
*This function initializes an event structure.*
- EXTERNEXI [OSEXIEvent](#) \* [rtEXINewEvent](#) (OSCTXT \*pctxt, [OSEXIEventType](#) type, const OSXMLFullQName \*qname)  
*This function allocates a new event structure and initializes it.*
- EXTERNEXI [OSEXIEvent](#) \* [rtEXINewEventDeepCopy](#) (OSCTXT \*pctxt, const [OSEXIEvent](#) \*pEvent)  
*This function creates a deep copy of the given event record.*
- EXTERNEXI void [rtEXIEventDeepCopy](#) (OSCTXT \*pctxt, [OSEXIEvent](#) \*pdest, const [OSEXIEvent](#) \*psrc)  
*This function makes a deep copy of the given Event including the strings inside.*
- EXTERNEXI void [rtEXIEventFreeMem](#) (OSCTXT \*pctxt, [OSEXIEvent](#) \*pEvent, OSBOOL dynamic)  
*This function frees all memory within an Event structure.*
- EXTERNEXI OSUINT32 [rtEXIEventHash](#) (const [OSEXIEvent](#) \*pEvent)  
*This function computes a hash code for an event structure.*
- EXTERNEXI OSBOOL [rtEXIEventsEqual](#) (const [OSEXIEvent](#) \*pEvent1, const [OSEXIEvent](#) \*pEvent2)  
*This function tests if two event structures are equal.*
- EXTERNEXI const [OSEXIEvent](#) \* [rtEXIGetBaseEvent](#) (const [OSEXIEvent](#) \*pEvent)  
*This function gets the base event corresponding to a given event.*
- EXTERNEXI const char \* [rtEXIEventTypeToString](#) ([OSEXIEventType](#) type)  
*This function returns the textual name for the given event type (for example, "SD", "SE", etc).*
- EXTERNEXI OSUTF8CHAR \* [rtEXIEventToString](#) (OSCTXT \*pctxt, const [OSEXIEvent](#) \*pEvent)

*This function returns the full textual description for the given event.*

- EXTERNEXI void **rtEXIEventPrint** (const **OSEXIEvent** \*pEvent)

*This function prints information on the given event to stdout.*

## 7.14.1 Detailed Description

EXI event definitions and functions.

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

## 7.14.2 Enumeration Type Documentation

### 7.14.2.1 enum **OSEXIEventType**

A structure representing an EXI event.

An event is a pair of an event type and a possibly null qname. Only SE and AT events accept a qname parameter. For all other events, the static constants defined in this class must be used. Note that SE(\*) and AT(\*) are represented by SE and AT, respectively.

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

## 7.14.3 Function Documentation

### 7.14.3.1 EXTERNEXI void **rtEXIEventDeepCopy** (OSCTXT \* *pctxt*, **OSEXIEvent** \* *pdest*, const **OSEXIEvent** \* *psrc*)

This function makes a deep copy of the given Event including the strings inside.

#### Parameters:

*pctxt* Pointer to a context structure.

*pdest* Pointer to Event to receive copied data.

*psrc* Pointer to Event to be copied.

### 7.14.3.2 EXTERNEXI void **rtEXIEventFreeMem** (OSCTXT \* *pctxt*, **OSEXIEvent** \* *pEvent*, OSBOOL *dynamic*)

This function frees all memory within an Event structure.

#### Parameters:

*pctxt* Pointer to a context structure.

*pEvent* Pointer to Event in which memory will be freed.

*dynamic* Boolean indicating if pEvent is dynamic. If true, the memory for pEvent is freed.

#### 7.14.3.3 EXTERNEXI OSUINT32 rtEXIEventHash (const OSEXIEvent \* pEvent)

This function computes a hash code for an event structure.

##### Parameters:

*pEvent* Pointer to event structure.

##### Returns:

Computed hash code.

#### 7.14.3.4 EXTERNEXI void rtEXIEventInit (OSCTXT \* pctxt, OSEXIEvent \* pEvent, OSEXIEventType type, const OSXMLFullQName \* qname)

This function initializes an event structure.

##### Parameters:

*pctxt* Pointer to a context block structure.

*pEvent* Pointer to event structure.

*type* Type of the event.

*qname* QName associated with the event (possibly NULL).

#### 7.14.3.5 EXTERNEXI void rtEXIEventPrint (const OSEXIEvent \* pEvent)

This function prints information on the given event to stdout.

##### Parameters:

*pEvent* Pointer to event structure.

#### 7.14.3.6 EXTERNEXI OSBOOL rtEXIEventsEqual (const OSEXIEvent \* pEvent1, const OSEXIEvent \* pEvent2)

This function tests if two event structures are equal.

##### Parameters:

*pEvent1* Pointer to event structure.

*pEvent2* Pointer to event structure.

##### Returns:

True if events are equal.

#### 7.14.3.7 EXTERNEXI OSUTF8CHAR\* rtEXIEventToString (OSCTXT \* *pctxt*, const OSEXIEvent \* *pEvent*)

This function returns the full textual description for the given event.

This includes the type (for example, "SD", "SE", etc.) and, if SE or AT, the local name or '\*'.

##### Parameters:

*pctxt* Pointer to a context structure.

*pEvent* Pointer to Event structure.

##### Returns:

Text corresponding to the type.

#### 7.14.3.8 EXTERNEXI const char\* rtEXIEventTypeToString (OSEXIEventType *type*)

This function returns the textual name for the given event type (for example, "SD", "SE", etc. ). The code is returned as string literal text.

##### Parameters:

*type* Enumerated event type.

##### Returns:

Text corresponding to the type.

#### 7.14.3.9 EXTERNEXI const OSEXIEvent\* rtEXIGetBaseEvent (const OSEXIEvent \* *pEvent*)

This function gets the base event corresponding to a given event.

For SE or AT events, this is SE(\*) or AT(\*) respectively. For all other events, it is the event itself.

##### Parameters:

*pEvent* Pointer to event structure.

##### Returns:

Pointer to base event.

#### 7.14.3.10 EXTERNEXI OSEXIEvent\* rtEXINewEvent (OSCTXT \* *pctxt*, OSEXIEventType *type*, const OSXMLFullQName \* *qname*)

This function allocates a new event structure and initializes it.

##### Parameters:

*pctxt* Pointer to a context block structure.

*type* Type of the event.

*qname* QName associated with the event (possibly NULL).

**Returns:**

Pointer to allocated event structure or NULL if no dynamic memory available.

**7.14.3.11 EXTERNEXI OSEXIEvent\* rtEXINewEventDeepCopy (OSCTXT \* *pctxt*, const OSEXIEvent \* *pEvent*)**

This function creates a deep copy of the given event record.

Memory for the record is allocated using `rtxMemAlloc`. The record should be freed using `rtEXIEventFreeMem`.

**Parameters:**

*pctxt* Pointer to a context block structure.

*pEvent* Pointer to event record to be copied.

**Returns:**

Pointer to allocated event structure or NULL if no dynamic memory available.

## 7.15 rtEXIEventCode.h File Reference

EXI event code definitions and functions.

```
#include "rtexisrc/rtEXIExternDefs.h"  
#include "rtxsrc/rtxContext.h"
```

### Classes

- struct [OSEXIEventCode](#)  
*A structure representing a production's event code.*

### Defines

- #define [rtEXISetEventCode1](#)(pEventCode, part1) [rtEXISetEventCode3](#)(pEventCode, part1, OSINT32\_MIN, OSINT32\_MIN)  
*This macro sets a one-part event code.*
- #define [rtEXISetEventCode2](#)(pEventCode, part1, part2) [rtEXISetEventCode3](#)(pEventCode, part1, part2, OSINT32\_MIN)  
*This macro sets a two-part event code.*

### Functions

- EXTERNEXI int [rtEXIEventCodeCompare](#) (const [OSEXIEventCode](#) \*pec1, const [OSEXIEventCode](#) \*pec2)  
*This function compares two event codes.*
- EXTERNEXI [OSEXIEventCode](#) \* [rtEXIEventCodeCopy](#) (OSCTXT \*pctxt, const [OSEXIEventCode](#) \*pec)  
*This function does a deep-copy of an event code structure.*
- EXTERNEXI OSBOOL [rtEXIEventCodesEqual](#) (const [OSEXIEventCode](#) \*pec1, const [OSEXIEventCode](#) \*pec2)  
*This function compares two event codes for equality.*
- EXTERNEXI char \* [rtEXIEventCodeToString](#) (OSCTXT \*pctxt, const [OSEXIEventCode](#) \*pec)  
*This function returns a string representation of the given event code in dot notation (part1.part2.part3).*
- EXTERNEXI OSUINT32 [rtEXIEventCodeLength](#) (const [OSEXIEventCode](#) \*pec)  
*This function returns the length of the given event code.*
- EXTERNEXI [OSEXIEventCode](#) \* [rtEXINewEventCode1](#) (OSCTXT \*pctxt, OSINT32 part1)  
*This function allocates and initializes a one-part event code.*
- EXTERNEXI [OSEXIEventCode](#) \* [rtEXINewEventCode2](#) (OSCTXT \*pctxt, OSINT32 part1, OSINT32 part2)  
*This function allocates and initializes a two-part event code.*

- EXTERNEXI [OSEXIEventCode](#) \* [rtEXINewEventCode3](#) (OSCTXT \*pctxt, OSINT32 part1, OSINT32 part2, OSINT32 part3)  
*This function allocates and initializes a three-part event code.*
- EXTERNEXI void [rtEXISetEventCode3](#) ([OSEXIEventCode](#) \*pEventCode, OSINT32 part1, OSINT32 part2, OSINT32 part3)  
*This function sets a three-part event code.*
- EXTERNEXI void [rtEXIEventCodePrint](#) (const [OSEXIEventCode](#) \*pEventCode)  
*This function prints information on the given event code to stdout.*

### 7.15.1 Detailed Description

EXI event code definitions and functions.

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

## 7.16 rtEXIEventCodeGroup.h File Reference

EXI event code definitions and functions.

```
#include "rtexisrc/rtEXIEventCode.h"  
#include "rtxsrc/rtxDynBitSet.h"  
#include "rtxsrc/rtxSList.h"
```

### Classes

- struct [OSEXIEventCodeGroup](#)  
*An EventCodeGroup is a group of related event codes.*

### Defines

- #define [rtEXIEventCodeGroupHasPart1](#)(pecgrp, part1) rtxDynBitSetTestBit(&pecgrp → part1Set,part1)  
*This macro returns true if there is an event code in this group whose length is 1 and whose first part is equal to part1.*
- #define [rtEXIEventCodeGroupHasPart2](#)(pecgrp, part2) rtxDynBitSetTestBit(&pecgrp → part2Set,part2)  
*This macro returns true if there is an event code in this group whose length is 2 and whose first part is equal to part2.*

### Functions

- EXTERNEXI void [rtEXIEventCodeGroupInit](#) (OSCTXT \*pctxt, [OSEXIEventCodeGroup](#) \*pecgrp)  
*This function initializes an event code group structure.*
- EXTERNEXI [OSEXIEventCodeGroup](#) \* [rtEXINewEventCodeGroup](#) (OSCTXT \*pctxt)  
*This function allocates and initializes a new event code group structure.*
- EXTERNEXI int [rtEXIEventCodeGroupAdd](#) ([OSEXIEventCodeGroup](#) \*pecgrp, const [OSEXIEventCode](#) \*pec)  
*This function adds an event code to an event code group and updates the maximum part variables.*
- EXTERNEXI [OSEXIEventCodeGroup](#) \* [rtEXIEventCodeGroupCopy](#) (const [OSEXIEventCodeGroup](#) \*pecgrp)  
*This function performs a deep copy of the given event group.*
- EXTERNEXI void [rtEXIEventCodeGroupFreeMem](#) ([OSEXIEventCodeGroup](#) \*pecgrp)  
*This function frees all memory associated with an event group.*
- EXTERNEXI int [rtEXIEventCodeGroupGetBitsPart1](#) ([OSEXIEventCodeGroup](#) \*pecgrp)  
*This function returns the number of bits necessary to encode the max part1 value in this group.*
- EXTERNEXI int [rtEXIEventCodeGroupGetBitsPart2](#) ([OSEXIEventCodeGroup](#) \*pecgrp)  
*This function returns the number of bits necessary to encode the max part2 value in this group.*
- EXTERNEXI int [rtEXIEventCodeGroupGetBitsPart3](#) ([OSEXIEventCodeGroup](#) \*pecgrp)

*This function returns the number of bits necessary to encode the max part3 value in this group.*

- EXTERNEXI void [rtEXIEventCodeGroupIncrPart1](#) (OSCTXT \*pctxt, [OSEXIEventCodeGroup](#) \*pecgrp)

*This function increments part1 in all event codes in this group, as well as the max part1 value.*

### **7.16.1 Detailed Description**

EXI event code definitions and functions.

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

## 7.17 `rtEXIExternDefs.h` File Reference

EXI external function declaration macros for building Windows DLL's.

### Defines

- #define [EXTERNEXI](#)

### 7.17.1 Detailed Description

EXI external function declaration macros for building Windows DLL's.

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

## 7.18 rtXML2EXI.h File Reference

Functions for serializing XML data into EXI format.

```
#include "rtexisrc/osrtexi.h"
```

### Functions

- EXTERNEXI int [rtXmlMem2ExiMem](#) (OSCTXT \*pctxt, OSOCTET \*outbuf, size\_t outbufsiz)  
*This function serializes XML data to EXI data and stores the result in the given memory buffer.*
- EXTERNEXI int [rtXmlFile2ExiStream](#) (const char \*xmlFileName, OSCTXT \*pExiCtxt)  
*This function serializes XML data from a file to EXI data and outputs the result to the output stream defined within the given output context.*
- EXTERNEXI int [rtXmlMem2ExiStream](#) (OSCTXT \*pctxt, OSCTXT \*pExiCtxt)  
*This function serializes XML data to EXI data and outputs the result to the output stream defined within the given output context.*

### 7.18.1 Detailed Description

Functions for serializing XML data into EXI format.

Definition in file [rtXML2EXI.h](#).

## 7.19 rtXML2EXISAX.h File Reference

SAX callback functions for serializing XML data into EXI format.

```
#include "rtexisrc/osrtexi.h"
```

### Classes

- struct [XML2EXISAXUserData](#)

### Functions

- EXTERNEXI int [xml2ExiStartElement](#) (void \*userData, const OSUTF8CHAR \*localname, const OSUTF8CHAR \*qname, const OSUTF8CHAR \*const \*attrs)  
*SAX start element handler callback function.*
- EXTERNEXI int [xml2ExiCharacters](#) (void \*userData, const OSUTF8CHAR \*chars, int length)  
*SAX characters handler callback function definition.*
- EXTERNEXI int [xml2ExiEndElement](#) (void \*userData, const OSUTF8CHAR \*localname, const OSUTF8CHAR \*qname)  
*SAX end element handler callback function definition.*

### 7.19.1 Detailed Description

SAX callback functions for serializing XML data into EXI format.

Definition in file [rtXML2EXISAX.h](#).

# Index

- [\\_OSEXIEvent](#), [59](#)
    - [qname](#), [59](#)
  - [CharacterDataHandler](#)
    - [rtEXI2XML](#), [15](#)
  - [EndElementHandler](#)
    - [rtEXI2XML](#), [16](#)
  - [eventCodeGroups](#)
    - [OSEXIAutomaton](#), [64](#)
  - [eventStates](#)
    - [OSEXIAutomaton](#), [64](#)
  - [EXI decode structures and functions.](#), [18](#)
  - [EXI encode structures and functions.](#), [39](#)
  - [EXI event code definitions and functions.](#), [49](#)
  - [EXI runtime library functions.](#), [5](#)
  - [EXI to XML serialization functions.](#), [15](#)
  - [EXI2SAXAttribute](#), [60](#)
  - [EXI2SAXReader](#), [61](#)
- [globalValueTable](#)
    - [OSEXIDecStringTables](#), [69](#)
    - [OSEXIEncStringTables](#), [72](#)
  - [isClosed](#)
    - [OSEXIAutomaton](#), [64](#)
  - [localNameTables](#)
    - [OSEXIDecStringTables](#), [68](#)
    - [OSEXIEncStringTables](#), [71](#)
  - [localValueTables](#)
    - [OSEXIDecStringTables](#), [69](#)
    - [OSEXIEncStringTables](#), [72](#)
  - [matchedBaseEvent](#)
    - [OSEXIAutomaton](#), [64](#)
  - [OSEXIAtmState](#), [62](#)
  - [OSEXIAutomaton](#), [63](#)
    - [eventCodeGroups](#), [64](#)
    - [eventStates](#), [64](#)
    - [isClosed](#), [64](#)
    - [matchedBaseEvent](#), [64](#)
    - [pDynEvent](#), [64](#)
  - [OSEXICtxtInfo](#), [65](#)
  - [OSEXICtxtInfo](#)
    - [pDecState](#), [66](#)
  - [OSEXIDecStringTable](#), [67](#)
  - [OSEXIDecStringTable](#)
    - [records](#), [67](#)
  - [OSEXIDecStringTables](#), [68](#)
  - [OSEXIDecStringTables](#)
    - [globalValueTable](#), [69](#)
    - [localNameTables](#), [68](#)
    - [localValueTables](#), [69](#)
    - [prefixTables](#), [68](#)
    - [uriTable](#), [68](#)
  - [OSEXIEncStringTable](#), [70](#)
  - [OSEXIEncStringTables](#), [71](#)
  - [OSEXIEncStringTables](#)
    - [globalValueTable](#), [72](#)
    - [localNameTables](#), [71](#)
    - [localValueTables](#), [72](#)
    - [prefixTables](#), [71](#)
    - [uriTable](#), [71](#)
  - [OSEXIEventCode](#), [73](#)
  - [OSEXIEventCodeGroup](#), [74](#)
  - [OSEXIEventType](#)
    - [rtEXIEvent.h](#), [110](#)
  - [OSEXIHeader](#), [75](#)
  - [OSEXIStateEvent](#), [76](#)
  - [OSEXIStateTableRecord](#), [77](#)
  - [osrtexi.h](#), [79](#)
- [pDecState](#)
    - [OSEXICtxtInfo](#), [66](#)
  - [pDynEvent](#)
    - [OSEXIAutomaton](#), [64](#)
  - [prefixTables](#)
    - [OSEXIDecStringTables](#), [68](#)
    - [OSEXIEncStringTables](#), [71](#)
  - [qname](#)
    - [\\_OSEXIEvent](#), [59](#)
  - [records](#)
    - [OSEXIDecStringTable](#), [67](#)
  - [rtEXI](#)
    - [rtEXIAtmAddUndeclaredContentItems](#), [9](#)
    - [rtEXIAtmAddUndeclaredItems](#), [10](#)
    - [rtEXIAtmAddUndeclaredStartTagItems](#), [10](#)
    - [rtEXIAtmGetCurrentEventCodeGroup](#), [10](#)

- rtEXIAutomatonAddTransition, 11
- rtEXIAutomatonCopy, 11
- rtEXIAutomatonFreeMem, 11
- rtEXIAutomatonInit, 12
- rtEXIAutomatonInitCopy, 12
- rtEXIAutomatonPop, 12
- rtEXIAutomatonPush, 12
- rtEXIClearOption, 8
- rtEXIEnableBitFieldTrace, 13
- rtEXIGetDocAutomaton, 13
- rtEXIGetElemAutomaton, 13
- rtEXIGetMsgLen, 8
- rtEXIGetMsgPtr, 8
- rtEXIInitContext, 13
- rtEXIInitCtxtAppInfo, 14
- rtEXINewAutomaton, 14
- rtEXISetBufPtr, 8
- rtEXISetOption, 9
- rtEXITestOption, 9
- rtEXI2SAX.h, 81
- rtEXI2SAXCreateReader
  - rtEXI2XML, 16
- rtEXI2SAXParse
  - rtEXI2XML, 17
- rtEXI2XML
  - CharacterDataHandler, 15
  - EndElementHandler, 16
  - rtEXI2SAXCreateReader, 16
  - rtEXI2SAXParse, 17
  - rtExi2XmlStream, 17
  - StartElementHandler, 16
- rtEXI2XML.h, 82
- rtExi2XmlStream
  - rtEXI2XML, 17
- rtEXIAtmAddUndeclaredContentItems
  - rtEXI, 9
- rtEXIAtmAddUndeclaredItems
  - rtEXI, 10
- rtEXIAtmAddUndeclaredStartTagItems
  - rtEXI, 10
- rtEXIAtmGetCurrentEventCodeGroup
  - rtEXI, 10
- rtEXIAutomaton.h, 83
- rtEXIAutomatonAddTransition
  - rtEXI, 11
- rtEXIAutomatonCopy
  - rtEXI, 11
- rtEXIAutomatonFreeMem
  - rtEXI, 11
- rtEXIAutomatonInit
  - rtEXI, 12
- rtEXIAutomatonInitCopy
  - rtEXI, 12
- rtEXIAutomatonPop
  - rtEXI, 12
- rtEXIAutomatonPush
  - rtEXI, 12
- rtEXIClearOption
  - rtEXI, 8
- rtEXIDec
  - rtEXIDec\_CH\_String\_EE, 22
  - rtEXIDecAtmAddTransition, 22
  - rtEXIDecAttribute, 22
  - rtEXIDecAutomatonAdvance, 23
  - rtEXIDecBoolValue, 23
  - rtEXIDecDate, 23
  - rtEXIDecDateString, 24
  - rtEXIDecDocumentType, 24
  - rtEXIDecEventCodePart1, 25
  - rtEXIDecGetDocAutomaton, 25
  - rtEXIDecGetElemAutomaton, 25
  - rtEXIDecHasNext, 26
  - rtEXIDecIntValue, 26
  - rtEXIDecLocalName, 26
  - rtEXIDecNamespaceURI, 26
  - rtEXIDecNBitUIntValue, 27
  - rtEXIDecNewStringTable, 27
  - rtEXIDecNextEventType, 27
  - rtEXIDecoderInit, 28
  - rtEXIDecPrefix, 28
  - rtEXIDecProcessingInstruction, 28
  - rtEXIDecQName, 29
  - rtEXIDecReset, 29
  - rtEXIDecString, 30
  - rtEXIDecStringCEvent, 30
  - rtEXIDecStringLength, 30
  - rtEXIDecStringTableAdd, 31
  - rtEXIDecStringTableClear, 31
  - rtEXIDecStringTableGetString, 31
  - rtEXIDecStringTableInit, 32
  - rtEXIDecStringToCharArray, 32
  - rtEXIDecStrTabsAddGlobalValue, 32
  - rtEXIDecStrTabsAddLocalName, 32
  - rtEXIDecStrTabsAddLocalValue, 33
  - rtEXIDecStrTabsAddPrefix, 33
  - rtEXIDecStrTabsAddURI, 33
  - rtEXIDecStrTabsClear, 34
  - rtEXIDecStrTabsGetGlobalValue, 34
  - rtEXIDecStrTabsGetGlobalValueTableSize, 34
  - rtEXIDecStrTabsGetLocalName, 34
  - rtEXIDecStrTabsGetLocalNameTableSize, 35
  - rtEXIDecStrTabsGetLocalValue, 35
  - rtEXIDecStrTabsGetLocalValueTableSize, 35
  - rtEXIDecStrTabsGetPrefix, 36
  - rtEXIDecStrTabsGetPrefixTableSize, 36
  - rtEXIDecStrTabsGetURI, 36
  - rtEXIDecStrTabsGetURITableSize, 36
  - rtEXIDecStrTabsInit, 37

- rtEXIDecUIntValue, [37](#)
- rtEXIDecUTF8Chars, [37](#)
- rtEXIDecUTF8Str, [37](#)
- rtEXIDec\_CH\_String\_EE
  - rtEXIDec, [22](#)
- rtEXIDecAtmAddTransition
  - rtEXIDec, [22](#)
- rtEXIDecAttribute
  - rtEXIDec, [22](#)
- rtEXIDecAutomaton.h, [85](#)
- rtEXIDecAutomatonAdvance
  - rtEXIDec, [23](#)
- rtEXIDecBitTrace.h, [86](#)
- rtEXIDecBoolValue
  - rtEXIDec, [23](#)
- rtEXIDecDate
  - rtEXIDec, [23](#)
- rtEXIDecDateString
  - rtEXIDec, [24](#)
- rtEXIDecDocumentType
  - rtEXIDec, [24](#)
- rtEXIDecEventCodePart1
  - rtEXIDec, [25](#)
- rtEXIDecGetDocAutomaton
  - rtEXIDec, [25](#)
- rtEXIDecGetElemAutomaton
  - rtEXIDec, [25](#)
- rtEXIDecHasNext
  - rtEXIDec, [26](#)
- rtEXIDecIntValue
  - rtEXIDec, [26](#)
- rtEXIDecLocalName
  - rtEXIDec, [26](#)
- rtEXIDecNamespaceURI
  - rtEXIDec, [26](#)
- rtEXIDecNBitUIntValue
  - rtEXIDec, [27](#)
- rtEXIDecNewStringTable
  - rtEXIDec, [27](#)
- rtEXIDecNextEventType
  - rtEXIDec, [27](#)
- rtEXIDecoder.h, [87](#)
- rtEXIDecoderInit
  - rtEXIDec, [28](#)
- rtEXIDecPrefix
  - rtEXIDec, [28](#)
- rtEXIDecProcessingInstruction
  - rtEXIDec, [28](#)
- rtEXIDecQName
  - rtEXIDec, [29](#)
- rtEXIDecReset
  - rtEXIDec, [29](#)
- rtEXIDecString
  - rtEXIDec, [30](#)

- rtEXIDecStringCHEvent
  - rtEXIDec, [30](#)
- rtEXIDecStringLength
  - rtEXIDec, [30](#)
- rtEXIDecStringTable.h, [90](#)
- rtEXIDecStringTableAdd
  - rtEXIDec, [31](#)
- rtEXIDecStringTableClear
  - rtEXIDec, [31](#)
- rtEXIDecStringTableGetString
  - rtEXIDec, [31](#)
- rtEXIDecStringTableInit
  - rtEXIDec, [32](#)
- rtEXIDecStringTables.h, [91](#)
- rtEXIDecStringToCharArray
  - rtEXIDec, [32](#)
- rtEXIDecStrTabsAddGlobalValue
  - rtEXIDec, [32](#)
- rtEXIDecStrTabsAddLocalName
  - rtEXIDec, [32](#)
- rtEXIDecStrTabsAddLocalValue
  - rtEXIDec, [33](#)
- rtEXIDecStrTabsAddPrefix
  - rtEXIDec, [33](#)
- rtEXIDecStrTabsAddURI
  - rtEXIDec, [33](#)
- rtEXIDecStrTabsClear
  - rtEXIDec, [34](#)
- rtEXIDecStrTabsGetGlobalValue
  - rtEXIDec, [34](#)
- rtEXIDecStrTabsGetGlobalValueTableSize
  - rtEXIDec, [34](#)
- rtEXIDecStrTabsGetLocalName
  - rtEXIDec, [34](#)
- rtEXIDecStrTabsGetLocalNameTableSize
  - rtEXIDec, [35](#)
- rtEXIDecStrTabsGetLocalValue
  - rtEXIDec, [35](#)
- rtEXIDecStrTabsGetLocalValueTableSize
  - rtEXIDec, [35](#)
- rtEXIDecStrTabsGetPrefix
  - rtEXIDec, [36](#)
- rtEXIDecStrTabsGetPrefixTableSize
  - rtEXIDec, [36](#)
- rtEXIDecStrTabsGetURI
  - rtEXIDec, [36](#)
- rtEXIDecStrTabsGetURITableSize
  - rtEXIDec, [36](#)
- rtEXIDecStrTabsInit
  - rtEXIDec, [37](#)
- rtEXIDecUIntValue
  - rtEXIDec, [37](#)
- rtEXIDecUTF8Chars
  - rtEXIDec, [37](#)

- rtEXIDecUTF8Str
  - rtEXIDec, [37](#)
- rtEXIEnableBitFieldTrace
  - rtEXI, [13](#)
- rtEXIEnc
  - rtEXIEncAtmAddTransition, [41](#)
  - rtEXIEncAutomatonAdvance, [41](#)
  - rtEXIEncGetDocAutomaton, [42](#)
  - rtEXIEncGetElemAutomaton, [42](#)
  - rtEXIEncNewStringTable, [42](#)
  - rtEXIEncStringTableAdd, [42](#)
  - rtEXIEncStringTableClear, [43](#)
  - rtEXIEncStringTableGetIndex, [43](#)
  - rtEXIEncStringTableInit, [43](#)
  - rtEXIEncStrTabsAddGlobalValue, [43](#)
  - rtEXIEncStrTabsAddLocalName, [44](#)
  - rtEXIEncStrTabsAddLocalValue, [44](#)
  - rtEXIEncStrTabsAddPrefix, [44](#)
  - rtEXIEncStrTabsAddURI, [45](#)
  - rtEXIEncStrTabsClear, [45](#)
  - rtEXIEncStrTabsGetGlobalValueID, [45](#)
  - rtEXIEncStrTabsGetGlobalValueTableSize, [45](#)
  - rtEXIEncStrTabsGetLocalNameID, [46](#)
  - rtEXIEncStrTabsGetLocalNameTableSize, [46](#)
  - rtEXIEncStrTabsGetLocalValueID, [46](#)
  - rtEXIEncStrTabsGetLocalValueTableSize, [47](#)
  - rtEXIEncStrTabsGetPrefixID, [47](#)
  - rtEXIEncStrTabsGetPrefixTableSize, [47](#)
  - rtEXIEncStrTabsGetURIID, [47](#)
  - rtEXIEncStrTabsGetURITableSize, [48](#)
  - rtEXIEncStrTabsInit, [48](#)
- rtEXIEncAtmAddTransition
  - rtEXIEnc, [41](#)
- rtEXIEncAttribute
  - rtEXIEncoder.h, [96](#)
- rtEXIEncAutomaton.h, [93](#)
- rtEXIEncAutomatonAdvance
  - rtEXIEnc, [41](#)
- rtEXIEncBinary
  - rtEXIEncoder.h, [96](#)
- rtEXIEncBoolValue
  - rtEXIEncoder.h, [97](#)
- rtEXIEncCharacters
  - rtEXIEncoder.h, [97](#)
- rtEXIEncCharArray
  - rtEXIEncoder.h, [97](#)
- rtEXIEncComment
  - rtEXIEncoder.h, [97](#)
- rtEXIEncDate
  - rtEXIEncoder.h, [98](#)
- rtEXIEncDateString
  - rtEXIEncoder.h, [98](#)
- rtEXIEncDTD
  - rtEXIEncoder.h, [98](#)
- rtEXIEncEndDocument
  - rtEXIEncoder.h, [99](#)
- rtEXIEncEndElement
  - rtEXIEncoder.h, [99](#)
- rtEXIEncEntityRef
  - rtEXIEncoder.h, [99](#)
- rtEXIEncEventCode
  - rtEXIEncoder.h, [100](#)
- rtEXIEncGetDocAutomaton
  - rtEXIEnc, [42](#)
- rtEXIEncGetElemAutomaton
  - rtEXIEnc, [42](#)
- rtEXIEncIntCHEvent
  - rtEXIEncoder.h, [100](#)
- rtEXIEncIntElem
  - rtEXIEncoder.h, [100](#)
- rtEXIEncIntValue
  - rtEXIEncoder.h, [101](#)
- rtEXIEncNamespace
  - rtEXIEncoder.h, [101](#)
- rtEXIEncNBitUIntValue
  - rtEXIEncoder.h, [101](#)
- rtEXIEncNewStringTable
  - rtEXIEnc, [42](#)
- rtEXIEncoder.h, [94](#)
- rtEXIEncoder.h
  - rtEXIEncAttribute, [96](#)
  - rtEXIEncBinary, [96](#)
  - rtEXIEncBoolValue, [97](#)
  - rtEXIEncCharacters, [97](#)
  - rtEXIEncCharArray, [97](#)
  - rtEXIEncComment, [97](#)
  - rtEXIEncDate, [98](#)
  - rtEXIEncDateString, [98](#)
  - rtEXIEncDTD, [98](#)
  - rtEXIEncEndDocument, [99](#)
  - rtEXIEncEndElement, [99](#)
  - rtEXIEncEntityRef, [99](#)
  - rtEXIEncEventCode, [100](#)
  - rtEXIEncIntCHEvent, [100](#)
  - rtEXIEncIntElem, [100](#)
  - rtEXIEncIntValue, [101](#)
  - rtEXIEncNamespace, [101](#)
  - rtEXIEncNBitUIntValue, [101](#)
  - rtEXIEncProcessingInstruction, [102](#)
  - rtEXIEncStartDocument, [102](#)
  - rtEXIEncStartElement, [102](#)
  - rtEXIEncString, [103](#)
  - rtEXIEncStringCHEvent, [103](#)
  - rtEXIEncStringElem, [103](#)
  - rtEXIEncUIntCHEvent, [104](#)
  - rtEXIEncUIntElem, [104](#)
  - rtEXIEncUIntValue, [104](#)
  - rtEXIEncUTF8Str, [105](#)

- rtEXIEncProcessingInstruction
  - rtEXIEncoder.h, [102](#)
- rtEXIEncStartDocument
  - rtEXIEncoder.h, [102](#)
- rtEXIEncStartElement
  - rtEXIEncoder.h, [102](#)
- rtEXIEncString
  - rtEXIEncoder.h, [103](#)
- rtEXIEncStringCEvent
  - rtEXIEncoder.h, [103](#)
- rtEXIEncStringElem
  - rtEXIEncoder.h, [103](#)
- rtEXIEncStringTable.h, [106](#)
- rtEXIEncStringTableAdd
  - rtEXIEnc, [42](#)
- rtEXIEncStringTableClear
  - rtEXIEnc, [43](#)
- rtEXIEncStringTableGetIndex
  - rtEXIEnc, [43](#)
- rtEXIEncStringTableInit
  - rtEXIEnc, [43](#)
- rtEXIEncStringTables.h, [107](#)
- rtEXIEncStrTabsAddGlobalValue
  - rtEXIEnc, [43](#)
- rtEXIEncStrTabsAddLocalName
  - rtEXIEnc, [44](#)
- rtEXIEncStrTabsAddLocalValue
  - rtEXIEnc, [44](#)
- rtEXIEncStrTabsAddPrefix
  - rtEXIEnc, [44](#)
- rtEXIEncStrTabsAddURI
  - rtEXIEnc, [45](#)
- rtEXIEncStrTabsClear
  - rtEXIEnc, [45](#)
- rtEXIEncStrTabsGetGlobalValueID
  - rtEXIEnc, [45](#)
- rtEXIEncStrTabsGetGlobalValueTableSize
  - rtEXIEnc, [45](#)
- rtEXIEncStrTabsGetLocalNameID
  - rtEXIEnc, [46](#)
- rtEXIEncStrTabsGetLocalNameTableSize
  - rtEXIEnc, [46](#)
- rtEXIEncStrTabsGetLocalValueID
  - rtEXIEnc, [46](#)
- rtEXIEncStrTabsGetLocalValueTableSize
  - rtEXIEnc, [47](#)
- rtEXIEncStrTabsGetPrefixID
  - rtEXIEnc, [47](#)
- rtEXIEncStrTabsGetPrefixTableSize
  - rtEXIEnc, [47](#)
- rtEXIEncStrTabsGetURIID
  - rtEXIEnc, [47](#)
- rtEXIEncStrTabsGetURITableSize
  - rtEXIEnc, [48](#)
- rtEXIEncStrTabsInit
  - rtEXIEnc, [48](#)
- rtEXIEncUIIntCEvent
  - rtEXIEncoder.h, [104](#)
- rtEXIEncUIIntElem
  - rtEXIEncoder.h, [104](#)
- rtEXIEncUIIntValue
  - rtEXIEncoder.h, [104](#)
- rtEXIEncUTF8Str
  - rtEXIEncoder.h, [105](#)
- rtEXIEvent
  - rtEXIEventCodeCompare, [51](#)
  - rtEXIEventCodeCopy, [51](#)
  - rtEXIEventCodeGroupAdd, [51](#)
  - rtEXIEventCodeGroupCopy, [52](#)
  - rtEXIEventCodeGroupFreeMem, [52](#)
  - rtEXIEventCodeGroupGetBitsPart1, [52](#)
  - rtEXIEventCodeGroupGetBitsPart2, [52](#)
  - rtEXIEventCodeGroupGetBitsPart3, [53](#)
  - rtEXIEventCodeGroupIncrPart1, [53](#)
  - rtEXIEventCodeGroupInit, [53](#)
  - rtEXIEventCodeLength, [53](#)
  - rtEXIEventCodePrint, [53](#)
  - rtEXIEventCodesEqual, [54](#)
  - rtEXIEventCodeToString, [54](#)
  - rtEXINewEventCode1, [54](#)
  - rtEXINewEventCode2, [54](#)
  - rtEXINewEventCode3, [55](#)
  - rtEXINewEventCodeGroup, [55](#)
  - rtEXISetEventCode1, [50](#)
  - rtEXISetEventCode2, [51](#)
  - rtEXISetEventCode3, [55](#)
- rtEXIEvent.h, [109](#)
- rtEXIEvent.h
  - OSEXIEventType, [110](#)
  - rtEXIEventDeepCopy, [110](#)
  - rtEXIEventFreeMem, [110](#)
  - rtEXIEventHash, [110](#)
  - rtEXIEventInit, [111](#)
  - rtEXIEventPrint, [111](#)
  - rtEXIEventsEqual, [111](#)
  - rtEXIEventToString, [111](#)
  - rtEXIEventTypeToString, [112](#)
  - rtEXIGetBaseEvent, [112](#)
  - rtEXINewEvent, [112](#)
  - rtEXINewEventDeepCopy, [113](#)
- rtEXIEventCode.h, [114](#)
- rtEXIEventCodeCompare
  - rtEXIEvent, [51](#)
- rtEXIEventCodeCopy
  - rtEXIEvent, [51](#)
- rtEXIEventCodeGroup.h, [116](#)
- rtEXIEventCodeGroupAdd
  - rtEXIEvent, [51](#)

- rtEXIEventCodeGroupCopy
  - rtEXIEvent, [52](#)
- rtEXIEventCodeGroupFreeMem
  - rtEXIEvent, [52](#)
- rtEXIEventCodeGroupGetBitsPart1
  - rtEXIEvent, [52](#)
- rtEXIEventCodeGroupGetBitsPart2
  - rtEXIEvent, [52](#)
- rtEXIEventCodeGroupGetBitsPart3
  - rtEXIEvent, [53](#)
- rtEXIEventCodeGroupIncrPart1
  - rtEXIEvent, [53](#)
- rtEXIEventCodeGroupInit
  - rtEXIEvent, [53](#)
- rtEXIEventCodeLength
  - rtEXIEvent, [53](#)
- rtEXIEventCodePrint
  - rtEXIEvent, [53](#)
- rtEXIEventCodesEqual
  - rtEXIEvent, [54](#)
- rtEXIEventCodeToString
  - rtEXIEvent, [54](#)
- rtEXIEventDeepCopy
  - rtEXIEvent.h, [110](#)
- rtEXIEventFreeMem
  - rtEXIEvent.h, [110](#)
- rtEXIEventHash
  - rtEXIEvent.h, [110](#)
- rtEXIEventInit
  - rtEXIEvent.h, [111](#)
- rtEXIEventPrint
  - rtEXIEvent.h, [111](#)
- rtEXIEventsEqual
  - rtEXIEvent.h, [111](#)
- rtEXIEventToString
  - rtEXIEvent.h, [111](#)
- rtEXIEventTypeToString
  - rtEXIEvent.h, [112](#)
- rtEXIExternDefs.h, [118](#)
- rtEXIGetBaseEvent
  - rtEXIEvent.h, [112](#)
- rtEXIGetDocAutomaton
  - rtEXI, [13](#)
- rtEXIGetElemAutomaton
  - rtEXI, [13](#)
- rtEXIGetMsgLen
  - rtEXI, [8](#)
- rtEXIGetMsgPtr
  - rtEXI, [8](#)
- rtEXIInitContext
  - rtEXI, [13](#)
- rtEXIInitCtxAppInfo
  - rtEXI, [14](#)
- rtEXINewAutomaton
  - rtEXI, [14](#)
- rtEXINewEvent
  - rtEXIEvent.h, [112](#)
- rtEXINewEventCode1
  - rtEXIEvent, [54](#)
- rtEXINewEventCode2
  - rtEXIEvent, [54](#)
- rtEXINewEventCode3
  - rtEXIEvent, [55](#)
- rtEXINewEventCodeGroup
  - rtEXIEvent, [55](#)
- rtEXINewEventDeepCopy
  - rtEXIEvent.h, [113](#)
- rtEXISetBufPtr
  - rtEXI, [8](#)
- rtEXISetEventCode1
  - rtEXIEvent, [50](#)
- rtEXISetEventCode2
  - rtEXIEvent, [51](#)
- rtEXISetEventCode3
  - rtEXIEvent, [55](#)
- rtEXISetOption
  - rtEXI, [9](#)
- rtEXITestOption
  - rtEXI, [9](#)
- rtXML2EXI
  - rtXmlFile2ExiStream, [56](#)
  - rtXmlMem2ExiMem, [56](#)
  - rtXmlMem2ExiStream, [57](#)
  - xml2ExiCharacters, [57](#)
  - xml2ExiEndElement, [57](#)
  - xml2ExiStartElement, [58](#)
- rtXML2EXI.h, [119](#)
- rtXML2EXISAX.h, [120](#)
- rtXmlFile2ExiStream
  - rtXML2EXI, [56](#)
- rtXmlMem2ExiMem
  - rtXML2EXI, [56](#)
- rtXmlMem2ExiStream
  - rtXML2EXI, [57](#)
- StartElementHandler
  - rtEXI2XML, [16](#)
- uriTable
  - OSEXIDecStringTables, [68](#)
  - OSEXIEncStringTables, [71](#)
- XML to EXI serialization functions., [56](#)
- xml2ExiCharacters
  - rtXML2EXI, [57](#)
- xml2ExiEndElement
  - rtXML2EXI, [57](#)
- XML2EXISAXUserData, [78](#)
- xml2ExiStartElement

rtXML2EXI, 58