XBinder JSON Runtime

Version 3.0

Objective Systems, Inc.

December 2024

XBinder JSON Runtime

Author's Contact Information. Comments, suggestions, and inquiries regarding XBinder or this document may be sent by electronic mail to `<info@obj-sys.com>`.

Chapter 1. C JSON Runtime Library Functions

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

Chapter 2. Module Documentation

JSON encode functions.

Detailed Description

Functions

- int rtJsonEncAnyAttr (OSCTXT * pctxt, const OSRTDList * pvalue)
- int rtJsonEncBase64StrValue (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 rtJsonEncBitStrValueV72 (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data)
- int rtJsonEncBitStrValueExtV72 (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data, OSSIZE dataSize, const OSOCTET * extData)
- int rtJsonEncBitStrValue (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data)
- int rtJsonEncFixedBitStrValue (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data)
- int rtJsonEncBitStrValueExt (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data, OSSIZE dataSize, const OSOCTET * extData)
- int rtJsonEncIndent (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 rtJsonEncStartObject (OSCTXT * pctxt, const OSUTF8CHAR * name, OSBOOL noComma)
- int rtJsonEncEndObject (OSCTXT * pctxt)
- int rtJsonEncBetweenObject (OSCTXT * pctxt)

Macros

- #define rtJsonEncIntValue rtxTxtWriteInt
- #define rtJsonEncInt64Value rtxTxtWriteInt64
- #define rtJsonEncDecrIndent rtxIndentDecr
- #define rtJsonEncIncrIndent rtxIndentIncr
- #define rtJsonEncResetIndent rtxIndentReset
- #define rtJsonGetIndentLevels rtxGetIndentLevels
- #define rtJsonEncUIntValue rtxTxtWriteUInt
- #define rtJsonEncUInt64Value rtxTxtWriteUInt64

Function Documentation

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.

Table 2.1. Parameters

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

Returns: . Completion status of operation:

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

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

This function encodes a variable of the XSD base64Binary type.

Table 2.2. Parameters

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

Returns: . Completion status of operation:

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

int rtJsonEncBoolValue (OSCTXT *pctxt, OSBOOL value)

This function encodes a variable of the XSD boolean type.

Table 2.3. Parameters

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

Returns: . Completion status of operation:

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

int rtJsonEncGYear (OSCTXT *pctxt, const OSXSDDateTime *pval- ue)

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

Table 2.4. Parameters

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

Returns: . Completion status of operation:

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

int rtJsonEncGYearMonth (OSCTXT *pctxt, const OSXSDDateTime *pvalue)

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

Table 2.5. Parameters

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

Returns: . Completion status of operation:

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

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

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

Table 2.6. Parameters

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

Returns: . Completion status of operation:

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

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

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

Table 2.7. Parameters

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

Returns: . Completion status of operation:

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

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

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

Table 2.8. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.9. Parameters

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

Returns: . Completion status of operation:

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

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

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

Table 2.10. Parameters

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

Returns: . Completion status of operation:

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

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

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

Table 2.11. Parameters

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

Returns: . Completion status of operation:

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

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

This function encodes a value of the XSD decimal type.

Table 2.12. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
pFmtSpec	Pointer to format specification structure.

Returns: . Completion status of operation:

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

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

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

Table 2.13. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
pFmtSpec	Pointer to format specification structure.

Returns: . Completion status of operation:

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

int rtJsonEncFloatValue (OSCTXT *pctxt, OSREAL value, const OS-DoubleFmt *pFmtSpec)

This function encodes a variable of the XSD float type.

Table 2.14. Parameters

pctxt	Pointer to context block structure.
value	Value to be encoded.
pFmtSpec	Pointer to format specification structure.

Returns: . Completion status of operation:

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

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.

Table 2.15. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.16. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.17. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.18. Parameters

pctxt	Pointer to context block structure.

nbits	Number of bits in the value string.
data	Value to be encoded.
dataSize	Size of data member.
extData	Value of extdata to be encoded.

Returns: . Completion status of operation:

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

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.

Table 2.19. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.20. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.21. Parameters

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

Returns: . Completion status of operation:

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

int rtJsonEncIndent (OSCTXT *pctxt)

This function:

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

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

Table 2.22. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

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

This function encodes a JSON object containing a string value.

Table 2.23. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.24. Parameters

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

Returns: . Completion status of operation:

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

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

This function encodes a name/value pair. The value is a character string.

Table 2.25. Parameters

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

Returns: . Completion status of operation:

- 0 = success,

- negative return value is error.

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.

Table 2.26. Parameters

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

Returns: . Completion status of operation:

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

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

This function encodes a UTF8 string value.

Table 2.27. Parameters

pctxt	Pointer to context block structure.
value	XML string value to be encoded.

Returns: . Completion status of operation:

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

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

This function encodes a UTF8 string value.

Table 2.28. Parameters

pctxt	Pointer to context block structure.

value	XML string value to be encoded.
valueLen	Length of the XML string to be encoded.

Returns: . Completion status of operation:

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

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.

Table 2.29. Parameters

pctxt	Pointer to context block structure.
value	Character string value to be encoded.
valueLen	Length of the XML string to be encoded.

Returns: . Completion status of operation:

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

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

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

Table 2.30. Parameters

pctxt	Pointer to context block structure.
value	Character string value to be encoded.

Returns: . Completion status of operation:

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

int rtJsonEncStringNull (OSCTXT *pctxt)

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

Table 2.31. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

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

This function encodes a raw string without any quotation.

Table 2.32. Parameters

pctxt	Pointer to context block structure.
value	String value to be written.

Returns: . Completion status of operation:

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

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

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

Table 2.33. Parameters

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

Returns: . Completion status of operation:

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

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

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

Table 2.34. Parameters

pctxt	Pointer to context block structure.

value	UCS-4 characters to be encoded.
nchars	Number of characters to be encoded.

Returns: . Completion status of operation:

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

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

This function encodes the beginning of a JSON object.

Table 2.35. Parameters

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

Returns: . Completion status of operation:

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

int rtJsonEncEndObject (OSCTXT *pctxt)

This function encodes the end of a JSON object.

Table 2.36. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

int rtJsonEncBetweenObject (OSCTXT *pctxt)

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

Table 2.37. Parameters

pctxt	Pointer to context block structure.

Returns: . Completion status of operation:

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

Macro Definition Documentation

#define rtJsonEncIntValue

This function encodes a variable of the XSD integer type.

Table 2.38. Parameters

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

Returns: . Completion status of operation:

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

Definition at line 118 of file osrtjson.h

The Documentation for this define was generated from the following file:

- osrtjson.h

#define rtJsonEncInt64Value

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

Table 2.39. Parameters

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

Returns: . Completion status of operation:

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

Definition at line 130 of file osrtjson.h

The Documentation for this define was generated from the following file:

- osrtjson.h

#define rtJsonEncDecrIndent

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

Table 2.40. Parameters

pctxt	Pointer to context block structure.

Definition at line 423 of file osrtjson.h

The Documentation for this define was generated from the following file:

- osrtjson.h

#define rtJsonEncIncrIndent

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

Table 2.41. Parameters

pctxt	Pointer to context block structure.

Definition at line 431 of file osrtjson.h

The Documentation for this define was generated from the following file:

- osrtjson.h

#define rtJsonEncResetIndent

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

Table 2.42. Parameters

pctxt	Pointer to context block structure.

Definition at line 438 of file osrtjson.h

The Documentation for this define was generated from the following file:

- osrtjson.h

#define rtJsonGetIndentLevels

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

Table 2.43. Parameters

pctxt	Pointer to context block structure.

Definition at line 447 of file osrtjson.h

The Documentation for this define was generated from the following file:

- osrtjson.h

#define rtJsonEncUIntValue

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

Table 2.44. Parameters

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

Returns: . Completion status of operation:

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

Definition at line 614 of file osrtjson.h

The Documentation for this define was generated from the following file:

- osrtjson.h

#define rtJsonEncUInt64Value

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.

Table 2.45. Parameters

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

Returns: . Completion status of operation:

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

Definition at line 627 of file osrtjson.h

The Documentation for this define was generated from the following file:

- osrtjson.h

JSON decode functions.

Detailed Description

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 * 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, OSOCTET ** extdata)
- int rtJsonDecBitStrValueExt64V72 (OSCTXT * pctxt, OSSIZE * nbits, OSOCTET * data, OSSIZE bufsize, OSOCTET ** 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 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 rtJsonDecStringObject (OSCTXT * pctxt, const OSUTF8CHAR * name, OSUTF8CHAR ** ppvalue)
- int rtJsonDecStringValue (OSCTXT * pctxt, OSUTF8CHAR ** ppvalue)

- int rtJsonDecStringValueArray (OSCTXT * pctxt, OSUTF8CHAR * pvalue, OSSIZE bufsize)
- int rtJsonDecXmlStringValue (OSCTXT * pctxt, OSXMLSTRING * pvalue)
- int rtJsonDecUCS2String (OSCTXT * pctxt, OSUNICHAR ** ppstr, OSSIZE * pnchars)
- int rtJsonDecUCS4String (OSCTXT * pctxt, OS32BITCHAR ** ppstr, OSSIZE * pnchars)
- size_t rtJsonGetElemIdx (OSCTXT * pctxt, const OSUTF8NameAndLen nameArray, size_t nrows)

Macros

- #define rtJsonDecInt8Value rtxTxtReadInt8
- #define rtJsonDecInt16Value rtxTxtReadInt16
- #define rtJsonDecInt32Value rtxTxtReadInt32
- #define rtJsonDecIntValue rtxTxtReadInt32
- #define rtJsonDecInt64Value rtxTxtReadInt64
- #define rtJsonDecUInt8Value rtxTxtReadUInt8
- #define rtJsonDecUInt16Value rtxTxtReadUInt16
- #define rtJsonDecUInt32Value rtxTxtReadUInt32
- #define rtJsonDecUIntValue rtxTxtReadUInt32
- #define rtJsonDecUInt64Value rtxTxtReadUInt64
- #define rtJsonDecPeekChar rtxTxtPeekChar(pctxt, pch, TRUE)
- #define rtJsonDecPeekChar2 rtxTxtPeekChar2(pctxt, TRUE)
- #define rtJsonDecSkipWhitespace rtxTxtSkipWhitespace

Function Documentation

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.

Table 2.46. Parameters

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

Returns: . Completion status of operation:

- 0 = success,

- negative return value is error.

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.

Table 2.47. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.48. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
ppvalue	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.

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.

Table 2.49. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.

pvalue	A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.
pnocts	A pointer to an integer value to receive the decoded number of octets.
bufsize	The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

Returns: . Completion status of operation:

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

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.

Table 2.50. Parameters

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

Returns: . Completion status of operation:

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

See also: . rtJsonDecBase64Str

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.

Table 2.51. Parameters

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

Returns: . Completion status of operation:

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

int rtJsonDecDynBase64Str64 (OSCTXT *pctxt, OSDynOctStr64 *pvalue)

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

Table 2.52. Parameters

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

Returns: . Completion status of operation:

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

int rtJsonDecBool (OSCTXT *pctxt, OSBOOL *pvalue)

This function decodes a variable of the boolean type.

Table 2.53. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to a variable to receive the decoded boolean value.

Returns: . Completion status of operation:

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

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.

Table 2.54. Parameters

pctxt	Pointer to context block structure.

pvalue	Pointer to a variable to receive the decoded boolean value.

Returns: . Completion status of operation:

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

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.

Table 2.55. Parameters

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

Returns: . Completion status of operation:

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

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
```

Table 2.56. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.57. Parameters

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

Returns: . Completion status of operation:

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

int rtJsonDecGYear (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'gYear' type. Input is expected to be a string of characters returned by a JSON parser. The string should have CCYY[-+hh:mm|Z] format.

Table 2.58. Parameters

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

Returns: . Completion status of operation:

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

int rtJsonDecGYearMonth (OSCTXT *pctxt, OSXSDDateTime *pvalue)

This function decodes a variable of the XSD 'gYearMonth' type. Input is expected to be a string of characters returned by a JSON parser. The string should have CCYY-MM[-+hh:mm|Z] format.

Table 2.59. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.60. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.61. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.62. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.63. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.64. Parameters

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

Returns: . Completion status of operation:

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

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

This function decodes a JSON string, interpreting the content as the hexadecimal represention 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.

Table 2.65. Parameters

pctxt	Pointer to context block structure.
ppvalue	Pointer to an 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.

int rtJsonDecHexStr (OSCTXT *pctxt, OSOCTET *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.

Table 2.66. Parameters

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

Returns: . Completion status of operation:

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

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

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

Table 2.67. Parameters

pctxt	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
pvalue	A pointer to a variable to receive the decoded bit string. This is assumed to be a static array large enough to hold the number of octets specified in the bufsize input parameter.
pnocts	A pointer to an integer value to receive the decoded number of octets.

bufsize	The size (in octets) of the sized octet string. An error will occur if the number of octets in the decoded string is larger than this value.

Returns: . Completion status of operation:

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

See also: . rtJsonDecHexStr

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

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

Table 2.68. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.69. Parameters

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

Returns: . Completion status of operation:

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

int rtJsonDecDynHexStr64 (OSCTXT *pctxt, OSDynOctStr64 *pvalue)

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

Table 2.70. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.71. Parameters

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

Returns: . Completion status of operation:

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

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

This function decodes a variable of the ASN.1 Bit string type. This provides ObjSys-specific behavior that predates ITU-T X.697.

Table 2.72. Parameters

pctxt	Pointer to context block structure.

nbits	A pointer to an unsigned integer to receive the number of bits in the bit string.
data	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.

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.

Table 2.73. Parameters

pctxt	Pointer to context block structure.
nbits	A pointer to an unsigned integer to receive the number of bits in the bit string.
data	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

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.

Table 2.74. Parameters

pctxt	Pointer to context block structure.
nbits	A pointer to an unsigned integer to receive the number of bits in the bit string.
data	A pointer to an OSOCTET array that will receive the decoded bit string. The array must be preallocated.
bufsize	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.

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.

Table 2.75. Parameters

pctxt	Pointer to context block structure.
nbits	A pointer to an unsigned integer to receive the number of bits in the bit string.
data	A pointer to an OSOCTET array that will receive the decoded bit string. The array must be preallocated.
bufsize	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

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.

Table 2.76. Parameters

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

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.

Table 2.77. Parameters

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

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.

Table 2.78. Parameters

pctxt	Pointer to context block structure.
nbits	A pointer to an unsigned integer to receive the number of bits in the bit string.
data	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.

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.

Table 2.79. Parameters

pctxt	Pointer to context block structure.
nbits	A pointer to an unsigned integer to receive the number of bits in the bit string.
data	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

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.

Table 2.80. Parameters

pctxt	Pointer to context block structure.
nbits	A pointer to an unsigned integer to receive the number of bits in the bit string.
data	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.

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.

Table 2.81. Parameters

pctxt	Pointer to context block structure.

nbits	A pointer to an unsigned integer to receive the number of bits in the bit string.
data	A pointer to an OSOCTET array that will receive the decoded bit string. The array must be preallocated.
bufsize	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.

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.

Table 2.82. Parameters

pctxt	Pointer to context block structure.
nbits	A pointer to an unsigned integer to receive the number of bits in the bit string.
data	A pointer to an OSOCTET array that will receive the decoded bit string. The array must be preallocated.
bufsize	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

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.

Table 2.83. Parameters

pctxt	Pointer to context block structure.
nbits	The number of bits the BIT STRING is fixed to.
data	A pointer to an OSOCTET array that will receive the decoded bit string. The array must be preallocated.

bufsize	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.

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.

Table 2.84. Parameters

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

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.

Table 2.85. Parameters

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

int rtJsonDecMatchChar (OSCTXT *pctxt, OSUTF8CHAR ch)

This function attempts to match the given character, skipping over any whitesapce, if necessary. If a different character is found, this function returns RTERR_INVCHAR and does not consume the non-matching character.

Table 2.86. Parameters

pctxt	Pointer to context block structure.
ch	The character to be matched.

Returns: . Completion status of operation:

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

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

This function decodes a JSON string and matches with a given token. This is equivalent to rtJsonDecMatchToken.

Table 2.87. Parameters

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

Returns: . Completion status of operation:

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

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.

Table 2.88. Parameters

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

Returns: . Completion status of operation:

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

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

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

Table 2.89. Parameters

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

Returns: . Completion status of operation:

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

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

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

Table 2.90. Parameters

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

Returns: . Completion status of operation:

- 0 = success,

- negative return value is error.

int rtJsonDecNameValuePair (OSCTXT *pctxt, OSUTF8NVP *pvalue)

This function decodes a name/value pair.

Table 2.91. Parameters

pctxt	Pointer to context block structure.
pvalue	Pointer to an structure to receive the decoded name and value.

Returns: . Completion status of operation:

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

int rtJsonDecNull (OSCTXT *pctxt)

This function decodes a JSON null.

Table 2.92. Parameters

pctxt	A pointer to a context data structure.

Returns: . 0 on success, less than zero otherwise.

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.

Table 2.93. Parameters

pctxt	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)

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

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

Table 2.94. Parameters

pctxt	Pointer to context block structure.
ppCharStr	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.

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.

Table 2.95. Parameters

pctxt	Pointer to context block structure.
name	The name token.
ppvalue	Pointer to an 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.

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 characer string. Input is expected to be a string of UTF-8 characters returned by a JSON parser.

Table 2.96. Parameters

pctxt	Pointer to context block structure.
ppvalue	Pointer to an 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.

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

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

Table 2.97. Parameters

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

Returns: . Completion status of operation:

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

int rtJsonDecXmlStringValue (OSCTXT *pctxt, OSXMLSTRING *pval- ue)

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

Table 2.98. Parameters

pctxt	Pointer to context block structure.
pvalue	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.

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.

Table 2.99. Parameters

pctxt	A pointer to the context block structure.
ppstr	A pointer to a UTF-16 string; memory will be allocated to hold the string using the run-time memory manager.

pnchars	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.

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.

Table 2.100. Parameters

pctxt	A pointer to the context block structure.
ppstr	A pointer to a UTF-32 string; memory will be allocated to hold the string using the run-time memory manager.
pnchars	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.

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.

Table 2.101. Parameters

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

Returns: . Completion status of operation:

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

Macro Definition Documentation

#define rtJsonDecPeekChar

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

Table 2.102. Parameters

pctxt	Pointer to OSCTXT structure
pch	A pointer to a variable to receive the next character.

Returns: . Completion status of operation:

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

Definition at line 1589 of file osrtjson.h

The Documentation for this define was generated from the following file:

- osrtjson.h

#define rtJsonDecPeekChar2

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

Table 2.103. Parameters

pctxt	Pointer to OSCTXT structure

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

Definition at line 1599 of file osrtjson.h

The Documentation for this define was generated from the following file:

- osrtjson.h

Chapter 3. Class Documentation

OSJSONDecodeBuffer class Reference

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

Protected Attributes

- OSRTInputStream * mpInputStream
- OSBOOL mbOwnStream

- OSJSONDecodeBuffer (const char * jsonFile)
- OSJSONDecodeBuffer (const OSOCTET * msgbuf, size_t bufsiz)
- OSJSONDecodeBuffer (OSRTInputStream & inputStream)
- ~OSJSONDecodeBuffer ( )
- virtual EXTJSONMETHOD int init ( )
- virtual OSBOOL isA (Type bufferType)

Detailed Description

The OSJSONDecodeBuffer class is derived from the OSJSONMessageBuffer base class. It contains variables and methods specific to decoding JSON messages. It is used to manage an input buffer or stream containing a message to be decoded.

Definition at line 40 of file OSJSONDecodeBuffer.h

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

- OSJSONDecodeBuffer.h

Member Data Documentation

OSRTInputStream* OSJSONDecodeBuffer::mpInputStream

Input source for message to be decoded.

Definition at line 45 of file OSJSONDecodeBuffer.h

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

- OSJSONDecodeBuffer.h

OSBOOL OSJSONDecodeBuffer::mbOwnStream

This is set to true if this object creates the underlying stream object. In this case, the stream will be deleted in the object's destructor.

Definition at line 52 of file OSJSONDecodeBuffer.h

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

- OSJSONDecodeBuffer.h

EXTJSONMETHOD 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.

Table 3.1. Parameters

jsonFile	A pointer to name of file to be decoded.

EXTJSONMETHOD 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.

Table 3.2. Parameters

msgbuf	A pointer to a buffer containing an JSON message.
bufsiz	Size of the message buffer.

EXTJSONMETHOD OSJSONDecodeBuffer::OSJSONDecodeBuffer (OSRTIn-putStream &inputStream)

This version of the OSJSONDecodeBuffer constructor takes a reference to the OSInputStream object. The stream is assumed to have been previuously initialized to point at an encoded JSON message.

Table 3.3. Parameters

inputStream	reference to the OSInputStream object

EXTJSONMETHOD int OSJSONDecodeBuffer::init ()

This method initializes the decode message buffer.

Returns: . Completion status of operation:

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

virtual OSBOOL OSJSONDecodeBuffer::isA (Type bufferType)

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.

Table 3.4. Parameters

bufferType	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.

OSJSONEncodeBuffer class Reference

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

- OSJSONEncodeBuffer (OSRTContext * pContext)
- EXTJSONMETHOD OSJSONEncodeBuffer ( )
- EXTJSONMETHOD OSJSONEncodeBuffer (OSOCTET * 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)

Detailed Description

The OSJSONEncodeBuffer class is derived from the OSJSONMessageBuffer base class. It contains variables and methods specific to encoding JSON messages. It is used to manage the buffer into which a message is to be encoded.

Definition at line 38 of file OSJSONEncodeBuffer.h

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

- OSJSONEncodeBuffer.h

EXTJSONMETHOD OSJSONEncodeBuffer::OSJSONEncodeBuffer ()

Default constructor

EXTJSONMETHOD OSJSONEncodeBuffer::OSJSONEncodeBuffer (OSOCTET *pMsgBuf, size_t msgBufLen)

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

Table 3.5. Parameters

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

virtual size_t OSJSONEncodeBuffer::getMsgLen ()

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

Returns: . Length of the JSON message encapsulated within this buffer object.

EXTJSONMETHOD int OSJSONEncodeBuffer::init ()

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.

virtual OSBOOL OSJSONEncodeBuffer::isA (Type bufferType)

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.

Table 3.6. Parameters

bufferType	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.

void OSJSONEncodeBuffer::nullTerminate ()

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

EXTJSONMETHOD long OSJSONEncodeBuffer::write (const char *filename)

This method writes the encoded message to the given file.

Table 3.7. Parameters

filename	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.

EXTJSONMETHOD long OSJSONEncodeBuffer::write (FILE *fp)

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

Table 3.8. Parameters

fp	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.

OSJSONEncodeStream class Reference

`#include <OSJSONEncodeStream.h>`

Protected Attributes

- OSRTOutputStream * mpStream
- OSBOOL mbOwnStream
- OSCTXT * mpCtxt

- EXTJSONMETHOD OSJSONEncodeStream (OSRTOutputStream & outputStream)
- OSJSONEncodeStream (OSRTOutputStream * pOutputStream, OSBOOL ownStream)

- ~OSJSONEncodeStream ( )
- 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 ( )

Detailed Description

The OSJSONEncodeStream class is derived from the OSJSONMessageBuffer base class. It contains variables and methods specific to streaming encoding JSON messages. It is used to manage the stream into which a message is to be encoded.

Definition at line 40 of file OSJSONEncodeStream.h

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

- OSJSONEncodeStream.h

Member Data Documentation

OSRTOutputStream* OSJSONEncodeStream::mpStream

A pointer to an OSRTOutputStream object.

Definition at line 43 of file OSJSONEncodeStream.h

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

- OSJSONEncodeStream.h

OSBOOL OSJSONEncodeStream::mbOwnStream

TRUE if the OSJSONEncodeStream object will close and free the stream in the destructor.

Definition at line 47 of file OSJSONEncodeStream.h

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

- OSJSONEncodeStream.h

OSCTXT* OSJSONEncodeStream::mpCtxt

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

Definition at line 51 of file OSJSONEncodeStream.h

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

- OSJSONEncodeStream.h

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.

Table 3.9. Parameters

outputStream	reference to the OSOutputStream object

EXTJSONMETHOD 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.

Table 3.10. Parameters

pOutputStream	reference to the OSOutputStream object
ownStream	set ownership for the passed stream object.

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.

Table 3.11. Parameters

name	Attribute name.
value	UTF-8 string value to be encoded.

Returns: . Completion status of operation:

- 0 = success,

- negative return value is error.

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.

Table 3.12. Parameters

value	UTF-8 string value to be encoded.

Returns: . Completion status of operation:

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

EXTJSONMETHOD int OSJSONEncodeStream::init ()

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.

virtual OSBOOL OSJSONEncodeStream::isA (Type bufferType)

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.

Table 3.13. Parameters

bufferType	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.

virtual const OSOCTET* OSJSONEncodeStream::getMsgPtr ()

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.

OSRTOutputStream* OSJSONEncodeStream::getStream () const

This method returns the output stream associated with the object.

Returns: . A pointer to the output stream.

OSJSONMessageBuffer class Reference

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

- EXTJSONMETHOD OSJSONMessageBuffer (Type bufferType, OSRTContext * pContext)

Detailed Description

The JSON message buffer class is derived from the OSMessageBuffer base class. It is the base class for the OSJSONEncodeBuffer and OSJSONDecodeBuffer classes. It contains variables and methods specific to encoding or decoding JSON messages. It is used to manage the buffer into which a message is to be encoded or decoded.

Definition at line 42 of file OSJSONMessageBuffer.h

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

- OSJSONMessageBuffer.h

EXTJSONMETHOD OSJSONMessageBuffer::OSJSONMessageBuffer (Type bufferType, OSRTContext *pContext=0)

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

Table 3.14. Parameters

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

OSRTMessageBuffer class Reference

Chapter 4. File Documentation

OSJSONDecodeBuffer.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "rtxsrc/OSRTFileInputStream.h"
#include "rtxsrc/OSRTMemoryInputStream.h"
#include "rtjsonsrc/OSJSONDecodeBuffer.h"
```

Detailed Description

Definition in file OSJSONDecodeBuffer.cpp

OSJSONDecodeBuffer.h File Reference

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

Classes

- struct OSJSONDecodeBuffer

Detailed Description

JSON decode buffer or stream class definition.

Definition in file OSJSONDecodeBuffer.h

OSJSONEncodeBuffer.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "rtjsonsrc/OSJSONEncodeBuffer.h"
#include "rtxsrc/osMacros.h"
#include "rtxsrc/rtxFile.h"
```

Detailed Description

Definition in file OSJSONEncodeBuffer.cpp

OSJSONEncodeBuffer.h File Reference

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

Classes

- struct OSJSONEncodeBuffer

Detailed Description

JSON encode message buffer class definition.

Definition in file OSJSONEncodeBuffer.h

OSJSONEncodeStream.cpp File Reference

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

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

Detailed Description

Definition in file OSJSONEncodeStream.cpp

OSJSONEncodeStream.h File Reference

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

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

Classes

- struct OSJSONEncodeStream

Detailed Description

JSON encode stream class definition.

Definition in file OSJSONEncodeStream.h

OSJSONMessageBuffer.cpp File Reference

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>
```

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

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

Detailed Description

Definition in file OSJSONMessageBuffer.cpp

OSJSONMessageBuffer.h File Reference

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

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

Classes

- struct OSJSONMessageBuffer

Detailed Description

JSON encode/decode buffer and stream base class.

Definition in file OSJSONMessageBuffer.h

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 "rtxsrc/rtxText.h"
```

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

Macros

- #define OSJSONNOWS OSNOWHITESPACE /* encode with NO WhiteSpace */
- #define OSJSONATTR 0x00008000 /* encode string value as attribute */
- #define OSJSONPUTCOMMA if ((pctxt)->lastChar != '{' && (pctxt)->lastChar != ',' \ && (pctxt)->lastChar != '\0' && (pctxt)->lastChar != '[') \ OSRTSAFEPUTCHAR (pctxt, ',');
- #define DEFAULT_DOUBLE_PRECISION 11
- #define DEFAULT_FLOAT_PRECISION 6

- #define OSUPCASE 0x00008000
- #define rtJsonEncIntValue rtxTxtWriteInt
- #define rtJsonEncInt64Value rtxTxtWriteInt64
- #define rtJsonEncDecrIndent rtxIndentDecr
- #define rtJsonEncIncrIndent rtxIndentIncr
- #define rtJsonEncResetIndent rtxIndentReset
- #define rtJsonGetIndentLevels rtxGetIndentLevels
- #define rtJsonEncUIntValue rtxTxtWriteUInt
- #define rtJsonEncUInt64Value rtxTxtWriteUInt64
- #define rtJsonDecInt8Value rtxTxtReadInt8
- #define rtJsonDecInt16Value rtxTxtReadInt16
- #define rtJsonDecInt32Value rtxTxtReadInt32
- #define rtJsonDecIntValue rtxTxtReadInt32
- #define rtJsonDecInt64Value rtxTxtReadInt64
- #define rtJsonDecUInt8Value rtxTxtReadUInt8
- #define rtJsonDecUInt16Value rtxTxtReadUInt16
- #define rtJsonDecUInt32Value rtxTxtReadUInt32
- #define rtJsonDecUIntValue rtxTxtReadUInt32
- #define rtJsonDecUInt64Value rtxTxtReadUInt64
- #define rtJsonDecPeekChar rtxTxtPeekChar(pctxt, pch, TRUE)
- #define rtJsonDecPeekChar2 rtxTxtPeekChar2(pctxt, TRUE)
- #define rtJsonDecSkipWhitespace rtxTxtSkipWhitespace

Enumerations

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

Functions

- int rtJsonEncAnyAttr (OSCTXT * pctxt, const OSRTDList * pvalue)
- int rtJsonEncBase64StrValue (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 rtJsonEncBitStrValueV72 (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data)
- int rtJsonEncBitStrValueExtV72 (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data, OSSIZE dataSize, const OSOCTET * extData)
- int rtJsonEncBitStrValue (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data)
- int rtJsonEncFixedBitStrValue (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data)
- int rtJsonEncBitStrValueExt (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data, OSSIZE dataSize, const OSOCTET * extData)
- int rtJsonEncIndent (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 rtJsonEncStartObject (OSCTXT * pctxt, const OSUTF8CHAR * name, OSBOOL noComma)
- int rtJsonEncEndObject (OSCTXT * pctxt)
- int rtJsonEncBetweenObject (OSCTXT * pctxt)
- 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, OSOCTET ** extdata)
- int rtJsonDecBitStrValueExt64V72 (OSCTXT * pctxt, OSSIZE * nbits, OSOCTET * data, OSSIZE bufsize, OSOCTET ** 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 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 rtJsonDecStringObject (OSCTXT * pctxt, const OSUTF8CHAR * name, OSUTF8CHAR ** ppvalue)
- int rtJsonDecStringValue (OSCTXT * pctxt, OSUTF8CHAR ** ppvalue)
- int rtJsonDecStringValueArray (OSCTXT * pctxt, OSUTF8CHAR * pvalue, OSSIZE bufsize)

- int rtJsonDecXmlStringValue (OSCTXT * pctxt, OSXMLSTRING * pvalue)
- int rtJsonDecUCS2String (OSCTXT * pctxt, OSUNICHAR ** ppstr, OSSIZE * pnchars)
- int rtJsonDecUCS4String (OSCTXT * pctxt, OS32BITCHAR ** ppstr, OSSIZE * pnchars)
- size_t rtJsonGetElemIdx (OSCTXT * pctxt, const OSUTF8NameAndLen nameArray, size_t nrows)

Detailed Description

JSON low-level C encode/decode functions.

Definition in file osrtjson.h

rtJsonCppMsgBuf.h File Reference

`#include "rtsrc/asn1CppTypes.h"`

`#include "rtjsonsrc/OSJSONEncodeBuffer.h"`

`#include "rtjsonsrc/OSJSONEncodeStream.h"`

`#include "rtjsonsrc/OSJSONDecodeBuffer.h"`

Detailed Description

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

Definition in file rtJsonCppMsgBuf.h

rtJsonDecAny.c File Reference

`#include "rtxsrc/rtxBuffer.h"`

`#include "rtxsrc/rtxCtype.h"`

`#include "rtxsrc/rtxDiag.h"`

`#include "rtxsrc/rtxError.h"`

`#include "rtxsrc/rtxIntStack.h"`

`#include "rtxsrc/rtxMemBuf.h"`

`#include "rtjsonsrc/osrtjson.h"`

Macros

- #define __ -1 /* the universal error code */
- #define IS_HIGH_SURROGATE (((uc) & 0xFC00) == 0xD800)
- #define IS_LOW_SURROGATE (((uc) & 0xFC00) == 0xDC00)
- #define DECODE_SURROGATE_PAIR ((((hi) & 0x3FF) << 10) + ((lo) & 0x3FF) + 0x10000)

- #define MATCHING_NUMBER ((pctxt)->state == ZE || \ (pctxt)->state == IT || \ (pctxt)->state == FR || \ (pctxt)->state == E3)

Enumerations

- enum classes {
  C_SPACE,
  C_WHITE,
  C_LCURB,
  C_RCURB,
  C_LSQRB,
  C_RSQRB,
  C_COLON,
  C_COMMA,
  C_QUOTE,
  C_BACKS,
  C_SLASH,
  C_PLUS,
  C_MINUS,
  C_POINT,
  C_ZERO,
  C_DIGIT,
  C_LOW_A,
  C_LOW_B,
  C_LOW_C,
  C_LOW_D,
  C_LOW_E,
  C_LOW_F,
  C_LOW_L,
  C_LOW_N,
  C_LOW_R,
  C_LOW_S,
  C_LOW_T,
  C_LOW_U,
  C_ABCDF,
  C_E,
  C_ETC,
  C_STAR,
  NR_CLASSES
  }

- enum JsonParserMode {
  MODE_UNDEF= 0,
  MODE_ARRAY= 1,
  MODE_DONE= 2,
  MODE_KEY= 3,
  MODE_OBJECT= 4
  }

- enum states {
  GO,
  OK,
  OB,
  KE,

CO,
VA,
AR,
ST,
ES,
U1,
U2,
U3,
U4,
MI,
ZE,
IT,
FR,
E1,
E2,
E3,
T1,
T2,
T3,
F1,
F2,
F3,
F4,
N1,
N2,
N3,
C1,
C2,
C3,
FX,
D1,
D2,
NR_STATES
}

- enum actions {
  CB= -10,
  CE= -11,
  FA= -12,
  TR= -13,
  NU= -14,
  DE= -15,
  DF= -16,
  SB= -17,
  MX= -18,
  ZX= -19,
  IX= -20,
  EX= -21,
  UC= -22
  }

Variables

- static const signed char ascii_class

- static const signed char state_transition_table

Functions

- static int getErrCode (OSCTXT * pctxt)
- static int parseUnicodeChar (OSCTXT * pctxt, const OSOCTET * buffer, size_t buflen, OSUINT16 * pUTF16hs)
- static int popMode (OSRTIntStack * pstack, JsonParserMode expectedMode)
- static int jsonParserDone (OSCTXT * pctxt, OSRTIntStack * pstack)
- static int jsonDecAnyType (OSCTXT * pctxt, OSRTMEMBUF * pmembuf)
- int rtJsonDecAnyElem (OSCTXT * pctxt, OSUTF8CHAR ** ppvalue)
- int rtJsonDecAnyElem2 (OSCTXT * pctxt, OSUTF8CHAR ** ppvalue)
- int rtJsonDecAnyType (OSCTXT * pctxt, OSUTF8CHAR ** ppvalue)

Detailed Description

Definition in file rtJsonDecAny.c

rtJsonDecBase64Str.c File Reference

```
#include "osrtjson.h"

#include "rtxsrc/rtxBase64.h"
```

Functions

- int rtJsonDecBase64Str (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocts, OSSIZE bufsize)
- int rtJsonDecBase64Str64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnocts, OSSIZE bufsize)

Detailed Description

Definition in file rtJsonDecBase64Str.c

rtJsonDecBitStr.c File Reference

```
#include "osrtjson.h"

#include "rtxsrc/rtxBitString.h"

#include "rtxsrc/rtxUTF8.h"
```

Functions

- static int decBitStrChars (OSCTXT * pctxt, const char * data, OSSIZE nchars, OSOCTET * outbuf, OSSIZE outbufsize)
- 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, OSOCTET ** extdata)
- int rtJsonDecBitStrValueExt64V72 (OSCTXT * pctxt, OSSIZE * nbits, OSOCTET * data, OSSIZE bufsize, OSOCTET ** extdata)
- static int decString (OSCTXT * pctxt, OSOCTET * parray, OSSIZE arraysize, OSDynOctStr64 * pext, OSSIZE * plen)
- static int decValueAndLength (OSCTXT * pctxt, OSOCTET * parray, OSSIZE arraysize, OSDynOctStr64 * pext, OSSIZE * plen)
- 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)

Detailed Description

Definition in file rtJsonDecBitStr.c

rtJsonDecBool.c File Reference

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

Functions

- static int decJsonBoolValue (OSCTXT * pctxt, OSBOOL * pvalue, OSBOOL logError)
- int rtJsonDecBool (OSCTXT * pctxt, OSBOOL * pvalue)
- int rtJsonTryDecBool (OSCTXT * pctxt, OSBOOL * pvalue)

Detailed Description

Definition in file rtJsonDecBool.c

rtJsonDecDateTime.c File Reference

```
#include "rtxsrc/rtxCtype.h"
```

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

Enumerations

- enum DecDateFrom {
  Years,
  Months,
  Days,
  TimeStart,
  TimeEnd= 8
  }

Variables

- static const int multiplier
- static const int tzh_multiplier
- static const int tzm_multiplier

Functions

- static int rtJsonDecDateTimeInner (OSCTXT * pctxt, OSXSDDateTime * pvalue, int decDateFrom, int decDateTo)
- EXTJSONMETHOD int rtJsonDecDate (OSCTXT * pctxt, OSXSDDateTime * pvalue)
- EXTJSONMETHOD int rtJsonDecTime (OSCTXT * pctxt, OSXSDDateTime * pvalue)
- EXTJSONMETHOD int rtJsonDecDateTime (OSCTXT * pctxt, OSXSDDateTime * pvalue)
- EXTJSONMETHOD int rtJsonDecGYear (OSCTXT * pctxt, OSXSDDateTime * pvalue)
- EXTJSONMETHOD int rtJsonDecGYearMonth (OSCTXT * pctxt, OSXSDDateTime * pvalue)
- EXTJSONMETHOD int rtJsonDecGMonth (OSCTXT * pctxt, OSXSDDateTime * pvalue)
- EXTJSONMETHOD int rtJsonDecGMonthDay (OSCTXT * pctxt, OSXSDDateTime * pvalue)
- EXTJSONMETHOD int rtJsonDecGDay (OSCTXT * pctxt, OSXSDDateTime * pvalue)

Detailed Description

Definition in file rtJsonDecDateTime.c

rtJsonDecDecimal.c File Reference

```
#include <math.h>
```

```
#include <limits.h>
```

```
#include "rtxsrc/rtxCtype.h"
```

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

Functions

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

Detailed Description

Definition in file rtJsonDecDecimal.c

rtJsonDecDouble.c File Reference

```
#include <math.h>
```

```
#include "rtxsrc/rtxCtype.h"
```

```
#include "rtxsrc/rtxReal.h"
```

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

Functions

- int rtJsonDecDouble (OSCTXT * pctxt, OSREAL * pvalue)

Detailed Description

Definition in file rtJsonDecDouble.c

rtJsonDecDynBase64Str.c File Reference

```
#include "rtxsrc/rtxMemBuf.h"
```

```
#include "rtxsrc/rtxBase64.h"
```

```
#include "rtxsrc/rtxCtype.h"
```

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

Macros

- #define JSON_DECODE_SEGSIZE 1

Functions

- int rtJsonDecDynBase64Str (OSCTXT * pctxt, OSDynOctStr * pvalue)
- int rtJsonDecDynBase64Str64 (OSCTXT * pctxt, OSDynOctStr64 * pvalue)

Detailed Description

Definition in file rtJsonDecDynBase64Str.c

rtJsonDecDynHexStr.c File Reference

```
#include "rtxsrc/rtxMemBuf.h"
```

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

Macros

- #define XMLP_DECODE_SEGSIZE 1

Functions

- int rtJsonDecDynHexStr (OSCTXT * pctxt, OSDynOctStr * pvalue)
- int decDynHexData (OSCTXT * pctxt, OSRTMEMBUF * pmembuf)
- int rtJsonDecDynHexData64 (OSCTXT * pctxt, OSDynOctStr64 * pvalue)
- int rtJsonDecHexToCharStr (OSCTXT * pctxt, OSUTF8CHAR ** ppvalue)
- int rtJsonDecDynHexStr64 (OSCTXT * pctxt, OSDynOctStr64 * pvalue)

Detailed Description

Definition in file rtJsonDecDynHexStr.c

rtJsonDecHexStr.c File Reference

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

Functions

- int rtJsonDecHexStr (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 * pnocts, size_t bufsize)
- int rtJsonDecHexData64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnocts, size_t bufsize)
- int rtJsonDecHexStr64 (OSCTXT * pctxt, OSOCTET * pvalue, OSSIZE * pnocts, size_t bufsize)

Detailed Description

Definition in file rtJsonDecHexStr.c

rtJsonDecNull.c File Reference

```
#include "osrtjson.h"
```

Functions

- int rtJsonDecNull (OSCTXT * pctxt)
- int rtJsonTryDecNull (OSCTXT * pctxt)

Detailed Description

Definition in file rtJsonDecNull.c

rtJsonDecNumberString.c File Reference

```
#include "rtjsonsrc/osrtjson.h"
#include "rtxsrc/rtxCharStr.h"
#include "rtxsrc/rtxCtype.h"
```

Functions

- static int parseDigits (OSCTXT * pctxt, char * outbuf, OSSIZE * pbufidx, OSSIZE bufsize)
- int rtJsonDecNumberString (OSCTXT * pctxt, char ** ppCharStr)

Detailed Description

Definition in file rtJsonDecNumberString.c

rtJsonDecString.c File Reference

```
#include "rtxsrc/rtxBuffer.h"
#include "rtxsrc/rtxCtype.h"
#include "rtxsrc/rtxError.h"
#include "rtxsrc/rtxMemBuf.h"
#include "rtjsonsrc/osrtjson.h"
```

Functions

- int rtJsonDecStringValueInternal (OSCTXT * pctxt, OSUTF8CHAR ** ppvalue, OSSIZE bufsize, OSBOOL dynamic)
- int rtJsonDecStringValue (OSCTXT * pctxt, OSUTF8CHAR ** ppvalue)
- int rtJsonDecStringValueArray (OSCTXT * pctxt, OSUTF8CHAR * pvalue, OSSIZE bufsize)
- int rtJsonDecXmlStringValue (OSCTXT * pctxt, OSXMLSTRING * pvalue)
- int rtJsonDecStringObject (OSCTXT * pctxt, const OSUTF8CHAR * name, OSUTF8CHAR ** ppvalue)

- int rtJsonDecNameValuePair (OSCTXT * pctxt, OSUTF8NVP * pvalue)

Detailed Description

Definition in file rtJsonDecString.c

rtJsonDecUCS2String.c File Reference

```
#include "rtxsrc/osMacros.h"
#include "rtxsrc/rtxErrCodes.h"
#include "osrtjson.h"
#include "rtJsonDecUTF.c"
```

Macros

- #define OSJSONUNIDECFUNC rtJsonDecUCS2String
- #define OSJSONUNICHAR OSUNICHAR

Detailed Description

Definition in file rtJsonDecUCS2String.c

rtJsonDecUCS4String.c File Reference

```
#include "rtxsrc/osMacros.h"
#include "rtxsrc/rtxErrCodes.h"
#include "osrtjson.h"
#include "rtJsonDecUTF.c"
```

Macros

- #define OSJSONUNIDECFUNC rtJsonDecUCS4String
- #define OSJSONUNICHAR OS32BITCHAR

Detailed Description

Definition in file rtJsonDecUCS4String.c

rtJsonDecUTF.c File Reference

```
#include "rtxsrc/osMacros.h"
#include "rtxsrc/rtxErrCodes.h"
```

```
#include "rtxsrc/rtxMemBuf.h"
```

```
#include "osrtjson.h"
```

Functions

- EXTJSONMETHOD int OSJSONUNIDECFUNC (OSCTXT * pctxt, OSJSONUNICHAR ** ppstr, OSSIZE * pnchars)

Detailed Description

Definition in file rtJsonDecUTF.c

rtJsonEncAnyAttr.c File Reference

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

Functions

- int rtJsonEncAnyAttr (OSCTXT * pctxt, const OSRTDList * pvalue)

Detailed Description

Definition in file rtJsonEncAnyAttr.c

rtJsonEncBase64Binary.c File Reference

```
#include "osrtjson.h"
```

Variables

- static const char encodeTable

Functions

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

Detailed Description

Definition in file rtJsonEncBase64Binary.c

rtJsonEncBitStr.c File Reference

```
#include "osrtjson.h"
```

Functions

- static void encBitStrContent (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data)

- int rtJsonEncBitStrValueV72 (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data)
- int rtJsonEncBitStrValueExtV72 (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data, OSSIZE dataSize, const OSOCTET * extData)
- int rtJsonEncBitStrValue (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data)
- int rtJsonEncFixedBitStrValue (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data)
- int rtJsonEncBitStrValueExt (OSCTXT * pctxt, OSSIZE nbits, const OSOCTET * data, OSSIZE dataSize, const OSOCTET * extData)

Detailed Description

Definition in file rtJsonEncBitStr.c

rtJsonEncBool.c File Reference

```
#include "osrtjson.h"
```

Functions

- int rtJsonEncBoolValue (OSCTXT * pctxt, OSBOOL value)

Detailed Description

Definition in file rtJsonEncBool.c

rtJsonEncDates.c File Reference

```
#include "rtxsrc/rtxErrCodes.h"
```

```
#include "rtxsrc/rtxDateTime.hh"
```

```
#include "rtxsrc/rtxBuffer.h"
```

Functions

- 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)

Detailed Description

Definition in file rtJsonEncDates.c

rtJsonEncDateTime.c File Reference

`#include "osrtjson.h"`

`#include "rtxsrc/osMacros.h"`

`#include "rtxsrc/rtxErrCodes.h"`

`#include "rtxsrc/rtxDateTime.hh"`

Functions

- int rtJsonEncDatePart (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)

Detailed Description

Definition in file rtJsonEncDateTime.c

rtJsonEncDecimal.c File Reference

`#include <math.h>`

`#include "osrtjson.h"`

`#include "rtxsrc/rtxCharStr.h"`

`#include "rtxsrc/rtxErrCodes.h"`

`#include "rtxsrc/rtxReal.h"`

Macros

- #define LG2 0.30102999566398119521373889472449
- #define PRECISION 15 /* the default number of fraction digits for normalized double */
- #define FRACTIONDIGITS 10

Functions

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

Detailed Description

Definition in file rtJsonEncDecimal.c

rtJsonEncDouble.c File Reference

`#include <math.h>`

`#include "osrtjson.h"`

`#include "rtxsrc/rtxReal.h"`

Macros

- #define RTJSONENCDOUBLEVALUEFUNC rtJsonEncDoubleValue
- #define PRECISION DEFAULT_DOUBLE_PRECISION
- #define LG2 0.30102999566398119521373889472449

Functions

- int RTJSONENCDOUBLEVALUEFUNC (OSCTXT * pctxt, OSREAL value, const OSDoubleFmt * pFmtSpec)

Detailed Description

Definition in file rtJsonEncDouble.c

rtJsonEncEdges.c File Reference

`#include "rtjsonsrc/osrtjson.h"`

Functions

- int rtJsonEncStartObject (OSCTXT * pctxt, const OSUTF8CHAR * name, OSBOOL noComma)
- int rtJsonEncEndObject (OSCTXT * pctxt)
- int rtJsonEncBetweenObject (OSCTXT * pctxt)

Detailed Description

Definition in file rtJsonEncEdges.c

rtJsonEncFloat.c File Reference

`#include "rtJsonEncDouble.c"`

Macros

- #define RTJSONENCDOUBLEVALUEFUNC rtJsonEncFloatValue
- #define PRECISION DEFAULT_FLOAT_PRECISION

Detailed Description

Definition in file rtJsonEncFloat.c

rtJsonEncHexBinary.c File Reference

```
#include "osrtjson.h"
```

Functions

- int rtJsonEncHexValue (OSCTXT * pctxt, OSSIZE nocts, const OSOCTET * data)
- int rtJsonEncHexStr (OSCTXT * pctxt, OSSIZE nocts, const OSOCTET * data)

Detailed Description

Definition in file rtJsonEncHexBinary.c

rtJsonEncIndent.c File Reference

```
#include "rtxsrc/osMacros.h"

#include "rtxsrc/rtxErrCodes.h"

#include "osrtjson.h"
```

Functions

- int rtJsonEncIndent (OSCTXT * pctxt)

Detailed Description

Definition in file rtJsonEncIndent.c

rtJsonEncString.c File Reference

```
#include "rtxsrc/osMacros.h"

#include "rtxsrc/rtxBuffer.h"

#include "rtxsrc/rtxError.h"

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

Functions

- int rtJsonEncStringValue2 (OSCTXT * pctxt, const OSUTF8CHAR * value, size_t valueLen)
- int rtJsonEncStringValue (OSCTXT * pctxt, const OSUTF8CHAR * value)
- int rtJsonEncChars (OSCTXT * pctxt, const char * value, OSSIZE valueLen)

- int rtJsonEncCharStr (OSCTXT * pctxt, const char * value)
- int rtJsonEncStringPair2 (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 rtJsonEncStringObject2 (OSCTXT * pctxt, const OSUTF8CHAR * name, size_t nameLen, const OSUTF8CHAR * value, size_t valueLen)
- int rtJsonEncStringObject (OSCTXT * pctxt, const OSUTF8CHAR * name, const OSUTF8CHAR * value)
- int rtJsonEncStringNull (OSCTXT * pctxt)
- int rtJsonEncStringRaw (OSCTXT * pctxt, const OSUTF8CHAR * value)

Detailed Description

Definition in file rtJsonEncString.c

rtJsonEncUCS4String.c File Reference

```
#include "rtxsrc/osMacros.h"

#include "rtxsrc/rtxErrCodes.h"

#include "osrtjson.h"

#include "rtJsonEncUTF.c"
```

Macros

- #define OSJSONUNIENCFUNC rtJsonEncUCS4Data
- #define OSJSONUNICHAR OS32BITCHAR

Detailed Description

Definition in file rtJsonEncUCS4String.c

rtJsonEncUnicode.c File Reference

```
#include "rtxsrc/osMacros.h"

#include "rtxsrc/rtxErrCodes.h"

#include "osrtjson.h"

#include "rtJsonEncUTF.c"
```

Macros

- #define OSJSONUNIENCFUNC rtJsonEncUnicodeData

- #define OSJSONUNICHAR OSUNICHAR

Detailed Description

Definition in file rtJsonEncUnicode.c

rtJsonEncUTF.c File Reference

`#include "rtxsrc/osMacros.h"`

`#include "rtxsrc/rtxErrCodes.h"`

`#include "osrtjson.h"`

Variables

- static const OSUINT32 encoding_mask
- static const unsigned char encoding_byte

Functions

- int OSJSONUNIENCFUNC (OSCTXT * pctxt, const OSJSONUNICHAR * value, OSSIZE nchars)

Detailed Description

Definition in file rtJsonEncUTF.c

rtJsonExternDefs.h File Reference

Macros

- #define EXTJSONMETHOD

Detailed Description

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

Definition in file rtJsonExternDefs.h

rtJsonParser.c File Reference

`#include "rtxsrc/osMacros.h"`

`#include "rtxsrc/rtxBuffer.h"`

`#include "rtxsrc/rtxCtype.h"`

`#include "rtxsrc/rtxDiag.h"`

`#include "rtxsrc/rtxError.h"`

```
#include "rtxsrc/rtxStream.h"
```

```
#include "rtxsrc/rtxUTF8.h"
```

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

Functions

- size_t rtJsonGetElemIdx (OSCTXT * pctxt, const OSUTF8NameAndLen nameArray, size_t nrows)
- int rtJsonDecMatchChar (OSCTXT * pctxt, OSUTF8CHAR ch)
- int rtJsonDecMatchToken2 (OSCTXT * pctxt, const OSUTF8CHAR * token, size_t tokenLen)
- int rtJsonDecMatchToken (OSCTXT * pctxt, const OSUTF8CHAR * token)
- int rtJsonDecMatchCharStr (OSCTXT * pctxt, const char * token)
- int rtJsonDecMatchObjectStart (OSCTXT * pctxt, const OSUTF8NameAndLen * nameArray, size_t numNames)
- int rtJsonDecSkipWhitespaceAndComma (OSCTXT * pctxt)

Detailed Description

Definition in file rtJsonParser.c