Package 'RcppMsgPack'

January 16, 2018

Type Package

Title 'MsgPack' C++ Header Files and Interface Functions for R

Version 0.2.1

Date 2018-01-15

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 2018-01-16 09:29:11 UTC

R topics documented:

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_simplify        Simplify 'MsgPack'
msgpack_timestamp_decode
                        'MsgPack' Timestamp
msgpack_timestamp_encode
                        'MsgPack' Timestamp
msgpack_unpack          'MsgPack' Unpack
```

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)

msgpackFormat(x)
```

Arguments

x	An r object.

Value

A formatted R object to use as input to msgpack_pack. Vectors are converted into Lists.

Examples

```
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

```
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

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

msgpack_pack *'MsgPack' Pack*

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)
```

---

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

```
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

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

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

Arguments

`x`	A raw vector with attriubte 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. 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)

msgpackTimestampEncode(posix = 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)
```

---

msgpack_unpack *'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)

msgpackUnpack(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

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

Index