Package 'janssonr'

September 12, 2026

Type Package

Title Strict JSON Encoding and Decoding via the 'Jansson' C Library

Version 0.1.2

Date 2026-09-01

Copyright cornball.ai, except the bundled Jansson sources under src/jansson, whose per-file copyright holders and license statements are listed in inst/COPYRIGHTS.

Description An R-safe profile of RFC 8259 JSON: parsing and generation backed by the 'Jansson' C library, linked as a system library where one is available and compiled from the bundled sources otherwise. The parser rejects, with classed conditions carrying line, column, and byte position: malformed or truncated input, trailing content, duplicate object keys at any depth, invalid UTF-8, escapes encoding a null character, reals overflowing double,
and integer literals whose magnitude exceeds 2^53, the range within which a double represents every integer exactly.
Number literals with a fraction or exponent convert by ordinary correctly rounded IEEE 754 double conversion. Objects decode to named lists in key order, arrays to unnamed lists, and scalars to length-one vectors. The encoder maps named lists to objects in insertion order, unnamed lists to arrays, guarantees that every finite double, signed zero included, round-trips to the exact same value (whole-number doubles are written as integers), and refuses values with no faithful JSON representation (NA, NaN, infinities, named atomic vectors, classed objects) instead of guessing. No R package dependencies.

License MIT + file LICENSE

Depends R (>= 4.4.0)

URL https://github.com/cornball-ai/janssonr

BugReports https://github.com/cornball-ai/janssonr/issues

SystemRequirements jansson (>= 2.11) is linked when present: libjansson-dev (deb), jansson-devel (rpm), jansson (brew). When it is absent, and always on Windows, the bundled Jansson 2.15.1 is compiled into the package.

Suggests tinytest

Encoding UTF-8

NeedsCompilation yes

Author Troy Hernandez [aut, cre] (ORCID:
<https://orcid.org/0009-0005-4248-604X>),
cornball.ai [cph],
Petri Lehtinen [ctb, cph] (Author of the Jansson C library, bundled under src/jansson),
Basile Starynkevitch [ctb, cph] (Jansson: src/jansson/memory.c),
Graeme Smecher [ctb, cph] (Jansson: src/jansson/pack_unpack.c),
Sean Bright [ctb, cph] (Jansson: src/jansson/version.c),
David M. Gay [ctb] (Author of dtoa.c, bundled by Jansson as src/jansson/dtoa.c),
Lucent Technologies [cph] (src/jansson/dtoa.c),
Bob Jenkins [ctb] (Author of lookup3.c (public domain), bundled by Jansson as src/jansson/lookup3.h)

Maintainer Troy Hernandez <troy@cornball.ai>

Repository CRAN

Date/Publication 2026-09-12 09:00:02 UTC

Contents

`from_json` *Parse JSON strictly*

Description

Parses one JSON document under janssonr's R-safe profile of RFC 8259. The parser refuses, with a classed condition: malformed or truncated input, trailing content, duplicate object keys at any depth, invalid UTF-8, embedded NUL escapes, numbers overflowing double, and integer literals whose magnitude exceeds 2^53. The exactness guarantee is for integer literals only: number literals with a fraction or exponent (`1.5`, `9007199254740993.0`, `1e-999`) convert by ordinary correctly rounded IEEE 754 double conversion.

Usage

```
from_json(x)
```

Arguments

`x`	A length-1, non-NA character vector, or a raw vector holding UTF-8 bytes.

Details

Mapping: a JSON object becomes a named list in key order (an empty object keeps a `character(0)` names attribute, distinguishing it from an empty array); an array becomes an unnamed list, never an atomic vector; a string becomes `character(1)`; an integer fitting R's integer range becomes `integer(1)` (except -2147483648, R's NA sentinel, which becomes double); any other number becomes `double(1)`; `true`/`false` become `logical(1)`; `null` becomes `NULL`. Nesting beyond 1024 containers refuses.

Character input is translated to UTF-8; raw input is taken byte for byte and must already be valid UTF-8.

Errors have class `c("janssonr_parse_error", "janssonr_error","error", "condition")` and carry `line`, `column`, `position` (byte offset) for lexical errors, plus `code`, a stable string such as `"duplicate_key"` or `"invalid_utf8"`. The `"integer_precision"` code covers every integer-form refusal: literals beyond 2^53 detected after parsing carry NA coordinates plus `path`, an RFC 6901 JSON Pointer to the offending value, while literals so large the lexer itself refuses them carry source coordinates and an NA `path` (no parse tree exists to point into). `"numeric_overflow"` is real-form overflow such as `1e999`; `"depth_limit"` carries NA coordinates.

Value

The corresponding R value; see the mapping above.

Examples

```
from_json('{"a": 1, "b": [true, null]}')
from_json("[1, 2, 3]")
```

---

to_json *Serialize to JSON strictly*

Description

Serializes an R value to one compact UTF-8 JSON document (no whitespace, no trailing newline). Values with no faithful JSON representation refuse with a classed condition instead of being coerced.

Usage

```
to_json(x)
```

Arguments

`x`	An R value made of named or unnamed lists, atomic vectors of type logical, integer, double or character, and `NULL`.

Details

Mapping: a list with a names attribute becomes an object in insertion order (`""` is a valid key, so unnamed slots of a partially-named list encode as `""`; duplicate keys refuse); an unnamed list becomes an array (a length-1 list stays a 1-element array); a length-1 unnamed atomic becomes a bare scalar; any other atomic becomes an array; `NULL` becomes `null`. Integral doubles with magnitude at most 2^53 are written as integers (`1`, not `1.0`), except `-0`, which stays a real (`-0.0`) so its sign bit survives; other doubles are written with 17 significant digits. Every finite double, signed zero included, round-trips to the exact same value. The lexical spelling of non-integral doubles is not guaranteed byte-stable across Jansson versions.

Refused: NA of any type, NaN, infinities, named atomic vectors, atomics or lists with other attributes, classed objects, duplicate or NA keys, raw, complex, functions, environments, and nesting beyond 1024 containers. Errors have class `c("janssonr_encode_error", "janssonr_error", "error","condition")`.

Value

A length-1 UTF-8 character vector holding the JSON document.

Examples

```
to_json(list(a = 1L, b = list(TRUE, NULL)))
to_json(list())
to_json(structure(list(), names = character(0)))
```

Index