

## **XBinder**

---

XML Schema Compiler  
Version 2.6  
C JSON 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 ©1997–2018 Objective Systems, Inc. All rights reserved.

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

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

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



# Contents

<b>1</b>	<b>Main Page</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>3</b>
2.1	Modules . . . . .	3
<b>3</b>	<b>Hierarchical Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List . . . . .	7
<b>5</b>	<b>File Index</b>	<b>9</b>
5.1	File List . . . . .	9
<b>6</b>	<b>Module Documentation</b>	<b>11</b>
6.1	JSON encode functions. . . . .	11
6.1.1	Detailed Description . . . . .	13
6.1.2	Function Documentation . . . . .	13
6.1.2.1	rtJsonEncAnyAttr() . . . . .	13
6.1.2.2	rtJsonEncBase64StrValue() . . . . .	13
6.1.2.3	rtJsonEncBetweenObject() . . . . .	14
6.1.2.4	rtJsonEncBitStrValue() . . . . .	14
6.1.2.5	rtJsonEncBitStrValueExt() . . . . .	15
6.1.2.6	rtJsonEncBoolValue() . . . . .	15

6.1.2.7	<code>rtJsonEncDate()</code>	16
6.1.2.8	<code>rtJsonEncDateTime()</code>	16
6.1.2.9	<code>rtJsonEncDecimalValue()</code>	17
6.1.2.10	<code>rtJsonEncDoubleValue()</code>	17
6.1.2.11	<code>rtJsonEncEndObject()</code>	18
6.1.2.12	<code>rtJsonEncFloatValue()</code>	18
6.1.2.13	<code>rtJsonEncGDay()</code>	19
6.1.2.14	<code>rtJsonEncGMonth()</code>	19
6.1.2.15	<code>rtJsonEncGMonthDay()</code>	20
6.1.2.16	<code>rtJsonEncGYear()</code>	20
6.1.2.17	<code>rtJsonEncGYearMonth()</code>	21
6.1.2.18	<code>rtJsonEncHexStr()</code>	21
6.1.2.19	<code>rtJsonEncIndent()</code>	22
6.1.2.20	<code>rtJsonEncInt64Value()</code>	22
6.1.2.21	<code>rtJsonEncIntValue()</code>	23
6.1.2.22	<code>rtJsonEncStartObject()</code>	23
6.1.2.23	<code>rtJsonEncStringNull()</code>	24
6.1.2.24	<code>rtJsonEncStringObject()</code>	24
6.1.2.25	<code>rtJsonEncStringObject2()</code>	24
6.1.2.26	<code>rtJsonEncStringPair()</code>	25
6.1.2.27	<code>rtJsonEncStringPair2()</code>	26
6.1.2.28	<code>rtJsonEncStringRaw()</code>	26
6.1.2.29	<code>rtJsonEncStringValue()</code>	27
6.1.2.30	<code>rtJsonEncStringValue2()</code>	27
6.1.2.31	<code>rtJsonEncTime()</code>	28
6.1.2.32	<code>rtJsonEncUCS4Data()</code>	28
6.1.2.33	<code>rtJsonEncUInt64Value()</code>	29
6.1.2.34	<code>rtJsonEncUIntValue()</code>	29

6.1.2.35	rtJsonEncUnicodeData()	30
6.2	JSON decode functions	31
6.2.1	Detailed Description	33
6.2.2	Function Documentation	33
6.2.2.1	rtJsonDecAnyElem()	33
6.2.2.2	rtJsonDecAnyElem2()	34
6.2.2.3	rtJsonDecAnyType()	34
6.2.2.4	rtJsonDecBase64Str()	35
6.2.2.5	rtJsonDecBase64Str64()	35
6.2.2.6	rtJsonDecBitStrValue()	36
6.2.2.7	rtJsonDecBitStrValue64()	36
6.2.2.8	rtJsonDecBitStrValueExt()	37
6.2.2.9	rtJsonDecBitStrValueExt64()	38
6.2.2.10	rtJsonDecBool()	38
6.2.2.11	rtJsonDecDate()	39
6.2.2.12	rtJsonDecDateTime()	39
6.2.2.13	rtJsonDecDecimal()	40
6.2.2.14	rtJsonDecDouble()	40
6.2.2.15	rtJsonDecDynBase64Str()	41
6.2.2.16	rtJsonDecDynBase64Str64()	41
6.2.2.17	rtJsonDecDynBitStr()	42
6.2.2.18	rtJsonDecDynBitStr64()	42
6.2.2.19	rtJsonDecDynHexStr()	43
6.2.2.20	rtJsonDecDynHexStr64()	44
6.2.2.21	rtJsonDecGDay()	44
6.2.2.22	rtJsonDecGMonth()	45
6.2.2.23	rtJsonDecGMonthDay()	45
6.2.2.24	rtJsonDecGYear()	46

6.2.2.25	<code>rtJsonDecGYearMonth()</code>	46
6.2.2.26	<code>rtJsonDecHexStr()</code>	47
6.2.2.27	<code>rtJsonDecHexStr64()</code>	47
6.2.2.28	<code>rtJsonDecInt16Value()</code>	48
6.2.2.29	<code>rtJsonDecInt32Value()</code>	48
6.2.2.30	<code>rtJsonDecInt64Value()</code>	49
6.2.2.31	<code>rtJsonDecInt8Value()</code>	49
6.2.2.32	<code>rtJsonDecMatchChar()</code>	50
6.2.2.33	<code>rtJsonDecMatchObjectStart()</code>	50
6.2.2.34	<code>rtJsonDecMatchToken()</code>	51
6.2.2.35	<code>rtJsonDecMatchToken2()</code>	52
6.2.2.36	<code>rtJsonDecNameValuePair()</code>	52
6.2.2.37	<code>rtJsonDecNumberString()</code>	53
6.2.2.38	<code>rtJsonDecPeekChar()</code>	53
6.2.2.39	<code>rtJsonDecPeekChar2()</code>	54
6.2.2.40	<code>rtJsonDecStringObject()</code>	54
6.2.2.41	<code>rtJsonDecStringValue()</code>	55
6.2.2.42	<code>rtJsonDecTime()</code>	55
6.2.2.43	<code>rtJsonDecUCS2String()</code>	56
6.2.2.44	<code>rtJsonDecUCS4String()</code>	56
6.2.2.45	<code>rtJsonDecUInt16Value()</code>	57
6.2.2.46	<code>rtJsonDecUInt32Value()</code>	57
6.2.2.47	<code>rtJsonDecUInt64Value()</code>	58
6.2.2.48	<code>rtJsonDecUInt8Value()</code>	58
6.2.2.49	<code>rtJsonDecXmlStringValue()</code>	59
6.2.2.50	<code>rtJsonGetElemIdx()</code>	59



<b>7</b>	<b>Class Documentation</b>	<b>61</b>
7.1	OSJSONDecodeBuffer Class Reference	61
7.1.1	Detailed Description	62
7.1.2	Constructor & Destructor Documentation	62
7.1.2.1	OSJSONDecodeBuffer() [1/3]	62
7.1.2.2	OSJSONDecodeBuffer() [2/3]	62
7.1.2.3	OSJSONDecodeBuffer() [3/3]	63
7.1.3	Member Function Documentation	63
7.1.3.1	init()	63
7.1.3.2	isA()	63
7.1.4	Member Data Documentation	64
7.1.4.1	mbOwnStream	64
7.2	OSJSONEncodeBuffer Class Reference	64
7.2.1	Detailed Description	65
7.2.2	Constructor & Destructor Documentation	65
7.2.2.1	OSJSONEncodeBuffer()	65
7.2.3	Member Function Documentation	66
7.2.3.1	getMsgLen()	66
7.2.3.2	init()	66
7.2.3.3	isA()	66
7.2.3.4	write() [1/2]	67
7.2.3.5	write() [2/2]	67
7.3	OSJSONEncodeStream Class Reference	68
7.3.1	Detailed Description	69
7.3.2	Constructor & Destructor Documentation	69
7.3.2.1	OSJSONEncodeStream() [1/2]	69
7.3.2.2	OSJSONEncodeStream() [2/2]	69
7.3.3	Member Function Documentation	70

7.3.3.1	encodeAttr()	70
7.3.3.2	encodeText()	70
7.3.3.3	getMsgPtr()	71
7.3.3.4	getStream()	71
7.3.3.5	init()	71
7.3.3.6	isA()	71
7.3.4	Member Data Documentation	72
7.3.4.1	mbOwnStream	72
7.3.4.2	mpCtxt	72
7.3.4.3	mpStream	72
7.4	OSJSONMessageBuffer Class Reference	73
7.4.1	Detailed Description	73
7.4.2	Constructor & Destructor Documentation	73
7.4.2.1	OSJSONMessageBuffer()	73
7.4.3	Member Function Documentation	74
7.4.3.1	getIndent()	74
7.4.3.2	getIndentChar()	74
7.4.3.3	setIndent()	74
<b>8</b>	<b>File Documentation</b>	<b>77</b>
8.1	OSJSONDecodeBuffer.h File Reference	77
8.1.1	Detailed Description	77
8.2	OSJSONEncodeBuffer.h File Reference	77
8.2.1	Detailed Description	78
8.3	OSJSONEncodeStream.h File Reference	78
8.3.1	Detailed Description	78
8.4	OSJSONMessageBuffer.h File Reference	78
8.4.1	Detailed Description	78
8.5	osrtjson.h File Reference	79
8.5.1	Detailed Description	83
8.5.2	Macro Definition Documentation	83
8.5.2.1	OSUPCASE	83
8.6	rtJsonCppMsgBuf.h File Reference	83
8.6.1	Detailed Description	83
8.7	rtJsonExternDefs.h File Reference	84
8.7.1	Detailed Description	84
<b>Index</b>		<b>85</b>

## Chapter 1

# Main Page

### C JSON Runtime Library Functions

The **C run-time JSON library** contains functions used to encode/decode data in Javascript object notation (JSON). These functions are identified by their *rtJson* prefixes.



# Chapter 2

## Module Index

### 2.1 Modules

Here is a list of all modules:

JSON encode functions. . . . .	11
JSON decode functions. . . . .	31



# Chapter 3

## Hierarchical Index

### 3.1 Class Hierarchy

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

- OSJSONMessageBuffer . . . . . 73
- OSJSONDecodeBuffer . . . . . 61
- OSJSONEncodeBuffer . . . . . 64
- OSJSONEncodeStream . . . . . 68





# Chapter 4

## Class Index

### 4.1 Class List

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

<a href="#">OSJSONDecodeBuffer</a>	Derived from the <a href="#">OSJSONMessageBuffer</a> base class . . . . .	61
<a href="#">OSJSONEncodeBuffer</a>	Derived from the <a href="#">OSJSONMessageBuffer</a> base class . . . . .	64
<a href="#">OSJSONEncodeStream</a>	Derived from the <a href="#">OSJSONMessageBuffer</a> base class . . . . .	68
<a href="#">OSJSONMessageBuffer</a>	The JSON message buffer class is derived from the <a href="#">OSMessageBuffer</a> base class . . . . .	73



# Chapter 5

## File Index

### 5.1 File List

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

<a href="#">OSJSONDecodeBuffer.h</a>	JSON decode buffer or stream class definition . . . . .	77
<a href="#">OSJSONEncodeBuffer.h</a>	JSON encode message buffer class definition . . . . .	77
<a href="#">OSJSONEncodeStream.h</a>	JSON encode stream class definition . . . . .	78
<a href="#">OSJSONMessageBuffer.h</a>	JSON encode/decode buffer and stream base class . . . . .	78
<a href="#">osrtjson.h</a>	JSON low-level C encode/decode functions . . . . .	79
<a href="#">rtJsonCppMsgBuf.h</a>	This file is deprecated . . . . .	83
<a href="#">rtJsonExternDefs.h</a>	JSON external definitions macro . . . . .	84



## Chapter 6

# Module Documentation

### 6.1 JSON encode functions.

#### Functions

- EXTERNJSON int [rtJsonEncAnyAttr](#) (OSCTXT \*pctxt, const OSRTDList \*pvalue)  
*This function encodes a list of OSAnyAttr attributes in which the name and value are given as a UTF-8 string.*
- EXTERNJSON int [rtJsonEncIntValue](#) (OSCTXT \*pctxt, OSINT32 value)  
*This function encodes a variable of the XSD integer type.*
- EXTERNJSON int [rtJsonEncInt64Value](#) (OSCTXT \*pctxt, OSINT64 value)  
*This function encodes a variable of the XSD integer type.*
- EXTERNJSON int [rtJsonEncBase64StrValue](#) (OSCTXT \*pctxt, OSSIZE nocts, const OSOCTET \*value)  
*This function encodes a variable of the XSD base64Binary type.*
- EXTERNJSON int [rtJsonEncBoolValue](#) (OSCTXT \*pctxt, OSBOOL value)  
*This function encodes a variable of the XSD boolean type.*
- EXTERNJSON int [rtJsonEncGYear](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gYear value into a JSON string representation.*
- EXTERNJSON int [rtJsonEncGYearMonth](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gYearMonth value into a JSON string representation.*
- EXTERNJSON int [rtJsonEncGMonth](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gMonth value into a JSON string representation.*
- EXTERNJSON int [rtJsonEncGMonthDay](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gMonthDay value into a JSON string representation.*
- EXTERNJSON int [rtJsonEncGDay](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gDay value into a JSON string representation.*
- EXTERNJSON int [rtJsonEncDate](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)  
*This function encodes a variable of the XSD 'date' type as a string.*
- EXTERNJSON int [rtJsonEncTime](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)  
*This function encodes a variable of the XSD 'time' type as a JSON string.*
- EXTERNJSON int [rtJsonEncDateTime](#) (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric date/time value into a string representation.*
- EXTERNJSON int [rtJsonEncDecimalValue](#) (OSCTXT \*pctxt, OSREAL value, const OSDecimalFmt \*pFmtSpec)

- This function encodes a value of the XSD decimal type.*

  - EXTERNJSON int [rtJsonEncDoubleValue](#) (OSCTXT \*pctxt, OSREAL value, const OSDoubleFmt \*pFmtSpec)
- This function encodes a value of the XSD double or float type.*

  - EXTERNJSON int [rtJsonEncFloatValue](#) (OSCTXT \*pctxt, OSREAL value, const OSDoubleFmt \*pFmtSpec)
- This function encodes a variable of the XSD float type.*

  - EXTERNJSON int [rtJsonEncHexStr](#) (OSCTXT \*pctxt, OSSIZE noctx, const OSOCTET \*data)
- This function encodes a variable of the XSD hexBinary type.*

  - EXTERNJSON int [rtJsonEncBitStrValue](#) (OSCTXT \*pctxt, OSSIZE nbits, const OSOCTET \*data)
- This function encodes a variable of the ASN.1 Bit string type.*

  - EXTERNJSON int [rtJsonEncBitStrValueExt](#) (OSCTXT \*pctxt, OSSIZE nbits, const OSOCTET \*data, OSSIZE dataSize, const OSOCTET \*extData)
- This function encodes a variable of the ASN.1 Bit string type.*

  - EXTERNJSON int [rtJsonEncIndent](#) (OSCTXT \*pctxt)
- This function adds indentation whitespace to the output stream.*

  - EXTERNJSON int [rtJsonEncStringObject](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*name, const OSUTF8CHAR \*value)
- This function encodes a JSON object containing a string value.*

  - EXTERNJSON int [rtJsonEncStringObject2](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*name, size\_t nameLen, const OSUTF8CHAR \*value, size\_t valueLen)
- This function encodes a JSON object containing a string value.*

  - EXTERNJSON int [rtJsonEncStringPair](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*name, const OSUTF8CHAR \*value)
- This function encodes a name/value pair.*

  - EXTERNJSON int [rtJsonEncStringPair2](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*name, size\_t nameLen, const OSUTF8CHAR \*value, size\_t valueLen)
- This function encodes a name/value pair.*

  - EXTERNJSON int [rtJsonEncStringValue](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*value)
- This function encodes a variable of the XSD string type.*

  - EXTERNJSON int [rtJsonEncStringValue2](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*value, size\_t valueLen)
- This function encodes a variable of the XSD string type.*

  - EXTERNJSON int [rtJsonEncStringNull](#) (OSCTXT \*pctxt)
- This function encodes an asn.1 NULL type as string.*

  - EXTERNJSON int [rtJsonEncStringRaw](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*value)
- This function encodes a raw string without any quotation.*

  - EXTERNJSON int [rtJsonEncUnicodeData](#) (OSCTXT \*pctxt, const OSUNICHAR \*value, OSSIZE nchars)
- This function encodes a variable that contains UCS-2 / UTF-16 characters.*

  - EXTERNJSON int [rtJsonEncUCS4Data](#) (OSCTXT \*pctxt, const OS32BITCHAR \*value, OSSIZE nchars)
- This function encodes a variable that contains UCS-4 / UTF-32 characters.*

  - EXTERNJSON int [rtJsonEncUIntValue](#) (OSCTXT \*pctxt, OSUINT32 value)
- This function encodes a variable of the XSD unsigned integer type.*

  - EXTERNJSON int [rtJsonEncUInt64Value](#) (OSCTXT \*pctxt, OSUINT64 value)
- This function encodes a variable of the XSD integer type.*

  - EXTERNJSON int [rtJsonEncStartObject](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*name, OSBOOL noComma)
- This function encodes the beginning of a JSON object.*

  - EXTERNJSON int [rtJsonEncEndObject](#) (OSCTXT \*pctxt)
- This function encodes the end of a JSON object.*

  - EXTERNJSON int [rtJsonEncBetweenObject](#) (OSCTXT \*pctxt)
- This function encodes the characters separating the JSON name and value.*

## 6.1.1 Detailed Description

## 6.1.2 Function Documentation

### 6.1.2.1 rtJsonEncAnyAttr()

```
EXTERNJSON int rtJsonEncAnyAttr (
    OSCTXT * pctxt,
    const OSRTDList * pvalue )
```

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	List of attributes.

#### Returns

Completion status of operation:

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

### 6.1.2.2 rtJsonEncBase64StrValue()

```
EXTERNJSON int rtJsonEncBase64StrValue (
    OSCTXT * pctxt,
    OSSIZE nocts,
    const OSOCTET * value )
```

This function encodes a variable of the XSD base64Binary type.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>nocts</i>	Number of octets in the value string.
<i>value</i>	Value to be encoded.

## Returns

Completion status of operation:

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

### 6.1.2.3 rtJsonEncBetweenObject()

```
EXTERNJSON int rtJsonEncBetweenObject (
    OSCTXT * pctxt )
```

This function encodes the characters separating the JSON name and value.

#### Parameters

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

## Returns

Completion status of operation:

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

### 6.1.2.4 rtJsonEncBitStrValue()

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

This function encodes a variable of the ASN.1 Bit string type.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>nbits</i>	Number of bits in the value string.
<i>data</i>	Value to be encoded.

## Returns

Completion status of operation:



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

#### 6.1.2.5 rtJsonEncBitStrValueExt()

```
EXTERNJSON int rtJsonEncBitStrValueExt (
    OSCTXT * pctxt,
    OSSIZE nbits,
    const OSOCTET * data,
    OSSIZE dataSize,
    const OSOCTET * extData )
```

This function encodes a variable of the ASN.1 Bit string type.

It handles bit strings with *extdata* member present.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>nbits</i>	Number of bits in the value string.
<i>data</i>	Value to be encoded.
<i>dataSize</i>	Size of data member.
<i>extData</i>	Value of <i>extdata</i> to be encoded.

##### Returns

Completion status of operation:

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

#### 6.1.2.6 rtJsonEncBoolValue()

```
EXTERNJSON int rtJsonEncBoolValue (
    OSCTXT * pctxt,
    OSBOOL value )
```

This function encodes a variable of the XSD boolean type.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Boolean value to be encoded.

## Returns

Completion status of operation:

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

### 6.1.2.7 rtJsonEncDate()

```
EXTERNJSON int rtJsonEncDate (
    OSCTXT * pctxt,
    const OSXSDDateTime * pvalue )
```

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

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	OSXSDDateTime type pointer points to a OSXSDDateTime value to be encoded.

## Returns

Completion status of operation:

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

### 6.1.2.8 rtJsonEncDateTime()

```
EXTERNJSON int rtJsonEncDateTime (
    OSCTXT * pctxt,
    const OSXSDDateTime * pvalue )
```

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to be encoded.

## Returns

Completion status of operation:

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

### 6.1.2.9 rtJsonEncDecimalValue()

```
EXTERNJSON int rtJsonEncDecimalValue (  
    OSCTXT * pctxt,  
    OSREAL value,  
    const OSDecimalFmt * pFmtSpec )
```

This function encodes a value of the XSD decimal type.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.
<i>pFmtSpec</i>	Pointer to format specification structure.

## Returns

Completion status of operation:

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

### 6.1.2.10 rtJsonEncDoubleValue()

```
EXTERNJSON int rtJsonEncDoubleValue (  
    OSCTXT * pctxt,  
    OSREAL value,  
    const OSDoubleFmt * pFmtSpec )
```

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.
<i>pFmtSpec</i>	Pointer to format specification structure.

## Returns

Completion status of operation:

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

### 6.1.2.11 rtJsonEncEndObject()

```
EXTERNJSON int rtJsonEncEndObject (
    OSCTXT * pctxt )
```

This function encodes the end of a JSON object.

#### Parameters

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

## Returns

Completion status of operation:

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

### 6.1.2.12 rtJsonEncFloatValue()

```
EXTERNJSON int rtJsonEncFloatValue (
    OSCTXT * pctxt,
    OSREAL value,
    const OSDoubleFmt * pFmtSpec )
```

This function encodes a variable of the XSD float type.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.
<i>pFmtSpec</i>	Pointer to format specification structure.

## Returns

Completion status of operation:

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

### 6.1.2.13 rtJsonEncGDay()

```
EXTERNJSON int rtJsonEncGDay (
    OSCTXT * pctxt,
    const OSXSDDateTime * pvalue )
```

This function encodes a numeric gDay value into a JSON string representation.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to be encoded.

#### Returns

Completion status of operation:

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

### 6.1.2.14 rtJsonEncGMonth()

```
EXTERNJSON int rtJsonEncGMonth (
    OSCTXT * pctxt,
    const OSXSDDateTime * pvalue )
```

This function encodes a numeric gMonth value into a JSON string representation.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to be encoded.

#### Returns

Completion status of operation:

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

### 6.1.2.15 rtJsonEncGMonthDay()

```
EXTERNJSON int rtJsonEncGMonthDay (
    OSCTXT * pctxt,
    const OSXSDDateTime * pvalue )
```

This function encodes a numeric gMonthDay value into a JSON string representation.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to be encoded.

#### Returns

Completion status of operation:

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

### 6.1.2.16 rtJsonEncGYear()

```
EXTERNJSON int rtJsonEncGYear (
    OSCTXT * pctxt,
    const OSXSDDateTime * pvalue )
```

This function encodes a numeric gYear value into a JSON string representation.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to be encoded.

#### Returns

Completion status of operation:

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

### 6.1.2.17 rtJsonEncGYearMonth()

```
EXTERNJSON int rtJsonEncGYearMonth (
    OSCTXT * pctxt,
    const OSXSDDatetime * pvalue )
```

This function encodes a numeric gYearMonth value into a JSON string representation.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to be encoded.

#### Returns

Completion status of operation:

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

### 6.1.2.18 rtJsonEncHexStr()

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

This function encodes a variable of the XSD hexBinary type.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>nocts</i>	Number of octets in the value string.
<i>data</i>	Value to be encoded.

#### Returns

Completion status of operation:

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

### 6.1.2.19 rtJsonEncIndent()

```
EXTERNJSON int rtJsonEncIndent (
    OSCTXT * pctxt )
```

This function adds indentation whitespace to the output stream.

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

#### Parameters

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

#### Returns

Completion status of operation:

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

### 6.1.2.20 rtJsonEncInt64Value()

```
EXTERNJSON int rtJsonEncInt64Value (
    OSCTXT * pctxt,
    OSINT64 value )
```

This function encodes a variable of the XSD integer type.

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.

#### Returns

Completion status of operation:

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



### 6.1.2.21 rtJsonEncIntValue()

```
EXTERNJSON int rtJsonEncIntValue (
    OSCTXT * pctxt,
    OSINT32 value )
```

This function encodes a variable of the XSD integer type.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.

#### Returns

Completion status of operation:

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

### 6.1.2.22 rtJsonEncStartObject()

```
EXTERNJSON int rtJsonEncStartObject (
    OSCTXT * pctxt,
    const OSUTF8CHAR * name,
    OSBOOL noComma )
```

This function encodes the beginning of a JSON object.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>name</i>	Object name token to be encoded.
<i>noComma</i>	If TRUE do not print comma at end of line in output.

#### Returns

Completion status of operation:

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

### 6.1.2.23 rtJsonEncStringNull()

```
EXTERNJSON int rtJsonEncStringNull (
    OSCTXT * pctxt )
```

This function encodes an asn.1 NULL type as string.

#### Parameters

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

#### Returns

Completion status of operation:

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

### 6.1.2.24 rtJsonEncStringObject()

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

This function encodes a JSON object containing a string value.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>name</i>	Name token to be encoded.
<i>value</i>	Value as a character string to be encoded.

#### Returns

Completion status of operation:

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

### 6.1.2.25 rtJsonEncStringObject2()

```
EXTERNJSON int rtJsonEncStringObject2 (
    OSCTXT * pctxt,
```

```

const OSUTF8CHAR * name,
size_t nameLen,
const OSUTF8CHAR * value,
size_t valueLen )

```

This function encodes a JSON object containing a string value.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>name</i>	Name token to be encoded.
<i>nameLen</i>	Length of the name token to be encoded.
<i>value</i>	Value as a character string to be encoded.
<i>valueLen</i>	Length of the value to be encoded.

#### Returns

Completion status of operation:

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

#### 6.1.2.26 rtJsonEncStringPair()

```

EXTERNJSON int rtJsonEncStringPair (
    OSCTXT * pctxt,
    const OSUTF8CHAR * name,
    const OSUTF8CHAR * value )

```

This function encodes a name/value pair.

The value is a character string.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>name</i>	Name token to be encoded.
<i>value</i>	Value as a character string to be encoded.

#### Returns

Completion status of operation:

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

### 6.1.2.27 rtJsonEncStringPair2()

```
EXTERNJSON int rtJsonEncStringPair2 (
    OSCTXT * pctxt,
    const OSUTF8CHAR * name,
    size_t nameLen,
    const OSUTF8CHAR * value,
    size_t valueLen )
```

This function encodes a name/value pair.

The value is a character string.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>name</i>	Name token to be encoded.
<i>nameLen</i>	Length of the name token to be encoded.
<i>value</i>	Value as a character string to be encoded.
<i>valueLen</i>	Length of the value to be encoded.

#### Returns

Completion status of operation:

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

### 6.1.2.28 rtJsonEncStringRaw()

```
EXTERNJSON int rtJsonEncStringRaw (
    OSCTXT * pctxt,
    const OSUTF8CHAR * value )
```

This function encodes a raw string without any quotation.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	String value to be written.

#### Returns

Completion status of operation:

- 0 = success,

- negative return value is error.

#### 6.1.2.29 rtJsonEncStringValue()

```
EXTERNJSON int rtJsonEncStringValue (
    OSCTXT * pctxt,
    const OSUTF8CHAR * value )
```

This function encodes a variable of the XSD string type.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	XML string value to be encoded.

##### Returns

Completion status of operation:

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

#### 6.1.2.30 rtJsonEncStringValue2()

```
EXTERNJSON int rtJsonEncStringValue2 (
    OSCTXT * pctxt,
    const OSUTF8CHAR * value,
    size_t valueLen )
```

This function encodes a variable of the XSD string type.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	XML string value to be encoded.
<i>valueLen</i>	Length of the XML string to be encoded.

##### Returns

Completion status of operation:

- 0 = success,

- negative return value is error.

### 6.1.2.31 rtJsonEncTime()

```
EXTERNJSON int rtJsonEncTime (
    OSCTXT * pctxt,
    const OSXSDDateTime * pvalue )
```

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	OSXSDDateTime type pointer points to a OSXSDDateTime value to be encoded.

#### Returns

Completion status of operation:

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

### 6.1.2.32 rtJsonEncUCS4Data()

```
EXTERNJSON int rtJsonEncUCS4Data (
    OSCTXT * pctxt,
    const OS32BITCHAR * value,
    OSSIZE nchars )
```

This function encodes a variable that contains UCS-4 / UTF-32 characters.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	UCS-4 characters to be encoded.
<i>nchars</i>	Number of characters to be encoded.

#### Returns

Completion status of operation:

- 0 = success,

- negative return value is error.

### 6.1.2.33 rtJsonEncUInt64Value()

```
EXTERNJSON int rtJsonEncUInt64Value (
    OSCTXT * pctxt,
    OSUINT64 value )
```

This function encodes a variable of the XSD integer type.

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.

#### Returns

Completion status of operation:

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

### 6.1.2.34 rtJsonEncUIntValue()

```
EXTERNJSON int rtJsonEncUIntValue (
    OSCTXT * pctxt,
    OSUINT32 value )
```

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.

#### Returns

Completion status of operation:

- 0 = success,

- negative return value is error.

#### 6.1.2.35 rtJsonEncUnicodeData()

```
EXTERNJSON int rtJsonEncUnicodeData (
    OSCTXT * pctxt,
    const OSUNICHAR * value,
    OSSIZE nchars )
```

This function encodes a variable that contains UCS-2 / UTF-16 characters.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	UCS-2 characters to be encoded.
<i>nchars</i>	Number of Unicode characters to be encoded.

##### Returns

Completion status of operation:

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



## 6.2 JSON decode functions.

### Functions

- EXTERNJSON int [rtJsonDecAnyElem](#) (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)  
*This function decodes an arbitrary block of JSON-encoded data into a string variable.*
- EXTERNJSON int [rtJsonDecAnyElem2](#) (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)  
*This version of [rtJsonDecAnyElem](#) assumes the element name has been pushed on the element name stack in the context.*
- EXTERNJSON int [rtJsonDecAnyType](#) (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)  
*This function decodes an arbitrary block of JSON-encoded data into a string variable.*
- EXTERNJSON int [rtJsonDecBase64Str](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, size\_t bufsize)  
*This function decodes a contents of a Base64-encode binary string into a static memory structure.*
- EXTERNJSON int [rtJsonDecBase64Str64](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocts, size\_t bufsize)  
*This function is identical to [rtJsonDecBase64Str](#) except that it supports lengths up to 64-bits in size on 64-bit machines.*
- EXTERNJSON int [rtJsonDecDynBase64Str](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)  
*This function decodes a contents of a Base64-encode binary string.*
- EXTERNJSON int [rtJsonDecDynBase64Str64](#) (OSCTXT \*pctxt, OSDynOctStr64 \*pvalue)  
*This function is identical to [rtJsonDecDynBase64Str64](#) except that it supports 64-bit integer lengths on 64-bit systems.*
- EXTERNJSON int [rtJsonDecBool](#) (OSCTXT \*pctxt, OSBOOL \*pvalue)  
*This function decodes a variable of the boolean type.*
- EXTERNJSON int [rtJsonDecDate](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'date' type.*
- EXTERNJSON int [rtJsonDecTime](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'time' type.*
- EXTERNJSON int [rtJsonDecDateTime](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'dateTime' type.*
- EXTERNJSON int [rtJsonDecGYear](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gYear' type.*
- EXTERNJSON int [rtJsonDecGYearMonth](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gYearMonth' type.*
- EXTERNJSON int [rtJsonDecGMonth](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gMonth' type.*
- EXTERNJSON int [rtJsonDecGMonthDay](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gMonthDay' type.*
- EXTERNJSON int [rtJsonDecGDay](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)  
*This function decodes a variable of the XSD 'gDay' type.*
- EXTERNJSON int [rtJsonDecDecimal](#) (OSCTXT \*pctxt, OSREAL \*pvalue, int totalDigits, int fractionDigits)  
*This function decodes the contents of a decimal data type.*
- EXTERNJSON int [rtJsonDecDouble](#) (OSCTXT \*pctxt, OSREAL \*pvalue)  
*This function decodes the contents of a float or double data type.*
- EXTERNJSON int [rtJsonDecHexStr](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, size\_t bufsize)  
*This function decodes the contents of a hexBinary string into a static memory structure.*
- EXTERNJSON int [rtJsonDecHexStr64](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocts, size\_t bufsize)  
*This function is identical to [rtJsonDecHexStr](#) except that it supports lengths up to 64-bits in size on 64-bit machines.*
- EXTERNJSON int [rtJsonDecDynHexStr](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)  
*This function decodes a contents of a hexBinary string.*

- EXTERNJSON int [rtJsonDecDynHexStr64](#) (OSCTXT \*pctxt, OSDynOctStr64 \*pvalue)  
*This function is identical to rtJsonDecDynHexStr except that it supports 64-bit integer lengths on 64-bit systems.*
- EXTERNJSON int [rtJsonDecDynBitStr](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*\*data)  
*This function decodes a variable of the ASN.1 Bit string type.*
- EXTERNJSON int [rtJsonDecDynBitStr64](#) (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*\*data)  
*This function is identical to rtJsonDecDynBitStr except that it supports lengths up to 64-bits in size on 64-bit machines.*
- EXTERNJSON int [rtJsonDecBitStrValue](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize)  
*This function decodes a variable of the ASN.1 Bit string type.*
- EXTERNJSON int [rtJsonDecBitStrValue64](#) (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*data, OSSIZE bufsize)  
*This function is identical to rtJsonDecBitStrValue except that it supports lengths up to 64-bits in size on 64-bit machines.*
- EXTERNJSON int [rtJsonDecBitStrValueExt](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize, OSOCTET \*\*extdata)  
*This function decodes a variable of the ASN.1 Bit string type.*
- EXTERNJSON int [rtJsonDecBitStrValueExt64](#) (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*data, OSSIZE bufsize, OSOCTET \*\*extdata)  
*This function is identical to rtJsonDecBitStrValueExt except that it supports lengths up to 64-bits in size on 64-bit machines.*
- EXTERNJSON int [rtJsonDecInt8Value](#) (OSCTXT \*pctxt, OSINT8 \*pvalue)  
*This function decodes the contents of an 8-bit integer data type (i.e.*
- EXTERNJSON int [rtJsonDecInt16Value](#) (OSCTXT \*pctxt, OSINT16 \*pvalue)  
*This function decodes the contents of a 16-bit integer data type.*
- EXTERNJSON int [rtJsonDecInt32Value](#) (OSCTXT \*pctxt, OSINT32 \*pvalue)  
*This function decodes the contents of a 32-bit integer data type.*
- EXTERNJSON int [rtJsonDecInt64Value](#) (OSCTXT \*pctxt, OSINT64 \*pvalue)  
*This function decodes the contents of a 64-bit integer data type.*
- EXTERNJSON int [rtJsonDecUInt8Value](#) (OSCTXT \*pctxt, OSUINT8 \*pvalue)  
*This function decodes the contents of an unsigned 8-bit integer data type (i.e.*
- EXTERNJSON int [rtJsonDecUInt16Value](#) (OSCTXT \*pctxt, OSUINT16 \*pvalue)  
*This function decodes the contents of an unsigned 16-bit integer data type.*
- EXTERNJSON int [rtJsonDecUInt32Value](#) (OSCTXT \*pctxt, OSUINT32 \*pvalue)  
*This function decodes the contents of an unsigned 32-bit integer data type.*
- EXTERNJSON int [rtJsonDecUInt64Value](#) (OSCTXT \*pctxt, OSUINT64 \*pvalue)  
*This function decodes the contents of an unsigned 64-bit integer data type.*
- EXTERNJSON int [rtJsonDecMatchChar](#) (OSCTXT \*pctxt, OSUTF8CHAR ch)  
*This function attempts to match the given character, skipping over any whitespace, if necessary.*
- EXTERNJSON int [rtJsonDecMatchObjectStart](#) (OSCTXT \*pctxt, const OSUTF8NameAndLen \*nameArray, size\_t numNames)  
*This function matches the start of a JSON object.*
- EXTERNJSON int [rtJsonDecMatchToken](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*token)  
*This function decodes a JSON string and matches with a given token.*
- EXTERNJSON int [rtJsonDecMatchToken2](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*token, size\_t tokenLen)  
*This function decodes a JSON string and matches with a given token.*
- EXTERNJSON int [rtJsonDecNameValuePair](#) (OSCTXT \*pctxt, OSUTF8NVP \*pvalue)  
*This function decodes a name/value pair.*
- EXTERNJSON int [rtJsonDecNumberString](#) (OSCTXT \*pctxt, char \*\*ppCharStr)  
*This function decodes a JSON number into a character string variable.*
- EXTERNJSON int [rtJsonDecPeekChar](#) (OSCTXT \*pctxt, OSUTF8CHAR \*pch)  
*This function determines the next non-whitespace character in the input.*

- EXTERNJSON char [rtJsonDecPeekChar2](#) (OSCTXT \*pctxt)  
*This function determines the next non-whitespace character in the input.*
- EXTERNJSON int [rtJsonDecStringObject](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*name, OSUTF8CHAR \*\*ppvalue)  
*This function decodes a JSON object containing a single entry with the given key (name), and returns the key's associated value, which must be a JSON string, via ppvalue.*
- EXTERNJSON int [rtJsonDecStringValue](#) (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)  
*This function decodes the contents of a string data type.*
- EXTERNJSON int [rtJsonDecXmlStringValue](#) (OSCTXT \*pctxt, OSXMLSTRING \*pvalue)  
*This function decodes the contents of an XML string data type.*
- EXTERNJSON int [rtJsonDecUCS2String](#) (OSCTXT \*pctxt, OSUNICHAR \*\*ppstr, OSSIZE \*pnchars)  
*This function is used to decode input UTF-8 data into a UCS-2 / UTF-16 character string.*
- EXTERNJSON int [rtJsonDecUCS4String](#) (OSCTXT \*pctxt, OS32BITCHAR \*\*ppstr, OSSIZE \*pnchars)  
*This function is used to decode input UTF-8 data into a UCS-4 / UTF-32 character string.*
- EXTERNJSON size\_t [rtJsonGetElemIdx](#) (OSCTXT \*pctxt, const OSUTF8NameAndLen nameArray[], size\_t nrows)  
*This function determines which of several possible JSON strings appears next in the input.*

## 6.2.1 Detailed Description

## 6.2.2 Function Documentation

### 6.2.2.1 rtJsonDecAnyElem()

```
EXTERNJSON int rtJsonDecAnyElem (
    OSCTXT * pctxt,
    OSUTF8CHAR ** ppvalue )
```

This function decodes an arbitrary block of JSON-encoded data into a string variable.

In this case, the expected format is element name : JSON encoded data.

#### Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ppvalue</i>	A pointer to a variable to receive the decoded JSON text.

#### Returns

Completion status of operation:

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

### 6.2.2.2 rtJsonDecAnyElem2()

```
EXTERNJSON int rtJsonDecAnyElem2 (
    OSCTXT * pctxt,
    OSUTF8CHAR ** ppvalue )
```

This version of `rtJsonDecAnyElem` assumes the element name has been pushed on the element name stack in the context.

This will be the case if `rtJsonGetElemIdx` is called prior to calling this function.

#### Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ppvalue</i>	A pointer to a variable to receive the decoded JSON text.

#### Returns

Completion status of operation:

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

### 6.2.2.3 rtJsonDecAnyType()

```
EXTERNJSON int rtJsonDecAnyType (
    OSCTXT * pctxt,
    OSUTF8CHAR ** ppvalue )
```

This function decodes an arbitrary block of JSON-encoded data into a string variable.

In this case, the expected format is a complete JSON encoded data fragment.

#### Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>ppvalue</i>	A pointer to a variable to receive the decoded JSON text.

#### Returns

Completion status of operation:

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

#### 6.2.2.4 rtJsonDecBase64Str()

```
EXTERNJSON int rtJsonDecBase64Str (
    OSCTXT * pctxt,
    OSOCTET * pvalue,
    OSUINT32 * pnocts,
    size_t bufsize )
```

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

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

##### Parameters

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

##### Returns

Completion status of operation:

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

#### 6.2.2.5 rtJsonDecBase64Str64()

```
EXTERNJSON int rtJsonDecBase64Str64 (
    OSCTXT * pctxt,
    OSOCTET * pvalue,
    OSSIZE * pnocts,
    size_t bufsize )
```

This function is identical to `rtJsonDecBase64Str` except that it supports lengths up to 64-bits in size on 64-bit machines.

##### Parameters

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

## Returns

Completion status of operation:

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

## See also

[rtJsonDecBase64Str](#)

### 6.2.2.6 rtJsonDecBitStrValue()

```
EXTERNJSON int rtJsonDecBitStrValue (  
    OSCTXT * pctxt,  
    OSUINT32 * nbits,  
    OSOCTET * data,  
    OSSIZE bufsize )
```

This function decodes a variable of the ASN.1 Bit string type.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>nbits</i>	A pointer to an unsigned integer to receive the number of bits in the bit string.
<i>data</i>	A pointer to an OSOCTET array that will receive the decoded bit string. The array must be preallocated.
<i>bufsize</i>	Size of the static buffer in bytes into which the data is to be decoded.

## Returns

Completion status of operation:

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

### 6.2.2.7 rtJsonDecBitStrValue64()

```
EXTERNJSON int rtJsonDecBitStrValue64 (  
    OSCTXT * pctxt,  
    OSSIZE * nbits,  
    OSOCTET * data,  
    OSSIZE bufsize )
```

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

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>nbits</i>	A pointer to an unsigned integer to receive the number of bits in the bit string.
<i>data</i>	A pointer to an OSOCTET array that will receive the decoded bit string. The array must be preallocated.
<i>bufsize</i>	Size of the static buffer in bytes into which the data is to be decoded.

## Returns

Completion status of operation:

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

## See also

[rtJsonDecBitStrValue](#)

### 6.2.2.8 rtJsonDecBitStrValueExt()

```
EXTERNJSON int rtJsonDecBitStrValueExt (
    OSCTXT * pctxt,
    OSUINT32 * nbits,
    OSOCTET * data,
    OSSIZE bufsize,
    OSOCTET ** extdata )
```

This function decodes a variable of the ASN.1 Bit string type.

It handles bit strings with extdata member present.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>nbits</i>	A pointer to an unsigned integer to receive the number of bits in the bit string.
<i>data</i>	A pointer to an OSOCTET array that will receive the decoded bit string. The array must be preallocated.
<i>bufsize</i>	Size of the static buffer in bytes into which the data is to be decoded.
<i>extdata</i>	A pointer to an OSOCTET array that will receive the decoded extdata value.

## Returns

Completion status of operation:

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

### 6.2.2.9 rtJsonDecBitStrValueExt64()

```
EXTERNJSON int rtJsonDecBitStrValueExt64 (
    OSCTXT * pctxt,
    OSSIZE * nbits,
    OSOCTET * data,
    OSSIZE bufsize,
    OSOCTET ** extdata )
```

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>nbits</i>	A pointer to an unsigned integer to receive the number of bits in the bit string.
<i>data</i>	A pointer to an OSOCTET array that will receive the decoded bit string. The array must be preallocated.
<i>bufsize</i>	Size of the static buffer in bytes into which the data is to be decoded.
<i>extdata</i>	A pointer to an OSOCTET array that will receive the decoded extdata value.

#### Returns

Completion status of operation:

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

#### See also

[rtJsonDecBitStrValueExt](#)

### 6.2.2.10 rtJsonDecBool()

```
EXTERNJSON int rtJsonDecBool (
    OSCTXT * pctxt,
    OSBOOL * pvalue )
```

This function decodes a variable of the boolean type.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to a variable to receive the decoded boolean value.



## Returns

Completion status of operation:

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

### 6.2.2.11 rtJsonDecDate()

```
EXTERNJSON int rtJsonDecDate (
    OSCTXT * pctxt,
    OSXSDDateTime * pvalue )
```

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

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

## Returns

Completion status of operation:

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

### 6.2.2.12 rtJsonDecDateTime()

```
EXTERNJSON int rtJsonDecDateTime (
    OSCTXT * pctxt,
    OSXSDDateTime * pvalue )
```

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

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

## Returns

Completion status of operation:

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

### 6.2.2.13 rtJsonDecDecimal()

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

This function decodes the contents of a decimal data type.

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 64-bit double value to receive decoded result.
<i>totalDigits</i>	The total number of digits in the decimal value.
<i>fractionDigits</i>	The number of fractional digits in the decimal value.

## Returns

Completion status of operation:

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

### 6.2.2.14 rtJsonDecDouble()

```
EXTERNJSON int rtJsonDecDouble (
    OSCTXT * pctxt,
    OSREAL * pvalue )
```

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

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

### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 64-bit double value to receive decoded result.

### Returns

Completion status of operation:

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

#### 6.2.2.15 rtJsonDecDynBase64Str()

```
EXTERNJSON int rtJsonDecDynBase64Str (  
    OSCTXT * pctxt,  
    OSDynOctStr * pvalue )
```

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

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

### Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pvalue</i>	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated for the string using the ::rtxMemAlloc function.

### Returns

Completion status of operation:

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

#### 6.2.2.16 rtJsonDecDynBase64Str64()

```
EXTERNJSON int rtJsonDecDynBase64Str64 (  
    OSCTXT * pctxt,  
    OSDynOctStr64 * pvalue )
```

This function is identical to `rtJsonDecDynBase64Str64` except that it supports 64-bit integer lengths on 64-bit systems.

## Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pvalue</i>	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated for the string using the <code>::rtxMemAlloc</code> function.

## Returns

Completion status of operation:

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

### 6.2.2.17 rtJsonDecDynBitStr()

```
EXTERNJSON int rtJsonDecDynBitStr (  
    OSCTXT * pctxt,  
    OSUINT32 * nbits,  
    OSOCTET ** data )
```

This function decodes a variable of the ASN.1 Bit string type.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>nbits</i>	A pointer to an unsigned integer to receive the number of bits in the bit string.
<i>data</i>	A pointer to an OSOCTET array that will receive the decoded bit string; the array will be allocated by the decoding function.

## Returns

Completion status of operation:

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

### 6.2.2.18 rtJsonDecDynBitStr64()

```
EXTERNJSON int rtJsonDecDynBitStr64 (  
    OSCTXT * pctxt,  
    OSSIZE * nbits,  
    OSOCTET ** data )
```

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

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>nbits</i>	A pointer to an unsigned integer to receive the number of bits in the bit string.
<i>data</i>	A pointer to an OSOCTET array that will receive the decoded bit string; the array will be allocated by the decoding function.

## Returns

Completion status of operation:

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

## See also

[rtJsonDecDynBitStr](#)

### 6.2.2.19 rtJsonDecDynHexStr()

```
EXTERNJSON int rtJsonDecDynHexStr (  
    OSCTXT * pctxt,  
    OSDynOctStr * pvalue )
```

This function decodes a contents of a hexBinary string.

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

## Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pvalue</i>	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the ::rtxMemAlloc function.

## Returns

Completion status of operation:

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

### 6.2.2.20 rtJsonDecDynHexStr64()

```
EXTERNJSON int rtJsonDecDynHexStr64 (
    OSCTXT * pctxt,
    OSDynOctStr64 * pvalue )
```

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

#### Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pvalue</i>	A pointer to a dynamic octet string structure to receive the decoded octet string. Dynamic memory is allocated to hold the string using the <code>::rtxMemAlloc</code> function.

#### Returns

Completion status of operation:

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

### 6.2.2.21 rtJsonDecGDay()

```
EXTERNJSON int rtJsonDecGDay (
    OSCTXT * pctxt,
    OSXSDDateTime * pvalue )
```

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

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

#### Returns

Completion status of operation:

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

### 6.2.2.22 rtJsonDecGMonth()

```
EXTERNJSON int rtJsonDecGMonth (  
    OSCTXT * pctxt,  
    OSXSDDateTime * pvalue )
```

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

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

#### Returns

Completion status of operation:

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

### 6.2.2.23 rtJsonDecGMonthDay()

```
EXTERNJSON int rtJsonDecGMonthDay (  
    OSCTXT * pctxt,  
    OSXSDDateTime * pvalue )
```

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

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

#### Returns

Completion status of operation:

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

#### 6.2.2.24 rtJsonDecGYear()

```
EXTERNJSON int rtJsonDecGYear (
    OSCTXT * pctxt,
    OSXSDDateTime * pvalue )
```

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

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

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

##### Returns

Completion status of operation:

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

#### 6.2.2.25 rtJsonDecGYearMonth()

```
EXTERNJSON int rtJsonDecGYearMonth (
    OSCTXT * pctxt,
    OSXSDDateTime * pvalue )
```

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

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

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

##### Returns

Completion status of operation:

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



### 6.2.2.26 rtJsonDecHexStr()

```
EXTERNJSON int rtJsonDecHexStr (
    OSCTXT * pctxt,
    OSOCTET * pvalue,
    OSUINT32 * pnocts,
    size_t bufsize )
```

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

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

#### Parameters

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

#### Returns

Completion status of operation:

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

### 6.2.2.27 rtJsonDecHexStr64()

```
EXTERNJSON int rtJsonDecHexStr64 (
    OSCTXT * pctxt,
    OSOCTET * pvalue,
    OSSIZE * pnocts,
    size_t bufsize )
```

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

#### Parameters

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

## Returns

Completion status of operation:

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

## See also

[rtJsonDecHexStr](#)

### 6.2.2.28 rtJsonDecInt16Value()

```
EXTERNJSON int rtJsonDecInt16Value (  
    OSCTXT * pctxt,  
    OSINT16 * pvalue )
```

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

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 16-bit integer value to receive decoded result.

## Returns

Completion status of operation:

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

### 6.2.2.29 rtJsonDecInt32Value()

```
EXTERNJSON int rtJsonDecInt32Value (  
    OSCTXT * pctxt,  
    OSINT32 * pvalue )
```

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

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

### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 32-bit integer value to receive decoded result.

### Returns

Completion status of operation:

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

#### 6.2.2.30 rtJsonDecInt64Value()

```
EXTERNJSON int rtJsonDecInt64Value (  
    OSCTXT * pctxt,  
    OSINT64 * pvalue )
```

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

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

### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 64-bit integer value to receive decoded result.

### Returns

Completion status of operation:

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

#### 6.2.2.31 rtJsonDecInt8Value()

```
EXTERNJSON int rtJsonDecInt8Value (  
    OSCTXT * pctxt,  
    OSINT8 * pvalue )
```

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

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

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 8-bit integer value to receive decoded result.

## Returns

Completion status of operation:

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

### 6.2.2.32 rtJsonDecMatchChar()

```
EXTERNJSON int rtJsonDecMatchChar (
    OSCTXT * pctxt,
    OSUTF8CHAR ch )
```

This function attempts to match the given character, skipping over any whitespace, if necessary.

If a different character is found, this function returns `RTERR_INVCHAR` and does not consume the non-matching character.

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>ch</i>	The character to be matched.

## Returns

Completion status of operation:

- 0 = success,
- `RTERR_INVCHAR` = different character found
- negative return value is error.

### 6.2.2.33 rtJsonDecMatchObjectStart()

```
EXTERNJSON int rtJsonDecMatchObjectStart (
    OSCTXT * pctxt,
    const OSUTF8NameAndLen * nameArray,
    size_t numNames )
```

This function matches the start of a JSON object.

This will skip leading whitespace, then match the opening '{'. It will then match a key that matches any of the values in `nameArray`, and, if successful, it then matches the subsequent ':' character after the key.

There is no indication of which name was matched, making this function not very useful. See also `rtJsonDecStringObject`.

It is an error if there is not an opening '{', if the key does not match any of the given names, or if the ':' character is not found.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>nameArray</i>	Array of names to be matched.
<i>numNames</i>	Number of names in the name array

#### Returns

Completion status of operation:

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

#### 6.2.2.34 `rtJsonDecMatchToken()`

```
EXTERNJSON int rtJsonDecMatchToken (  
    OSCTXT * pctxt,  
    const OSUTF8CHAR * token )
```

This function decodes a JSON string and matches with a given token.

#### Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>token</i>	The token to be matched.

#### Returns

Completion status of operation:

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

### 6.2.2.35 rtJsonDecMatchToken2()

```
EXTERNJSON int rtJsonDecMatchToken2 (
    OSCTXT * pctxt,
    const OSUTF8CHAR * token,
    size_t tokenLen )
```

This function decodes a JSON string and matches with a given token.

#### Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>token</i>	The token to be matched.
<i>tokenLen</i>	The length of the token to be matched.

#### Returns

Completion status of operation:

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

### 6.2.2.36 rtJsonDecNameValuePair()

```
EXTERNJSON int rtJsonDecNameValuePair (
    OSCTXT * pctxt,
    OSUTF8NVP * pvalue )
```

This function decodes a name/value pair.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to an structure to receive the decoded name and value.

#### Returns

Completion status of operation:

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

### 6.2.2.37 rtJsonDecNumberString()

```
EXTERNJSON int rtJsonDecNumberString (
    OSCTXT * pctxt,
    char ** ppCharStr )
```

This function decodes a JSON number into a character string variable.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>ppCharStr</i>	Pointer to character string pointer to receive decoded value. Dynamic memory is allocated for the string using the <code>rtxMemAlloc</code> function.

#### Returns

Completion status of operation:

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

### 6.2.2.38 rtJsonDecPeekChar()

```
EXTERNJSON int rtJsonDecPeekChar (
    OSCTXT * pctxt,
    OSUTF8CHAR * pch )
```

This function determines the next non-whitespace character in the input.

The non-whitespace character is not consumed.

#### Parameters

<i>pctxt</i>	Pointer to OSCTXT structure
<i>pch</i>	A pointer to a variable to receive the next character.

#### Returns

Completion status of operation:

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

### 6.2.2.39 rtJsonDecPeekChar2()

```
EXTERNJSON char rtJsonDecPeekChar2 (
    OSCTXT * pctx )
```

This function determines the next non-whitespace character in the input.

The non-whitespace character is not consumed.

#### Parameters

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

#### Returns

The peeked character, or null if there is a failure. The error will be logged in the context.

### 6.2.2.40 rtJsonDecStringObject()

```
EXTERNJSON int rtJsonDecStringObject (
    OSCTXT * pctx,
    const OSUTF8CHAR * name,
    OSUTF8CHAR ** pvalue )
```

This function decodes a JSON object containing a single entry with the given key (*name*), and returns the key's associated value, which must be a JSON string, via *pvalue*.

#### Parameters

<i>pctx</i>	Pointer to context block structure.
<i>name</i>	The name token.
<i>pvalue</i>	Pointer to a string structure to receive the decoded string. Memory is allocated for the string using the run-time memory manager.

#### Returns

Completion status of operation:

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



#### 6.2.2.41 rtJsonDecStringValue()

```
EXTERNJSON int rtJsonDecStringValue (
    OSCTXT * pctxt,
    OSUTF8CHAR ** pvalue )
```

This function decodes the contents of a string data type.

This type contains a pointer to a UTF-8 character string. Input is expected to be a string of UTF-8 characters returned by a JSON parser.

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>ppvalue</i>	Pointer to a string structure to receive the decoded string. Memory is allocated for the string using the run-time memory manager.

##### Returns

Completion status of operation:

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

#### 6.2.2.42 rtJsonDecTime()

```
EXTERNJSON int rtJsonDecTime (
    OSCTXT * pctxt,
    OSXSDDateTime * pvalue )
```

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

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

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

##### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	OSXSDDateTime type pointer points to a OSXSDDateTime value to receive decoded result.

## Returns

Completion status of operation:

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

### 6.2.2.43 rtJsonDecUCS2String()

```
EXTERNJSON int rtJsonDecUCS2String (
    OSCTXT * pctxt,
    OSUNICHAR ** ppstr,
    OSSIZE * pnchars )
```

This function is used to decode input UTF-8 data into a UCS-2 / UTF-16 character string.

#### Parameters

<i>pctxt</i>	A pointer to the context block structure.
<i>ppstr</i>	A pointer to a UTF-16 string; memory will be allocated to hold the string using the run-time memory manager.
<i>pnchars</i>	A pointer to an integer to hold the number of characters in the string. (The number of octets may be found by multiplying by two.)

## Returns

0 on success; less than zero on error.

### 6.2.2.44 rtJsonDecUCS4String()

```
EXTERNJSON int rtJsonDecUCS4String (
    OSCTXT * pctxt,
    OS32BITCHAR ** ppstr,
    OSSIZE * pnchars )
```

This function is used to decode input UTF-8 data into a UCS-4 / UTF-32 character string.

#### Parameters

<i>pctxt</i>	A pointer to the context block structure.
<i>ppstr</i>	A pointer to a UTF-32 string; memory will be allocated to hold the string using the run-time memory manager.
<i>pnchars</i>	A pointer to an integer to hold the number of characters in the string. (The number of octets may be found by multiplying by two.)

## Returns

0 on success; less than zero on error.

### 6.2.2.45 rtJsonDecUInt16Value()

```
EXTERNJSON int rtJsonDecUInt16Value (  
    OSCTXT * pctxt,  
    OSUINT16 * pvalue )
```

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

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned 16-bit integer value to receive decoded result.

## Returns

Completion status of operation:

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

### 6.2.2.46 rtJsonDecUInt32Value()

```
EXTERNJSON int rtJsonDecUInt32Value (  
    OSCTXT * pctxt,  
    OSUINT32 * pvalue )
```

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

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned 32-bit integer value to receive decoded result.

## Returns

Completion status of operation:

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

### 6.2.2.47 rtJsonDecUInt64Value()

```
EXTERNJSON int rtJsonDecUInt64Value (  
    OSCTXT * pctxt,  
    OSUINT64 * pvalue )
```

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

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned 64-bit integer value to receive decoded result.

## Returns

Completion status of operation:

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

### 6.2.2.48 rtJsonDecUInt8Value()

```
EXTERNJSON int rtJsonDecUInt8Value (  
    OSCTXT * pctxt,  
    OSUINT8 * pvalue )
```

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

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

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned 8-bit integer value to receive decoded result.

## Returns

Completion status of operation:

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

### 6.2.2.49 rtJsonDecXmlStringValue()

```
EXTERNJSON int rtJsonDecXmlStringValue (
    OSCTXT * pctxt,
    OSXMLSTRING * pvalue )
```

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

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

## Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to an XML string structure to receive the decoded string. Memory is allocated for the string using the run-time memory manager.

## Returns

Completion status of operation:

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

### 6.2.2.50 rtJsonGetElemIdx()

```
EXTERNJSON size_t rtJsonGetElemIdx (
    OSCTXT * pctxt,
    const OSUTF8NameAndLen nameArray[],
    size_t nrows )
```

This function determines which of several possible JSON strings appears next in the input.

This will skip any leading whitespace and then parses a JSON string. It is an error if the input does not have a JSON string. The value of the JSON string is then matched against one of the values in *nameArray* and the corresponding index is returned.

### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>nameArray</i>	Elements descriptor table.
<i>nrows</i>	Number of descriptors in table.

### Returns

Completion status of operation:

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

# Chapter 7

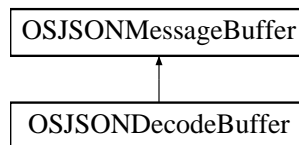
## Class Documentation

### 7.1 OSJSONDecodeBuffer Class Reference

The [OSJSONDecodeBuffer](#) class is derived from the [OSJSONMessageBuffer](#) base class.

```
#include <OSJSONDecodeBuffer.h>
```

Inheritance diagram for OSJSONDecodeBuffer:



#### Public Member Functions

- [OSJSONDecodeBuffer](#) (const char \*jsonFile)  
*This version of the [OSJSONDecodeBuffer](#) constructor takes a name of a file that contains JSON data to be decoded and constructs a buffer.*
- [OSJSONDecodeBuffer](#) (const OSOCTET \*msgbuf, size\_t bufsiz)  
*This version of the [OSJSONDecodeBuffer](#) constructor takes parameters describing a message in memory to be decoded and constructs a buffer.*
- [OSJSONDecodeBuffer](#) (OSRTInputStream &inputStream)  
*This version of the [OSJSONDecodeBuffer](#) constructor takes a reference to the OSInputStream object.*
- virtual EXTJSONMETHOD int [init](#) ()  
*This method initializes the decode message buffer.*
- virtual OSBOOL [isA](#) (Type bufferType)  
*This is a virtual method that must be overridden by derived classes to allow identification of the class.*

## Protected Attributes

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

## Additional Inherited Members

### 7.1.1 Detailed Description

The [OSJSONDecodeBuffer](#) class is derived from the [OSJSONMessageBuffer](#) base class.

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

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

### 7.1.2 Constructor & Destructor Documentation

#### 7.1.2.1 OSJSONDecodeBuffer() [1/3]

```
OSJSONDecodeBuffer::OSJSONDecodeBuffer (  
    const char * jsonFile )
```

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

#### Parameters

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

#### 7.1.2.2 OSJSONDecodeBuffer() [2/3]

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

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



#### Parameters

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

#### 7.1.2.3 OSJSONDecodeBuffer() [3/3]

```
OSJSONDecodeBuffer::OSJSONDecodeBuffer (
    OSRTInputStream & inputStream )
```

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

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

#### Parameters

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

### 7.1.3 Member Function Documentation

#### 7.1.3.1 init()

```
virtual EXTJSONMETHOD int OSJSONDecodeBuffer::init ( ) [virtual]
```

This method initializes the decode message buffer.

#### Returns

Completion status of operation:

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

#### 7.1.3.2 isA()

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

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

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

## Parameters

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

## Returns

Boolean result of the match operation. True if the `bufferType` argument is `JSONDecode`. argument.

Definition at line 108 of file OSJSONDecodeBuffer.h.

## 7.1.4 Member Data Documentation

### 7.1.4.1 mbOwnStream

```
OSBOOL OSJSONDecodeBuffer::mbOwnStream [protected]
```

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

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

Definition at line 52 of file OSJSONDecodeBuffer.h.

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

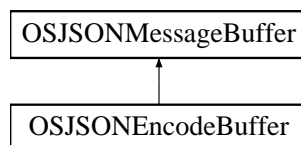
- [OSJSONDecodeBuffer.h](#)

## 7.2 OSJSONEncodeBuffer Class Reference

The [OSJSONEncodeBuffer](#) class is derived from the [OSJSONMessageBuffer](#) base class.

```
#include <OSJSONEncodeBuffer.h>
```

Inheritance diagram for OSJSONEncodeBuffer:



## Public Member Functions

- EXTJSONMETHOD [OSJSONEncodeBuffer](#) ()  
*Default constructor.*
- EXTJSONMETHOD [OSJSONEncodeBuffer](#) (OSOCKET \*pMsgBuf, size\_t msgBufLen)  
*This constructor allows a static message buffer to be specified to receive the encoded message.*
- virtual size\_t [getMsgLen](#) ()  
*This method returns the length of a previously encoded JSON message.*
- virtual EXTJSONMETHOD int [init](#) ()  
*This method reinitializes the encode buffer to allow a new message to be encoded.*
- virtual OSBOOL [isA](#) (Type bufferType)  
*This is a virtual method that must be overridden by derived classes to allow identification of the class.*
- void [nullTerminate](#) ()  
*This method adds a null-terminator character ('\0') at the current buffer position.*
- virtual EXTJSONMETHOD long [write](#) (const char \*filename)  
*This method writes the encoded message to the given file.*
- virtual EXTJSONMETHOD long [write](#) (FILE \*fp)  
*This version of the write method writes to a file that is specified by a FILE pointer.*

## Additional Inherited Members

### 7.2.1 Detailed Description

The [OSJSONEncodeBuffer](#) class is derived from the [OSJSONMessageBuffer](#) base class.

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

Definition at line 38 of file [OSJSONEncodeBuffer.h](#).

### 7.2.2 Constructor & Destructor Documentation

#### 7.2.2.1 OSJSONEncodeBuffer()

```
EXTJSONMETHOD OSJSONEncodeBuffer::OSJSONEncodeBuffer (  
    OSOCKET * pMsgBuf,  
    size_t msgBufLen )
```

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

#### Parameters

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

## 7.2.3 Member Function Documentation

### 7.2.3.1 getMsgLen()

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

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

#### Returns

Length of the JSON message encapsulated within this buffer object.

Definition at line 67 of file OSJSONEncodeBuffer.h.

### 7.2.3.2 init()

```
virtual EXTJSONMETHOD int OSJSONEncodeBuffer::init ( ) [virtual]
```

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

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

### 7.2.3.3 isA()

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

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

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

#### Parameters

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

## Returns

Boolean result of the match operation. True if the `bufferType` argument is `JSONEncode`. argument.

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

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

```
virtual EXTJSONMETHOD long OSJSONEncodeBuffer::write (  
    const char * filename ) [virtual]
```

This method writes the encoded message to the given file.

#### Parameters

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

## Returns

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

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

```
virtual EXTJSONMETHOD long OSJSONEncodeBuffer::write (  
    FILE * fp ) [virtual]
```

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

#### Parameters

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

## Returns

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

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

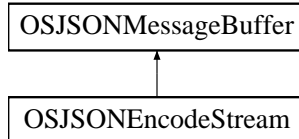
- [OSJSONEncodeBuffer.h](#)

## 7.3 OSJSONEncodeStream Class Reference

The [OSJSONEncodeStream](#) class is derived from the [OSJSONMessageBuffer](#) base class.

```
#include <OSJSONEncodeStream.h>
```

Inheritance diagram for OSJSONEncodeStream:



### Public Member Functions

- EXTJSONMETHOD [OSJSONEncodeStream](#) (OSRTOutputStream &outputStream)  
*This version of the [OSJSONEncodeStream](#) constructor takes a reference to the OSOutputStream object.*
- [OSJSONEncodeStream](#) (OSRTOutputStream \*pOutputStream, OSBOOL ownStream=TRUE)  
*This version of the [OSJSONEncodeStream](#) constructor takes a pointer to the OSRTOutputStream object.*
- EXTJSONMETHOD int [encodeAttr](#) (const OSUTF8CHAR \*name, const OSUTF8CHAR \*value)  
*This function encodes an attribute in which the name and value are given as null-terminated UTF-8 strings.*
- EXTJSONMETHOD int [encodeText](#) (const OSUTF8CHAR \*value)  
*This method encodes JSON textual content.*
- virtual EXTJSONMETHOD int [init](#) ()  
*This method reinitializes the encode stream to allow a new message to be encoded.*
- virtual OSBOOL [isA](#) (Type bufferType)  
*This is a virtual method that must be overridden by derived classes to allow identification of the class.*
- virtual const OSOCTET \* [getMsgPtr](#) ()  
*This is a virtual method that must be overridden by derived classes to allow access to the stored message.*
- OSRTOutputStream \* [getStream](#) () const  
*This method returns the output stream associated with the object.*

### Protected Attributes

- OSRTOutputStream \* [mpStream](#)  
*A pointer to an OSRTOutputStream object.*
- OSBOOL [mbOwnStream](#)  
*TRUE if the [OSJSONEncodeStream](#) object will close and free the stream in the destructor.*
- OSCTXT \* [mpCtxt](#)  
*Internal pointer to the context structure associated with the stream for making C function calls.*

## Additional Inherited Members

### 7.3.1 Detailed Description

The [OSJSONEncodeStream](#) class is derived from the [OSJSONMessageBuffer](#) base class.

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

Definition at line 40 of file `OSJSONEncodeStream.h`.

### 7.3.2 Constructor & Destructor Documentation

#### 7.3.2.1 OSJSONEncodeStream() [1/2]

```
EXTJSONMETHOD OSJSONEncodeStream::OSJSONEncodeStream (  
    OSRTOutputStream & outputStream )
```

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

The stream is assumed to have been previously initialized.

#### Parameters

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

#### 7.3.2.2 OSJSONEncodeStream() [2/2]

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

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

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

#### Parameters

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

### 7.3.3 Member Function Documentation

#### 7.3.3.1 encodeAttr()

```
EXTJSONMETHOD int OSJSONEncodeStream::encodeAttr (
    const OSUTF8CHAR * name,
    const OSUTF8CHAR * value )
```

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

##### Parameters

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

##### Returns

Completion status of operation:

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

#### 7.3.3.2 encodeText()

```
EXTJSONMETHOD int OSJSONEncodeStream::encodeText (
    const OSUTF8CHAR * value )
```

This method encodes JSON textual content.

JSON metadata characters are escaped. The input value is specified in UTF-8 character format.

##### Parameters

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

##### Returns

Completion status of operation:

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



### 7.3.3.3 getMsgPtr()

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

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

The base class implementation returns a null value.

#### Returns

A pointer to the stored message.

Definition at line 135 of file OSJSONEncodeStream.h.

### 7.3.3.4 getStream()

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

This method returns the output stream associated with the object.

#### Returns

A pointer to the output stream.

Definition at line 142 of file OSJSONEncodeStream.h.

### 7.3.3.5 init()

```
virtual EXTJSONMETHOD int OSJSONEncodeStream::init ( ) [virtual]
```

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

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

### 7.3.3.6 isA()

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

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

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

## Parameters

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

## Returns

Boolean result of the match operation. True if the `bufferType` argument is `JSONEncode`. argument.

Definition at line 124 of file OSJSONEncodeStream.h.

## 7.3.4 Member Data Documentation

### 7.3.4.1 mbOwnStream

```
OSBOOL OSJSONEncodeStream::mbOwnStream [protected]
```

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

Definition at line 47 of file OSJSONEncodeStream.h.

### 7.3.4.2 mpCtxt

```
OSCTXT* OSJSONEncodeStream::mpCtxt [protected]
```

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

Definition at line 51 of file OSJSONEncodeStream.h.

### 7.3.4.3 mpStream

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

A pointer to an OSRTOutputStream object.

Definition at line 43 of file OSJSONEncodeStream.h.

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

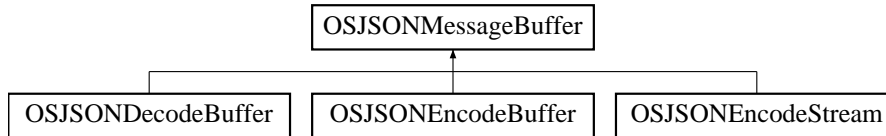
- [OSJSONEncodeStream.h](#)

## 7.4 OSJSONMessageBuffer Class Reference

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

```
#include <OSJSONMessageBuffer.h>
```

Inheritance diagram for OSJSONMessageBuffer:



### Public Member Functions

- EXTJSONMETHOD int [getIndent](#) ()  
*This method returns current JSON output indent value.*
- EXTJSONMETHOD int [getIndentChar](#) ()  
*This method returns current JSON output indent character value (default is space).*
- EXTJSONMETHOD void [setIndent](#) (OSUINT8 indent)  
*This method sets JSON output indent to the given value.*

### Protected Member Functions

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

#### 7.4.1 Detailed Description

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

It is the base class for the [OSJSONEncodeBuffer](#) and [OSJSONDecodeBuffer](#) classes. It contains variables and methods specific to encoding or decoding JSON messages. It is used to manage the buffer into which a message is to be encoded or decoded.

Definition at line 42 of file OSJSONMessageBuffer.h.

#### 7.4.2 Constructor & Destructor Documentation

##### 7.4.2.1 OSJSONMessageBuffer()

```
EXTJSONMETHOD OSJSONMessageBuffer::OSJSONMessageBuffer (  
    Type bufferSize,  
    OSRTContext * pContext = 0 ) [protected]
```

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

## Parameters

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

## 7.4.3 Member Function Documentation

### 7.4.3.1 getIndent()

```
EXTJSONMETHOD int OSJSONMessageBuffer::getIndent ( )
```

This method returns current JSON output indent value.

#### Returns

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

### 7.4.3.2 getIndentChar()

```
EXTJSONMETHOD int OSJSONMessageBuffer::getIndentChar ( )
```

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

#### Returns

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

### 7.4.3.3 setIndent()

```
EXTJSONMETHOD void OSJSONMessageBuffer::setIndent (
    OSUINT8 indent )
```

This method sets JSON output indent to the given value.

#### Parameters

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

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

- [OSJSONMessageBuffer.h](#)



# Chapter 8

## File Documentation

### 8.1 OSJSONDecodeBuffer.h File Reference

JSON decode buffer or stream class definition.

```
#include "rtxsrc/OSRTInputStream.h"  
#include "rtjsonsrc/OSJSONMessageBuffer.h"
```

#### Classes

- class [OSJSONDecodeBuffer](#)

*The [OSJSONDecodeBuffer](#) class is derived from the [OSJSONMessageBuffer](#) base class.*

#### 8.1.1 Detailed Description

JSON decode buffer or stream class definition.

### 8.2 OSJSONEncodeBuffer.h File Reference

JSON encode message buffer class definition.

```
#include "rtjsonsrc/OSJSONMessageBuffer.h"
```

#### Classes

- class [OSJSONEncodeBuffer](#)

*The [OSJSONEncodeBuffer](#) class is derived from the [OSJSONMessageBuffer](#) base class.*

### 8.2.1 Detailed Description

JSON encode message buffer class definition.

## 8.3 OSJSONEncodeStream.h File Reference

JSON encode stream class definition.

```
#include "rtxsrc/OSRTOutputStream.h"  
#include "rtjsonsrc/OSJSONMessageBuffer.h"
```

### Classes

- class [OSJSONEncodeStream](#)

*The [OSJSONEncodeStream](#) class is derived from the [OSJSONMessageBuffer](#) base class.*

### 8.3.1 Detailed Description

JSON encode stream class definition.

## 8.4 OSJSONMessageBuffer.h File Reference

JSON encode/decode buffer and stream base class.

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

### Classes

- class [OSJSONMessageBuffer](#)

*The [JSON message buffer](#) class is derived from the [OSMessageBuffer](#) base class.*

### 8.4.1 Detailed Description

JSON encode/decode buffer and stream base class.



## 8.5 osrtjson.h File Reference

JSON low-level C encode/decode functions.

```
#include "rtxsrc/osMacros.h"
#include "rtxsrc/osSysTypes.h"
#include "rtxsrc/rtxCommon.h"
#include "rtxsrc/rtxError.h"
#include "rtxsrc/rtxBuffer.h"
#include "rtxsrc/rtxMemory.h"
#include "rtjsonsrc/rtJsonExternDefs.h"
```

### Macros

- #define `OSUPCASE` 0x00008000  
*The upper-case flag: if set, hex strings will be encoded in upper case.*

### Functions

- EXTERNJSON int `rtJsonEncAnyAttr` (OSCTXT \*pctx, const OSRTDList \*pvalue)  
*This function encodes a list of OSAnyAttr attributes in which the name and value are given as a UTF-8 string.*
- EXTERNJSON int `rtJsonEncIntValue` (OSCTXT \*pctx, OSINT32 value)  
*This function encodes a variable of the XSD integer type.*
- EXTERNJSON int `rtJsonEncInt64Value` (OSCTXT \*pctx, OSINT64 value)  
*This function encodes a variable of the XSD integer type.*
- EXTERNJSON int `rtJsonEncBase64StrValue` (OSCTXT \*pctx, OSSIZE noctx, const OSOCTET \*value)  
*This function encodes a variable of the XSD base64Binary type.*
- EXTERNJSON int `rtJsonEncBoolValue` (OSCTXT \*pctx, OSBOOL value)  
*This function encodes a variable of the XSD boolean type.*
- EXTERNJSON int `rtJsonEncGYear` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gYear value into a JSON string representation.*
- EXTERNJSON int `rtJsonEncGYearMonth` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gYearMonth value into a JSON string representation.*
- EXTERNJSON int `rtJsonEncGMonth` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gMonth value into a JSON string representation.*
- EXTERNJSON int `rtJsonEncGMonthDay` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gMonthDay value into a JSON string representation.*
- EXTERNJSON int `rtJsonEncGDay` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric gDay value into a JSON string representation.*
- EXTERNJSON int `rtJsonEncDate` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a variable of the XSD 'date' type as a string.*
- EXTERNJSON int `rtJsonEncTime` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a variable of the XSD 'time' type as a JSON string.*
- EXTERNJSON int `rtJsonEncDateTime` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)  
*This function encodes a numeric date/time value into a string representation.*

- EXTERNJSON int [rtJsonEncDecimalValue](#) (OSCTXT \*pctx, OSREAL value, const OSDecimalFmt \*pFmtSpec)  
*This function encodes a value of the XSD decimal type.*
- EXTERNJSON int [rtJsonEncDoubleValue](#) (OSCTXT \*pctx, OSREAL value, const OSDoubleFmt \*pFmtSpec)  
*This function encodes a value of the XSD double or float type.*
- EXTERNJSON int [rtJsonEncFloatValue](#) (OSCTXT \*pctx, OSREAL value, const OSDoubleFmt \*pFmtSpec)  
*This function encodes a variable of the XSD float type.*
- EXTERNJSON int [rtJsonEncHexStr](#) (OSCTXT \*pctx, OSSIZE noctx, const OSOCTET \*data)  
*This function encodes a variable of the XSD hexBinary type.*
- EXTERNJSON int [rtJsonEncBitStrValue](#) (OSCTXT \*pctx, OSSIZE nbits, const OSOCTET \*data)  
*This function encodes a variable of the ASN.1 Bit string type.*
- EXTERNJSON int [rtJsonEncBitStrValueExt](#) (OSCTXT \*pctx, OSSIZE nbits, const OSOCTET \*data, OSSIZE dataSize, const OSOCTET \*extData)  
*This function encodes a variable of the ASN.1 Bit string type.*
- EXTERNJSON int [rtJsonEncIndent](#) (OSCTXT \*pctx)  
*This function adds indentation whitespace to the output stream.*
- EXTERNJSON int [rtJsonEncStringObject](#) (OSCTXT \*pctx, const OSUTF8CHAR \*name, const OSUTF8CHAR \*value)  
*This function encodes a JSON object containing a string value.*
- EXTERNJSON int [rtJsonEncStringObject2](#) (OSCTXT \*pctx, const OSUTF8CHAR \*name, size\_t nameLen, const OSUTF8CHAR \*value, size\_t valueLen)  
*This function encodes a JSON object containing a string value.*
- EXTERNJSON int [rtJsonEncStringPair](#) (OSCTXT \*pctx, const OSUTF8CHAR \*name, const OSUTF8CHAR \*value)  
*This function encodes a name/value pair.*
- EXTERNJSON int [rtJsonEncStringPair2](#) (OSCTXT \*pctx, const OSUTF8CHAR \*name, size\_t nameLen, const OSUTF8CHAR \*value, size\_t valueLen)  
*This function encodes a name/value pair.*
- EXTERNJSON int [rtJsonEncStringValue](#) (OSCTXT \*pctx, const OSUTF8CHAR \*value)  
*This function encodes a variable of the XSD string type.*
- EXTERNJSON int [rtJsonEncStringValue2](#) (OSCTXT \*pctx, const OSUTF8CHAR \*value, size\_t valueLen)  
*This function encodes a variable of the XSD string type.*
- EXTERNJSON int [rtJsonEncStringNull](#) (OSCTXT \*pctx)  
*This function encodes an asn.1 NULL type as string.*
- EXTERNJSON int [rtJsonEncStringRaw](#) (OSCTXT \*pctx, const OSUTF8CHAR \*value)  
*This function encodes a raw string without any quotation.*
- EXTERNJSON int [rtJsonEncUnicodeData](#) (OSCTXT \*pctx, const OSUNICHAR \*value, OSSIZE nchars)  
*This function encodes a variable that contains UCS-2 / UTF-16 characters.*
- EXTERNJSON int [rtJsonEncUCS4Data](#) (OSCTXT \*pctx, const OS32BITCHAR \*value, OSSIZE nchars)  
*This function encodes a variable that contains UCS-4 / UTF-32 characters.*
- EXTERNJSON int [rtJsonEncUIntValue](#) (OSCTXT \*pctx, OSUINT32 value)  
*This function encodes a variable of the XSD unsigned integer type.*
- EXTERNJSON int [rtJsonEncUInt64Value](#) (OSCTXT \*pctx, OSUINT64 value)  
*This function encodes a variable of the XSD integer type.*
- EXTERNJSON int [rtJsonEncStartObject](#) (OSCTXT \*pctx, const OSUTF8CHAR \*name, OSBOOL noComma)  
*This function encodes the beginning of a JSON object.*
- EXTERNJSON int [rtJsonEncEndObject](#) (OSCTXT \*pctx)  
*This function encodes the end of a JSON object.*
- EXTERNJSON int [rtJsonEncBetweenObject](#) (OSCTXT \*pctx)

- This function encodes the characters separating the JSON name and value.*

  - EXTERNJSON int [rtJsonDecAnyElem](#) (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)
- This function decodes an arbitrary block of JSON-encoded data into a string variable.*

  - EXTERNJSON int [rtJsonDecAnyElem2](#) (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)
- This version of [rtJsonDecAnyElem](#) assumes the element name has been pushed on the element name stack in the context.*

  - EXTERNJSON int [rtJsonDecAnyType](#) (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)
- This function decodes an arbitrary block of JSON-encoded data into a string variable.*

  - EXTERNJSON int [rtJsonDecBase64Str](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUIN32 \*pnocets, size\_t bufsize)
- This function decodes a contents of a Base64-encode binary string into a static memory structure.*

  - EXTERNJSON int [rtJsonDecBase64Str64](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocets, size\_t bufsize)
- This function is identical to [rtJsonDecBase64Str](#) except that it supports lengths up to 64-bits in size on 64-bit machines.*

  - EXTERNJSON int [rtJsonDecDynBase64Str](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)
- This function decodes a contents of a Base64-encode binary string.*

  - EXTERNJSON int [rtJsonDecDynBase64Str64](#) (OSCTXT \*pctxt, OSDynOctStr64 \*pvalue)
- This function is identical to [rtJsonDecDynBase64Str64](#) except that it supports 64-bit integer lengths on 64-bit systems.*

  - EXTERNJSON int [rtJsonDecBool](#) (OSCTXT \*pctxt, OSBOOL \*pvalue)
- This function decodes a variable of the boolean type.*

  - EXTERNJSON int [rtJsonDecDate](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- This function decodes a variable of the XSD 'date' type.*

  - EXTERNJSON int [rtJsonDecTime](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- This function decodes a variable of the XSD 'time' type.*

  - EXTERNJSON int [rtJsonDecDateTime](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- This function decodes a variable of the XSD 'dateTime' type.*

  - EXTERNJSON int [rtJsonDecGYear](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- This function decodes a variable of the XSD 'gYear' type.*

  - EXTERNJSON int [rtJsonDecGYearMonth](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- This function decodes a variable of the XSD 'gYearMonth' type.*

  - EXTERNJSON int [rtJsonDecGMonth](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- This function decodes a variable of the XSD 'gMonth' type.*

  - EXTERNJSON int [rtJsonDecGMonthDay](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- This function decodes a variable of the XSD 'gMonthDay' type.*

  - EXTERNJSON int [rtJsonDecGDay](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- This function decodes a variable of the XSD 'gDay' type.*

  - EXTERNJSON int [rtJsonDecDecimal](#) (OSCTXT \*pctxt, OSREAL \*pvalue, int totalDigits, int fractionDigits)
- This function decodes the contents of a decimal data type.*

  - EXTERNJSON int [rtJsonDecDouble](#) (OSCTXT \*pctxt, OSREAL \*pvalue)
- This function decodes the contents of a float or double data type.*

  - EXTERNJSON int [rtJsonDecHexStr](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUIN32 \*pnocets, size\_t bufsize)
- This function decodes the contents of a hexBinary string into a static memory structure.*

  - EXTERNJSON int [rtJsonDecHexStr64](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocets, size\_t bufsize)
- This function is identical to [rtJsonDecHexStr](#) except that it supports lengths up to 64-bits in size on 64-bit machines.*

  - EXTERNJSON int [rtJsonDecDynHexStr](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)
- This function decodes a contents of a hexBinary string.*

  - EXTERNJSON int [rtJsonDecDynHexStr64](#) (OSCTXT \*pctxt, OSDynOctStr64 \*pvalue)
- This function is identical to [rtJsonDecDynHexStr](#) except that it supports 64-bit integer lengths on 64-bit systems.*

  - EXTERNJSON int [rtJsonDecDynBitStr](#) (OSCTXT \*pctxt, OSUIN32 \*nbits, OSOCTET \*\*data)

- This function decodes a variable of the ASN.1 Bit string type.*

  - EXTERNJSON int [rtJsonDecDynBitStr64](#) (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*\*data)

*This function is identical to rtJsonDecDynBitStr except that it supports lengths up to 64-bits in size on 64-bit machines.*
- EXTERNJSON int [rtJsonDecBitStrValue](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize)

*This function decodes a variable of the ASN.1 Bit string type.*
- EXTERNJSON int [rtJsonDecBitStrValue64](#) (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*data, OSSIZE bufsize)

*This function is identical to rtJsonDecBitStrValue except that it supports lengths up to 64-bits in size on 64-bit machines.*
- EXTERNJSON int [rtJsonDecBitStrValueExt](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize, OSOCTET \*\*extdata)

*This function decodes a variable of the ASN.1 Bit string type.*
- EXTERNJSON int [rtJsonDecBitStrValueExt64](#) (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*data, OSSIZE bufsize, OSOCTET \*\*extdata)

*This function is identical to rtJsonDecBitStrValueExt except that it supports lengths up to 64-bits in size on 64-bit machines.*
- EXTERNJSON int [rtJsonDecInt8Value](#) (OSCTXT \*pctxt, OSINT8 \*pvalue)

*This function decodes the contents of an 8-bit integer data type (i.e.*
- EXTERNJSON int [rtJsonDecInt16Value](#) (OSCTXT \*pctxt, OSINT16 \*pvalue)

*This function decodes the contents of a 16-bit integer data type.*
- EXTERNJSON int [rtJsonDecInt32Value](#) (OSCTXT \*pctxt, OSINT32 \*pvalue)

*This function decodes the contents of a 32-bit integer data type.*
- EXTERNJSON int [rtJsonDecInt64Value](#) (OSCTXT \*pctxt, OSINT64 \*pvalue)

*This function decodes the contents of a 64-bit integer data type.*
- EXTERNJSON int [rtJsonDecUInt8Value](#) (OSCTXT \*pctxt, OSUINT8 \*pvalue)

*This function decodes the contents of an unsigned 8-bit integer data type (i.e.*
- EXTERNJSON int [rtJsonDecUInt16Value](#) (OSCTXT \*pctxt, OSUINT16 \*pvalue)

*This function decodes the contents of an unsigned 16-bit integer data type.*
- EXTERNJSON int [rtJsonDecUInt32Value](#) (OSCTXT \*pctxt, OSUINT32 \*pvalue)

*This function decodes the contents of an unsigned 32-bit integer data type.*
- EXTERNJSON int [rtJsonDecUInt64Value](#) (OSCTXT \*pctxt, OSUINT64 \*pvalue)

*This function decodes the contents of an unsigned 64-bit integer data type.*
- EXTERNJSON int [rtJsonDecMatchChar](#) (OSCTXT \*pctxt, OSUTF8CHAR ch)

*This function attempts to match the given character, skipping over any whitespace, if necessary.*
- EXTERNJSON int [rtJsonDecMatchObjectStart](#) (OSCTXT \*pctxt, const OSUTF8NameAndLen \*nameArray, size\_t numNames)

*This function matches the start of a JSON object.*
- EXTERNJSON int [rtJsonDecMatchToken](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*token)

*This function decodes a JSON string and matches with a given token.*
- EXTERNJSON int [rtJsonDecMatchToken2](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*token, size\_t tokenLen)

*This function decodes a JSON string and matches with a given token.*
- EXTERNJSON int [rtJsonDecNameValuePair](#) (OSCTXT \*pctxt, OSUTF8NVP \*pvalue)

*This function decodes a name/value pair.*
- EXTERNJSON int [rtJsonDecNumberString](#) (OSCTXT \*pctxt, char \*\*ppCharStr)

*This function decodes a JSON number into a character string variable.*
- EXTERNJSON int [rtJsonDecPeekChar](#) (OSCTXT \*pctxt, OSUTF8CHAR \*pch)

*This function determines the next non-whitespace character in the input.*
- EXTERNJSON char [rtJsonDecPeekChar2](#) (OSCTXT \*pctxt)

*This function determines the next non-whitespace character in the input.*
- EXTERNJSON int [rtJsonDecStringObject](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*name, OSUTF8CHAR \*\*ppvalue)

*This function decodes a JSON object containing a single entry with the given key (name), and returns the key's associated value, which must be a JSON string, via ppvalue.*

- EXTERNJSON int [rtJsonDecStringValue](#) (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)

*This function decodes the contents of a string data type.*

- EXTERNJSON int [rtJsonDecXmlStringValue](#) (OSCTXT \*pctxt, OSXMLSTRING \*pvalue)

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

- EXTERNJSON int [rtJsonDecUCS2String](#) (OSCTXT \*pctxt, OSUNICHAR \*\*ppstr, OSSIZE \*pnchars)

*This function is used to decode input UTF-8 data into a UCS-2 / UTF-16 character string.*

- EXTERNJSON int [rtJsonDecUCS4String](#) (OSCTXT \*pctxt, OS32BITCHAR \*\*ppstr, OSSIZE \*pnchars)

*This function is used to decode input UTF-8 data into a UCS-4 / UTF-32 character string.*

- EXTERNJSON size\_t [rtJsonGetElemIdx](#) (OSCTXT \*pctxt, const OSUTF8NameAndLen nameArray[], size\_t nrows)

*This function determines which of several possible JSON strings appears next in the input.*

## 8.5.1 Detailed Description

JSON low-level C encode/decode functions.

## 8.5.2 Macro Definition Documentation

### 8.5.2.1 OSUPCASE

```
#define OSUPCASE 0x00008000
```

The upper-case flag: if set, hex strings will be encoded in upper case.

Definition at line 86 of file osrtjson.h.

## 8.6 rtJsonCppMsgBuf.h File Reference

This file is deprecated.

```
#include "rtsrc/asn1CppTypes.h"  
#include "rtjsonsrc/OSJSONEncodeBuffer.h"  
#include "rtjsonsrc/OSJSONEncodeStream.h"  
#include "rtjsonsrc/OSJSONDecodeBuffer.h"
```

### 8.6.1 Detailed Description

This file is deprecated.

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

## 8.7 rtJsonExternDefs.h File Reference

JSON external definitions macro.

### 8.7.1 Detailed Description

JSON external definitions macro.

This is used for Windows to properly declare function scope within DLL's.

# Index

- encodeAttr
  - OSJSONEncodeStream, 70
- encodeText
  - OSJSONEncodeStream, 70
- getIndent
  - OSJSONMessageBuffer, 74
- getIndentChar
  - OSJSONMessageBuffer, 74
- getMsgLen
  - OSJSONEncodeBuffer, 66
- getMsgPtr
  - OSJSONEncodeStream, 70
- getStream
  - OSJSONEncodeStream, 71
- init
  - OSJSONDecodeBuffer, 63
  - OSJSONEncodeBuffer, 66
  - OSJSONEncodeStream, 71
- isA
  - OSJSONDecodeBuffer, 63
  - OSJSONEncodeBuffer, 66
  - OSJSONEncodeStream, 71
- JSON decode functions., 31
  - rtJsonDecAnyElem, 33
  - rtJsonDecAnyElem2, 33
  - rtJsonDecAnyType, 34
  - rtJsonDecBase64Str, 34
  - rtJsonDecBase64Str64, 35
  - rtJsonDecBitStrValue, 36
  - rtJsonDecBitStrValue64, 36
  - rtJsonDecBitStrValueExt, 37
  - rtJsonDecBitStrValueExt64, 38
  - rtJsonDecBool, 38
  - rtJsonDecDate, 39
  - rtJsonDecDateTime, 39
  - rtJsonDecDecimal, 40
  - rtJsonDecDouble, 40
  - rtJsonDecDynBase64Str, 41
  - rtJsonDecDynBase64Str64, 41
  - rtJsonDecDynBitStr, 42
  - rtJsonDecDynBitStr64, 42
  - rtJsonDecDynHexStr, 43
  - rtJsonDecDynHexStr64, 43
  - rtJsonDecGDay, 44
  - rtJsonDecGMonth, 44
  - rtJsonDecGMonthDay, 45
  - rtJsonDecGYear, 45
  - rtJsonDecGYearMonth, 46
  - rtJsonDecHexStr, 46
  - rtJsonDecHexStr64, 47
  - rtJsonDecInt16Value, 48
  - rtJsonDecInt32Value, 48
  - rtJsonDecInt64Value, 49
  - rtJsonDecInt8Value, 49
  - rtJsonDecMatchChar, 50
  - rtJsonDecMatchObjectStart, 50
  - rtJsonDecMatchToken, 51
  - rtJsonDecMatchToken2, 51
  - rtJsonDecNameValuePair, 52
  - rtJsonDecNumberString, 52
  - rtJsonDecPeekChar, 53
  - rtJsonDecPeekChar2, 53
  - rtJsonDecStringObject, 54
  - rtJsonDecStringValue, 54
  - rtJsonDecTime, 55
  - rtJsonDecUCS2String, 56
  - rtJsonDecUCS4String, 56
  - rtJsonDecUInt16Value, 57
  - rtJsonDecUInt32Value, 57
  - rtJsonDecUInt64Value, 58
  - rtJsonDecUInt8Value, 58
  - rtJsonDecXmlStringValue, 59
  - rtJsonGetElemIdx, 59
- JSON encode functions., 11
  - rtJsonEncAnyAttr, 13
  - rtJsonEncBase64StrValue, 13
  - rtJsonEncBetweenObject, 14
  - rtJsonEncBitStrValue, 14
  - rtJsonEncBitStrValueExt, 15
  - rtJsonEncBoolValue, 15
  - rtJsonEncDate, 16
  - rtJsonEncDateTime, 16
  - rtJsonEncDecimalValue, 17
  - rtJsonEncDoubleValue, 17
  - rtJsonEncEndObject, 18
  - rtJsonEncFloatValue, 18
  - rtJsonEncGDay, 19
  - rtJsonEncGMonth, 19



- rtJsonEncGMonthDay, [20](#)
- rtJsonEncGYear, [20](#)
- rtJsonEncGYearMonth, [20](#)
- rtJsonEncHexStr, [21](#)
- rtJsonEncIndent, [21](#)
- rtJsonEncInt64Value, [22](#)
- rtJsonEncIntValue, [22](#)
- rtJsonEncStartObject, [23](#)
- rtJsonEncStringNull, [23](#)
- rtJsonEncStringObject, [24](#)
- rtJsonEncStringObject2, [24](#)
- rtJsonEncStringPair, [25](#)
- rtJsonEncStringPair2, [25](#)
- rtJsonEncStringRaw, [26](#)
- rtJsonEncStringValue, [27](#)
- rtJsonEncStringValue2, [27](#)
- rtJsonEncTime, [28](#)
- rtJsonEncUCS4Data, [28](#)
- rtJsonEncUInt64Value, [29](#)
- rtJsonEncUIntValue, [29](#)
- rtJsonEncUnicodeData, [30](#)

mbOwnStream

- OSJSONDecodeBuffer, [64](#)
- OSJSONEncodeStream, [72](#)

mpCtxt

- OSJSONEncodeStream, [72](#)

mpStream

- OSJSONEncodeStream, [72](#)

OSJSONDecodeBuffer, [61](#)

- init, [63](#)
- isA, [63](#)
- mbOwnStream, [64](#)
- OSJSONDecodeBuffer, [62, 63](#)

OSJSONDecodeBuffer.h, [77](#)

OSJSONEncodeBuffer, [64](#)

- getMsgLen, [66](#)
- init, [66](#)
- isA, [66](#)
- OSJSONEncodeBuffer, [65](#)
- write, [67](#)

OSJSONEncodeBuffer.h, [77](#)

OSJSONEncodeStream, [68](#)

- encodeAttr, [70](#)
- encodeText, [70](#)
- getMsgPtr, [70](#)
- getStream, [71](#)
- init, [71](#)
- isA, [71](#)
- mbOwnStream, [72](#)
- mpCtxt, [72](#)
- mpStream, [72](#)
- OSJSONEncodeStream, [69](#)

OSJSONEncodeStream.h, [78](#)

OSJSONMessageBuffer, [73](#)

- getIndent, [74](#)
- getIndentChar, [74](#)
- OSJSONMessageBuffer, [73](#)
- setIndent, [74](#)

OSJSONMessageBuffer.h, [78](#)

OSUPCASE

- osrtjson.h, [83](#)

osrtjson.h, [79](#)

OSUPCASE, [83](#)

rtJsonCppMsgBuf.h, [83](#)

rtJsonDecAnyElem

- JSON decode functions., [33](#)

rtJsonDecAnyElem2

- JSON decode functions., [33](#)

rtJsonDecAnyType

- JSON decode functions., [34](#)

rtJsonDecBase64Str

- JSON decode functions., [34](#)

rtJsonDecBase64Str64

- JSON decode functions., [35](#)

rtJsonDecBitStrValue

- JSON decode functions., [36](#)

rtJsonDecBitStrValue64

- JSON decode functions., [36](#)

rtJsonDecBitStrValueExt

- JSON decode functions., [37](#)

rtJsonDecBitStrValueExt64

- JSON decode functions., [38](#)

rtJsonDecBool

- JSON decode functions., [38](#)

rtJsonDecDate

- JSON decode functions., [39](#)

rtJsonDecDateTime

- JSON decode functions., [39](#)

rtJsonDecDecimal

- JSON decode functions., [40](#)

rtJsonDecDouble

- JSON decode functions., [40](#)

rtJsonDecDynBase64Str

- JSON decode functions., [41](#)

rtJsonDecDynBase64Str64

- JSON decode functions., [41](#)

rtJsonDecDynBitStr

- JSON decode functions., [42](#)

rtJsonDecDynBitStr64

- JSON decode functions., [42](#)

rtJsonDecDynHexStr

- JSON decode functions., [43](#)

rtJsonDecDynHexStr64

- JSON decode functions., [43](#)

rtJsonDecGDay

- JSON decode functions., [44](#)



rt.JsonDecGMonth	JSON decode functions., 44	rt.JsonDecXmlStringValue	JSON decode functions., 59
rt.JsonDecGMonthDay	JSON decode functions., 45	rt.JsonEncAnyAttr	JSON encode functions., 13
rt.JsonDecGYear	JSON decode functions., 45	rt.JsonEncBase64StrValue	JSON encode functions., 13
rt.JsonDecGYearMonth	JSON decode functions., 46	rt.JsonEncBetweenObject	JSON encode functions., 14
rt.JsonDecHexStr	JSON decode functions., 46	rt.JsonEncBitStrValue	JSON encode functions., 14
rt.JsonDecHexStr64	JSON decode functions., 47	rt.JsonEncBitStrValueExt	JSON encode functions., 15
rt.JsonDecInt16Value	JSON decode functions., 48	rt.JsonEncBoolValue	JSON encode functions., 15
rt.JsonDecInt32Value	JSON decode functions., 48	rt.JsonEncDate	JSON encode functions., 16
rt.JsonDecInt64Value	JSON decode functions., 49	rt.JsonEncDateTime	JSON encode functions., 16
rt.JsonDecInt8Value	JSON decode functions., 49	rt.JsonEncDecimalValue	JSON encode functions., 17
rt.JsonDecMatchChar	JSON decode functions., 50	rt.JsonEncDoubleValue	JSON encode functions., 17
rt.JsonDecMatchObjectStart	JSON decode functions., 50	rt.JsonEncEndObject	JSON encode functions., 18
rt.JsonDecMatchToken	JSON decode functions., 51	rt.JsonEncFloatValue	JSON encode functions., 18
rt.JsonDecMatchToken2	JSON decode functions., 51	rt.JsonEncGDay	JSON encode functions., 19
rt.JsonDecNameValuePair	JSON decode functions., 52	rt.JsonEncGMonth	JSON encode functions., 19
rt.JsonDecNumberString	JSON decode functions., 52	rt.JsonEncGMonthDay	JSON encode functions., 20
rt.JsonDecPeekChar	JSON decode functions., 53	rt.JsonEncGYear	JSON encode functions., 20
rt.JsonDecPeekChar2	JSON decode functions., 53	rt.JsonEncGYearMonth	JSON encode functions., 20
rt.JsonDecStringObject	JSON decode functions., 54	rt.JsonEncHexStr	JSON encode functions., 21
rt.JsonDecStringValue	JSON decode functions., 54	rt.JsonEncIndent	JSON encode functions., 21
rt.JsonDecTime	JSON decode functions., 55	rt.JsonEncInt64Value	JSON encode functions., 22
rt.JsonDecUCS2String	JSON decode functions., 56	rt.JsonEncIntValue	JSON encode functions., 22
rt.JsonDecUCS4String	JSON decode functions., 56	rt.JsonEncStartObject	JSON encode functions., 23
rt.JsonDecUInt16Value	JSON decode functions., 57	rt.JsonEncStringNull	JSON encode functions., 23
rt.JsonDecUInt32Value	JSON decode functions., 57	rt.JsonEncStringObject	JSON encode functions., 24
rt.JsonDecUInt64Value	JSON decode functions., 58	rt.JsonEncStringObject2	JSON encode functions., 24
rt.JsonDecUInt8Value	JSON decode functions., 58	rt.JsonEncStringPair	JSON encode functions., 25

- rt.JsonEncStringPair2
  - JSON encode functions., [25](#)
- rt.JsonEncStringRaw
  - JSON encode functions., [26](#)
- rt.JsonEncStringValue
  - JSON encode functions., [27](#)
- rt.JsonEncStringValue2
  - JSON encode functions., [27](#)
- rt.JsonEncTime
  - JSON encode functions., [28](#)
- rt.JsonEncUCS4Data
  - JSON encode functions., [28](#)
- rt.JsonEncUInt64Value
  - JSON encode functions., [29](#)
- rt.JsonEncUIntValue
  - JSON encode functions., [29](#)
- rt.JsonEncUnicodeData
  - JSON encode functions., [30](#)
- rt.JsonExternDefs.h, [84](#)
- rt.JsonGetElemIdx
  - JSON decode functions., [59](#)
  
- setIndent
  - OSJSONMessageBuffer, [74](#)
  
- write
  - OSJSONEncodeBuffer, [67](#)