

# **XBinder**

---

XML Schema Compiler  
Version 2.7  
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 ©1997–2021 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>C JSON Runtime Library Functions</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	Macro Definition Documentation . . . . .	13
6.1.2.1	rtJsonEncDecrIndent . . . . .	13
6.1.2.2	rtJsonEncIncrIndent . . . . .	14
6.1.2.3	rtJsonEncInt64Value . . . . .	14
6.1.2.4	rtJsonEncIntValue . . . . .	14
6.1.2.5	rtJsonEncResetIndent . . . . .	15
6.1.2.6	rtJsonEncUInt64Value . . . . .	15

6.1.2.7	<code>rtJsonEncUIntValue</code>	16
6.1.2.8	<code>rtJsonGetIndentLevels</code>	16
6.1.3	Function Documentation	17
6.1.3.1	<code>rtJsonEncAnyAttr()</code>	17
6.1.3.2	<code>rtJsonEncBase64StrValue()</code>	17
6.1.3.3	<code>rtJsonEncBetweenObject()</code>	18
6.1.3.4	<code>rtJsonEncBitStrValue()</code>	18
6.1.3.5	<code>rtJsonEncBitStrValueExt()</code>	19
6.1.3.6	<code>rtJsonEncBitStrValueExtV72()</code>	19
6.1.3.7	<code>rtJsonEncBitStrValueV72()</code>	20
6.1.3.8	<code>rtJsonEncBoolValue()</code>	20
6.1.3.9	<code>rtJsonEncChars()</code>	21
6.1.3.10	<code>rtJsonEncCharStr()</code>	21
6.1.3.11	<code>rtJsonEncDate()</code>	22
6.1.3.12	<code>rtJsonEncDateTime()</code>	22
6.1.3.13	<code>rtJsonEncDecimalValue()</code>	23
6.1.3.14	<code>rtJsonEncDoubleValue()</code>	23
6.1.3.15	<code>rtJsonEncEndObject()</code>	24
6.1.3.16	<code>rtJsonEncFixedBitStrValue()</code>	24
6.1.3.17	<code>rtJsonEncFloatValue()</code>	25
6.1.3.18	<code>rtJsonEncGDay()</code>	25
6.1.3.19	<code>rtJsonEncGMonth()</code>	26
6.1.3.20	<code>rtJsonEncGMonthDay()</code>	26
6.1.3.21	<code>rtJsonEncGYear()</code>	27
6.1.3.22	<code>rtJsonEncGYearMonth()</code>	27
6.1.3.23	<code>rtJsonEncHexStr()</code>	28
6.1.3.24	<code>rtJsonEncHexValue()</code>	28
6.1.3.25	<code>rtJsonEncIndent()</code>	29

6.1.3.26	rtJsonEncStartObject()	29
6.1.3.27	rtJsonEncStringNull()	30
6.1.3.28	rtJsonEncStringObject()	30
6.1.3.29	rtJsonEncStringObject2()	31
6.1.3.30	rtJsonEncStringPair()	31
6.1.3.31	rtJsonEncStringPair2()	32
6.1.3.32	rtJsonEncStringRaw()	32
6.1.3.33	rtJsonEncStringValue()	33
6.1.3.34	rtJsonEncStringValue2()	33
6.1.3.35	rtJsonEncTime()	34
6.1.3.36	rtJsonEncUCS4Data()	34
6.1.3.37	rtJsonEncUnicodeData()	35
6.2	JSON decode functions.	36
6.2.1	Detailed Description	39
6.2.2	Macro Definition Documentation	39
6.2.2.1	rtJsonDecPeekChar	39
6.2.2.2	rtJsonDecPeekChar2	39
6.2.3	Function Documentation	40
6.2.3.1	rtJsonDecAnyElem()	40
6.2.3.2	rtJsonDecAnyElem2()	40
6.2.3.3	rtJsonDecAnyType()	41
6.2.3.4	rtJsonDecBase64Str()	41
6.2.3.5	rtJsonDecBase64Str64()	43
6.2.3.6	rtJsonDecBitStrValue()	44
6.2.3.7	rtJsonDecBitStrValue64()	44
6.2.3.8	rtJsonDecBitStrValue64V72()	45
6.2.3.9	rtJsonDecBitStrValueExt()	46
6.2.3.10	rtJsonDecBitStrValueExt64()	46

6.2.3.11	<code>rtJsonDecBitStrValueExt64V72()</code>	47
6.2.3.12	<code>rtJsonDecBitStrValueExtV72()</code>	48
6.2.3.13	<code>rtJsonDecBitStrValueV72()</code>	48
6.2.3.14	<code>rtJsonDecBool()</code>	49
6.2.3.15	<code>rtJsonDecDate()</code>	49
6.2.3.16	<code>rtJsonDecDateTime()</code>	50
6.2.3.17	<code>rtJsonDecDecimal()</code>	50
6.2.3.18	<code>rtJsonDecDouble()</code>	51
6.2.3.19	<code>rtJsonDecDynBase64Str()</code>	51
6.2.3.20	<code>rtJsonDecDynBase64Str64()</code>	52
6.2.3.21	<code>rtJsonDecDynBitStr()</code>	52
6.2.3.22	<code>rtJsonDecDynBitStr64()</code>	53
6.2.3.23	<code>rtJsonDecDynBitStr64V72()</code>	53
6.2.3.24	<code>rtJsonDecDynBitStrV72()</code>	54
6.2.3.25	<code>rtJsonDecDynHexData64()</code>	55
6.2.3.26	<code>rtJsonDecDynHexStr()</code>	55
6.2.3.27	<code>rtJsonDecDynHexStr64()</code>	56
6.2.3.28	<code>rtJsonDecFixedBitStrValue()</code>	56
6.2.3.29	<code>rtJsonDecFixedDynBitStr()</code>	57
6.2.3.30	<code>rtJsonDecGDay()</code>	57
6.2.3.31	<code>rtJsonDecGMonth()</code>	58
6.2.3.32	<code>rtJsonDecGMonthDay()</code>	58
6.2.3.33	<code>rtJsonDecGYear()</code>	59
6.2.3.34	<code>rtJsonDecGYearMonth()</code>	59
6.2.3.35	<code>rtJsonDecHexData64()</code>	60
6.2.3.36	<code>rtJsonDecHexStr()</code>	60
6.2.3.37	<code>rtJsonDecHexStr64()</code>	61
6.2.3.38	<code>rtJsonDecHexToCharStr()</code>	62



6.2.3.39	<code>rtJsonDecMatchChar()</code>	62
6.2.3.40	<code>rtJsonDecMatchCharStr()</code>	63
6.2.3.41	<code>rtJsonDecMatchObjectStart()</code>	63
6.2.3.42	<code>rtJsonDecMatchToken()</code>	64
6.2.3.43	<code>rtJsonDecMatchToken2()</code>	65
6.2.3.44	<code>rtJsonDecNameValuePair()</code>	65
6.2.3.45	<code>rtJsonDecNull()</code>	66
6.2.3.46	<code>rtJsonDecNumberString()</code>	66
6.2.3.47	<code>rtJsonDecStringObject()</code>	66
6.2.3.48	<code>rtJsonDecStringValue()</code>	67
6.2.3.49	<code>rtJsonDecStringValueArray()</code>	67
6.2.3.50	<code>rtJsonDecTime()</code>	68
6.2.3.51	<code>rtJsonDecUCS2String()</code>	69
6.2.3.52	<code>rtJsonDecUCS4String()</code>	69
6.2.3.53	<code>rtJsonDecXmlStringValue()</code>	70
6.2.3.54	<code>rtJsonGetElemIdx()</code>	70
6.2.3.55	<code>rtJsonTryDecBool()</code>	71
6.2.3.56	<code>rtJsonTryDecNull()</code>	71

## **7 Class Documentation 73**

7.1	<code>OSJSONDecodeBuffer</code> Class Reference	73
7.1.1	Detailed Description	74
7.1.2	Constructor & Destructor Documentation	74
7.1.2.1	<code>OSJSONDecodeBuffer()</code> [1/3]	74
7.1.2.2	<code>OSJSONDecodeBuffer()</code> [2/3]	74
7.1.2.3	<code>OSJSONDecodeBuffer()</code> [3/3]	75
7.1.3	Member Function Documentation	75
7.1.3.1	<code>init()</code>	75

7.1.3.2	isA()	75
7.1.4	Member Data Documentation	76
7.1.4.1	mbOwnStream	76
7.2	OSJSONEncodeBuffer Class Reference	76
7.2.1	Detailed Description	77
7.2.2	Constructor & Destructor Documentation	77
7.2.2.1	OSJSONEncodeBuffer()	77
7.2.3	Member Function Documentation	78
7.2.3.1	getMsgLen()	78
7.2.3.2	init()	78
7.2.3.3	isA()	78
7.2.3.4	write() [1/2]	79
7.2.3.5	write() [2/2]	79
7.3	OSJSONEncodeStream Class Reference	80
7.3.1	Detailed Description	81
7.3.2	Constructor & Destructor Documentation	81
7.3.2.1	OSJSONEncodeStream() [1/2]	81
7.3.2.2	OSJSONEncodeStream() [2/2]	81
7.3.3	Member Function Documentation	82
7.3.3.1	encodeAttr()	82
7.3.3.2	encodeText()	82
7.3.3.3	getMsgPtr()	83
7.3.3.4	getStream()	83
7.3.3.5	init()	83
7.3.3.6	isA()	83
7.3.4	Member Data Documentation	84
7.3.4.1	mbOwnStream	84
7.3.4.2	mpCtxt	84
7.3.4.3	mpStream	84
7.4	OSJSONMessageBuffer Class Reference	85
7.4.1	Detailed Description	85
7.4.2	Constructor & Destructor Documentation	85
7.4.2.1	OSJSONMessageBuffer()	85

<b>8 File Documentation</b>	<b>87</b>
8.1 OSJSONDecodeBuffer.h File Reference	87
8.1.1 Detailed Description	87
8.2 OSJSONEncodeBuffer.h File Reference	87
8.2.1 Detailed Description	88
8.3 OSJSONEncodeStream.h File Reference	88
8.3.1 Detailed Description	88
8.4 OSJSONMessageBuffer.h File Reference	88
8.4.1 Detailed Description	88
8.5 osrtjson.h File Reference	89
8.5.1 Detailed Description	94
8.5.2 Macro Definition Documentation	94
8.5.2.1 OSUPCASE	94
8.6 rtJsonCppMsgBuf.h File Reference	94
8.6.1 Detailed Description	95
8.7 rtJsonExternDefs.h File Reference	95
8.7.1 Detailed Description	95
<b>Index</b>	<b>97</b>



## Chapter 1

# 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. . . . .	36





# Chapter 3

## Hierarchical Index

### 3.1 Class Hierarchy

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

- OSJSONMessageBuffer . . . . . 85
- OSJSONDecodeBuffer . . . . . 73
- OSJSONEncodeBuffer . . . . . 76
- OSJSONEncodeStream . . . . . 80



# 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 . . . . .	73
<a href="#">OSJSONEncodeBuffer</a>	Derived from the <a href="#">OSJSONMessageBuffer</a> base class . . . . .	76
<a href="#">OSJSONEncodeStream</a>	Derived from the <a href="#">OSJSONMessageBuffer</a> base class . . . . .	80
<a href="#">OSJSONMessageBuffer</a>	The JSON message buffer class is derived from the <a href="#">OSMessageBuffer</a> base class . . . . .	85



# 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 . . . . .	87
<a href="#">OSJSONEncodeBuffer.h</a>	JSON encode message buffer class definition . . . . .	87
<a href="#">OSJSONEncodeStream.h</a>	JSON encode stream class definition . . . . .	88
<a href="#">OSJSONMessageBuffer.h</a>	JSON encode/decode buffer and stream base class . . . . .	88
<a href="#">osrtjson.h</a>	JSON low-level C encode/decode functions . . . . .	89
<a href="#">rtJsonCppMsgBuf.h</a>	This file is deprecated . . . . .	94
<a href="#">rtJsonExternDefs.h</a>	JSON external definitions macro . . . . .	95



## Chapter 6

# Module Documentation

### 6.1 JSON encode functions.

#### Macros

- #define [rtJsonEncIntValue](#) `rtTxtWriteInt`  
*This function encodes a variable of the XSD integer type.*
- #define [rtJsonEncInt64Value](#) `rtTxtWriteInt64`  
*This function encodes a variable of the XSD integer type.*
- #define [rtJsonEncDecrIndent](#) `rtxIndentDecr`  
*This decreases the indentation level set in the given context by updating the indent member.*
- #define [rtJsonEncIncrIndent](#) `rtxIndentIncr`  
*This increases the indentation level set in the given context by updating the indent member.*
- #define [rtJsonEncResetIndent](#) `rtxIndentReset`  
*This resets the indentation level in the given context to zero.*
- #define [rtJsonGetIndentLevels](#) `rtxGetIndentLevels`  
*This returns the number of levels of indentation `rtJsonEncIndent` is using.*
- #define [rtJsonEncUIntValue](#) `rtTxtWriteUInt`  
*This function encodes a variable of the XSD unsigned integer type.*
- #define [rtJsonEncUInt64Value](#) `rtTxtWriteUInt64`  
*This function encodes a variable of the XSD integer type.*

#### 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 [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 \*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 the given data as a JSON string, using a hexadecimal representation of the data.*

  - EXTERNJSON int [rtJsonEncHexValue](#) (OSCTXT \*pctx, OSSIZE noctx, const OSOCTET \*data)
- This function encodes the given data as a sequence of hexadecimal characters.*

  - EXTERNJSON int [rtJsonEncBitStrValueV72](#) (OSCTXT \*pctx, OSSIZE nbits, const OSOCTET \*data)
- This function encodes a variable of the ASN.1 Bit string type.*

  - EXTERNJSON int [rtJsonEncBitStrValueExtV72](#) (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 [rtJsonEncBitStrValue](#) (OSCTXT \*pctx, OSSIZE nbits, const OSOCTET \*data)
- This function encodes a variable of the ASN.1 Bit string type as a JSON object, as required by X.697 for a BIT STRING without a fixed length.*

  - EXTERNJSON int [rtJsonEncFixedBitStrValue](#) (OSCTXT \*pctx, OSSIZE nbits, const OSOCTET \*data)
- This function encodes a variable of the ASN.1 Bit string type as a JSON string, as required by X.697 for a BIT STRING with a fixed length.*

  - 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 as a JSON object, as required by X.697 for a BIT STRING without a fixed length.*

  - EXTERNJSON int [rtJsonEncIndent](#) (OSCTXT \*pctx)
- This function:*

  - 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 \*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 UTF8 string value.*
- EXTERNJSON int [rtJsonEncStringValue2](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*value, size\_t valueLen)  
*This function encodes a UTF8 string value.*
- EXTERNJSON int [rtJsonEncChars](#) (OSCTXT \*pctxt, const char \*value, OSSIZE valueLen)  
*This function encodes the given number of characters from a character string value as a JSON string.*
- EXTERNJSON int [rtJsonEncCharStr](#) (OSCTXT \*pctxt, const char \*value)  
*This function encodes a character string value as a JSON string.*
- 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 [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 Macro Definition Documentation

### 6.1.2.1 [rtJsonEncDecrIndent](#)

```
#define rtJsonEncDecrIndent rtxIndentDecr
```

This decreases the indentation level set in the given context by updating the indent member.

#### Parameters

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

Definition at line 423 of file osrtjson.h.

#### 6.1.2.2 rtJsonEncIncrIndent

```
#define rtJsonEncIncrIndent rtxIndentIncr
```

This increases the indentation level set in the given context by updating the indent member.

##### Parameters

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

Definition at line 431 of file osrtjson.h.

#### 6.1.2.3 rtJsonEncInt64Value

```
#define rtJsonEncInt64Value rtxTxtWriteInt64
```

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.

Definition at line 130 of file osrtjson.h.

#### 6.1.2.4 rtJsonEncIntValue

```
#define rtJsonEncIntValue rtxTxtWriteInt
```

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.

Definition at line 118 of file osrtjson.h.

### 6.1.2.5 rtJsonEncResetIndent

```
#define rtJsonEncResetIndent rtxIndentReset
```

This resets the indentation level in the given context to zero.

### Parameters

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

Definition at line 438 of file osrtjson.h.

### 6.1.2.6 rtJsonEncUInt64Value

```
#define rtJsonEncUInt64Value rtxTxtWriteUInt64
```

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.

Definition at line 627 of file osrtjson.h.

### 6.1.2.7 rtJsonEncUIntValue

```
#define rtJsonEncUIntValue rtxTxtWriteUInt
```

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

#### Parameters

<i>pctx</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.

Definition at line 614 of file osrtjson.h.

### 6.1.2.8 rtJsonGetIndentLevels

```
#define rtJsonGetIndentLevels rtxGetIndentLevels
```

This returns the number of levels of indentation rtJsonEnclndent is using.

#### Parameters

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

Definition at line 447 of file osrtjson.h.

## 6.1.3 Function Documentation

### 6.1.3.1 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.

### 6.1.3.2 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.

### 6.1.3.3 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.

### 6.1.3.4 rtJsonEncBitStrValue()

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

This function encodes a variable of the ASN.1 Bit string type as a JSON object, as required by X.697 for a BIT STRING without a fixed length.

#### 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.

### 6.1.3.5 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 as a JSON object, as required by X.697 for a BIT STRING without a fixed length.

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.

### 6.1.3.6 rtJsonEncBitStrValueExtV72()

```
EXTERNJSON int rtJsonEncBitStrValueExtV72 (
    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. This provides ObjSys-specific behavior that predates ITU-T X.697.

#### 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.

### 6.1.3.7 rtJsonEncBitStrValueV72()

```
EXTERNJSON int rtJsonEncBitStrValueV72 (  
    OSCTXT * pctxt,  
    OSSIZE nbits,  
    const OSOCTET * data )
```

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

This provides ObjSys-specific behavior that predates ITU-T X.697.

#### 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.

### 6.1.3.8 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.

### 6.1.3.9 rtJsonEncChars()

```
EXTERNJSON int rtJsonEncChars (
    OSCTXT * pctxt,
    const char * value,
    OSSIZE valueLen )
```

This function encodes the given number of characters from a character string value as a JSON string.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Character 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.

### 6.1.3.10 rtJsonEncCharStr()

```
EXTERNJSON int rtJsonEncCharStr (
    OSCTXT * pctxt,
    const char * value )
```

This function encodes a character string value as a JSON string.

#### Parameters

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

## Returns

Completion status of operation:

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

### 6.1.3.11 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.

### 6.1.3.12 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.

### 6.1.3.13 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.

### 6.1.3.14 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.

### 6.1.3.15 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.

### 6.1.3.16 rtJsonEncFixedBitStrValue()

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

This function encodes a variable of the ASN.1 Bit string type as a JSON string, as required by X.697 for a BIT STRING with a fixed length.

#### 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.

### 6.1.3.17 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.

### 6.1.3.18 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.

### 6.1.3.19 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.

### 6.1.3.20 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.

### 6.1.3.21 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.

### 6.1.3.22 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.

### 6.1.3.23 rtJsonEncHexStr()

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

This function encodes the given data as a JSON string, using a hexadecimal representation of the data.

#### 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.

### 6.1.3.24 rtJsonEncHexValue()

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

This function encodes the given data as a sequence of hexadecimal characters.

Unlike `rtJsonHexStr`, it does not encode quotation marks.

#### 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.

### 6.1.3.25 rtJsonEncIndent()

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

This function:

- Writes a comma unless OSJSONNOCOMMA is set or the previous character is a comma, null terminator, {, or [ character.
- Writes a newline and indentation unless OSNOWHITEPSACE is set.

The amount of indentation to add is determined by the indent member variable in the context structure.

#### Parameters

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

#### Returns

Completion status of operation:

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

### 6.1.3.26 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.

### 6.1.3.27 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.

### 6.1.3.28 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.

### 6.1.3.29 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.

### 6.1.3.30 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.

### 6.1.3.31 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.

### 6.1.3.32 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.

#### 6.1.3.33 rtJsonEncStringValue()

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

This function encodes a UTF8 string value.

#### 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.

#### 6.1.3.34 rtJsonEncStringValue2()

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

This function encodes a UTF8 string value.

### 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.

### 6.1.3.35 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.

### 6.1.3.36 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.

### 6.1.3.37 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.

### Macros

- #define `rtJsonDecPeekChar`(pctxt, pch) `rtxTxtPeekChar`(pctxt, pch, TRUE)  
*This function determines the next non-whitespace character in the input.*
- #define `rtJsonDecPeekChar2`(pctxt) `rtxTxtPeekChar2`(pctxt, TRUE)  
*This function determines the next non-whitespace character in the input.*

### 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 `rtJsonTryDecBool` (OSCTXT \*pctxt, OSBOOL \*pvalue)  
*This function will attempt to decode a boolean variable at the current buffer or stream position.*
- 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 [rtJsonDecHexToCharStr](#) (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)  
*This function decodes a JSON string, interpreting the content as the hexadecimal representation of the bytes for a character string, which it returns.*
- EXTERNJSON int [rtJsonDecHexStr](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocets, size\_t bufsize)  
*This function decodes a JSON string consisting of hexadecimal characters 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 [rtJsonDecHexData64](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocets, size\_t bufsize)  
*This function decodes a sequence of hexadecimal characters until a non-hexadecimal character (which is not consumed) is found OR the preallocated array is filled.*
- EXTERNJSON int [rtJsonDecDynHexStr](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)  
*This function decodes a JSON string consisting of hexadecimal characters.*
- 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 [rtJsonDecDynHexData64](#) (OSCTXT \*pctxt, OSDynOctStr64 \*pvalue)  
*This function decodes a sequence of hexadecimal characters until a non-hexadecimal character (which is not consumed) is found.*
- EXTERNJSON int [rtJsonDecDynBitStrV72](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*\*data)  
*This function decodes a variable of the ASN.1 Bit string type.*
- EXTERNJSON int [rtJsonDecDynBitStr64V72](#) (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*\*data)  
*This function is identical to [rtJsonDecDynBitStrV72](#) except that it supports lengths up to 64-bits in size on 64-bit machines.*
- EXTERNJSON int [rtJsonDecBitStrValueV72](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize)  
*This function decodes a variable of the ASN.1 Bit string type.*
- EXTERNJSON int [rtJsonDecBitStrValue64V72](#) (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*data, OSSIZE bufsize)  
*This function is identical to [rtJsonDecBitStrValueV72](#) except that it supports lengths up to 64-bits in size on 64-bit machines.*
- EXTERNJSON int [rtJsonDecBitStrValueExtV72](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize, OSOCTET \*\*extdata)  
*This function decodes a variable of the ASN.1 Bit string type.*
- EXTERNJSON int [rtJsonDecBitStrValueExt64V72](#) (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*data, OSSIZE bufsize, OSOCTET \*\*extdata)  
*This function is identical to [rtJsonDecBitStrValueExtV72](#) except that it supports lengths up to 64-bits in size on 64-bit machines.*
- EXTERNJSON int [rtJsonDecDynBitStr](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*\*data)  
*This function decodes a value of an ASN.1 BIT STRING type, not constrained to a fixed length, encoded according to X.697 as a JSON object.*
- 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 [rtJsonDecFixedDynBitStr](#) (OSCTXT \*pctxt, OSSIZE nbits, OSOCTET \*\*data)  
*This function decodes a value of an ASN.1 BIT STRING type constrained to a fixed length, encoded according to X.697 as a JSON string.*
- EXTERNJSON int [rtJsonDecBitStrValue](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize)

- This function decodes a value of an ASN.1 BIT STRING type that is not constrained to a fixed length, encoded according to X.697 as a JSON object.*
- 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 [rtJsonDecFixedBitStrValue](#) (OSCTXT \*pctxt, OSSIZE nbits, OSOCTET \*data, OSSIZE bufsize)
 

*This function decodes a value of an ASN.1 BIT STRING type constrained to have the given fixed length, encoded according to X.697 as a JSON string.*
  - EXTERNJSON int [rtJsonDecBitStrValueExt](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize, OSOCTET \*\*extdata)
 

*This function decodes a value of a ASN.1 BIT STRING type without a fixed-length constraint, encoded according to X.697 as JSON object.*
  - 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 [rtJsonDecMatchChar](#) (OSCTXT \*pctxt, OSUTF8CHAR ch)
 

*This function attempts to match the given character, skipping over any whitesapce, if necessary.*
  - EXTERNJSON int [rtJsonDecMatchCharStr](#) (OSCTXT \*pctxt, const char \*token)
 

*This function decodes a JSON string and matches with a given token.*
  - 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 [rtJsonDecNull](#) (OSCTXT \*pctxt)
 

*This function decodes a JSON null.*
  - EXTERNJSON int [rtJsonTryDecNull](#) (OSCTXT \*pctxt)
 

*This function will attempt to decode a null value at the current buffer or stream position.*
  - EXTERNJSON int [rtJsonDecNumberString](#) (OSCTXT \*pctxt, char \*\*ppCharStr)
 

*This function decodes a JSON number into a character string variable.*
  - 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 [rtJsonDecStringValueArray](#) (OSCTXT \*pctxt, OSUTF8CHAR \*pvalue, OSSIZE bufsize)
 

*This function is the same as rtJsonDecStringValue except that it decodes into a character array.*
  - 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 Macro Definition Documentation

### 6.2.2.1 rtJsonDecPeekChar

```
#define rtJsonDecPeekChar(  
    pctxt,  
    pch ) rtTxtPeekChar(pctxt, pch, TRUE)
```

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.

Definition at line 1589 of file osrtjson.h.

### 6.2.2.2 rtJsonDecPeekChar2

```
#define rtJsonDecPeekChar2(  
    pctxt ) rtTxtPeekChar2(pctxt, TRUE)
```

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
--------------	-----------------------------

## Returns

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

Definition at line 1599 of file osrtjson.h.

## 6.2.3 Function Documentation

### 6.2.3.1 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.

### 6.2.3.2 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.

### 6.2.3.3 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. You may pass null if not interested in receiving the decoded text.

## Returns

Completion status of operation:

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

### 6.2.3.4 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.

### 6.2.3.5 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](#)

### 6.2.3.6 rtJsonDecBitStrValue()

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

This function decodes a value of an ASN.1 BIT STRING type that is not constrained to a fixed length, encoded according to X.697 as a JSON object.

#### 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.

### 6.2.3.7 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](#)

### 6.2.3.8 rtJsonDecBitStrValue64V72()

```
EXTERNJSON int rtJsonDecBitStrValue64V72 (  
    OSCTXT * pctxt,  
    OSSIZE * nbits,  
    OSOCTET * data,  
    OSSIZE bufsize )
```

This function is identical to `rtJsonDecBitStrValueV72` except that it supports lengths up to 64-bits in size on 64-bit machines.

This provides ObjSys-specific behavior that predates ITU-T X.697.

## 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

[rtJsonDecBitStrValueV72](#)

### 6.2.3.9 rtJsonDecBitStrValueExt()

```
EXTERNJSON int rtJsonDecBitStrValueExt (
    OSCTXT * pctxt,
    OSUIN32 * nbits,
    OSOCTET * data,
    OSSIZE bufsize,
    OSOCTET ** extdata )
```

This function decodes a value of a ASN.1 BIT STRING type without a fixed-length constraint, encoded according to X.697 as JSON object.

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 <i>extdata</i> value.

#### Returns

Completion status of operation:

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

### 6.2.3.10 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 <i>extdata</i> value.

## Returns

Completion status of operation:

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

## See also

[rtJsonDecBitStrValueExt](#)

### 6.2.3.11 `rtJsonDecBitStrValueExt64V72()`

```
EXTERNJSON int rtJsonDecBitStrValueExt64V72 (  
    OSCTXT * pctxt,  
    OSSIZE * nbits,  
    OSOCTET * data,  
    OSSIZE bufsize,  
    OSOCTET ** extdata )
```

This function is identical to `rtJsonDecBitStrValueExtV72` except that it supports lengths up to 64-bits in size on 64-bit machines.

This provides ObjSys-specific behavior that predates ITU-T X.697.

## 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

[rtJsonDecBitStrValueExtV72](#)

### 6.2.3.12 rtJsonDecBitStrValueExtV72()

```
EXTERNJSON int rtJsonDecBitStrValueExtV72 (
    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. This provides ObjSys-specific behavior that predates ITU-T X.697.

#### 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 <i>extdata</i> value.

#### Returns

Completion status of operation:

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

### 6.2.3.13 rtJsonDecBitStrValueV72()

```
EXTERNJSON int rtJsonDecBitStrValueV72 (
    OSCTXT * pctxt,
    OSUINT32 * nbits,
    OSOCTET * data,
    OSSIZE bufsize )
```

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

This provides ObjSys-specific behavior that predates ITU-T X.697.

#### 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.

### 6.2.3.14 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.

### 6.2.3.15 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.

### 6.2.3.16 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 a 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.

### 6.2.3.17 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.

### 6.2.3.18 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.

### 6.2.3.19 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.

### 6.2.3.20 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.

### 6.2.3.21 rtJsonDecDynBitStr()

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

This function decodes a value of an ASN.1 BIT STRING type, not constrained to a fixed length, encoded according to X.697 as a JSON object.

## 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.

### 6.2.3.22 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](#)

### 6.2.3.23 rtJsonDecDynBitStr64V72()

```
EXTERNJSON int rtJsonDecDynBitStr64V72 (  
    OSCTXT * pctxt,  
    OSSIZE * nbits,  
    OSOCTET ** data )
```

This function is identical to `rtJsonDecDynBitStrV72` except that it supports lengths up to 64-bits in size on 64-bit machines.

This provides ObjSys-specific behavior that predates ITU-T X.697.

## 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

[rtJsonDecDynBitStrV72](#)

### 6.2.3.24 rtJsonDecDynBitStrV72()

```
EXTERNJSON int rtJsonDecDynBitStrV72 (  
    OSCTXT * pctxt,  
    OSUINT32 * nbits,  
    OSOCTET ** data )
```

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

This provides ObjSys-specific behavior that predates ITU-T X.697.

## 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.

### 6.2.3.25 rtJsonDecDynHexData64()

```
EXTERNJSON int rtJsonDecDynHexData64 (
    OSCTXT * pctxt,
    OSDynOctStr64 * pvalue )
```

This function decodes a sequence of hexadecimal characters until a non-hexadecimal character (which is not consumed) is found.

#### 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.

### 6.2.3.26 rtJsonDecDynHexStr()

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

This function decodes a JSON string consisting of hexadecimal characters.

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.

### 6.2.3.27 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.

### 6.2.3.28 rtJsonDecFixedBitStrValue()

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

This function decodes a value of an ASN.1 BIT STRING type constrained to have the given fixed length, encoded according to X.697 as a JSON string.

This ensures the encoded data is the given number of bits and that unused bits are zeros.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>nbits</i>	The number of bits the BIT STRING is fixed to.
<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.

### 6.2.3.29 rtJsonDecFixedDynBitStr()

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

This function decodes a value of an ASN.1 BIT STRING type constrained to a fixed length, encoded according to X.697 as a JSON string.

This ensures the encoded data is the given number of bits and that unused bits are zeros.

## 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.

### 6.2.3.30 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.

### 6.2.3.31 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.

### 6.2.3.32 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.

### 6.2.3.33 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.

### 6.2.3.34 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.

### 6.2.3.35 rtJsonDecHexData64()

```
EXTERNJSON int rtJsonDecHexData64 (  
    OSCTXT * pctxt,  
    OSOCTET * pvalue,  
    OSSIZE * pnocts,  
    size_t bufsize )
```

This function decodes a sequence of hexadecimal characters until a non-hexadecimal character (which is not consumed) is found OR the preallocated array is filled.

### 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 preallocated 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 pvalue.

### Returns

Completion status of operation:

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

### 6.2.3.36 rtJsonDecHexStr()

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



```

OSOCKET * pvalue,
OSUINT32 * pnocts,
size_t bufsize )

```

This function decodes a JSON string consisting of hexadecimal characters 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.

#### 6.2.3.37 rtJsonDecHexStr64()

```

EXTERNJSON int rtJsonDecHexStr64 (
    OSCTXT * pctxt,
    OSOCKET * 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](#)

### 6.2.3.38 rtJsonDecHexToCharStr()

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

This function decodes a JSON string, interpreting the content as the hexadecimal representation of the bytes for a character string, which it returns.

This function is used to support the JSON encoding specified in X.697 for the following ASN.1 types: TeletexString, T61String, VideotexString, GraphicString, and GeneralString.

## 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. If null, the JSON string is decoded but not retained.

## Returns

Completion status of operation:

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

### 6.2.3.39 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.

#### 6.2.3.40 rtJsonDecMatchCharStr()

```
EXTERNJSON int rtJsonDecMatchCharStr (  
    OSCTXT * pctxt,  
    const char * token )
```

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

This is equivalent to rtJsonDecMatchToken.

### 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.

#### 6.2.3.41 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.

#### 6.2.3.42 `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.

### 6.2.3.43 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.

### 6.2.3.44 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.

#### 6.2.3.45 rtJsonDecNull()

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

This function decodes a JSON null.

##### Parameters

<i>pctxt</i>	A pointer to a context data structure.
--------------	--

##### Returns

0 on success, less than zero otherwise.

#### 6.2.3.46 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.

#### 6.2.3.47 rtJsonDecStringObject()

```
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*.

#### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>name</i>	The name token.
<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.

#### 6.2.3.48 rtJsonDecStringValue()

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

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.

#### 6.2.3.49 rtJsonDecStringValueArray()

```
EXTERNJSON int rtJsonDecStringValueArray (
    OSCTXT * pctxt,
    OSUTF8CHAR * pvalue,
    OSSIZE bufsize )
```

This function is the same as `rtJsonDecStringValue` except that it decodes into a character array.

### Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to character array to decode into.
<i>bufize</i>	Size of the given array.

### Returns

Completion status of operation:

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

### 6.2.3.50 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.



### 6.2.3.51 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.

### 6.2.3.52 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.

### 6.2.3.53 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.

### 6.2.3.54 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.

### 6.2.3.55 rtJsonTryDecBool()

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

This function will attempt to decode a boolean variable at the current buffer or stream position.

If the attempt fails, the decode cursor is reset to the position it was at when the function was called and an RTERR\_IDNOTFOU status is returned. The error is not logged in the context.

#### 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,
- RTERR\_IDNOTFOU = not boolean value (not logged)
- negative return value if other error (logged)

### 6.2.3.56 rtJsonTryDecNull()

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

This function will attempt to decode a null value at the current buffer or stream position.

If the attempt fails, the decode cursor is reset to the position it was at when the function was called and an RTERR\_IDNOTFOU status is returned. The error is not logged in the context.

#### Parameters

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

## Returns

Completion status of operation:

- 0 = success,
- RTERR\_IDNOTFOU = not boolean value (not logged)
- negative return value if other error (logged)

# Chapter 7

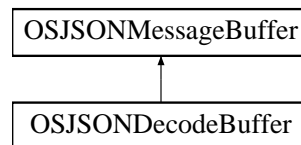
## Class Documentation

### 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:

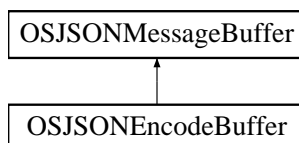
- [OSJSONDecodeBuffer.h](#)

## 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:

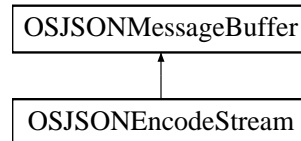
- [OSJSONEncodeBuffer.h](#)

## 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](#)

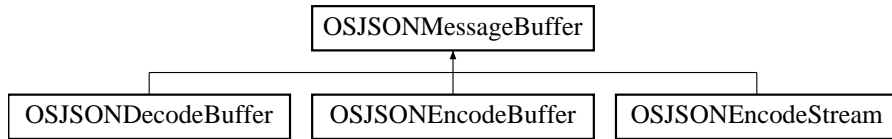


## 7.4 OSJSONMessageBuffer Class Reference

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

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

Inheritance diagram for OSJSONMessageBuffer:



### Protected Member Functions

- EXTJSONMETHOD [OSJSONMessageBuffer](#) (Type *bufferType*, 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 bufferType,  
    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.

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 "rtxsrc/rtxText.h"
#include "rtjsonsrc/rJsonExternDefs.h"
```

### Macros

- #define [OSUPCASE](#) 0x00008000  
*The upper-case flag: if set, hex strings will be encoded in upper case.*
- #define [rtJsonEncIntValue](#) rtxTxtWriteInt  
*This function encodes a variable of the XSD integer type.*
- #define [rtJsonEncInt64Value](#) rtxTxtWriteInt64  
*This function encodes a variable of the XSD integer type.*
- #define [rtJsonEncDecrIndent](#) rtxIndentDecr  
*This decreases the indentation level set in the given context by updating the indent member.*
- #define [rtJsonEncIncrIndent](#) rtxIndentIncr  
*This increases the indentation level set in the given context by updating the indent member.*
- #define [rtJsonEncResetIndent](#) rtxIndentReset  
*This resets the indentation level in the given context to zero.*
- #define [rtJsonGetIndentLevels](#) rtxGetIndentLevels  
*This returns the number of levels of indentation rtJsonEncIndent is using.*
- #define [rtJsonEncUIntValue](#) rtxTxtWriteUInt  
*This function encodes a variable of the XSD unsigned integer type.*
- #define [rtJsonEncUInt64Value](#) rtxTxtWriteUInt64  
*This function encodes a variable of the XSD integer type.*
- #define [rtJsonDecPeekChar](#)(pctxt, pch) rtxTxtPeekChar(pctxt, pch, TRUE)  
*This function determines the next non-whitespace character in the input.*
- #define [rtJsonDecPeekChar2](#)(pctxt) rtxTxtPeekChar2(pctxt, TRUE)  
*This function determines the next non-whitespace character in the input.*

### 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 [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 \*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 the given data as a JSON string, using a hexadecimal representation of the data.*
  - EXTERNJSON int [rtJsonEncHexValue](#) (OSCTXT \*pctx, OSSIZE noctx, const OSOCTET \*data)
 

*This function encodes the given data as a sequence of hexadecimal characters.*
  - EXTERNJSON int [rtJsonEncBitStrValueV72](#) (OSCTXT \*pctx, OSSIZE nbits, const OSOCTET \*data)
 

*This function encodes a variable of the ASN.1 Bit string type.*
  - EXTERNJSON int [rtJsonEncBitStrValueExtV72](#) (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 [rtJsonEncBitStrValue](#) (OSCTXT \*pctx, OSSIZE nbits, const OSOCTET \*data)
 

*This function encodes a variable of the ASN.1 Bit string type as a JSON object, as required by X.697 for a BIT STRING without a fixed length.*
  - EXTERNJSON int [rtJsonEncFixedBitStrValue](#) (OSCTXT \*pctx, OSSIZE nbits, const OSOCTET \*data)
 

*This function encodes a variable of the ASN.1 Bit string type as a JSON string, as required by X.697 for a BIT STRING with a fixed length.*
  - 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 as a JSON object, as required by X.697 for a BIT STRING without a fixed length.*
  - EXTERNJSON int [rtJsonEncIndent](#) (OSCTXT \*pctx)
 

*This function:*
  - 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 UTF8 string value.*
- EXTERNJSON int [rtJsonEncStringValue2](#) (OSCTXT \*pctx, const OSUTF8CHAR \*value, size\_t valueLen)  
*This function encodes a UTF8 string value.*
- EXTERNJSON int [rtJsonEncChars](#) (OSCTXT \*pctx, const char \*value, OSSIZE valueLen)  
*This function encodes the given number of characters from a character string value as a JSON string.*
- EXTERNJSON int [rtJsonEncCharStr](#) (OSCTXT \*pctx, const char \*value)  
*This function encodes a character string value as a JSON string.*
- 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 [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 \*pctx, OSUTF8CHAR \*\*ppvalue)  
*This function decodes an arbitrary block of JSON-encoded data into a string variable.*
- EXTERNJSON int [rtJsonDecAnyElem2](#) (OSCTXT \*pctx, 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 \*pctx, OSUTF8CHAR \*\*ppvalue)  
*This function decodes an arbitrary block of JSON-encoded data into a string variable.*
- EXTERNJSON int [rtJsonDecBase64Str](#) (OSCTXT \*pctx, OSOCTET \*pvalue, OSUINT32 \*pnocets, size\_t bufsize)  
*This function decodes a contents of a Base64-encode binary string into a static memory structure.*
- EXTERNJSON int [rtJsonDecBase64Str64](#) (OSCTXT \*pctx, 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 \*pctx, OSDynOctStr \*pvalue)  
*This function decodes a contents of a Base64-encode binary string.*
- EXTERNJSON int [rtJsonDecDynBase64Str64](#) (OSCTXT \*pctx, OSDynOctStr64 \*pvalue)  
*This function is identical to rtJsonDecDynBase64Str64 except that it supports 64-bit integer lengths on 64-bit systems.*
- EXTERNJSON int [rtJsonDecBool](#) (OSCTXT \*pctx, OSBOOL \*pvalue)

- This function decodes a variable of the boolean type.*

  - EXTERNJSON int [rtJsonTryDecBool](#) (OSCTXT \*pctx, OSBOOL \*pvalue)

*This function will attempt to decode a boolean variable at the current buffer or stream position.*
- EXTERNJSON int [rtJsonDecDate](#) (OSCTXT \*pctx, OSXSDDateTime \*pvalue)

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

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

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

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

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

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

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

*This function decodes a variable of the XSD 'gDay' type.*
- EXTERNJSON int [rtJsonDecDecimal](#) (OSCTXT \*pctx, OSREAL \*pvalue, int totalDigits, int fractionDigits)

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

*This function decodes the contents of a float or double data type.*
- EXTERNJSON int [rtJsonDecHexToCharStr](#) (OSCTXT \*pctx, OSUTF8CHAR \*\*ppvalue)

*This function decodes a JSON string, interpreting the content as the hexadecimal representation of the bytes for a character string, which it returns.*
- EXTERNJSON int [rtJsonDecHexStr](#) (OSCTXT \*pctx, OSOCTET \*pvalue, OSUINT32 \*pnocets, size\_t bufsize)

*This function decodes a JSON string consisting of hexadecimal characters into a static memory structure.*
- EXTERNJSON int [rtJsonDecHexStr64](#) (OSCTXT \*pctx, 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 [rtJsonDecHexData64](#) (OSCTXT \*pctx, OSOCTET \*pvalue, OSSIZE \*pnocets, size\_t bufsize)

*This function decodes a sequence of hexadecimal characters until a non-hexadecimal character (which is not consumed) is found OR the preallocated array is filled.*
- EXTERNJSON int [rtJsonDecDynHexStr](#) (OSCTXT \*pctx, OSDynOctStr \*pvalue)

*This function decodes a JSON string consisting of hexadecimal characters.*
- EXTERNJSON int [rtJsonDecDynHexStr64](#) (OSCTXT \*pctx, OSDynOctStr64 \*pvalue)

*This function is identical to [rtJsonDecDynHexStr](#) except that it supports 64-bit integer lengths on 64-bit systems.*
- EXTERNJSON int [rtJsonDecDynHexData64](#) (OSCTXT \*pctx, OSDynOctStr64 \*pvalue)

*This function decodes a sequence of hexadecimal characters until a non-hexadecimal character (which is not consumed) is found.*
- EXTERNJSON int [rtJsonDecDynBitStrV72](#) (OSCTXT \*pctx, OSUINT32 \*nbits, OSOCTET \*\*data)

*This function decodes a variable of the ASN.1 Bit string type.*
- EXTERNJSON int [rtJsonDecDynBitStr64V72](#) (OSCTXT \*pctx, OSSIZE \*nbits, OSOCTET \*\*data)

*This function is identical to [rtJsonDecDynBitStrV72](#) except that it supports lengths up to 64-bits in size on 64-bit machines.*
- EXTERNJSON int [rtJsonDecBitStrValueV72](#) (OSCTXT \*pctx, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize)

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



- EXTERNJSON int [rtJsonDecBitStrValue64V72](#) (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*data, OSSIZE bufsize)
 

*This function is identical to [rtJsonDecBitStrValueV72](#) except that it supports lengths up to 64-bits in size on 64-bit machines.*
- EXTERNJSON int [rtJsonDecBitStrValueExtV72](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize, OSOCTET \*\*extdata)
 

*This function decodes a variable of the ASN.1 Bit string type.*
- EXTERNJSON int [rtJsonDecBitStrValueExt64V72](#) (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*data, OSSIZE bufsize, OSOCTET \*\*extdata)
 

*This function is identical to [rtJsonDecBitStrValueExtV72](#) except that it supports lengths up to 64-bits in size on 64-bit machines.*
- EXTERNJSON int [rtJsonDecDynBitStr](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*\*data)
 

*This function decodes a value of an ASN.1 BIT STRING type, not constrained to a fixed length, encoded according to X.697 as a JSON object.*
- 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 [rtJsonDecFixedDynBitStr](#) (OSCTXT \*pctxt, OSSIZE nbits, OSOCTET \*\*data)
 

*This function decodes a value of an ASN.1 BIT STRING type constrained to a fixed length, encoded according to X.697 as a JSON string.*
- EXTERNJSON int [rtJsonDecBitStrValue](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize)
 

*This function decodes a value of an ASN.1 BIT STRING type that is not constrained to a fixed length, encoded according to X.697 as a JSON object.*
- 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 [rtJsonDecFixedBitStrValue](#) (OSCTXT \*pctxt, OSSIZE nbits, OSOCTET \*data, OSSIZE bufsize)
 

*This function decodes a value of an ASN.1 BIT STRING type constrained to have the given fixed length, encoded according to X.697 as a JSON string.*
- EXTERNJSON int [rtJsonDecBitStrValueExt](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize, OSOCTET \*\*extdata)
 

*This function decodes a value of a ASN.1 BIT STRING type without a fixed-length constraint, encoded according to X.697 as JSON object.*
- 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 [rtJsonDecMatchChar](#) (OSCTXT \*pctxt, OSUTF8CHAR ch)
 

*This function attempts to match the given character, skipping over any whitespace, if necessary.*
- EXTERNJSON int [rtJsonDecMatchCharStr](#) (OSCTXT \*pctxt, const char \*token)
 

*This function decodes a JSON string and matches with a given token.*
- 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 [rtJsonDecNull](#) (OSCTXT \*pctxt)

- This function decodes a JSON null.*

  - EXTERNJSON int [rtJsonTryDecNull](#) (OSCTXT \*pctx)

*This function will attempt to decode a null value at the current buffer or stream position.*
- EXTERNJSON int [rtJsonDecNumberString](#) (OSCTXT \*pctx, char \*\*ppCharStr)

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

*This function decodes the contents of a string data type.*
- EXTERNJSON int [rtJsonDecStringValueArray](#) (OSCTXT \*pctx, OSUTF8CHAR \*pvalue, OSSIZE bufsize)

*This function is the same as [rtJsonDecStringValue](#) except that it decodes into a character array.*
- EXTERNJSON int [rtJsonDecXmlStringValue](#) (OSCTXT \*pctx, OSXMLSTRING \*pvalue)

*This function decodes the contents of an XML string data type.*
- EXTERNJSON int [rtJsonDecUCS2String](#) (OSCTXT \*pctx, 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 \*pctx, 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 \*pctx, 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/asnlCppType.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, [82](#)
- encodeText
  - OSJSONEncodeStream, [82](#)
- getMsgLen
  - OSJSONEncodeBuffer, [78](#)
- getMsgPtr
  - OSJSONEncodeStream, [82](#)
- getStream
  - OSJSONEncodeStream, [83](#)
- init
  - OSJSONDecodeBuffer, [75](#)
  - OSJSONEncodeBuffer, [78](#)
  - OSJSONEncodeStream, [83](#)
- isA
  - OSJSONDecodeBuffer, [75](#)
  - OSJSONEncodeBuffer, [78](#)
  - OSJSONEncodeStream, [83](#)
- JSON decode functions., [36](#)
  - rtJsonDecAnyElem, [40](#)
  - rtJsonDecAnyElem2, [40](#)
  - rtJsonDecAnyType, [41](#)
  - rtJsonDecBase64Str, [41](#)
  - rtJsonDecBase64Str64, [43](#)
  - rtJsonDecBitStrValue, [44](#)
  - rtJsonDecBitStrValue64, [44](#)
  - rtJsonDecBitStrValue64V72, [45](#)
  - rtJsonDecBitStrValueExt, [45](#)
  - rtJsonDecBitStrValueExt64, [46](#)
  - rtJsonDecBitStrValueExt64V72, [47](#)
  - rtJsonDecBitStrValueExtV72, [47](#)
  - rtJsonDecBitStrValueV72, [48](#)
  - rtJsonDecBool, [49](#)
  - rtJsonDecDate, [49](#)
  - rtJsonDecDateTime, [50](#)
  - rtJsonDecDecimal, [50](#)
  - rtJsonDecDouble, [51](#)
  - rtJsonDecDynBase64Str, [51](#)
  - rtJsonDecDynBase64Str64, [52](#)
  - rtJsonDecDynBitStr, [52](#)
  - rtJsonDecDynBitStr64, [53](#)
  - rtJsonDecDynBitStr64V72, [53](#)
  - rtJsonDecDynBitStrV72, [54](#)
  - rtJsonDecDynHexData64, [54](#)
  - rtJsonDecDynHexStr, [55](#)
  - rtJsonDecDynHexStr64, [56](#)
  - rtJsonDecFixedBitStrValue, [56](#)
  - rtJsonDecFixedDynBitStr, [57](#)
  - rtJsonDecGDay, [57](#)
  - rtJsonDecGMonth, [58](#)
  - rtJsonDecGMonthDay, [58](#)
  - rtJsonDecGYear, [59](#)
  - rtJsonDecGYearMonth, [59](#)
  - rtJsonDecHexData64, [60](#)
  - rtJsonDecHexStr, [60](#)
  - rtJsonDecHexStr64, [61](#)
  - rtJsonDecHexToCharStr, [62](#)
  - rtJsonDecMatchChar, [62](#)
  - rtJsonDecMatchCharStr, [63](#)
  - rtJsonDecMatchObjectStart, [63](#)
  - rtJsonDecMatchToken, [64](#)
  - rtJsonDecMatchToken2, [64](#)
  - rtJsonDecNameValuePair, [65](#)
  - rtJsonDecNull, [65](#)
  - rtJsonDecNumberString, [66](#)
  - rtJsonDecPeekChar, [39](#)
  - rtJsonDecPeekChar2, [39](#)
  - rtJsonDecStringObject, [66](#)
  - rtJsonDecStringValue, [67](#)
  - rtJsonDecStringValueArray, [67](#)
  - rtJsonDecTime, [68](#)
  - rtJsonDecUCS2String, [68](#)
  - rtJsonDecUCS4String, [69](#)
  - rtJsonDecXmlStringValue, [69](#)
  - rtJsonGetElemIdx, [70](#)
  - rtJsonTryDecBool, [71](#)
  - rtJsonTryDecNull, [71](#)
- JSON encode functions., [11](#)
  - rtJsonEncAnyAttr, [17](#)
  - rtJsonEncBase64StrValue, [17](#)
  - rtJsonEncBetweenObject, [18](#)
  - rtJsonEncBitStrValue, [18](#)
  - rtJsonEncBitStrValueExt, [18](#)
  - rtJsonEncBitStrValueExtV72, [19](#)
  - rtJsonEncBitStrValueV72, [20](#)
  - rtJsonEncBoolValue, [20](#)
  - rtJsonEncCharStr, [21](#)
  - rtJsonEncChars, [21](#)

- rtJsonEncDate, [22](#)
- rtJsonEncDateTime, [22](#)
- rtJsonEncDecimalValue, [23](#)
- rtJsonEncDecrIndent, [13](#)
- rtJsonEncDoubleValue, [23](#)
- rtJsonEncEndObject, [24](#)
- rtJsonEncFixedBitStrValue, [24](#)
- rtJsonEncFloatValue, [25](#)
- rtJsonEncGDay, [25](#)
- rtJsonEncGMonth, [26](#)
- rtJsonEncGMonthDay, [26](#)
- rtJsonEncGYear, [27](#)
- rtJsonEncGYearMonth, [27](#)
- rtJsonEncHexStr, [28](#)
- rtJsonEncHexValue, [28](#)
- rtJsonEncIncrIndent, [14](#)
- rtJsonEncIndent, [29](#)
- rtJsonEncInt64Value, [14](#)
- rtJsonEncIntValue, [14](#)
- rtJsonEncResetIndent, [15](#)
- rtJsonEncStartObject, [29](#)
- rtJsonEncStringNull, [30](#)
- rtJsonEncStringObject, [30](#)
- rtJsonEncStringObject2, [31](#)
- rtJsonEncStringPair, [31](#)
- rtJsonEncStringPair2, [32](#)
- rtJsonEncStringRaw, [32](#)
- rtJsonEncStringValue, [33](#)
- rtJsonEncStringValue2, [33](#)
- rtJsonEncTime, [34](#)
- rtJsonEncUCS4Data, [34](#)
- rtJsonEncUInt64Value, [15](#)
- rtJsonEncUIntValue, [16](#)
- rtJsonEncUnicodeData, [35](#)
- rtJsonGetIndentLevels, [16](#)

- mbOwnStream
  - OSJSONDecodeBuffer, [76](#)
  - OSJSONEncodeStream, [84](#)
- mpCtxt
  - OSJSONEncodeStream, [84](#)
- mpStream
  - OSJSONEncodeStream, [84](#)
- OSJSONDecodeBuffer, [73](#)
  - init, [75](#)
  - isA, [75](#)
  - mbOwnStream, [76](#)
  - OSJSONDecodeBuffer, [74, 75](#)
- OSJSONDecodeBuffer.h, [87](#)
- OSJSONEncodeBuffer, [76](#)
  - getMsgLen, [78](#)
  - init, [78](#)
  - isA, [78](#)
  - OSJSONEncodeBuffer, [77](#)
  - write, [79](#)
- OSJSONEncodeBuffer.h, [87](#)
- OSJSONEncodeStream, [80](#)
  - encodeAttr, [82](#)
  - encodeText, [82](#)
  - getMsgPtr, [82](#)
  - getStream, [83](#)
  - init, [83](#)
  - isA, [83](#)
  - mbOwnStream, [84](#)
  - mpCtxt, [84](#)
  - mpStream, [84](#)
  - OSJSONEncodeStream, [81](#)
- OSJSONEncodeStream.h, [88](#)
- OSJSONMessageBuffer, [85](#)
  - OSJSONMessageBuffer, [85](#)
- OSJSONMessageBuffer.h, [88](#)
- OSUPCASE
  - osrtjson.h, [94](#)
- osrtjson.h, [89](#)
  - OSUPCASE, [94](#)
- rtJsonCppMsgBuf.h, [94](#)
- rtJsonDecAnyElem
  - JSON decode functions., [40](#)
- rtJsonDecAnyElem2
  - JSON decode functions., [40](#)
- rtJsonDecAnyType
  - JSON decode functions., [41](#)
- rtJsonDecBase64Str
  - JSON decode functions., [41](#)
- rtJsonDecBase64Str64
  - JSON decode functions., [43](#)
- rtJsonDecBitStrValue
  - JSON decode functions., [44](#)
- rtJsonDecBitStrValue64
  - JSON decode functions., [44](#)
- rtJsonDecBitStrValue64V72
  - JSON decode functions., [45](#)
- rtJsonDecBitStrValueExt
  - JSON decode functions., [45](#)
- rtJsonDecBitStrValueExt64
  - JSON decode functions., [46](#)
- rtJsonDecBitStrValueExt64V72
  - JSON decode functions., [47](#)
- rtJsonDecBitStrValueExtV72
  - JSON decode functions., [47](#)
- rtJsonDecBitStrValueV72
  - JSON decode functions., [48](#)
- rtJsonDecBool
  - JSON decode functions., [49](#)
- rtJsonDecDate
  - JSON decode functions., [49](#)
- rtJsonDecDateTime

- JSON decode functions., 50
- rt.JsonDecDecimal
  - JSON decode functions., 50
- rt.JsonDecDouble
  - JSON decode functions., 51
- rt.JsonDecDynBase64Str
  - JSON decode functions., 51
- rt.JsonDecDynBase64Str64
  - JSON decode functions., 52
- rt.JsonDecDynBitStr
  - JSON decode functions., 52
- rt.JsonDecDynBitStr64
  - JSON decode functions., 53
- rt.JsonDecDynBitStr64V72
  - JSON decode functions., 53
- rt.JsonDecDynBitStrV72
  - JSON decode functions., 54
- rt.JsonDecDynHexData64
  - JSON decode functions., 54
- rt.JsonDecDynHexStr
  - JSON decode functions., 55
- rt.JsonDecDynHexStr64
  - JSON decode functions., 56
- rt.JsonDecFixedBitStrValue
  - JSON decode functions., 56
- rt.JsonDecFixedDynBitStr
  - JSON decode functions., 57
- rt.JsonDecGDay
  - JSON decode functions., 57
- rt.JsonDecGMonth
  - JSON decode functions., 58
- rt.JsonDecGMonthDay
  - JSON decode functions., 58
- rt.JsonDecGYear
  - JSON decode functions., 59
- rt.JsonDecGYearMonth
  - JSON decode functions., 59
- rt.JsonDecHexData64
  - JSON decode functions., 60
- rt.JsonDecHexStr
  - JSON decode functions., 60
- rt.JsonDecHexStr64
  - JSON decode functions., 61
- rt.JsonDecHexToCharStr
  - JSON decode functions., 62
- rt.JsonDecMatchChar
  - JSON decode functions., 62
- rt.JsonDecMatchCharStr
  - JSON decode functions., 63
- rt.JsonDecMatchObjectStart
  - JSON decode functions., 63
- rt.JsonDecMatchToken
  - JSON decode functions., 64
- rt.JsonDecMatchToken2

- JSON decode functions., 64
- rt.JsonDecNameValuePair
  - JSON decode functions., 65
- rt.JsonDecNull
  - JSON decode functions., 65
- rt.JsonDecNumberString
  - JSON decode functions., 66
- rt.JsonDecPeekChar
  - JSON decode functions., 39
- rt.JsonDecPeekChar2
  - JSON decode functions., 39
- rt.JsonDecStringObject
  - JSON decode functions., 66
- rt.JsonDecStringValue
  - JSON decode functions., 67
- rt.JsonDecStringValueArray
  - JSON decode functions., 67
- rt.JsonDecTime
  - JSON decode functions., 68
- rt.JsonDecUCS2String
  - JSON decode functions., 68
- rt.JsonDecUCS4String
  - JSON decode functions., 69
- rt.JsonDecXmlStringValue
  - JSON decode functions., 69
- rt.JsonEncAnyAttr
  - JSON encode functions., 17
- rt.JsonEncBase64StrValue
  - JSON encode functions., 17
- rt.JsonEncBetweenObject
  - JSON encode functions., 18
- rt.JsonEncBitStrValue
  - JSON encode functions., 18
- rt.JsonEncBitStrValueExt
  - JSON encode functions., 18
- rt.JsonEncBitStrValueExtV72
  - JSON encode functions., 19
- rt.JsonEncBitStrValueV72
  - JSON encode functions., 20
- rt.JsonEncBoolValue
  - JSON encode functions., 20
- rt.JsonEncCharStr
  - JSON encode functions., 21
- rt.JsonEncChars
  - JSON encode functions., 21
- rt.JsonEncDate
  - JSON encode functions., 22
- rt.JsonEncDateTime
  - JSON encode functions., 22
- rt.JsonEncDecimalValue
  - JSON encode functions., 23
- rt.JsonEncDecrIndent
  - JSON encode functions., 13
- rt.JsonEncDoubleValue

- JSON encode functions., [23](#)
- rt.JsonEncEndObject
  - JSON encode functions., [24](#)
- rt.JsonEncFixedBitStrValue
  - JSON encode functions., [24](#)
- rt.JsonEncFloatValue
  - JSON encode functions., [25](#)
- rt.JsonEncGDay
  - JSON encode functions., [25](#)
- rt.JsonEncGMonth
  - JSON encode functions., [26](#)
- rt.JsonEncGMonthDay
  - JSON encode functions., [26](#)
- rt.JsonEncGYear
  - JSON encode functions., [27](#)
- rt.JsonEncGYearMonth
  - JSON encode functions., [27](#)
- rt.JsonEncHexStr
  - JSON encode functions., [28](#)
- rt.JsonEncHexValue
  - JSON encode functions., [28](#)
- rt.JsonEncIncrIndent
  - JSON encode functions., [14](#)
- rt.JsonEncIndent
  - JSON encode functions., [29](#)
- rt.JsonEncInt64Value
  - JSON encode functions., [14](#)
- rt.JsonEncIntValue
  - JSON encode functions., [14](#)
- rt.JsonEncResetIndent
  - JSON encode functions., [15](#)
- rt.JsonEncStartObject
  - JSON encode functions., [29](#)
- rt.JsonEncStringNull
  - JSON encode functions., [30](#)
- rt.JsonEncStringObject
  - JSON encode functions., [30](#)
- rt.JsonEncStringObject2
  - JSON encode functions., [31](#)
- rt.JsonEncStringPair
  - JSON encode functions., [31](#)
- rt.JsonEncStringPair2
  - JSON encode functions., [32](#)
- rt.JsonEncStringRaw
  - JSON encode functions., [32](#)
- rt.JsonEncStringValue
  - JSON encode functions., [33](#)
- rt.JsonEncStringValue2
  - JSON encode functions., [33](#)
- rt.JsonEncTime
  - JSON encode functions., [34](#)
- rt.JsonEncUCS4Data
  - JSON encode functions., [34](#)
- rt.JsonEncUInt64Value

- JSON encode functions., [15](#)
- rt.JsonEncUIntValue
  - JSON encode functions., [16](#)
- rt.JsonEncUnicodeData
  - JSON encode functions., [35](#)
- rt.JsonExternDefs.h, [95](#)
- rt.JsonGetElemIdx
  - JSON decode functions., [70](#)
- rt.JsonGetIndentLevels
  - JSON encode functions., [16](#)
- rt.JsonTryDecBool
  - JSON decode functions., [71](#)
- rt.JsonTryDecNull
  - JSON decode functions., [71](#)
- write
  - OSJSONEncodeBuffer, [79](#)