

# Package: protolite (via r-universe)

October 5, 2024

**Type** Package

**Title** Highly Optimized Protocol Buffer Serializers

**Description** Pure C++ implementations for reading and writing several common data formats based on Google protocol-buffers. Currently supports 'rexp.proto' for serialized R objects, 'geobuf.proto' for binary geojson, and 'mvt.proto' for vector tiles. This package uses the auto-generated C++ code by protobuf-compiler, hence the entire serialization is optimized at compile time. The 'RProtoBuf' package on the other hand uses the protobuf runtime library to provide a general-purpose toolkit for reading and writing arbitrary protocol-buffer data in R.

**Version** 2.3.1

**License** MIT + file LICENSE

**URL** <https://github.com/jeroen/protolite>  
<https://jeroen.r-universe.dev/protolite>

**BugReports** <https://github.com/jeroen/protolite/issues>

**SystemRequirements** libprotobuf and protobuf-compiler

**LinkingTo** Rcpp

**Imports** Rcpp (>= 0.12.12), jsonlite

**Suggests** spelling, curl, testthat, sf

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.2.1

**NeedsCompilation** yes

**Author** Jeroen Ooms [aut, cre]  
(<<https://orcid.org/0000-0002-4035-0289>>)

**Maintainer** Jeroen Ooms <jeroenooms@gmail.com>

**Repository** CRAN

**Date/Publication** 2024-10-04 11:50:02 UTC

## Contents

geobuf . . . . .	2
mapbox . . . . .	2
serialize_pb . . . . .	3

<b>Index</b>	<b>5</b>
--------------	----------

---

geobuf	<i>Geobuf</i>
--------	---------------

---

### Description

The **geobuf** format is an optimized binary format for storing geojson data with protocol buffers. These functions are compatible with the `geobuf2json` and `json2geobuf` utilities from the `geobuf` [npm package](#).

### Usage

```
read_geobuf(x, as_data_frame = TRUE)
```

```
geobuf2json(x, pretty = FALSE)
```

```
json2geobuf(json, decimals = 6)
```

### Arguments

<code>x</code>	file path or raw vector with the serialized <code>geobuf.proto</code> message
<code>as_data_frame</code>	simplify geojson data into data frames
<code>pretty</code>	indent json, see <a href="#">jsonlite::toJSON</a>
<code>json</code>	a text string with geojson data
<code>decimals</code>	how many decimals (digits behind the dot) to store for numbers

---

mapbox	<i>Mapbox Vector Tiles</i>
--------	----------------------------

---

### Description

Read Mapbox vector-tile (mvt) files and returns the list of layers.

### Usage

```
read_mvt_data(data, as_latlon = TRUE, zxy = NULL)
```

```
read_mvt_sf(data, crs = 4326, zxy = NULL)
```

**Arguments**

data	url, path or raw vector with the mvt data
as_latlon	return the data as lat/lon instead of raw EPSG:3857 positions
zxy	vector of length 3 with respectively z (zoom), x (column) and y (row). For file/url in the standard <code>./{z}/{x}/{y}.mvt</code> format, these are automatically inferred from the input path.
crs	desired output coordinate system (passed to <code>sf::st_transform</code> ). Note that mvt input is always by definition 3857.

---

serialize_pb	<i>Serialize to Protocol Buffers</i>
--------------	--------------------------------------

---

**Description**

Serializes R objects to a general purpose protobuf message. It uses the same `rexp.proto` descriptor and mapping between R objects and protobuf messages as RHIPE and the `RProtoBuf` package.

**Usage**

```
serialize_pb(object, connection = NULL, skip_native = FALSE)
```

```
unserialize_pb(msg)
```

**Arguments**

object	an R object to serialize
connection	a connection, file, or NULL for a raw vector
skip_native	do not serialize 'native' (non-data) R objects. Setting to TRUE will only serialize <i>data</i> types (numeric, boolean, string, raw, list). The default behavior is to fall back on base R <code>serialize</code> for non-data objects.
msg	raw vector with the serialized <code>rexp.proto</code> message

**Details**

The `serialize_pb` and `unserialize_pb` reimplement the identically named functions from the `RProtoBuf` package in pure C++. This makes the function faster and simpler, but the output should be identical.

**Examples**

```
# Serialize and unserialize an object
buf <- serialize_pb(iris)
out <- unserialize_pb(buf)
stopifnot(identical(iris, out))

## Not run: #Fully compatible with RProtoBuf
```

```
buf <- RProtoBuf::serialize_pb(iris, NULL)
out <- protolite::unserialize_pb(buf)
stopifnot(identical(iris, out))

# Other way around
buf <- protolite::serialize_pb(mtcars, NULL)
out <- RProtoBuf::unserialize_pb(buf)
stopifnot(identical(mtcars, out))

## End(Not run)
```

# Index

`geobuf`, [2](#)  
`geobuf2json` (`geobuf`), [2](#)  
  
`json2geobuf` (`geobuf`), [2](#)  
`jsonlite::toJSON`, [2](#)  
  
`mapbox`, [2](#)  
  
`protolite` (`serialize_pb`), [3](#)  
  
`read_geobuf` (`geobuf`), [2](#)  
`read_mvt_data` (`mapbox`), [2](#)  
`read_mvt_sf` (`mapbox`), [2](#)  
`RProtoBuf`, [3](#)  
  
`serialize`, [3](#)  
`serialize_pb`, [3](#)  
`sf::st_transform`, [3](#)  
  
`unserialize_pb` (`serialize_pb`), [3](#)