

# ASN1C

---

ASN.1 Compiler  
Version 7.3  
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 ©1997–2019 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 ASN1C may be submitted via electronic mail to [info@obj-sys.com](mailto:info@obj-sys.com).



# Contents

<b>1</b>	<b>Main Page</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>3</b>
2.1	Modules . . . . .	3
<b>3</b>	<b>Hierarchical Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List . . . . .	7
<b>5</b>	<b>File Index</b>	<b>9</b>
5.1	File List . . . . .	9
<b>6</b>	<b>Module Documentation</b>	<b>11</b>
6.1	JSON runtime encoding functions for ASN.1 data types. . . . .	11
6.1.1	Detailed Description . . . . .	11
6.1.2	Function Documentation . . . . .	11
6.1.2.1	rtJsonAsn1EncExtElem() . . . . .	11
6.1.2.2	rtJsonAsn1EncOID() . . . . .	12
6.1.2.3	rtJsonAsn1EncOpenType() . . . . .	12
6.1.2.4	rtJsonAsn1EncOpenTypeExt() . . . . .	12
6.1.2.5	rtJsonAsn1EncReal() . . . . .	13
6.1.2.6	rtJsonAsn1EncReal10() . . . . .	13

6.2	JSON runtime decoding functions for ASN.1 data types.	14
6.2.1	Detailed Description	14
6.2.2	Function Documentation	14
6.2.2.1	rtJsonAsn1DecBMPString()	14
6.2.2.2	rtJsonAsn1DecOIDValue()	14
6.2.2.3	rtJsonAsn1DecReal()	15
6.2.2.4	rtJsonAsn1DecReal10()	15
6.3	JSON encode functions.	17
6.3.1	Detailed Description	18
6.3.2	Function Documentation	18
6.3.2.1	rtJsonEncAnyAttr()	18
6.3.2.2	rtJsonEncBase64StrValue()	18
6.3.2.3	rtJsonEncBetweenObject()	19
6.3.2.4	rtJsonEncBitStrValue()	19
6.3.2.5	rtJsonEncBitStrValueExt()	20
6.3.2.6	rtJsonEncBitStrValueExtV72()	20
6.3.2.7	rtJsonEncBitStrValueV72()	21
6.3.2.8	rtJsonEncBoolValue()	21
6.3.2.9	rtJsonEncChars()	22
6.3.2.10	rtJsonEncCharStr()	22
6.3.2.11	rtJsonEncDate()	23
6.3.2.12	rtJsonEncDateTime()	23
6.3.2.13	rtJsonEncDecimalValue()	24
6.3.2.14	rtJsonEncDecrIndent()	24
6.3.2.15	rtJsonEncDoubleValue()	25
6.3.2.16	rtJsonEncEndObject()	25
6.3.2.17	rtJsonEncFixedBitStrValue()	26
6.3.2.18	rtJsonEncFloatValue()	26

6.3.2.19	rtJsonEncGDay()	27
6.3.2.20	rtJsonEncGMonth()	27
6.3.2.21	rtJsonEncGMonthDay()	28
6.3.2.22	rtJsonEncGYear()	28
6.3.2.23	rtJsonEncGYearMonth()	29
6.3.2.24	rtJsonEncHexStr()	29
6.3.2.25	rtJsonEncHexValue()	30
6.3.2.26	rtJsonEncIncrIndent()	30
6.3.2.27	rtJsonEncIndent()	30
6.3.2.28	rtJsonEncInt64Value()	31
6.3.2.29	rtJsonEncIntValue()	31
6.3.2.30	rtJsonEncResetIndent()	32
6.3.2.31	rtJsonEncStartObject()	32
6.3.2.32	rtJsonEncStringNull()	33
6.3.2.33	rtJsonEncStringObject()	33
6.3.2.34	rtJsonEncStringObject2()	33
6.3.2.35	rtJsonEncStringPair()	34
6.3.2.36	rtJsonEncStringPair2()	35
6.3.2.37	rtJsonEncStringRaw()	35
6.3.2.38	rtJsonEncStringValue()	36
6.3.2.39	rtJsonEncStringValue2()	36
6.3.2.40	rtJsonEncTime()	37
6.3.2.41	rtJsonEncUCS4Data()	37
6.3.2.42	rtJsonEncUInt64Value()	38
6.3.2.43	rtJsonEncUIntValue()	38
6.3.2.44	rtJsonEncUnicodeData()	39
6.3.2.45	rtJsonGetIndentLevels()	39

6.4	JSON decode functions	40
-----	-----------------------	----

6.4.1	Detailed Description	41
6.4.2	Function Documentation	41
6.4.2.1	rtJsonDecAnyElem()	41
6.4.2.2	rtJsonDecAnyElem2()	42
6.4.2.3	rtJsonDecAnyType()	42
6.4.2.4	rtJsonDecBase64Str()	43
6.4.2.5	rtJsonDecBase64Str64()	44
6.4.2.6	rtJsonDecBitStrValue()	44
6.4.2.7	rtJsonDecBitStrValue64()	45
6.4.2.8	rtJsonDecBitStrValue64V72()	45
6.4.2.9	rtJsonDecBitStrValueExt()	46
6.4.2.10	rtJsonDecBitStrValueExt64()	47
6.4.2.11	rtJsonDecBitStrValueExt64V72()	47
6.4.2.12	rtJsonDecBitStrValueExtV72()	48
6.4.2.13	rtJsonDecBitStrValueV72()	49
6.4.2.14	rtJsonDecBool()	49
6.4.2.15	rtJsonDecDate()	50
6.4.2.16	rtJsonDecDateTime()	50
6.4.2.17	rtJsonDecDecimal()	51
6.4.2.18	rtJsonDecDouble()	51
6.4.2.19	rtJsonDecDynBase64Str()	52
6.4.2.20	rtJsonDecDynBase64Str64()	52
6.4.2.21	rtJsonDecDynBitStr()	53
6.4.2.22	rtJsonDecDynBitStr64()	53
6.4.2.23	rtJsonDecDynBitStr64V72()	54
6.4.2.24	rtJsonDecDynBitStrV72()	54
6.4.2.25	rtJsonDecDynHexData64()	55
6.4.2.26	rtJsonDecDynHexStr()	55



6.4.2.27	<code>rtJsonDecDynHexStr64()</code>	56
6.4.2.28	<code>rtJsonDecFixedBitStrValue()</code>	56
6.4.2.29	<code>rtJsonDecFixedDynBitStr()</code>	58
6.4.2.30	<code>rtJsonDecGDay()</code>	58
6.4.2.31	<code>rtJsonDecGMonth()</code>	59
6.4.2.32	<code>rtJsonDecGMonthDay()</code>	59
6.4.2.33	<code>rtJsonDecGYear()</code>	60
6.4.2.34	<code>rtJsonDecGYearMonth()</code>	60
6.4.2.35	<code>rtJsonDecHexData64()</code>	61
6.4.2.36	<code>rtJsonDecHexStr()</code>	61
6.4.2.37	<code>rtJsonDecHexStr64()</code>	62
6.4.2.38	<code>rtJsonDecHexToCharStr()</code>	63
6.4.2.39	<code>rtJsonDecInt16Value()</code>	63
6.4.2.40	<code>rtJsonDecInt32Value()</code>	64
6.4.2.41	<code>rtJsonDecInt64Value()</code>	64
6.4.2.42	<code>rtJsonDecInt8Value()</code>	65
6.4.2.43	<code>rtJsonDecMatchChar()</code>	65
6.4.2.44	<code>rtJsonDecMatchCharStr()</code>	66
6.4.2.45	<code>rtJsonDecMatchObjectStart()</code>	66
6.4.2.46	<code>rtJsonDecMatchToken()</code>	67
6.4.2.47	<code>rtJsonDecMatchToken2()</code>	67
6.4.2.48	<code>rtJsonDecNameValuePair()</code>	68
6.4.2.49	<code>rtJsonDecNull()</code>	68
6.4.2.50	<code>rtJsonDecNumberString()</code>	69
6.4.2.51	<code>rtJsonDecPeekChar()</code>	69
6.4.2.52	<code>rtJsonDecPeekChar2()</code>	70
6.4.2.53	<code>rtJsonDecStringObject()</code>	70
6.4.2.54	<code>rtJsonDecStringValue()</code>	71
6.4.2.55	<code>rtJsonDecTime()</code>	71
6.4.2.56	<code>rtJsonDecUCS2String()</code>	72
6.4.2.57	<code>rtJsonDecUCS4String()</code>	72
6.4.2.58	<code>rtJsonDecUInt16Value()</code>	73
6.4.2.59	<code>rtJsonDecUInt32Value()</code>	73
6.4.2.60	<code>rtJsonDecUInt64Value()</code>	74
6.4.2.61	<code>rtJsonDecUInt8Value()</code>	74
6.4.2.62	<code>rtJsonDecXmlStringValue()</code>	75
6.4.2.63	<code>rtJsonGetElemIdx()</code>	75
6.4.2.64	<code>rtJsonTryDecBool()</code>	76
6.4.2.65	<code>rtJsonTryDecNull()</code>	76

<b>7</b>	<b>Class Documentation</b>	<b>79</b>
7.1	OSJSONDecodeBuffer Class Reference	79
7.1.1	Detailed Description	80
7.1.2	Constructor & Destructor Documentation	80
7.1.2.1	OSJSONDecodeBuffer() [1/3]	80
7.1.2.2	OSJSONDecodeBuffer() [2/3]	80
7.1.2.3	OSJSONDecodeBuffer() [3/3]	81
7.1.3	Member Function Documentation	81
7.1.3.1	init()	81
7.1.3.2	isA()	81
7.1.4	Member Data Documentation	82
7.1.4.1	mbOwnStream	82
7.1.4.2	mpInputStream	82
7.2	OSJSONEncodeBuffer Class Reference	82
7.2.1	Detailed Description	83
7.2.2	Constructor & Destructor Documentation	83
7.2.2.1	OSJSONEncodeBuffer() [1/2]	83
7.2.2.2	OSJSONEncodeBuffer() [2/2]	83
7.2.3	Member Function Documentation	84
7.2.3.1	getMsgLen()	84
7.2.3.2	init()	84
7.2.3.3	isA()	84
7.2.3.4	nullTerminate()	85
7.2.3.5	write() [1/2]	85
7.2.3.6	write() [2/2]	85
7.3	OSJSONEncodeStream Class Reference	86
7.3.1	Detailed Description	86
7.3.2	Constructor & Destructor Documentation	86

7.3.2.1	OSJSONEncodeStream() [1/2]	86
7.3.2.2	OSJSONEncodeStream() [2/2]	87
7.3.3	Member Function Documentation	87
7.3.3.1	encodeAttr()	87
7.3.3.2	encodeText()	88
7.3.3.3	getMsgPtr()	88
7.3.3.4	getStream()	88
7.3.3.5	init()	89
7.3.3.6	isA()	89
7.3.4	Member Data Documentation	89
7.3.4.1	mbOwnStream	89
7.3.4.2	mpCtxt	89
7.3.4.3	mpStream	90
7.4	OSJSONMessageBuffer Class Reference	90
7.4.1	Detailed Description	90
7.4.2	Constructor & Destructor Documentation	90
7.4.2.1	OSJSONMessageBuffer()	90
<b>8</b>	<b>File Documentation</b>	<b>93</b>
8.1	asn1json.h File Reference	93
8.1.1	Detailed Description	93
8.2	OSJSONDecodeBuffer.h File Reference	93
8.2.1	Detailed Description	94
8.3	OSJSONEncodeBuffer.h File Reference	94
8.3.1	Detailed Description	94
8.4	OSJSONEncodeStream.h File Reference	94
8.4.1	Detailed Description	94
8.5	OSJSONMessageBuffer.h File Reference	94
8.5.1	Detailed Description	95
8.6	osrtjson.h File Reference	95
8.6.1	Detailed Description	98
8.6.2	Macro Definition Documentation	98
8.6.2.1	OSJSONPUTCOMMA	98
8.6.2.2	OSUPCASE	98
8.7	rtJsonCppMsgBuf.h File Reference	99
8.7.1	Detailed Description	99
8.8	rtJsonExternDefs.h File Reference	99
8.8.1	Detailed Description	99
<b>Index</b>		<b>101</b>



## Chapter 1

# Main Page

### C JSON Runtime Library Functions

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



# Chapter 2

## Module Index

### 2.1 Modules

Here is a list of all modules:

JSON runtime encoding functions for ASN.1 data types. . . . .	11
JSON runtime decoding functions for ASN.1 data types. . . . .	14
JSON encode functions. . . . .	17
JSON decode functions. . . . .	40





# Chapter 3

## Hierarchical Index

### 3.1 Class Hierarchy

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

- OSRTMessageBuffer
- OSJSONMessageBuffer . . . . . 90
- OSJSONDecodeBuffer . . . . . 79
- OSJSONEncodeBuffer . . . . . 82
- OSJSONEncodeStream . . . . . 86



# Chapter 4

## Class Index

### 4.1 Class List

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

<a href="#">OSJSONDecodeBuffer</a>	79
<a href="#">OSJSONEncodeBuffer</a>	82
<a href="#">OSJSONEncodeStream</a>	86
<a href="#">OSJSONMessageBuffer</a>	90



# Chapter 5

## File Index

### 5.1 File List

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

<a href="#">asn1json.h</a>	93
<a href="#">OSJSONDecodeBuffer.h</a>	93
<a href="#">OSJSONEncodeBuffer.h</a>	94
<a href="#">OSJSONEncodeStream.h</a>	94
<a href="#">OSJSONMessageBuffer.h</a>	94
<a href="#">osrtjson.h</a>	95
<a href="#">rtJsonCppMsgBuf.h</a>	99
<a href="#">rtJsonExternDefs.h</a>	99



# Chapter 6

## Module Documentation

### 6.1 JSON runtime encoding functions for ASN.1 data types.

#### Functions

- int [rtJsonAsn1EncReal](#) (OSCTXT \*pctxt, OSREAL value)
- int [rtJsonAsn1EncReal10](#) (OSCTXT \*pctxt, const OSUTF8CHAR \*pvalue)
- int [rtJsonAsn1EncOID](#) (OSCTXT \*pctxt, ASN1OBJID \*pOID)
- int [rtJsonAsn1EncOpenType](#) (OSCTXT \*pctxt, OSOCTET \*pdata, OSSIZE numocts)
- int [rtJsonAsn1EncExtElem](#) (OSCTXT \*pctxt, ASN1OpenType \*pElem)
- int [rtJsonAsn1EncOpenTypeExt](#) (OSCTXT \*pctxt, const OSRTDList \*pvalue, OSBOOL asArray)

#### 6.1.1 Detailed Description

#### 6.1.2 Function Documentation

##### 6.1.2.1 [rtJsonAsn1EncExtElem\(\)](#)

```
int rtJsonAsn1EncExtElem (  
    OSCTXT * pctxt,  
    ASN1OpenType * pElem )
```

This function encodes an unknown extension element.

#### Parameters

<i>pctxt</i>	A pointer to a context data structure.
<i>pElem</i>	The unknown extension element value. If the encoding is not set or is set to JSON, the data is expected to be a JSONNamedValue, which will be written to the encoding as-is. Otherwise, the data will be written in hexadecimal form, wrapped inside an element.

### 6.1.2.2 rtJsonAsn1EncOID()

```
int rtJsonAsn1EncOID (
    OSCTXT * pctxt,
    ASN1OBJID * pOID )
```

This function encodes the JSON representation of an OID value.

#### Parameters

<i>pctxt</i>	A pointer to a context data structure.
<i>pOID</i>	A pointer to an ASN1OBJID data structure.

#### Returns

0 on success, less than zero otherwise.

### 6.1.2.3 rtJsonAsn1EncOpenType()

```
int rtJsonAsn1EncOpenType (
    OSCTXT * pctxt,
    OSOCTET * pdata,
    OSSIZE numocts )
```

This function encodes an ASN.1 open type to JSON.

If the input data are valid JSON data, then the output is encoded as a JSON string object. If the input data are not valid (e.g., binary data encoded with different rules, they are dropped.

### 6.1.2.4 rtJsonAsn1EncOpenTypeExt()

```
int rtJsonAsn1EncOpenTypeExt (
    OSCTXT * pctxt,
    const OSRTDList * pvalue,
    OSBOOL asArray )
```

This function encodes a list of unknown extension elements.

#### Parameters

<i>pctxt</i>	A pointer to a context data structure.
<i>pvalue</i>	A list of ASN1OpenType objects, each representing an unknown extension element. These will be encoded as by rtJsonEncExtElem.
<i>asArray</i>	TRUE if the enclosing type is [ARRAY] SEQUENCE; FALSE otherwise.



### 6.1.2.5 rtJsonAsn1EncReal()

```
int rtJsonAsn1EncReal (
    OSCTXT * pctxt,
    OSREAL value )
```

This function encodes the JSON representation of an ASN.1 REAL value.

#### Parameters

<i>pctxt</i>	A pointer to a context data structure.
<i>value</i>	The value to encode

#### Returns

0 on success, less than zero otherwise.

### 6.1.2.6 rtJsonAsn1EncReal10()

```
int rtJsonAsn1EncReal10 (
    OSCTXT * pctxt,
    const OSUTF8CHAR * pvalue )
```

This function encodes the JSON representation of an ASN.1 REAL value.

#### Parameters

<i>pctxt</i>	A pointer to a context data structure.
<i>pvalue</i>	The value to encode. This should be a valid JSON number, or may also be "INF", "INFINITY", "NAN", with an optional leading sign, and case ignored. For "NAN", "NAN(n-char-sequence)" is also allowed.

#### Returns

0 on success, less than zero otherwise.

## 6.2 JSON runtime decoding functions for ASN.1 data types.

### Functions

- int [rtJsonAsn1DecReal](#) (OSCTXT \*pctxt, OSREAL \*pvalue)
- int [rtJsonAsn1DecReal10](#) (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)
- int [rtJsonAsn1DecOIDValue](#) (OSCTXT \*pctxt, ASN1OBJID \*pOID)
- int [rtJsonAsn1DecBMPString](#) (OSCTXT \*pctxt, ASN1BMPString \*pBMPStr)

### 6.2.1 Detailed Description

### 6.2.2 Function Documentation

#### 6.2.2.1 rtJsonAsn1DecBMPString()

```
int rtJsonAsn1DecBMPString (
    OSCTXT * pctxt,
    ASN1BMPString * pBMPStr )
```

This function decodes the ASN.1 BMPString type. We assume that the input data are encoded in UTF-8. The data are internally represented as two-byte characters.

#### Parameters

<i>pctxt</i>	A pointer to a context data structure.
<i>pBMPStr</i>	A pointer to an ASN.1 BMP string.

#### Returns

0 on success, less than zero otherwise.

#### 6.2.2.2 rtJsonAsn1DecOIDValue()

```
int rtJsonAsn1DecOIDValue (
    OSCTXT * pctxt,
    ASN1OBJID * pOID )
```

This function decodes the JSON representation of an OID value.

#### Parameters

<i>pctxt</i>	A pointer to a context data structure.
<i>pOID</i>	A pointer to an ASN1OBJID data structure.

#### Returns

0 on success, less than zero otherwise.

#### 6.2.2.3 rtJsonAsn1DecReal()

```
int rtJsonAsn1DecReal (
    OSCTXT * pctxt,
    OSREAL * pvalue )
```

This function decodes the JSON representation of an ASN.1 REAL value.

#### Parameters

<i>pctxt</i>	A pointer to a context data structure.
<i>pvalue</i>	The value to decode into.

#### Returns

0 on success, less than zero otherwise.

#### 6.2.2.4 rtJsonAsn1DecReal10()

```
int rtJsonAsn1DecReal10 (
    OSCTXT * pctxt,
    OSUTF8CHAR ** ppvalue )
```

This function decodes the JSON representation of an ASN.1 REAL value.

#### Parameters

<i>pctxt</i>	A pointer to a context data structure.
<i>ppvalue</i>	Receives a pointer to the decoded value. Dynamic memory is allocated for the string using the rtxMemAlloc function.

## Returns

0 on success, less than zero otherwise.

## 6.3 JSON encode functions.

### Functions

- `int rtJsonEncAnyAttr` (OSCTXT \*pctxt, const OSRTDList \*pvalue)
- `int rtJsonEncIntValue` (OSCTXT \*pctxt, OSINT32 value)
- `int rtJsonEncInt64Value` (OSCTXT \*pctxt, OSINT64 value)
- `int rtJsonEncBase64StringValue` (OSCTXT \*pctxt, OSSIZE nocts, const OSOCTET \*value)
- `int rtJsonEncBoolValue` (OSCTXT \*pctxt, OSBOOL value)
- `int rtJsonEncGYear` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)
- `int rtJsonEncGYearMonth` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)
- `int rtJsonEncGMonth` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)
- `int rtJsonEncGMonthDay` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)
- `int rtJsonEncGDay` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)
- `int rtJsonEncDate` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)
- `int rtJsonEncTime` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)
- `int rtJsonEncDateTime` (OSCTXT \*pctxt, const OSXSDDateTime \*pvalue)
- `int rtJsonEncDecimalValue` (OSCTXT \*pctxt, OSREAL value, const OSDecimalFmt \*pFmtSpec)
- `int rtJsonEncDoubleValue` (OSCTXT \*pctxt, OSREAL value, const OSDoubleFmt \*pFmtSpec)
- `int rtJsonEncFloatValue` (OSCTXT \*pctxt, OSREAL value, const OSDoubleFmt \*pFmtSpec)
- `int rtJsonEncHexStr` (OSCTXT \*pctxt, OSSIZE nocts, const OSOCTET \*data)
- `int rtJsonEncHexValue` (OSCTXT \*pctxt, OSSIZE nocts, const OSOCTET \*data)
- `int rtJsonEncBitStringValueV72` (OSCTXT \*pctxt, OSSIZE nbits, const OSOCTET \*data)
- `int rtJsonEncBitStringValueExtV72` (OSCTXT \*pctxt, OSSIZE nbits, const OSOCTET \*data, OSSIZE dataSize, const OSOCTET \*extData)
- `int rtJsonEncBitStringValue` (OSCTXT \*pctxt, OSSIZE nbits, const OSOCTET \*data)
- `int rtJsonEncFixedBitStringValue` (OSCTXT \*pctxt, OSSIZE nbits, const OSOCTET \*data)
- `int rtJsonEncBitStringValueExt` (OSCTXT \*pctxt, OSSIZE nbits, const OSOCTET \*data, OSSIZE dataSize, const OSOCTET \*extData)
- `int rtJsonEncIndent` (OSCTXT \*pctxt)
- `void rtJsonEncDecrIndent` (OSCTXT \*pctxt)
- `void rtJsonEncIncrIndent` (OSCTXT \*pctxt)
- `void rtJsonEncResetIndent` (OSCTXT \*pctxt)
- `size_t rtJsonGetIndentLevels` (OSCTXT \*pctxt)
- `int rtJsonEncStringObject` (OSCTXT \*pctxt, const OSUTF8CHAR \*name, const OSUTF8CHAR \*value)
- `int rtJsonEncStringObject2` (OSCTXT \*pctxt, const OSUTF8CHAR \*name, size\_t nameLen, const OSUTF8CHAR \*value, size\_t valueLen)
- `int rtJsonEncStringPair` (OSCTXT \*pctxt, const OSUTF8CHAR \*name, const OSUTF8CHAR \*value)
- `int rtJsonEncStringPair2` (OSCTXT \*pctxt, const OSUTF8CHAR \*name, size\_t nameLen, const OSUTF8CHAR \*value, size\_t valueLen)
- `int rtJsonEncStringValue` (OSCTXT \*pctxt, const OSUTF8CHAR \*value)
- `int rtJsonEncStringValue2` (OSCTXT \*pctxt, const OSUTF8CHAR \*value, size\_t valueLen)
- `int rtJsonEncChars` (OSCTXT \*pctxt, const char \*value, OSSIZE valueLen)
- `int rtJsonEncCharStr` (OSCTXT \*pctxt, const char \*value)
- `int rtJsonEncStringNull` (OSCTXT \*pctxt)
- `int rtJsonEncStringRaw` (OSCTXT \*pctxt, const OSUTF8CHAR \*value)
- `int rtJsonEncUnicodeData` (OSCTXT \*pctxt, const OSUNICHAR \*value, OSSIZE nchars)
- `int rtJsonEncUCS4Data` (OSCTXT \*pctxt, const OS32BITCHAR \*value, OSSIZE nchars)
- `int rtJsonEncUIntValue` (OSCTXT \*pctxt, OSUINT32 value)
- `int rtJsonEncUInt64Value` (OSCTXT \*pctxt, OSUINT64 value)
- `int rtJsonEncStartObject` (OSCTXT \*pctxt, const OSUTF8CHAR \*name, OSBOOL noComma)
- `int rtJsonEncEndObject` (OSCTXT \*pctxt)
- `int rtJsonEncBetweenObject` (OSCTXT \*pctxt)

### 6.3.1 Detailed Description

### 6.3.2 Function Documentation

#### 6.3.2.1 rtJsonEncAnyAttr()

```
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.

#### 6.3.2.2 rtJsonEncBase64StrValue()

```
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.

### 6.3.2.3 rtJsonEncBetweenObject()

```
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.

### 6.3.2.4 rtJsonEncBitStrValue()

```
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.

### 6.3.2.5 rtJsonEncBitStrValueExt()

```
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.

### 6.3.2.6 rtJsonEncBitStrValueExtV72()

```
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 extdata to be encoded.

### Returns

Completion status of operation:

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

#### 6.3.2.7 rtJsonEncBitStrValueV72()

```
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.

#### 6.3.2.8 rtJsonEncBoolValue()

```
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.

#### 6.3.2.9 rtJsonEncChars()

```
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.

#### 6.3.2.10 rtJsonEncCharStr()

```
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.

#### 6.3.2.11 rtJsonEncDate()

```
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.

#### 6.3.2.12 rtJsonEncDateTime()

```
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.

#### 6.3.2.13 rtJsonEncDecimalValue()

```
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.

#### 6.3.2.14 rtJsonEncDecrIndent()

```
void rtJsonEncDecrIndent (
    OSCTXT * pctxt )
```

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

## Parameters

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

### 6.3.2.15 rtJsonEncDoubleValue()

```
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.

### 6.3.2.16 rtJsonEncEndObject()

```
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.

### 6.3.2.17 rtJsonEncFixedBitStrValue()

```
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.

### 6.3.2.18 rtJsonEncFloatValue()

```
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.

### 6.3.2.19 rtJsonEncGDay()

```
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.

### 6.3.2.20 rtJsonEncGMonth()

```
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.

### 6.3.2.21 rtJsonEncGMonthDay()

```
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.

### 6.3.2.22 rtJsonEncGYear()

```
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.



### 6.3.2.23 rtJsonEncGYearMonth()

```
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.

### 6.3.2.24 rtJsonEncHexStr()

```
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.

### 6.3.2.25 rtJsonEncHexValue()

```
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.

### 6.3.2.26 rtJsonEncIncrIndent()

```
void rtJsonEncIncrIndent (
    OSCTXT * pctxt )
```

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

#### Parameters

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

### 6.3.2.27 rtJsonEncIndent()

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

This function adds indentation whitespace to the output stream. 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.

#### 6.3.2.28 rtJsonEncInt64Value()

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

This function encodes a variable of the XSD integer type. This version of the function is used for 64-bit integer values.

#### Parameters

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

#### Returns

Completion status of operation:

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

#### 6.3.2.29 rtJsonEncIntValue()

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

This function encodes a variable of the XSD integer type.

#### Parameters

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

## Returns

Completion status of operation:

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

### 6.3.2.30 rtJsonEncResetIndent()

```
void rtJsonEncResetIndent (
    OSCTXT * pctxt )
```

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

#### Parameters

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

### 6.3.2.31 rtJsonEncStartObject()

```
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.

### 6.3.2.32 rtJsonEncStringNull()

```
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.

### 6.3.2.33 rtJsonEncStringObject()

```
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.

### 6.3.2.34 rtJsonEncStringObject2()

```
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.

#### 6.3.2.35 rtJsonEncStringPair()

```

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.

### 6.3.2.36 rtJsonEncStringPair2()

```
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.

### 6.3.2.37 rtJsonEncStringRaw()

```
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.

### 6.3.2.38 rtJsonEncStringValue()

```
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.

### 6.3.2.39 rtJsonEncStringValue2()

```
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.



### 6.3.2.40 rtJsonEncTime()

```
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.

### 6.3.2.41 rtJsonEncUCS4Data()

```
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.

### 6.3.2.42 rtJsonEncUInt64Value()

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

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

#### Parameters

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

#### Returns

Completion status of operation:

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

### 6.3.2.43 rtJsonEncUIntValue()

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

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

#### Parameters

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

#### Returns

Completion status of operation:

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

#### 6.3.2.44 rtJsonEncUnicodeData()

```
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.

#### 6.3.2.45 rtJsonGetIndentLevels()

```
size_t rtJsonGetIndentLevels (
    OSCTXT * pctxt )
```

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

##### Parameters

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

## 6.4 JSON decode functions.

### Macros

- #define **rtJsonDeclntValue** [rtJsonDeclnt32Value](#)
- #define **rtJsonDecUIntValue** [rtJsonDecUInt32Value](#)

### Functions

- int [rtJsonDecAnyElem](#) (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)
- int [rtJsonDecAnyElem2](#) (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)
- int [rtJsonDecAnyType](#) (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)
- int [rtJsonDecBase64Str](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocets, size\_t bufsize)
- int [rtJsonDecBase64Str64](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocets, size\_t bufsize)
- int [rtJsonDecDynBase64Str](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)
- int [rtJsonDecDynBase64Str64](#) (OSCTXT \*pctxt, OSDynOctStr64 \*pvalue)
- int [rtJsonDecBool](#) (OSCTXT \*pctxt, OSBOOL \*pvalue)
- int [rtJsonTryDecBool](#) (OSCTXT \*pctxt, OSBOOL \*pvalue)
- int [rtJsonDecDate](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- int [rtJsonDecTime](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- int [rtJsonDecDateTime](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- int [rtJsonDecGYear](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- int [rtJsonDecGYearMonth](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- int [rtJsonDecGMonth](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- int [rtJsonDecGMonthDay](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- int [rtJsonDecGDay](#) (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- int [rtJsonDecDecimal](#) (OSCTXT \*pctxt, OSREAL \*pvalue, int totalDigits, int fractionDigits)
- int [rtJsonDecDouble](#) (OSCTXT \*pctxt, OSREAL \*pvalue)
- int [rtJsonDecHexToCharStr](#) (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)
- int [rtJsonDecHexStr](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocets, size\_t bufsize)
- int [rtJsonDecHexStr64](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocets, size\_t bufsize)
- int [rtJsonDecHexData64](#) (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocets, size\_t bufsize)
- int [rtJsonDecDynHexStr](#) (OSCTXT \*pctxt, OSDynOctStr \*pvalue)
- int [rtJsonDecDynHexStr64](#) (OSCTXT \*pctxt, OSDynOctStr64 \*pvalue)
- int [rtJsonDecDynHexData64](#) (OSCTXT \*pctxt, OSDynOctStr64 \*pvalue)
- int [rtJsonDecDynBitStrV72](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*\*data)
- int [rtJsonDecDynBitStr64V72](#) (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*\*data)
- int [rtJsonDecBitStrValueV72](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize)
- int [rtJsonDecBitStrValue64V72](#) (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*data, OSSIZE bufsize)
- int [rtJsonDecBitStrValueExtV72](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize, OSO↵CTET \*\*extdata)
- int [rtJsonDecBitStrValueExt64V72](#) (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*data, OSSIZE bufsize, OSO↵CTET \*\*extdata)
- int [rtJsonDecDynBitStr](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*\*data)
- int [rtJsonDecDynBitStr64](#) (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*\*data)
- int [rtJsonDecFixedDynBitStr](#) (OSCTXT \*pctxt, OSSIZE nbits, OSOCTET \*\*data)
- int [rtJsonDecBitStrValue](#) (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize)
- int [rtJsonDecBitStrValue64](#) (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*data, OSSIZE bufsize)
- int [rtJsonDecFixedBitStrValue](#) (OSCTXT \*pctxt, OSSIZE nbits, OSOCTET \*data, OSSIZE bufsize)

- `int rtJsonDecBitStrValueExt` (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize, OSOCTET \*\*extdata)
- `int rtJsonDecBitStrValueExt64` (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*data, OSSIZE bufsize, OSOCTET \*\*extdata)
- `int rtJsonDecInt8Value` (OSCTXT \*pctxt, OSINT8 \*pvalue)
- `int rtJsonDecInt16Value` (OSCTXT \*pctxt, OSINT16 \*pvalue)
- `int rtJsonDecInt32Value` (OSCTXT \*pctxt, OSINT32 \*pvalue)
- `int rtJsonDecInt64Value` (OSCTXT \*pctxt, OSINT64 \*pvalue)
- `int rtJsonDecUInt8Value` (OSCTXT \*pctxt, OSUINT8 \*pvalue)
- `int rtJsonDecUInt16Value` (OSCTXT \*pctxt, OSUINT16 \*pvalue)
- `int rtJsonDecUInt32Value` (OSCTXT \*pctxt, OSUINT32 \*pvalue)
- `int rtJsonDecUInt64Value` (OSCTXT \*pctxt, OSUINT64 \*pvalue)
- `int rtJsonDecMatchChar` (OSCTXT \*pctxt, OSUTF8CHAR ch)
- `int rtJsonDecMatchCharStr` (OSCTXT \*pctxt, const char \*token)
- `int rtJsonDecMatchObjectStart` (OSCTXT \*pctxt, const OSUTF8NameAndLen \*nameArray, size\_t numNames)
- `int rtJsonDecMatchToken` (OSCTXT \*pctxt, const OSUTF8CHAR \*token)
- `int rtJsonDecMatchToken2` (OSCTXT \*pctxt, const OSUTF8CHAR \*token, size\_t tokenLen)
- `int rtJsonDecNameValuePair` (OSCTXT \*pctxt, OSUTF8NVP \*pvalue)
- `int rtJsonDecNull` (OSCTXT \*pctxt)
- `int rtJsonTryDecNull` (OSCTXT \*pctxt)
- `int rtJsonDecNumberString` (OSCTXT \*pctxt, char \*\*ppCharStr)
- `int rtJsonDecPeekChar` (OSCTXT \*pctxt, OSUTF8CHAR \*pch)
- `char rtJsonDecPeekChar2` (OSCTXT \*pctxt)
- `int rtJsonDecStringObject` (OSCTXT \*pctxt, const OSUTF8CHAR \*name, OSUTF8CHAR \*\*ppvalue)
- `int rtJsonDecStringValue` (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)
- `int rtJsonDecXmlStringValue` (OSCTXT \*pctxt, OSXMLSTRING \*pvalue)
- `int rtJsonDecUCS2String` (OSCTXT \*pctxt, OSUNICHAR \*\*ppstr, OSSIZE \*pnchars)
- `int rtJsonDecUCS4String` (OSCTXT \*pctxt, OS32BITCHAR \*\*ppstr, OSSIZE \*pnchars)
- `int rtJsonDecSkipWhitespace` (OSCTXT \*pctxt)
- `size_t rtJsonGetElemIdx` (OSCTXT \*pctxt, const OSUTF8NameAndLen nameArray[], size\_t nrows)

## 6.4.1 Detailed Description

## 6.4.2 Function Documentation

### 6.4.2.1 `rtJsonDecAnyElem()`

```
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.

### 6.4.2.2 rtJsonDecAnyElem2()

```
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.

### 6.4.2.3 rtJsonDecAnyType()

```
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.

### 6.4.2.4 rtJsonDecBase64Str()

```
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.

#### 6.4.2.5 rtJsonDecBase64Str64()

```
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 <i>bufsize</i> 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](#)

#### 6.4.2.6 rtJsonDecBitStrValue()

```
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 <i>data</i> is to be decoded.



## Returns

Completion status of operation:

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

### 6.4.2.7 rtJsonDecBitStrValue64()

```
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](#)

### 6.4.2.8 rtJsonDecBitStrValue64V72()

```
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](#)

#### 6.4.2.9 rtJsonDecBitStrValueExt()

```
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.

It handles bit strings with extdata member present.

#### Parameters

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

#### Returns

Completion status of operation:

- 0 = success,

- negative return value is error.

#### 6.4.2.10 rtJsonDecBitStrValueExt64()

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

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

#### Parameters

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

#### Returns

Completion status of operation:

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

#### See also

[rtJsonDecBitStrValueExt](#)

#### 6.4.2.11 rtJsonDecBitStrValueExt64V72()

```
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](#)

### 6.4.2.12 rtJsonDecBitStrValueExtV72()

```
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 extdata value.

## Returns

Completion status of operation:

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

#### 6.4.2.13 rtJsonDecBitStrValueV72()

```
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.

#### 6.4.2.14 rtJsonDecBool()

```
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.

#### 6.4.2.15 rtJsonDecDate()

```
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.

#### 6.4.2.16 rtJsonDecDateTime()

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

This function decodes a variable of the XSD 'dateTime' type. Input is expected to be a string of characters returned by an JSON parser.

##### Parameters

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

##### Returns

Completion status of operation:

- 0 = success,

- negative return value is error.

#### 6.4.2.17 rtJsonDecDecimal()

```
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.

#### 6.4.2.18 rtJsonDecDouble()

```
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.

### 6.4.2.19 rtJsonDecDynBase64Str()

```
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.

### 6.4.2.20 rtJsonDecDynBase64Str64()

```
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 ::rtxMemAlloc function.



## Returns

Completion status of operation:

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

### 6.4.2.21 rtJsonDecDynBitStr()

```
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.

### 6.4.2.22 rtJsonDecDynBitStr64()

```
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](#)

### 6.4.2.23 rtJsonDecDynBitStr64V72()

```
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](#)

### 6.4.2.24 rtJsonDecDynBitStrV72()

```
int rtJsonDecDynBitStrV72 (
    OSCTXT * pctxt,
    OSUIN32 * 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.

#### 6.4.2.25 rtJsonDecDynHexData64()

```
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.

#### 6.4.2.26 rtJsonDecDynHexStr()

```
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 <code>::rtxMemAlloc</code> function.

## Returns

Completion status of operation:

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

### 6.4.2.27 `rtJsonDecDynHexStr64()`

```
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.

### 6.4.2.28 `rtJsonDecFixedBitStrValue()`

```
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.

#### 6.4.2.29 rtJsonDecFixedDynBitStr()

```
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.

#### 6.4.2.30 rtJsonDecGDay()

```
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.

#### 6.4.2.31 rtJsonDecGMonth()

```
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.

#### 6.4.2.32 rtJsonDecGMonthDay()

```
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.

#### 6.4.2.33 rtJsonDecYear()

```
int rtJsonDecYear (
    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.

#### 6.4.2.34 rtJsonDecYearMonth()

```
int rtJsonDecYearMonth (
    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.

#### 6.4.2.35 rtJsonDecHexData64()

```
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.

#### 6.4.2.36 rtJsonDecHexStr()

```
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.

**6.4.2.37 rtJsonDecHexStr64()**

```

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](#)

### 6.4.2.38 rtJsonDecHexToCharStr()

```
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.

### 6.4.2.39 rtJsonDecInt16Value()

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

This function decodes the contents of a 16-bit integer data type. Input is expected to be a string of OSUTF8CHAR characters returned by a JSON parser.

#### Parameters

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

#### Returns

Completion status of operation:

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

#### 6.4.2.40 rtJsonDecInt32Value()

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

This function decodes the contents of a 32-bit integer data type. Input is expected to be a string of OSUTF8CHAR characters returned by a JSON parser.

#### Parameters

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

#### Returns

Completion status of operation:

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

#### 6.4.2.41 rtJsonDecInt64Value()

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

This function decodes the contents of a 64-bit integer data type. Input is expected to be a string of OSUTF8CHAR characters returned by a JSON parser.

#### Parameters

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

#### Returns

Completion status of operation:

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

#### 6.4.2.42 rtJsonDecInt8Value()

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

This function decodes the contents of an 8-bit integer data type (i.e. a signed byte type in the range of -128 to 127). Input is expected to be a string of OSUTF8CHAR characters returned by a JSON parser.

#### Parameters

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

#### Returns

Completion status of operation:

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

#### 6.4.2.43 rtJsonDecMatchChar()

```
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.

#### 6.4.2.44 rtJsonDecMatchCharStr()

```
int rtJsonDecMatchCharStr (
    OSCTXT * pctxt,
    const char * 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.

#### 6.4.2.45 rtJsonDecMatchObjectStart()

```
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.

#### 6.4.2.46 `rtJsonDecMatchToken()`

```
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.

#### 6.4.2.47 `rtJsonDecMatchToken2()`

```
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.

#### 6.4.2.48 rtJsonDecNameValuePair()

```

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.

#### 6.4.2.49 rtJsonDecNull()

```

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.

#### 6.4.2.50 rtJsonDecNumberString()

```
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 rtxMemAlloc function.

#### Returns

Completion status of operation:

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

#### 6.4.2.51 rtJsonDecPeekChar()

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

This function determines the next non-whitespace character in the input. The non-whitespace character is not consumed.

#### Parameters

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

## Returns

Completion status of operation:

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

### 6.4.2.52 rtJsonDecPeekChar2()

```
char rtJsonDecPeekChar2 (
    OSCTXT * pctxt )
```

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.

### 6.4.2.53 rtJsonDecStringObject()

```
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.

#### 6.4.2.54 rtJsonDecStringValue()

```
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.

#### 6.4.2.55 rtJsonDecTime()

```
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.

### 6.4.2.56 rtJsonDecUCS2String()

```
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.

### 6.4.2.57 rtJsonDecUCS4String()

```
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.

### 6.4.2.58 rtJsonDecUInt16Value()

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

This function decodes the contents of an unsigned 16-bit integer data type. Input is expected to be a string of OSUT↔F8CHAR characters returned by a JSON parser.

#### Parameters

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

## Returns

Completion status of operation:

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

### 6.4.2.59 rtJsonDecUInt32Value()

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

This function decodes the contents of an unsigned 32-bit integer data type. Input is expected to be a string of OSUT↔F8CHAR characters returned by a JSON parser.

#### Parameters

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

## Returns

Completion status of operation:

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

#### 6.4.2.60 rtJsonDecUInt64Value()

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

This function decodes the contents of an unsigned 64-bit integer data type. Input is expected to be a string of OSUTF8CHAR characters returned by a JSON parser.

##### Parameters

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

##### Returns

Completion status of operation:

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

#### 6.4.2.61 rtJsonDecUInt8Value()

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

This function decodes the contents of an unsigned 8-bit integer data type (i.e. a signed byte type in the range of 0 to 255). Input is expected to be a string of OSUTF8CHAR characters returned by a JSON parser.

##### Parameters

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

##### Returns

Completion status of operation:

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

#### 6.4.2.62 rtJsonDecXmlStringValue()

```
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.

#### 6.4.2.63 rtJsonGetElemIdx()

```
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.

### 6.4.2.64 rtJsonTryDecBool()

```
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)

### 6.4.2.65 rtJsonTryDecNull()

```
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)



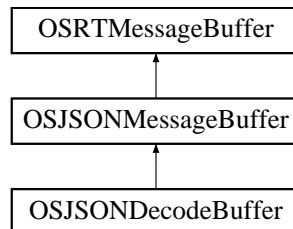
## Chapter 7

# Class Documentation

### 7.1 OSJSONDecodeBuffer Class Reference

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

Inheritance diagram for OSJSONDecodeBuffer:



#### Public Member Functions

- [OSJSONDecodeBuffer](#) (const char \*jsonFile)
- [OSJSONDecodeBuffer](#) (const OSOCTET \*msgbuf, size\_t bufsiz)
- [OSJSONDecodeBuffer](#) (OSRTInputStream &inputStream)
- virtual EXTJSONMETHOD int [init](#) ()
- virtual OSBOOL [isA](#) (Type bufferType)

#### Protected Attributes

- OSRTInputStream \* [mplInputStream](#)
- OSBOOL [mbOwnStream](#)

## 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.

### 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.

## 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.

### 7.1.4.2 `mpInputStream`

```
OSRTInputStream* OSJSONDecodeBuffer::mpInputStream [protected]
```

Input source for message to be decoded.

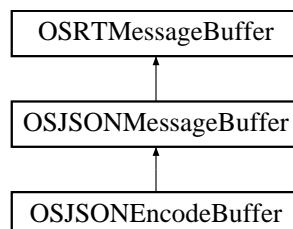
The documentation for this class was generated from the following file:

- [OSJSONDecodeBuffer.h](#)

## 7.2 OSJSONEncodeBuffer Class Reference

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

Inheritance diagram for OSJSONEncodeBuffer:



## Public Member Functions

- EXTJSONMETHOD [OSJSONEncodeBuffer](#) ()
- EXTJSONMETHOD [OSJSONEncodeBuffer](#) (OSOCKET \*pMsgBuf, size\_t msgBufLen)
- virtual size\_t [getMsgLen](#) ()
- virtual EXTJSONMETHOD int [init](#) ()
- virtual OSBOOL [isA](#) (Type bufferType)
- void [nullTerminate](#) ()
- virtual EXTJSONMETHOD long [write](#) (const char \*filename)
- virtual EXTJSONMETHOD long [write](#) (FILE \*fp)

## Protected Member Functions

- **OSJSONEncodeBuffer** (OSRTContext \*pContext)

### 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.

### 7.2.2 Constructor & Destructor Documentation

#### 7.2.2.1 OSJSONEncodeBuffer() [1/2]

```
EXTJSONMETHOD OSJSONEncodeBuffer::OSJSONEncodeBuffer ( )
```

Default constructor

#### 7.2.2.2 OSJSONEncodeBuffer() [2/2]

```
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.

### 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.



#### 7.2.3.4 nullTerminate()

```
void OSJSONEncodeBuffer::nullTerminate ( ) [inline]
```

This method adds a null-terminator character ('\0') at the current buffer position.

#### 7.2.3.5 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.

#### 7.2.3.6 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.

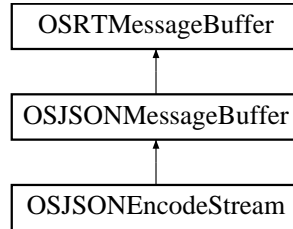
The documentation for this class was generated from the following file:

- [OSJSONEncodeBuffer.h](#)

## 7.3 OSJSONEncodeStream Class Reference

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

Inheritance diagram for OSJSONEncodeStream:



### Public Member Functions

- EXTJSONMETHOD [OSJSONEncodeStream](#) (OSRTOutputStream &outputStream)
- [OSJSONEncodeStream](#) (OSRTOutputStream \*pOutputStream, OSBOOL ownStream=TRUE)
- EXTJSONMETHOD int [encodeAttr](#) (const OSUTF8CHAR \*name, const OSUTF8CHAR \*value)
- EXTJSONMETHOD int [encodeText](#) (const OSUTF8CHAR \*value)
- virtual EXTJSONMETHOD int [init](#) ()
- virtual OSBOOL [isA](#) (Type bufferType)
- virtual const OSOCTET \* [getMsgPtr](#) ()
- OSRTOutputStream \* [getStream](#) () const

### Protected Attributes

- OSRTOutputStream \* [mpStream](#)
- OSBOOL [mbOwnStream](#)
- OSCTXT \* [mpCtxt](#)

### 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.

#### 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 OSOutputStream 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 OSOutputStream 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.

### 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.

### 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.

## 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.

### 7.3.4.2 mpCtxt

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

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

### 7.3.4.3 mpStream

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

A pointer to an OSRTOutputStream object.

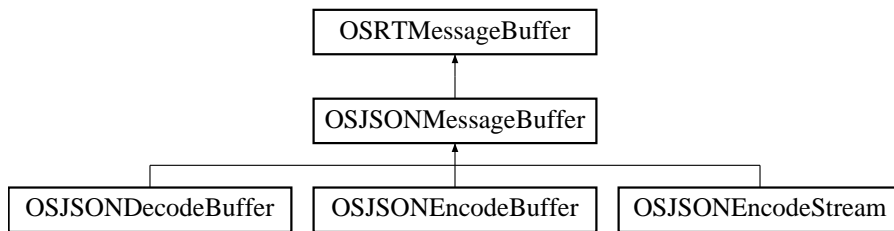
The documentation for this class was generated from the following file:

- [OSJSONEncodeStream.h](#)

## 7.4 OSJSONMessageBuffer Class Reference

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

Inheritance diagram for OSJSONMessageBuffer:



### Protected Member Functions

- EXTJSONMETHOD [OSJSONMessageBuffer](#) (Type bufferType, OSRTContext \*pContext=0)

### 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.

### 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 asn1json.h File Reference

```
#include "rtsrc/asn1type.h"  
#include "rtxsrc/osMacros.h"  
#include "rtxsrc/rtxCommonDefs.h"  
#include "rtxsrc/rtxError.h"  
#include "rtjsonsrc/rtJsonExternDefs.h"  
#include "osrtjson.h"
```

#### Functions

- int [rtJsonAsn1EncReal](#) (OSCTXT \*pctx, OSREAL value)
- int [rtJsonAsn1EncReal10](#) (OSCTXT \*pctx, const OSUTF8CHAR \*pvalue)
- int [rtJsonAsn1EncOID](#) (OSCTXT \*pctx, ASN1OBJID \*pOID)
- int [rtJsonAsn1EncOpenType](#) (OSCTXT \*pctx, OSOCTET \*pdata, OSSIZE numocts)
- int [rtJsonAsn1EncExtElem](#) (OSCTXT \*pctx, ASN1OpenType \*pElem)
- int [rtJsonAsn1EncOpenTypeExt](#) (OSCTXT \*pctx, const OSRTDList \*pvalue, OSBOOL asArray)
- int [rtJsonAsn1DecReal](#) (OSCTXT \*pctx, OSREAL \*pvalue)
- int [rtJsonAsn1DecReal10](#) (OSCTXT \*pctx, OSUTF8CHAR \*\*ppvalue)
- int [rtJsonAsn1DecOIDValue](#) (OSCTXT \*pctx, ASN1OBJID \*pOID)
- int [rtJsonAsn1DecBMPString](#) (OSCTXT \*pctx, ASN1BMPString \*pBMPStr)

#### 8.1.1 Detailed Description

JSON low-level C encode/decode functions for ASN.1.

### 8.2 OSJSONDecodeBuffer.h File Reference

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

## Classes

- class [OSJSONDecodeBuffer](#)

### 8.2.1 Detailed Description

JSON decode buffer or stream class definition.

## 8.3 OSJSONEncodeBuffer.h File Reference

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

## Classes

- class [OSJSONEncodeBuffer](#)

### 8.3.1 Detailed Description

JSON encode message buffer class definition.

## 8.4 OSJSONEncodeStream.h File Reference

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

## Classes

- class [OSJSONEncodeStream](#)

### 8.4.1 Detailed Description

JSON encode stream class definition.

## 8.5 OSJSONMessageBuffer.h File Reference

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

## Classes

- class [OSJSONMessageBuffer](#)

### 8.5.1 Detailed Description

JSON encode/decode buffer and stream base class.

## 8.6 osrtjson.h File Reference

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

## Macros

- #define **OSJSONNOWS** 0x00010000 /\* encode with NO WhiteSpace \*/
- #define **OSJSONATTR** 0x00008000 /\* encode string value as attribute \*/
- #define **OSJSONPUTCOMMA**(pctxt)
- #define **DEFAULT\_DOUBLE\_PRECISION** 11
- #define **DEFAULT\_FLOAT\_PRECISION** 6
- #define **OSUPCASE** 0x00008000
- #define **rtJsonDeclntValue** [rtJsonDeclnt32Value](#)
- #define **rtJsonDecUIntValue** [rtJsonDecUInt32Value](#)

## Enumerations

- enum **OSJSONState** {  
  **OSJSONNOCOMMA**, **OSJSONNORMAL**, **OSJSONSEQOF**, **OSJSONRAW**,  
  **OSJSONARRAYEDGE** }

## Functions

- `int rtJsonEncAnyAttr` (OSCTXT \*pctx, const OSRTDList \*pvalue)
- `int rtJsonEncIntValue` (OSCTXT \*pctx, OSINT32 value)
- `int rtJsonEncInt64Value` (OSCTXT \*pctx, OSINT64 value)
- `int rtJsonEncBase64StrValue` (OSCTXT \*pctx, OSSIZE noctx, const OSOCTET \*value)
- `int rtJsonEncBoolValue` (OSCTXT \*pctx, OSBOOL value)
- `int rtJsonEncGYear` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)
- `int rtJsonEncGYearMonth` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)
- `int rtJsonEncGMonth` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)
- `int rtJsonEncGMonthDay` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)
- `int rtJsonEncGDay` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)
- `int rtJsonEncDate` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)
- `int rtJsonEncTime` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)
- `int rtJsonEncDateTime` (OSCTXT \*pctx, const OSXSDDateTime \*pvalue)
- `int rtJsonEncDecimalValue` (OSCTXT \*pctx, OSREAL value, const OSDecimalFmt \*pFmtSpec)
- `int rtJsonEncDoubleValue` (OSCTXT \*pctx, OSREAL value, const OSDoubleFmt \*pFmtSpec)
- `int rtJsonEncFloatValue` (OSCTXT \*pctx, OSREAL value, const OSDoubleFmt \*pFmtSpec)
- `int rtJsonEncHexStr` (OSCTXT \*pctx, OSSIZE noctx, const OSOCTET \*data)
- `int rtJsonEncHexValue` (OSCTXT \*pctx, OSSIZE noctx, const OSOCTET \*data)
- `int rtJsonEncBitStrValueV72` (OSCTXT \*pctx, OSSIZE nbits, const OSOCTET \*data)
- `int rtJsonEncBitStrValueExtV72` (OSCTXT \*pctx, OSSIZE nbits, const OSOCTET \*data, OSSIZE dataSize, const OSOCTET \*extData)
- `int rtJsonEncBitStrValue` (OSCTXT \*pctx, OSSIZE nbits, const OSOCTET \*data)
- `int rtJsonEncFixedBitStrValue` (OSCTXT \*pctx, OSSIZE nbits, const OSOCTET \*data)
- `int rtJsonEncBitStrValueExt` (OSCTXT \*pctx, OSSIZE nbits, const OSOCTET \*data, OSSIZE dataSize, const OSOCTET \*extData)
- `int rtJsonEncIndent` (OSCTXT \*pctx)
- `void rtJsonEncDecrIndent` (OSCTXT \*pctx)
- `void rtJsonEncIncrIndent` (OSCTXT \*pctx)
- `void rtJsonEncResetIndent` (OSCTXT \*pctx)
- `size_t rtJsonGetIndentLevels` (OSCTXT \*pctx)
- `int rtJsonEncStringObject` (OSCTXT \*pctx, const OSUTF8CHAR \*name, const OSUTF8CHAR \*value)
- `int rtJsonEncStringObject2` (OSCTXT \*pctx, const OSUTF8CHAR \*name, size\_t nameLen, const OSUTF8CHAR \*value, size\_t valueLen)
- `int rtJsonEncStringPair` (OSCTXT \*pctx, const OSUTF8CHAR \*name, const OSUTF8CHAR \*value)
- `int rtJsonEncStringPair2` (OSCTXT \*pctx, const OSUTF8CHAR \*name, size\_t nameLen, const OSUTF8CHAR \*value, size\_t valueLen)
- `int rtJsonEncStringValue` (OSCTXT \*pctx, const OSUTF8CHAR \*value)
- `int rtJsonEncStringValue2` (OSCTXT \*pctx, const OSUTF8CHAR \*value, size\_t valueLen)
- `int rtJsonEncChars` (OSCTXT \*pctx, const char \*value, OSSIZE valueLen)
- `int rtJsonEncCharStr` (OSCTXT \*pctx, const char \*value)
- `int rtJsonEncStringNull` (OSCTXT \*pctx)
- `int rtJsonEncStringRaw` (OSCTXT \*pctx, const OSUTF8CHAR \*value)
- `int rtJsonEncUnicodeData` (OSCTXT \*pctx, const OSUNICHAR \*value, OSSIZE nchars)
- `int rtJsonEncUCS4Data` (OSCTXT \*pctx, const OS32BITCHAR \*value, OSSIZE nchars)
- `int rtJsonEncUIntValue` (OSCTXT \*pctx, OSUINT32 value)
- `int rtJsonEncUInt64Value` (OSCTXT \*pctx, OSUINT64 value)
- `int rtJsonEncStartObject` (OSCTXT \*pctx, const OSUTF8CHAR \*name, OSBOOL noComma)
- `int rtJsonEncEndObject` (OSCTXT \*pctx)
- `int rtJsonEncBetweenObject` (OSCTXT \*pctx)

- `int rtJsonDecAnyElem` (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)
- `int rtJsonDecAnyElem2` (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)
- `int rtJsonDecAnyType` (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)
- `int rtJsonDecBase64Str` (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, size\_t bufsize)
- `int rtJsonDecBase64Str64` (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocts, size\_t bufsize)
- `int rtJsonDecDynBase64Str` (OSCTXT \*pctxt, OSDynOctStr \*pvalue)
- `int rtJsonDecDynBase64Str64` (OSCTXT \*pctxt, OSDynOctStr64 \*pvalue)
- `int rtJsonDecBool` (OSCTXT \*pctxt, OSBOOL \*pvalue)
- `int rtJsonTryDecBool` (OSCTXT \*pctxt, OSBOOL \*pvalue)
- `int rtJsonDecDate` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- `int rtJsonDecTime` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- `int rtJsonDecDateTime` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- `int rtJsonDecGYear` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- `int rtJsonDecGYearMonth` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- `int rtJsonDecGMonth` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- `int rtJsonDecGMonthDay` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- `int rtJsonDecGDay` (OSCTXT \*pctxt, OSXSDDateTime \*pvalue)
- `int rtJsonDecDecimal` (OSCTXT \*pctxt, OSREAL \*pvalue, int totalDigits, int fractionDigits)
- `int rtJsonDecDouble` (OSCTXT \*pctxt, OSREAL \*pvalue)
- `int rtJsonDecHexToCharStr` (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)
- `int rtJsonDecHexStr` (OSCTXT \*pctxt, OSOCTET \*pvalue, OSUINT32 \*pnocts, size\_t bufsize)
- `int rtJsonDecHexStr64` (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocts, size\_t bufsize)
- `int rtJsonDecHexData64` (OSCTXT \*pctxt, OSOCTET \*pvalue, OSSIZE \*pnocts, size\_t bufsize)
- `int rtJsonDecDynHexStr` (OSCTXT \*pctxt, OSDynOctStr \*pvalue)
- `int rtJsonDecDynHexStr64` (OSCTXT \*pctxt, OSDynOctStr64 \*pvalue)
- `int rtJsonDecDynHexData64` (OSCTXT \*pctxt, OSDynOctStr64 \*pvalue)
- `int rtJsonDecDynBitStrV72` (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*\*data)
- `int rtJsonDecDynBitStr64V72` (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*\*data)
- `int rtJsonDecBitStrValueV72` (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize)
- `int rtJsonDecBitStrValue64V72` (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*data, OSSIZE bufsize)
- `int rtJsonDecBitStrValueExtV72` (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize, OSO←CTET \*\*extdata)
- `int rtJsonDecBitStrValueExt64V72` (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*data, OSSIZE bufsize, OSO←CTET \*\*extdata)
- `int rtJsonDecDynBitStr` (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*\*data)
- `int rtJsonDecDynBitStr64` (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*\*data)
- `int rtJsonDecFixedDynBitStr` (OSCTXT \*pctxt, OSSIZE nbits, OSOCTET \*\*data)
- `int rtJsonDecBitStrValue` (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize)
- `int rtJsonDecBitStrValue64` (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*data, OSSIZE bufsize)
- `int rtJsonDecFixedBitStrValue` (OSCTXT \*pctxt, OSSIZE nbits, OSOCTET \*data, OSSIZE bufsize)
- `int rtJsonDecBitStrValueExt` (OSCTXT \*pctxt, OSUINT32 \*nbits, OSOCTET \*data, OSSIZE bufsize, OSOCTET \*\*extdata)
- `int rtJsonDecBitStrValueExt64` (OSCTXT \*pctxt, OSSIZE \*nbits, OSOCTET \*data, OSSIZE bufsize, OSOCTET \*\*extdata)
- `int rtJsonDecInt8Value` (OSCTXT \*pctxt, OSINT8 \*pvalue)
- `int rtJsonDecInt16Value` (OSCTXT \*pctxt, OSINT16 \*pvalue)
- `int rtJsonDecInt32Value` (OSCTXT \*pctxt, OSINT32 \*pvalue)
- `int rtJsonDecInt64Value` (OSCTXT \*pctxt, OSINT64 \*pvalue)
- `int rtJsonDecUInt8Value` (OSCTXT \*pctxt, OSUINT8 \*pvalue)
- `int rtJsonDecUInt16Value` (OSCTXT \*pctxt, OSUINT16 \*pvalue)
- `int rtJsonDecUInt32Value` (OSCTXT \*pctxt, OSUINT32 \*pvalue)

- `int rtJsonDecUInt64Value` (OSCTXT \*pctxt, OSUINT64 \*pvalue)
- `int rtJsonDecMatchChar` (OSCTXT \*pctxt, OSUTF8CHAR ch)
- `int rtJsonDecMatchCharStr` (OSCTXT \*pctxt, const char \*token)
- `int rtJsonDecMatchObjectStart` (OSCTXT \*pctxt, const OSUTF8NameAndLen \*nameArray, size\_t numNames)
- `int rtJsonDecMatchToken` (OSCTXT \*pctxt, const OSUTF8CHAR \*token)
- `int rtJsonDecMatchToken2` (OSCTXT \*pctxt, const OSUTF8CHAR \*token, size\_t tokenLen)
- `int rtJsonDecNameValuePair` (OSCTXT \*pctxt, OSUTF8NVP \*pvalue)
- `int rtJsonDecNull` (OSCTXT \*pctxt)
- `int rtJsonTryDecNull` (OSCTXT \*pctxt)
- `int rtJsonDecNumberString` (OSCTXT \*pctxt, char \*\*ppCharStr)
- `int rtJsonDecPeekChar` (OSCTXT \*pctxt, OSUTF8CHAR \*pch)
- `char rtJsonDecPeekChar2` (OSCTXT \*pctxt)
- `int rtJsonDecStringObject` (OSCTXT \*pctxt, const OSUTF8CHAR \*name, OSUTF8CHAR \*\*ppvalue)
- `int rtJsonDecStringValue` (OSCTXT \*pctxt, OSUTF8CHAR \*\*ppvalue)
- `int rtJsonDecXmlStringValue` (OSCTXT \*pctxt, OSXMLSTRING \*pvalue)
- `int rtJsonDecUCS2String` (OSCTXT \*pctxt, OSUNICHAR \*\*ppstr, OSSIZE \*pnchars)
- `int rtJsonDecUCS4String` (OSCTXT \*pctxt, OS32BITCHAR \*\*ppstr, OSSIZE \*pnchars)
- `int rtJsonDecSkipWhitespace` (OSCTXT \*pctxt)
- `size_t rtJsonGetElemIdx` (OSCTXT \*pctxt, const OSUTF8NameAndLen nameArray[], size\_t nrows)

## 8.6.1 Detailed Description

JSON low-level C encode/decode functions.

## 8.6.2 Macro Definition Documentation

### 8.6.2.1 OSJSONPUTCOMMA

```
#define OSJSONPUTCOMMA(  
    pctxt )
```

#### Value:

```
if ((pctxt)->lastChar != '{' && (pctxt)->lastChar != ',' \
    && (pctxt)->lastChar != '\0' && (pctxt)->lastChar != '[') \
    OSRTSAFEPUTCHAR (pctxt, ',');
```

### 8.6.2.2 OSUPCASE

```
#define OSUPCASE 0x00008000
```

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

## 8.7 rtJsonCppMsgBuf.h File Reference

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

### 8.7.1 Detailed Description

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

## 8.8 rtJsonExternDefs.h File Reference

### 8.8.1 Detailed Description

JSON external definitions macro. This is used for Windows to properly declare function scope within DLL's.





# Index

- asn1json.h, [93](#)
  
- encodeAttr
  - [OSJSONEncodeStream](#), [87](#)
- encodeText
  - [OSJSONEncodeStream](#), [87](#)
  
- getMsgLen
  - [OSJSONEncodeBuffer](#), [84](#)
- getMsgPtr
  - [OSJSONEncodeStream](#), [88](#)
- getStream
  - [OSJSONEncodeStream](#), [88](#)
  
- init
  - [OSJSONDecodeBuffer](#), [81](#)
  - [OSJSONEncodeBuffer](#), [84](#)
  - [OSJSONEncodeStream](#), [88](#)
  
- isA
  - [OSJSONDecodeBuffer](#), [81](#)
  - [OSJSONEncodeBuffer](#), [84](#)
  - [OSJSONEncodeStream](#), [89](#)
  
- JSON decode functions., [40](#)
  - [rtJsonDecAnyElem](#), [41](#)
  - [rtJsonDecAnyElem2](#), [42](#)
  - [rtJsonDecAnyType](#), [42](#)
  - [rtJsonDecBase64Str](#), [43](#)
  - [rtJsonDecBase64Str64](#), [43](#)
  - [rtJsonDecBitStrValue](#), [44](#)
  - [rtJsonDecBitStrValue64](#), [45](#)
  - [rtJsonDecBitStrValue64V72](#), [45](#)
  - [rtJsonDecBitStrValueExt](#), [46](#)
  - [rtJsonDecBitStrValueExt64](#), [47](#)
  - [rtJsonDecBitStrValueExt64V72](#), [47](#)
  - [rtJsonDecBitStrValueExtV72](#), [48](#)
  - [rtJsonDecBitStrValueV72](#), [49](#)
  - [rtJsonDecBool](#), [49](#)
  - [rtJsonDecDate](#), [50](#)
  - [rtJsonDecDateTime](#), [50](#)
  - [rtJsonDecDecimal](#), [51](#)
  - [rtJsonDecDouble](#), [51](#)
  - [rtJsonDecDynBase64Str](#), [52](#)
  - [rtJsonDecDynBase64Str64](#), [52](#)
  - [rtJsonDecDynBitStr](#), [53](#)
  - [rtJsonDecDynBitStr64](#), [53](#)
  - [rtJsonDecDynBitStr64V72](#), [54](#)
  - [rtJsonDecDynBitStrV72](#), [54](#)
  - [rtJsonDecDynHexData64](#), [55](#)
  - [rtJsonDecDynHexStr](#), [55](#)
  - [rtJsonDecDynHexStr64](#), [56](#)
  - [rtJsonDecFixedBitStrValue](#), [56](#)
  - [rtJsonDecFixedDynBitStr](#), [58](#)
  - [rtJsonDecGDay](#), [58](#)
  - [rtJsonDecGMonth](#), [59](#)
  - [rtJsonDecGMonthDay](#), [59](#)
  - [rtJsonDecGYear](#), [60](#)
  - [rtJsonDecGYearMonth](#), [60](#)
  - [rtJsonDecHexData64](#), [61](#)
  - [rtJsonDecHexStr](#), [61](#)
  - [rtJsonDecHexStr64](#), [62](#)
  - [rtJsonDecHexToCharStr](#), [63](#)
  - [rtJsonDecInt16Value](#), [63](#)
  - [rtJsonDecInt32Value](#), [64](#)
  - [rtJsonDecInt64Value](#), [64](#)
  - [rtJsonDecInt8Value](#), [65](#)
  - [rtJsonDecMatchChar](#), [65](#)
  - [rtJsonDecMatchCharStr](#), [66](#)
  - [rtJsonDecMatchObjectStart](#), [66](#)
  - [rtJsonDecMatchToken](#), [67](#)
  - [rtJsonDecMatchToken2](#), [67](#)
  - [rtJsonDecNameValuePair](#), [68](#)
  - [rtJsonDecNull](#), [68](#)
  - [rtJsonDecNumberString](#), [69](#)
  - [rtJsonDecPeekChar](#), [69](#)
  - [rtJsonDecPeekChar2](#), [70](#)
  - [rtJsonDecStringObject](#), [70](#)
  - [rtJsonDecStringValue](#), [71](#)
  - [rtJsonDecTime](#), [71](#)
  - [rtJsonDecUCS2String](#), [72](#)
  - [rtJsonDecUCS4String](#), [72](#)
  - [rtJsonDecUInt16Value](#), [73](#)
  - [rtJsonDecUInt32Value](#), [73](#)
  - [rtJsonDecUInt64Value](#), [74](#)
  - [rtJsonDecUInt8Value](#), [74](#)
  - [rtJsonDecXmlStringValue](#), [75](#)
  - [rtJsonGetElemIdx](#), [75](#)
  - [rtJsonTryDecBool](#), [76](#)
  - [rtJsonTryDecNull](#), [76](#)
  
- JSON encode functions., [17](#)
  - [rtJsonEncAnyAttr](#), [18](#)

- rtJsonEncBase64StrValue, [18](#)
- rtJsonEncBetweenObject, [19](#)
- rtJsonEncBitStrValue, [19](#)
- rtJsonEncBitStrValueExt, [20](#)
- rtJsonEncBitStrValueExtV72, [20](#)
- rtJsonEncBitStrValueV72, [21](#)
- rtJsonEncBoolValue, [21](#)
- rtJsonEncCharStr, [22](#)
- rtJsonEncChars, [22](#)
- rtJsonEncDate, [23](#)
- rtJsonEncDateTime, [23](#)
- rtJsonEncDecimalValue, [24](#)
- rtJsonEncDecrIndent, [24](#)
- rtJsonEncDoubleValue, [25](#)
- rtJsonEncEndObject, [25](#)
- rtJsonEncFixedBitStrValue, [26](#)
- rtJsonEncFloatValue, [26](#)
- rtJsonEncGDay, [27](#)
- rtJsonEncGMonth, [27](#)
- rtJsonEncGMonthDay, [28](#)
- rtJsonEncGYear, [28](#)
- rtJsonEncGYearMonth, [29](#)
- rtJsonEncHexStr, [29](#)
- rtJsonEncHexValue, [29](#)
- rtJsonEncIncrIndent, [30](#)
- rtJsonEncIndent, [30](#)
- rtJsonEncInt64Value, [31](#)
- rtJsonEncIntValue, [31](#)
- rtJsonEncResetIndent, [32](#)
- rtJsonEncStartObject, [32](#)
- rtJsonEncStringNull, [32](#)
- rtJsonEncStringObject, [33](#)
- rtJsonEncStringObject2, [33](#)
- rtJsonEncStringPair, [34](#)
- rtJsonEncStringPair2, [34](#)
- rtJsonEncStringRaw, [35](#)
- rtJsonEncStringValue, [35](#)
- rtJsonEncStringValue2, [36](#)
- rtJsonEncTime, [36](#)
- rtJsonEncUCS4Data, [37](#)
- rtJsonEncUInt64Value, [37](#)
- rtJsonEncUIntValue, [38](#)
- rtJsonEncUnicodeData, [38](#)
- rtJsonGetIndentLevels, [39](#)
- JSON runtime decoding functions for ASN.1 data types., [14](#)
  - rtJsonAsn1DecBMPString, [14](#)
  - rtJsonAsn1DecOIDValue, [14](#)
  - rtJsonAsn1DecReal, [15](#)
  - rtJsonAsn1DecReal10, [15](#)
- JSON runtime encoding functions for ASN.1 data types., [11](#)
  - rtJsonAsn1EncExtElem, [11](#)
  - rtJsonAsn1EncOID, [12](#)
  - rtJsonAsn1EncOpenType, [12](#)
  - rtJsonAsn1EncOpenTypeExt, [12](#)
  - rtJsonAsn1EncReal, [13](#)
  - rtJsonAsn1EncReal10, [13](#)
- mbOwnStream
  - OSJSONDecodeBuffer, [82](#)
  - OSJSONEncodeStream, [89](#)
- mpCtxt
  - OSJSONEncodeStream, [89](#)
- mpInputStream
  - OSJSONDecodeBuffer, [82](#)
- mpStream
  - OSJSONEncodeStream, [89](#)
- nullTerminate
  - OSJSONEncodeBuffer, [84](#)
- OSJSONDecodeBuffer, [79](#)
  - init, [81](#)
  - isA, [81](#)
  - mbOwnStream, [82](#)
  - mpInputStream, [82](#)
  - OSJSONDecodeBuffer, [80](#)
- OSJSONDecodeBuffer.h, [93](#)
- OSJSONEncodeBuffer, [82](#)
  - getMsgLen, [84](#)
  - init, [84](#)
  - isA, [84](#)
  - nullTerminate, [84](#)
  - OSJSONEncodeBuffer, [83](#)
  - write, [85](#)
- OSJSONEncodeBuffer.h, [94](#)
- OSJSONEncodeStream, [86](#)
  - encodeAttr, [87](#)
  - encodeText, [87](#)
  - getMsgPtr, [88](#)
  - getStream, [88](#)
  - init, [88](#)
  - isA, [89](#)
  - mbOwnStream, [89](#)
  - mpCtxt, [89](#)
  - mpStream, [89](#)
  - OSJSONEncodeStream, [86, 87](#)
- OSJSONEncodeStream.h, [94](#)
- OSJSONMessageBuffer, [90](#)
  - OSJSONMessageBuffer, [90](#)
- OSJSONMessageBuffer.h, [94](#)
- OSJSONPUTCOMMA
  - osrtjson.h, [98](#)
- OSUPCASE
  - osrtjson.h, [98](#)
- osrtjson.h, [95](#)
  - OSJSONPUTCOMMA, [98](#)
  - OSUPCASE, [98](#)

rt.JsonAsn1DecBMPString  
     JSON runtime decoding functions for ASN.1 data types., 14

rt.JsonAsn1DecOIDValue  
     JSON runtime decoding functions for ASN.1 data types., 14

rt.JsonAsn1DecReal  
     JSON runtime decoding functions for ASN.1 data types., 15

rt.JsonAsn1DecReal10  
     JSON runtime decoding functions for ASN.1 data types., 15

rt.JsonAsn1EncExtElem  
     JSON runtime encoding functions for ASN.1 data types., 11

rt.JsonAsn1EncOID  
     JSON runtime encoding functions for ASN.1 data types., 12

rt.JsonAsn1EncOpenType  
     JSON runtime encoding functions for ASN.1 data types., 12

rt.JsonAsn1EncOpenTypeExt  
     JSON runtime encoding functions for ASN.1 data types., 12

rt.JsonAsn1EncReal  
     JSON runtime encoding functions for ASN.1 data types., 13

rt.JsonAsn1EncReal10  
     JSON runtime encoding functions for ASN.1 data types., 13

rt.JsonCppMsgBuf.h, 99

rt.JsonDecAnyElem  
     JSON decode functions., 41

rt.JsonDecAnyElem2  
     JSON decode functions., 42

rt.JsonDecAnyType  
     JSON decode functions., 42

rt.JsonDecBase64Str  
     JSON decode functions., 43

rt.JsonDecBase64Str64  
     JSON decode functions., 43

rt.JsonDecBitStrValue  
     JSON decode functions., 44

rt.JsonDecBitStrValue64  
     JSON decode functions., 45

rt.JsonDecBitStrValue64V72  
     JSON decode functions., 45

rt.JsonDecBitStrValueExt  
     JSON decode functions., 46

rt.JsonDecBitStrValueExt64  
     JSON decode functions., 47

rt.JsonDecBitStrValueExt64V72  
     JSON decode functions., 47

rt.JsonDecBitStrValueExtV72  
     JSON decode functions., 48

rt.JsonDecBitStrValueV72  
     JSON decode functions., 49

rt.JsonDecBool  
     JSON decode functions., 49

rt.JsonDecDate  
     JSON decode functions., 50

rt.JsonDecDateTime  
     JSON decode functions., 50

rt.JsonDecDecimal  
     JSON decode functions., 51

rt.JsonDecDouble  
     JSON decode functions., 51

rt.JsonDecDynBase64Str  
     JSON decode functions., 52

rt.JsonDecDynBase64Str64  
     JSON decode functions., 52

rt.JsonDecDynBitStr  
     JSON decode functions., 53

rt.JsonDecDynBitStr64  
     JSON decode functions., 53

rt.JsonDecDynBitStr64V72  
     JSON decode functions., 54

rt.JsonDecDynBitStrV72  
     JSON decode functions., 54

rt.JsonDecDynHexData64  
     JSON decode functions., 55

rt.JsonDecDynHexStr  
     JSON decode functions., 55

rt.JsonDecDynHexStr64  
     JSON decode functions., 56

rt.JsonDecFixedBitStrValue  
     JSON decode functions., 56

rt.JsonDecFixedDynBitStr  
     JSON decode functions., 58

rt.JsonDecGDay  
     JSON decode functions., 58

rt.JsonDecGMonth  
     JSON decode functions., 59

rt.JsonDecGMonthDay  
     JSON decode functions., 59

rt.JsonDecGYear  
     JSON decode functions., 60

rt.JsonDecGYearMonth  
     JSON decode functions., 60

rt.JsonDecHexData64  
     JSON decode functions., 61

rt.JsonDecHexStr  
     JSON decode functions., 61

rt.JsonDecHexStr64  
     JSON decode functions., 62

rt.JsonDecHexToCharStr  
     JSON decode functions., 63

rt.JsonDeclnt16Value

- JSON decode functions., 63
- rt.JsonDecInt32Value
  - JSON decode functions., 64
- rt.JsonDecInt64Value
  - JSON decode functions., 64
- rt.JsonDecInt8Value
  - JSON decode functions., 65
- rt.JsonDecMatchChar
  - JSON decode functions., 65
- rt.JsonDecMatchCharStr
  - JSON decode functions., 66
- rt.JsonDecMatchObjectStart
  - JSON decode functions., 66
- rt.JsonDecMatchToken
  - JSON decode functions., 67
- rt.JsonDecMatchToken2
  - JSON decode functions., 67
- rt.JsonDecNameValuePair
  - JSON decode functions., 68
- rt.JsonDecNull
  - JSON decode functions., 68
- rt.JsonDecNumberString
  - JSON decode functions., 69
- rt.JsonDecPeekChar
  - JSON decode functions., 69
- rt.JsonDecPeekChar2
  - JSON decode functions., 70
- rt.JsonDecStringObject
  - JSON decode functions., 70
- rt.JsonDecStringValue
  - JSON decode functions., 71
- rt.JsonDecTime
  - JSON decode functions., 71
- rt.JsonDecUCS2String
  - JSON decode functions., 72
- rt.JsonDecUCS4String
  - JSON decode functions., 72
- rt.JsonDecUInt16Value
  - JSON decode functions., 73
- rt.JsonDecUInt32Value
  - JSON decode functions., 73
- rt.JsonDecUInt64Value
  - JSON decode functions., 74
- rt.JsonDecUInt8Value
  - JSON decode functions., 74
- rt.JsonDecXmlStringValue
  - JSON decode functions., 75
- rt.JsonEncAnyAttr
  - JSON encode functions., 18
- rt.JsonEncBase64StrValue
  - JSON encode functions., 18
- rt.JsonEncBetweenObject
  - JSON encode functions., 19
- rt.JsonEncBitStrValue

- JSON encode functions., 19
- rt.JsonEncBitStrValueExt
  - JSON encode functions., 20
- rt.JsonEncBitStrValueExtV72
  - JSON encode functions., 20
- rt.JsonEncBitStrValueV72
  - JSON encode functions., 21
- rt.JsonEncBoolValue
  - JSON encode functions., 21
- rt.JsonEncCharStr
  - JSON encode functions., 22
- rt.JsonEncChars
  - JSON encode functions., 22
- rt.JsonEncDate
  - JSON encode functions., 23
- rt.JsonEncDateTime
  - JSON encode functions., 23
- rt.JsonEncDecimalValue
  - JSON encode functions., 24
- rt.JsonEncDecrIndent
  - JSON encode functions., 24
- rt.JsonEncDoubleValue
  - JSON encode functions., 25
- rt.JsonEncEndObject
  - JSON encode functions., 25
- rt.JsonEncFixedBitStrValue
  - JSON encode functions., 26
- rt.JsonEncFloatValue
  - JSON encode functions., 26
- rt.JsonEncGDay
  - JSON encode functions., 27
- rt.JsonEncGMonth
  - JSON encode functions., 27
- rt.JsonEncGMonthDay
  - JSON encode functions., 28
- rt.JsonEncGYear
  - JSON encode functions., 28
- rt.JsonEncGYearMonth
  - JSON encode functions., 29
- rt.JsonEncHexStr
  - JSON encode functions., 29
- rt.JsonEncHexValue
  - JSON encode functions., 29
- rt.JsonEncIncrIndent
  - JSON encode functions., 30
- rt.JsonEncIndent
  - JSON encode functions., 30
- rt.JsonEncInt64Value
  - JSON encode functions., 31
- rt.JsonEncIntValue
  - JSON encode functions., 31
- rt.JsonEncResetIndent
  - JSON encode functions., 32
- rt.JsonEncStartObject

- JSON encode functions., [32](#)
- rt.JsonEncStringNull
  - JSON encode functions., [32](#)
- rt.JsonEncStringObject
  - JSON encode functions., [33](#)
- rt.JsonEncStringObject2
  - JSON encode functions., [33](#)
- rt.JsonEncStringPair
  - JSON encode functions., [34](#)
- rt.JsonEncStringPair2
  - JSON encode functions., [34](#)
- rt.JsonEncStringRaw
  - JSON encode functions., [35](#)
- rt.JsonEncStringValue
  - JSON encode functions., [35](#)
- rt.JsonEncStringValue2
  - JSON encode functions., [36](#)
- rt.JsonEncTime
  - JSON encode functions., [36](#)
- rt.JsonEncUCS4Data
  - JSON encode functions., [37](#)
- rt.JsonEncUInt64Value
  - JSON encode functions., [37](#)
- rt.JsonEncUIntValue
  - JSON encode functions., [38](#)
- rt.JsonEncUnicodeData
  - JSON encode functions., [38](#)
- rt.JsonExternDefs.h, [99](#)
- rt.JsonGetElemIdx
  - JSON decode functions., [75](#)
- rt.JsonGetIndentLevels
  - JSON encode functions., [39](#)
- rt.JsonTryDecBool
  - JSON decode functions., [76](#)
- rt.JsonTryDecNull
  - JSON decode functions., [76](#)
- write
  - OSJSONEncodeBuffer, [85](#)