Package: zutoml (via r-universe)

November 15, 2026

Title Read and Write 'TOML' Documents

Version 0.1.0

Description Parses 'TOML' (Tom's Obvious Minimal Language) documents, versions 1.0.0 and 1.1.0 <https://toml.io/en/v1.1.0>, into ordinary R values, and writes R values as 'TOML'. Every 'TOML' type maps to one R type, including the four date and time types, and strings are never reinterpreted. Invalid documents are refused with the line and column at fault, under configurable size, depth, item and string limits, and output is deterministic. Both directions pass the 'toml-test' conformance suite <https://github.com/toml-lang/toml-test>. The parser and writer are written in C, with no system library required.

License MIT + file LICENSE

URL https://github.com/pedrobtz/zutoml, https://pedrobtz.github.io/zutoml/

BugReports https://github.com/pedrobtz/zutoml/issues

Depends R (>= 4.1)

LinkingTo zufast (>= 0.1.0)

Suggests knitr, rmarkdown, testthat (>= 3.0.0), withr, zujson

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Repository https://pedrobtz.r-universe.dev

Date/Publication 2026-11-15 06:29:20 UTC

RemoteUrl https://github.com/pedrobtz/zutoml

RemoteRef HEAD

RemoteSha 9d68d2cef8c781f32baba4076cf490e17d844543

Contents

zutoml-package . . . 2
toml-markers . . . 3
toml_bigint . . . 4
toml_emit . . . 4
toml_parse . . . 6
toml_read . . . 9
toml_validate . . . 10
zutoml-conditions . . . 11
zutoml_info . . . 12

Index **13**

`zutoml-package` *zutoml: Read and Write 'TOML' Documents*

Description

Parses 'TOML' (Tom's Obvious Minimal Language) documents, versions 1.0.0 and 1.1.0 https://toml.io/en/v1.1.0, into ordinary R values, and writes R values as 'TOML'. Every 'TOML' type maps to one R type, including the four date and time types, and strings are never reinterpreted. Invalid documents are refused with the line and column at fault, under configurable size, depth, item and string limits, and output is deterministic. Both directions pass the 'toml-test' conformance suite https://github.com/toml-lang/toml-test. The parser and writer are written in C, with no system library required.

Author(s)

Maintainer: Pedro Baltazar <pedrobtz@gmail.com> [copyright holder]

Authors:

- Pedro Baltazar <pedrobtz@gmail.com> [copyright holder]

See Also

`toml_parse()` to read, `toml_emit()` to write, and `vignette("zutoml")` for a tour.

toml-markers *Say how one value is written as TOML*

Description

Markers for `toml_emit()`, which otherwise chooses each value's form by fixed rules. They change how a value is written, never what it is: `toml_parse()` reads the text back as the unmarked value.

Usage

```
toml_inline(x)

toml_literal(x)

toml_multiline(x)
```

Arguments

x	For `toml_inline()`, a list; for the others, a character vector.

Details

- `toml_inline()` writes a named list as an inline table (`point = { x = 1, y = 2 }`) rather than under its own `[header]`, and an unnamed list of named lists as an array of inline tables rather than an array of tables. A `Cargo.toml` dependency such as `serde = { version = "1.0", features = ["derive"] }` is written this way.
- `toml_literal()` writes strings as literal strings, which need no escapes: `'C:\Users\tom'`. A string with a single quote or a newline is a multi-line literal string. One that no literal string can hold (a control character other than tab and newline, or three single quotes in a row) raises `zutoml_invalid_argument`.
- `toml_multiline()` writes strings as multi-line basic strings, even without a newline.

Like `I()`, which writes a vector of one as an array, the markers are classes on the value; any other class it has is kept.

Value

`x`, with the marker class added.

Examples

```
cat(toml_emit(list(
  dependencies = list(
    serde = toml_inline(list(version = "1.0", features = c("derive", "std"))),
    rand = "0.8"
  ),
  path = toml_literal("C:\\Users\\tom"),
  motd = toml_multiline("Welcome!")
)))
```

toml_bigint *Integers beyond what a double holds exactly*

Description

TOML integers are 64-bit. Those beyond 2^53 cannot be held exactly by a double, so `toml_parse()` returns them, by default, as a `toml_bigint` vector: their exact decimal text, with a class.

Usage

```
toml_bigint(x)
```

Arguments

`x`	Decimal strings (`"-9223372036854775808"`), or whole numbers within 2^53. `NA` is kept.

Value

A character vector of class `toml_bigint`, holding canonical decimal text: no leading zeros or plus sign.

Examples

```
x <- toml_bigint(c("9007199254740993", "-42"))
x
as.numeric(x)   # loses the precision a bigint keeps
toml_parse("big = 9223372036854775807")$big
```

toml_emit *Write R values as TOML*

Description

`toml_emit()` turns a named list into the text of a TOML document; `toml_write()` writes that text to a file or connection. The output is deterministic (the same value gives the same bytes on every platform) and is always text `toml_parse()` reads back.

Usage

```
toml_emit(
  x,
  indent = 2L,
  inline = 0L,
  width = 80L,
  na = c("error", "omit"),
```

```
  strings = c("basic", "literal"),
  max_depth = 128L
)

toml_write(x, file, ...)
```

Arguments

x	A named list: the document's top-level table.
indent	Spaces before each element of an array written one element per line.
inline	A nested named list with at most this many keys, none of them a list, is written as an inline table (`{ x = 1, y = 2 }`) rather than under its own header. `0`, the default, never does.
width	An array whose line would pass this column is written one element per line. `0` never wraps.
na	`"error"` (the default) refuses `NA` and `NULL`; `"omit"` leaves out a key whose value is `NA` or `NULL`, and drops such elements from arrays with a warning.
strings	`"basic"` (the default) writes double-quoted strings; `"literal"` writes single-quoted strings where that needs no escapes.
max_depth	The deepest nesting of tables and arrays written, counted as `toml_parse()` counts it; at most 1023.
file	For `toml_write()`, a path or a connection.
...	For `toml_write()`, passed to `toml_emit()`.

Value

`toml_emit()`: a single string, UTF-8, with `\n` line endings and a final newline. `toml_write()`: `file`, invisibly.

R to TOML

R	TOML
`character`	basic string, or literal with `strings = "literal"`
`integer`	integer
`double`, whole, within 64-bit range, not `-0`	integer
any other `double`	float, in the shortest digits that read back exactly
`logical`	boolean
`POSIXct` with a time zone	offset date-time, in UTC (`Z`)
`POSIXct` with no time zone (`tzone = ""`)	local date-time
`Date`	local date; a fractional one as its day
`difftime` in seconds, from 0 to a day	local time
`toml_bigint()`	integer
`factor`	its labels, as strings
a vector of length one	the value; `I()` makes it an array of one
a longer vector	array
named `list`	table, or an inline table inside an array

unnamed `list` of named lists	array of tables (`[[key]]`)
other unnamed `list`	array
`data.frame`	array of tables, one per row; `NA` cells are left out

A table's own keys are written first, in the list's order, then its sub-tables: TOML has no other way to say which keys belong to which table. A sub-table that holds only sub-tables gets no header of its own. A string with a newline is written as a multi-line string.

Refused with `zutoml_unsupported_type`: complex and raw vectors, functions, environments, `POSIXlt`, matrices and arrays with `dim`, and atomic vectors of other classes. `NA` and `NULL`, which TOML has no form for, raise `zutoml_na_error` unless `na = "omit"`. A value that breaks its class's invariants raises `zutoml_invalid_argument`: a data frame without a name for every column, or with a column that is not a vector of one cell per row, and an infinite date or time. Each condition carries `path`, the key path of the value at fault, as `a.b[3].c`.

Examples

```
cfg <- list(
  title = "Example",
  port = 8080,
  tags = c("a", "b"),
  owner = list(name = "Tom", since = as.Date("1979-05-27")),
  servers = list(
    list(name = "alpha", ip = "10.0.0.1"),
    list(name = "beta", ip = "10.0.0.2")
  )
)
cat(toml_emit(cfg))

identical(toml_parse(toml_emit(cfg))$port, 8080L)

path <- tempfile(fileext = ".toml")
toml_write(cfg, path)
toml_parse(readLines(path) |> paste(collapse = "\n"))$owner$name
```

`toml_parse` *Parse a TOML document*

Description

Reads a TOML document into ordinary R values: each table becomes a named list with its keys in document order, each array a vector or a list, and each value the R type below. Types come from the document's syntax: `"2024-01-01"` stays a string and `2024-01-01` becomes a `Date`. The whole document is checked, as by `toml_validate()`, before any R value is built.

Usage

```
toml_parse(
  x,
  select = NULL,
  version = c("1.1.0", "1.0.0"),
  simplify = c("preserve", "none"),
  data_frame = FALSE,
  big_integers = c("bigint", "double", "error"),
  datetimes = c("convert", "keep"),
  local_time = c("character", "difftime"),
  positions = FALSE,
  max_size = 64 * 1024^2,
  max_depth = 128L,
  max_items = 1e+06,
  max_string = 2^31 - 1
)
```

Arguments

x	A TOML document: a single string, or a raw vector of UTF-8 bytes.
select	NULL (the default) returns the whole document. A path returns only the value there: a string in TOML key syntax, as the positions table writes it ("tool.poetry"; '"a.b".c' for a key with a dot; "products[2].name" into arrays, counting from 1), or a vector of two or more keys taken as they are (c("tool", "poetry")). The whole document is still read and checked, since TOML lets a table be added to anywhere in it, but only the selected part becomes R values. A path the document does not have raises zutoml_missing_key, with path and found, the part of the path that does exist.
version	The TOML version to read: "1.1.0" (the default) or "1.0.0", which refuses the forms 1.1.0 added (\e and \xHH escapes, times without seconds, newlines and a trailing comma in inline tables).
simplify	"preserve" (the default) turns arrays whose elements share a type into vectors; "none" makes every array a list.
data_frame	If TRUE, an array whose elements are all tables becomes a data frame: one row per table, columns in first-seen key order, missing keys NA. Columns whose values do not share a type are lists. More than getOption("zutoml.max_df_cells", 1e7) cells raises zutoml_limit_error, since rows that share no keys make the frame quadratic in size.
big_integers	For integers beyond 2^53, which a double cannot hold exactly: "bigint" (the default) returns a toml_bigint() vector of exact decimal text; "double" the nearest double; "error" raises zutoml_unrepresentable.
datetimes	"convert" (the default) returns POSIXct and Date; "keep" returns the datetimes as text.
local_time	"character" (the default) returns a local time as its text; "difftime" as seconds since midnight.

positions	If TRUE, the result carries an attribute "toml_positions": a data frame with one row per key and array element, in the order of the parsed value, giving its path in TOML key syntax (servers[2].ip, with keys quoted where TOML needs it, and array elements numbered from 1), its type ("table", "inline_table", "array_of_tables", "array", "string", "integer", "float", "bool", "datetime", "datetime-local", "date-local" or "time-local"), and the line, column (1-based, in characters) and byte offset (0-based) where it is defined: a key's position for a keyed value, the [[header]] for each table of an array of tables. Code that checks a configuration can then say where a value is wrong.
max_size	The largest document accepted, in bytes, or Inf.
max_depth	The deepest nesting of tables and arrays, counting both; at most 1023.
max_items	The most keys plus array elements, or Inf.
max_string	The longest string or key, in decoded bytes, or Inf.

Value

A named list, one element per top-level key; with select, the value at that path.

TOML to R

TOML	R
string	character (UTF-8)
integer	integer; double beyond R's integer range (and for -2^31); beyond 2^53, see big_integers
float	double; inf and nan as Inf and NaN
boolean	logical
offset date-time	POSIXct in UTC
local date-time	POSIXct in the session's time zone (tzone = "")
local date	Date
local time	character ("07:32:00"), or difftime in seconds
table, inline table	named list
array of tables	unnamed list of named lists, or a data.frame
array	a vector when its elements share a type, else a list

An array becomes a vector when all its elements are strings, all booleans, all numbers, all dates, or all date-times of one kind. Numbers widen: integers with floats give a double vector. Booleans never join the numbers. An empty array is logical(0). A one-element array that becomes a vector is marked with I(), so that writing it back gives an array again. Anything else is a list. simplify = "none" makes every array a list.

With datetimes = "keep", date-times are returned as RFC 3339 text instead ("1979-05-27T07:32:00Z", a zero offset written Z, the fraction in 0, 3, 6 or 9 digits). Fractions past nanoseconds are truncated, as TOML requires; a POSIXct holds microseconds.

Valid TOML that R cannot hold raises zutoml_unrepresentable: a string or key containing U+0000, a float beyond the range of a double, or an integer beyond 2^53 with big_integers = "error".

See Also

toml_validate() to check a document without building it; zutoml-conditions for the errors.

Examples

```
doc <- '
title = "TOML Example"
ports = [8000, 8001, 8002]

[owner]
name = "Tom Preston-Werner"
dob = 1979-05-27T07:32:00-08:00

[[products]]
name = "Hammer"
sku = 738594937

[[products]]
name = "Nail"
sku = 284758393
color = "gray"
'
x <- toml_parse(doc)
x$owner$dob
x$ports

toml_parse(doc, data_frame = TRUE)$products

# Only part of it:
toml_parse(doc, select = "owner")
toml_parse(doc, select = "products[2].color")

# Where each value is, for messages about the document:
pos <- attr(toml_parse(doc, positions = TRUE), "toml_positions")
pos[pos$path == "owner.dob", ]
```

---

toml_read *Read a TOML document from a file or connection*

Description

Reads the input and parses it as toml_parse() would. At most max_size + 1 bytes are ever read, so an oversized file or an endless connection fails with zutoml_size_limit rather than exhausting memory.

Usage

```
toml_read(file, ..., max_size = 64 * 1024^2)
```

Arguments

`file`	A file path, a URL, or a connection.
`...`	Arguments passed on to `toml_parse()`.
`max_size`	The largest document accepted, in bytes, or `Inf`.

Details

A string with an `http`, `https`, `ftp`, `ftps` or `file` scheme is read with `url()`, any other string as a file path. A connection that is not open is opened in `"rb"` mode and closed afterwards; an open one must be binary, is read from its current position, and is left open. A URL or file that cannot be opened, such as one that answers HTTP 404, fails with `zutoml_io_error`.

Value

As `toml_parse()`.

Examples

```
path <- tempfile(fileext = ".toml")
writeLines(c("[package]", 'name = "zutoml"', "version = 1"), path)
toml_read(path)
unlink(path)
```

`toml_validate` *Check that a document is valid TOML*

Description

Runs the whole check phase of `toml_parse()` (the lexer, the grammar, the rules for defining tables and keys, every value, and the limits) without building any R value. It is exactly the check `toml_parse()` makes, with the same limits: the two disagree only where a document is valid TOML that R cannot hold, such as a string containing U+0000.

Usage

```
toml_validate(
  x,
  version = c("1.1.0", "1.0.0"),
  max_size = 64 * 1024^2,
  max_depth = 128L,
  max_items = 1e+06,
  max_string = 2^31 - 1,
  error = FALSE
)
```

Arguments

x	A TOML document: a single string, or a raw vector of UTF-8 bytes.
version	The TOML version to read: "1.1.0" (the default) or "1.0.0", which refuses the forms 1.1.0 added (\e and \xHH escapes, times without seconds, newlines and a trailing comma in inline tables).
max_size	The largest document accepted, in bytes, or Inf.
max_depth	The deepest nesting of tables and arrays, counting both; at most 1023.
max_items	The most keys plus array elements, or Inf.
max_string	The longest string or key, in decoded bytes, or Inf.
error	If FALSE (the default), return FALSE for a document that is not valid. If TRUE, raise the condition instead, so the caller learns why and where.

Value

TRUE when x is valid TOML within the limits. Otherwise FALSE, or with error = TRUE a condition of class zutoml_parse_error or zutoml_limit_error (see zutoml-conditions) carrying the line, column, byte offset and kind of the fault.

Examples

```
toml_validate('title = "TOML"\n[owner]\nname = "Tom"\n')
toml_validate("a = 1\na = 2\n")

# Why not, and where:
tryCatch(
  toml_validate("a = 1\na = 2\n", error = TRUE),
  zutoml_parse_error = function(e) c(e$kind, e$line, e$column)
)
```

---

zutoml-conditions *Conditions raised by zutoml*

Description

Every error zutoml raises carries a condition class, so it can be caught by kind rather than by matching the message, which may change. Every class below inherits from zutoml_error.

Details

zutoml_invalid_argument An argument was unusable, or a value cannot be written as TOML as given, such as a list with partial names or a string that is not valid UTF-8. Carries arg, the argument at fault.

zutoml_parse_error The document is not TOML. Carries line and column (1-based, in characters), offset (0-based, in bytes) and kind, the name of the rule broken, such as "bad_escape" or "table_redefined".

`zutoml_unrepresentable` The document is valid TOML but holds a value R cannot: a string containing U+0000, a string longer than an R string, or an integer beyond 2^53 with `big_integers = "error"`.

`zutoml_unsupported_type` An R value has no TOML form, such as a function, a raw vector or a matrix. Carries `path`, the key path of the value, as `a.b[3].c`.

`zutoml_na_error` An `NA` or `NULL` value with `na = "error"`. Carries `path`.

`zutoml_limit_error` A limit was reached. The subclasses `zutoml_size_limit`, `zutoml_depth_limit`, `zutoml_item_limit` and `zutoml_string_limit` name which one. Carries `limit`, the argument's name, such as `"max_depth"`, and `limit_value`.

`zutoml_io_error` A file, URL or connection could not be opened, read or written.

`zutoml_missing_key` `toml_parse()`'s `select` names a path the document does not have. Carries `path`, the path asked for, and `found`, the part of it that exists.

Examples

```
tryCatch(
  stop(structure(
    class = c("zutoml_parse_error", "zutoml_error", "error", "condition"),
    list(message = "TOML parse error", call = NULL)
  )),
  zutoml_error = function(e) class(e)[1]
)
```

---

zutoml_info *Information about the zutoml build*

Description

Information about the zutoml build

Usage

```
zutoml_info()
```

Value

A list: `version`, the package version; `zufast`, the version of the 'zufast' headers it was compiled against; `toml_test`, the version of the `toml-test` conformance suite it passes; `toml_versions`, the TOML versions `toml_parse()` reads; `max_depth_cap`, the largest `max_depth` accepted; and `ndebug`, whether the C code was compiled without assertions.

Examples

```
zutoml_info()$toml_test
```

Index

∗ **package**
zutoml-package, 2

I(), *3*, *5*, *8*

toml-markers, 3
toml_bigint, 4
toml_bigint(), *5*, *7*
toml_emit, 4
toml_emit(), *2, 3*
toml_inline *(*toml-markers*)*, 3
toml_literal *(*toml-markers*)*, 3
toml_multiline *(*toml-markers*)*, 3
toml_parse, 6
toml_parse(), *2–5*, *9, 10*, *12*
toml_read, 9
toml_validate, 10
toml_validate(), *6*, *9*
toml_write *(*toml_emit*)*, 4

url(), *10*

zutoml *(*zutoml-package*)*, 2
zutoml-conditions, *9*, 11, *11*
zutoml-package, 2
zutoml_info, 12