Package ‘RcppMsgPack’

November 18, 2018

Type Package
Title ‘MsgPack’ C++ Header Files and Interface Functions for R
Version 0.2.3
Date 2018-11-18
Author Travers Ching and Dirk Eddelbuettel; the authors and contributors of MsgPack
Maintainer Dirk Eddelbuettel <edd@debian.org>
Description ‘MsgPack’ header files are provided for use by R packages, along with the ability to access, create and alter ‘MsgPack’ objects directly from R. ‘MsgPack’ is an efficient binary serialization format. It lets you exchange data among multiple languages like ’JSON’ but it is faster and smaller. Small integers are encoded into a single byte, and typical short strings require only one extra byte in addition to the strings themselves. This package provides headers from the ‘msgpack-c’ implementation for C and C++(11) for use by R, particularly ‘Rcpp’. The included ‘msgpack-c’ headers are licensed under the Boost Software License (Version 1.0); the code added by this package as well the R integration are licensed under the GPL (>= 2). See the files ’COPYRIGHTS’ and ’AUTHORS’ for a full list of copyright holders and contributors to ‘msgpack-c’.

Copyright file inst/COPYRIGHTS
License GPL (>= 2)
Imports Rcpp
LinkingTo Rcpp, BH

BugReports https://github.com/eddelbuettel/rcppmsgpack/issues
RoxygenNote 6.0.1
Suggests knitr, rmarkdown, microbenchmark
VignetteBuilder knitr
NeedsCompilation yes
Repository CRAN
Date/Publication 2018-11-18 21:10:03 UTC
R topics documented:

RcppMsgPack-package .................................................. 2
arrayEx ................................................................. 3
enumEx ................................................................. 4
msgpack_format .......................................................... 4
msgpack_map ............................................................. 5
msgpack_pack ............................................................ 6
msgpack_read ............................................................. 6
msgpack_simplify ........................................................ 7
msgpack_timestamp_decode ............................................. 8
msgpack_timestamp_encode ............................................ 8
msgpack_unpack ........................................................ 9
msgpack_write .......................................................... 10

Index 11

RcppMsgPack-package 'MsgPack’ C++ Header Files and Interface Functions for R

Description

'MsgPack' header files are provided for use by R packages, along with the ability to access, create and alter 'MsgPack' objects directly from R. 'MsgPack' is an efficient binary serialization format. It lets you exchange data among multiple languages like 'JSON' but it is faster and smaller. Small integers are encoded into a single byte, and typical short strings require only one extra byte in addition to the strings themselves. This package provides headers from the 'msgpack-c' implementation for C and C++(11) for use by R, particularly 'Rcpp'. The included 'msgpack-c' headers are licensed under the Boost Software License (Version 1.0); the code added by this package as well the R integration are licensed under the GPL (>= 2). See the files 'COPYRIGHTS' and 'AUTHORS' for a full list of copyright holders and contributors to 'msgpack-c'.

Package Content

Index of help topics:

RcppMsgPack-package 'MsgPack' C++ Header Files and Interface Functions for R
arrayEx Simple MsgPack Example
enumEx Second simple MsgPack Example
msgpack_format Format data for 'MsgPack'
msgpack_map 'MsgPack' Map
msgpack_pack 'MsgPack' Pack
msgpack_read 'MsgPack' read
msgpack_simplify Simplify 'MsgPack'
msgpack_timestamp_decode 'MsgPack' Timestamp
arrayEx

- msgpack_unpack
- msgpack_write

'MsgPack' Timestamp
'MsgPack' Unpack
'MsgPack' write

Maintainer
Dirk Eddelbuettel <edd@debian.org>

Author(s)
Travers Ching and Dirk Eddelbuettel; the authors and contributors of MsgPack

arrayEx | Simple MsgPack Example

Description
Simple MsgPack Example

Usage
arrayEx()

Details
The function provides a simple illustration of MessagePack.

Value
A boolean value of TRUE is returned, but the function exists for its side effect.

See Also
The MessagePack documentation, particularly the msgpack-c examples.
**enumEx**  
*Second simple MsgPack Example*

**Description**  
Second simple MsgPack Example

**Usage**  
`enumEx()`

**Details**  
The function provides a simple illustration of MessagePack.

**Value**  
A boolean value of `TRUE` is returned, but the function exists for its side effect.

**See Also**  
The MessagePack documentation, particularly the `msgpack-c` examples.

---

**msgpack_format**  
*Format data for 'MsgPack'*

**Description**  
A helper function to format R data for input to 'MsgPack'.

**Usage**  
`msgpack_format(x)`

**Arguments**  
`x`  
An r object.

**Value**  
A formatted R object to use as input to `msgpack_pack`. Vectors are converted into Lists.
msgpack_map

Examples

```r
x <- msgpack_format(1:10)
x_packed <- msgpack_pack(x)
x_unpacked <- msgpack_unpack(x_packed)
x_simplified <- msgpack_simplify(x_unpacked)
```

---

msgpack_map  'MsgPack' Map

Description

A helper function to create a map object for input to 'MsgPack'.

Usage

```r
msgpack_map(key, value)
msgpackMap(key, value)
```

Arguments

- **key**: A list or vector of keys (coerced to list). Duplicate keys are fine (connects to std::multimap in C++).
- **value**: A list or vector of values (coerced to list). This should be the same length as key.

Value

An data.frame also of class "map" that can be used as input to msgpack_pack.

Examples

```r
x <- msgpack_map(key=letters[1:10], value=1:10)
x_packed <- msgpack_pack(x)
x_unpacked <- msgpack_unpack(x_packed)
```
Description

Serialize any number of objects into a single message. Unnamed List is converted into Array, Map/Data.frame and Named Lists are converted into Maps. Integer, Double, Character, Raw vectors and NULL are converted into Int types (depending on size), Float types, String, Raw and Nil respectively. Raw vectors with EXT attribute are converted into Extension types. The EXT attribute should be an integer from 0 to 127.

Usage

msgpack_pack(...)

msgpackPack(...)

Arguments

... Any R objects that have corresponding msgpack types.

Value

A raw vector containing the message.

See Also

See examples/tests.r for more examples.

Examples

x <- msgpack_format(1:10)
x_packed <- msgpack_pack(x)
x_unpacked <- msgpack_unpack(x_packed)
x_simplified <- msgpack_simplify(x_unpacked)

Description

A helper function to de-serialize an object read from a file or a connection.
Usage

msgpack_read(file, simplify = F, mode = "auto", nbytes = 16777216)

msgpackRead(file, simplify = F, mode = "auto", nbytes = 16777216)

Arguments

file A connection, or a string describing the file or pipe to write to, depending on the mode.

simplify Passed to msgpack_unpack. Default: FALSE.

mode One of "auto", "file", "gzip" or "pipe". If "auto", detects based on the file string (any space == pipe, ".gz" == gzip, file otherwise). Ignored if file is a connection.

nbytes If reading from a pipe or gzip, how many bytes to read at a time. Default: 16777216

Examples

tmp <- tempfile(fileext=".gz")
msgpack_write(1:10, file=tmp)
x <- msgpack_read(tmp, simplify=TRUE)

msgpack_simplify

Simplify 'MsgPack'

Description

A helper function for simplifying a 'MsgPack' return object.

Usage

msgpack_simplify(x)

msgpackSimplify(x)

Arguments

x Return object from msgpack_unpack.

Value

A simplified return object from msgpack_unpack. Lists of all the same type are concatenated into an atomic vector. Maps are simplified to named lists or named vectors as appropriate. NULLs are converted to NAs if simplified to vector.
Examples

```r
x <- msgpack_format(1:10)
x_packed <- msgpack_pack(x)
x_unpacked <- msgpack_unpack(x_packed)
x_simplified <- msgpack_simplify(x_unpacked)
```

---

**msgpack_timestamp_decode**

'`MsgPack' Timestamp`

**Description**

Decodes a timestamp from the 'MsgPack' extension specifications.

**Usage**

```r
msgpack_timestamp_decode(x, posix = T, tz = "UTC")
msgpackTimestampDecode(x, posix = T, tz = "UTC")
```

**Arguments**

- `x`  
  A raw vector with attribute EXT = -1, following the 'MsgPack' timestamp specifications.

- `posix`  
  Return a POSIXct object. Otherwise, return a list with seconds and nanoseconds since 1970-01-01 00:00:00.

- `tz`  
  If returning a POSIXct, set the timezone. Note that this doesn’t change the underlying value.

**Value**

A POSIXct or list.

```r
mt <- Sys.time()
attr(mt, "tzone") <- "UTC"
mp <- msgpack_pack(msgpack_timestamp_encode(mt))
mtu <- msgpack_timestamp_decode(msgpack_unpack(mp))
identical(mt, mtu)
```

---

**msgpack_timestamp_encode**

'`MsgPack' Timestamp`

**Description**

Encodes a timestamp to the 'MsgPack' specifications.
Usage

msgpack_timestamp_encode(posix = NULL, seconds = NULL, nanoseconds = NULL)

msgpack_timestamp_encodeHposix = NULL, seconds = NULL, nanoseconds = NULL)

Arguments

posix A POSIXct or POSIXlt or anything that can be coerced to a numeric.
seconds The number of seconds since 1970-01-01 00:00:00 UTC. Can be negative. Don’t use seconds and nanoseconds if you use posix (and vice versa).
nanoseconds The number of nanoseconds since 1970-01-01 00:00:00 UTC. Must be less than 1,000,000,000 and greater than 0.

Value

A serialized timestamp that can be used as input to msgpack_pack. Briefly, this is an extension type -1 that is variable length, depending on the desired range and precision.

Examples

mt <- Sys.time()
attr(mt, "tzone") <- "UTC"
mp <- msgpack_pack(msgpack_timestamp_encode(mt))
mtu <- msgpack_timestamp_decode(msgpack_unpack(mp))
identical(mt, mtu)

message 'MsgPack' Unpack

Description

De-serialize a 'MsgPack' message. Array is converted into List. Map is converted into Map/Data.frame. Extension types are converted into raw vectors with EXT attribute. Integers, Floats, Strings, Raw and Nil are converted into Integer, Float, Character, Raw and NULL respectively.

Usage

msgpack_unpack(message, simplify = F)

msgpack_unpack(message, simplify = F)

Arguments

message A raw vector containing the message.
simplify Default false. Should the return object be simplified? This is generally faster and more memory efficient.
**Value**

The message pack object(s) converted into R types. If more than one object exists in the message, a list of class "msgpack_set" containing the objects is returned.

**See Also**

See examples/tests.r for more examples.

**Examples**

```r
x <- msgpack_format(1:10)
x_packed <- msgpack_pack(x)
x_unpacked <- msgpack_unpack(x_packed)
x_simplified <- msgpack_simplify(x_unpacked)
```

---

**Description**

A helper function to serialize an object and write it to a file, or a connection.

**Usage**

```r
msgpack_write(..., msg = NULL, file, mode = "auto")
```

```r
msgpackWrite(..., msg = NULL, file, mode = "auto")
```

**Arguments**

- `...` Serializable R objects.
- `msg` Message to write to file. If not NULL and a raw vector, write it instead of the R objects. Default: NULL.
- `file` A connection, or a string describing the file or pipe to write to, depending on the mode.
- `mode` One of "auto", "file", "gzip" or "pipe". If "auto", detects based on the file string (any space == pipe, ".gz" == gzip, file otherwise). Ignored if file is a connection.

**Examples**

```r
tmp <- tempfile(fileext=".gz")
msgpack_write(1:10, file=tmp)
x <- msgpack_read(tmp, simplify=TRUE)
```
Index

*Topic package
   RcppMsgPack-package, 2

arrayEx, 3

enumEx, 4

msgpack_format, 4
msgpack_map, 5
msgpack_pack, 6
msgpack_read, 6
msgpack_simplify, 7
msgpack_timestamp_decode, 8
msgpack_timestamp_encode, 8
msgpack_unpack, 9
msgpack_write, 10
msgpackFormat (msgpack_format), 4
msgpackMap (msgpack_map), 5
msgpackPack (msgpack_pack), 6
msgpackRead (msgpack_read), 6
msgpackSimplify (msgpack_simplify), 7
msgpackTimestampDecode
   (msgpack_timestamp_decode), 8
msgpackTimestampEncode
   (msgpack_timestamp_encode), 8
msgpackUnpack (msgpack_unpack), 9
msgpackWrite (msgpack_write), 10

RcppMsgPack (RcppMsgPack-package), 2
RcppMsgPack-package, 2