Package: toonlite (via r-universe)

October 23, 2026

Title Read, Write, Validate, Stream, and Convert 'TOON' Data

Version 0.2.0

Description Reads, writes, validates, streams and converts 'TOON' (Token-Oriented Object Notation) data, a text format for passing structured data to language models. Implements version 4.4 of the specification <https://github.com/toon-format/spec>. C++ backend with no required dependencies. Converts to and from CSV, and, with 'jsonlite' or 'arrow' installed, JSON, Parquet and Feather.

License MIT + file LICENSE

URL https://github.com/aljrico/toonlite

BugReports https://github.com/aljrico/toonlite/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

SystemRequirements C++17

Depends R (>= 3.5.0)

Suggests testthat (>= 3.0.0), jsonlite, arrow

Config/testthat/edition 3

NeedsCompilation yes

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

Date/Publication 2026-10-23 11:39:28 UTC

RemoteUrl https://github.com/aljrico/toonlite

RemoteRef HEAD

RemoteSha 9954f951efb5e38c4fcf46b83a88e591b534ef14

Contents

as_tabular_toon . . . 2
assert_toon . . . 3
conditions . . . 4
conversions . . . 5
core-io . . . 5
csv_to_toon . . . 5
feather_to_toon . . . 6
format_toon . . . 7
from_toon . . . 8
json_to_toon . . . 9
parquet_to_toon . . . 10
print.toonlite_toon . . . 11
read_toon . . . 12
read_toon_df . . . 13
streaming . . . 15
tabular-io . . . 15
to_toon . . . 15
toon_info . . . 16
toon_peek . . . 17
toon_stream_items . . . 18
toon_stream_rows . . . 19
toon_stream_write_rows . . . 21
toon_to_csv . . . 22
toon_to_feather . . . 23
toon_to_json . . . 24
toon_to_parquet . . . 25
validate_toon . . . 26
validation . . . 27
write_toon . . . 27
write_toon_df . . . 28

Index **30**

`as_tabular_toon` *Convert array-of-objects to tabular representation*

Description

Convert array-of-objects to tabular representation

Usage

```
as_tabular_toon(x, strict = TRUE, warn = TRUE)
```

Arguments

x	List of lists/named lists (array of objects).
strict	Logical. If TRUE (default), validate input structure.
warn	Logical. If TRUE (default), emit warnings for type promotions.

Details

In permissive mode:

- Union schema across all records
- Missing fields filled with NA
- Type promotion to accommodate all values

Value

A data.frame suitable for `write_toon_df()`.

Examples

```
# Convert array of objects to data.frame
records <- list(
  list(name = "Alice", age = 30),
  list(name = "Bob", age = 25)
)
df <- as_tabular_toon(records)
```

---

assert_toon *Assert TOON validity*

Description

Assert TOON validity

Usage

```
assert_toon(
  x,
  is_file = FALSE,
  strict = TRUE,
  allow_comments = TRUE,
  allow_duplicate_keys = TRUE,
  indent = 2L
)
```

Arguments

x	Character scalar containing TOON text, or file path if is_file=TRUE.
is_file	Logical. If TRUE, x is treated as a file path.
strict	Logical. If TRUE (default), apply the spec's strict decoding rules: declared lengths must match, indentation must be a multiple of indent with no tabs, blank lines inside arrays and duplicate keys are errors. If FALSE, the spec's five non-strict recoveries apply instead (count mismatches accepted, duplicate keys resolve last-write-wins, tab/odd indentation, blank lines and depth jumps tolerated).
allow_comments	Deprecated and ignored. Full-line # comments are part of TOON and are always skipped; passing anything other than TRUE warns.
allow_duplicate_keys	Deprecated and ignored. Duplicate keys are an error when strict = TRUE and resolve last-write-wins when strict = FALSE, as the spec requires. Supplying it warns.
indent	Integer. Spaces per indentation level (the spec's indentSize, default 2).

Details

Throws toonlite_parse_error condition if invalid, with attached location and snippet information.

Value

Invisibly returns TRUE if valid.

Examples

```
# Valid TOON (returns invisibly)
assert_toon("name: Alice")

# Invalid TOON (throws error)
try(assert_toon("items[2]: a"))
```

---

conditions *toonlite error and warning conditions*

Description

Functions for creating and signaling toonlite-specific conditions.

conversions *TOON Conversion Functions*

Description

Functions for converting between TOON and other formats.

core-io *Core TOON I/O functions*

Description

Functions for parsing and serializing TOON data.

csv_to_toon *Convert CSV to TOON*

Description

Convert CSV to TOON

Usage

```
csv_to_toon(
  path_csv,
  path_toon,
  tabular = TRUE,
  strict = TRUE,
  col_types = NULL,
  delimiter = ","
)
```

Arguments

path_csv	Character scalar. Path to input CSV file.
path_toon	Character scalar. Path to output TOON file.
tabular	Deprecated and ignored. Data frames are always written as a tabular TOON array; passing anything other than TRUE gives a warning.
strict	Deprecated and ignored (see to_toon()).
col_types	Named character vector specifying column types.
delimiter	Document delimiter: "," (default), "\t" or "|".

Value

Invisibly returns NULL.

Examples

```
## Not run:
csv_to_toon("data.csv", "data.toon")

## End(Not run)
```

`feather_to_toon` *Convert Feather to TOON*

Description

Convert Feather to TOON

Usage

```
feather_to_toon(
  path_feather,
  path_toon,
  tabular = TRUE,
  strict = TRUE,
  delimiter = ","
)
```

Arguments

`path_feather`	Character scalar. Path to input Feather file.
`path_toon`	Character scalar. Path to output TOON file.
`tabular`	Deprecated and ignored. Data frames are always written as a tabular TOON array; passing anything other than TRUE gives a warning.
`strict`	Deprecated and ignored (see `to_toon()`).
`delimiter`	Document delimiter: `","` (default), `"\t"` or `"|"`.

Details

Requires the arrow package. If not installed, an error is thrown.

Value

Invisibly returns NULL.

Examples

```
## Not run:
feather_to_toon("data.feather", "data.toon")

## End(Not run)
```

`format_toon` *Format/pretty-print TOON*

Description

Decodes TOON and encodes it again with `to_toon()`, which gives the canonical spec form: normalised quoting and numbers, inline primitive arrays, tabular forms where they apply.

Usage

```
format_toon(
  x,
  is_file = FALSE,
  indent = 2L,
  canonical = FALSE,
  allow_comments = TRUE,
  delimiter = ","
)
```

Arguments

`x`	Character scalar containing TOON text, or file path if is_file=TRUE.
`is_file`	Logical. If TRUE, x is treated as a file path.
`indent`	Integer. Spaces per indentation level in the output (default 2).
`canonical`	Logical. If TRUE, sort object keys (byte order, at every level) for a stable representation. Default FALSE (preserve order).
`allow_comments`	Deprecated and ignored. Full-line # comments are always skipped (and never written).
`delimiter`	Document delimiter for the output: `","` (default), `"\t"` or `"|"`.

Value

Character scalar with formatted TOON. If is_file=TRUE, returns formatted text (does not rewrite file).

Examples

```
format_toon('name: "Alice"')

# Format with canonical key ordering
format_toon('b: 1\na: 2', canonical = TRUE)
```

`from_toon` *Parse TOON from string*

Description

Decodes a TOON document (spec v4.4) into R values.

Usage

```
from_toon(
  text,
  strict = TRUE,
  simplify = TRUE,
  allow_comments = TRUE,
  allow_duplicate_keys = TRUE,
  indent = 2L
)
```

Arguments

`text`	Character scalar or raw vector containing TOON data. If raw, treated as UTF-8 bytes.
`strict`	Logical. If TRUE (default), apply the spec's strict decoding rules: declared lengths must match, indentation must be a multiple of `indent` with no tabs, blank lines inside arrays and duplicate keys are errors. If FALSE, the spec's five non-strict recoveries apply instead (count mismatches accepted, duplicate keys resolve last-write-wins, tab/odd indentation, blank lines and depth jumps tolerated).
`simplify`	Logical. If TRUE (default), arrays whose elements are all primitives of one type become atomic vectors (`null` becomes `NA`; integers mixed with doubles give a double vector).
`allow_comments`	Deprecated and ignored. Full-line # comments are part of TOON and are always skipped; passing anything other than TRUE warns.
`allow_duplicate_keys`	Deprecated and ignored. Duplicate keys are an error when `strict = TRUE` and resolve last-write-wins when `strict = FALSE`, as the spec requires. Supplying it warns.
`indent`	Integer. Spaces per indentation level (the spec's `indentSize`, default 2).

Details

Nesting deeper than 1000 levels is an error. Strings containing NUL (`\u0000`) cannot be represented in R and are an error.

With `strict = FALSE`, a declared length that does not match the items found is accepted with a warning (the first five are reported).

Value

R object representing the parsed TOON data:

- object -> named list (an empty object is a named list of length 0)
- array -> list, or atomic vector when simplified
- string -> character; `true`/`false` -> logical; `null` -> `NULL`
- number -> integer when it is a whole number between -2147483647 and 2147483647, otherwise double. Numbers beyond the double range become `-Inf`/`Inf`.

Examples

```
from_toon("name: Alice\nage: 30")

from_toon("[3]: 1,2,3")

from_toon("users[2]{id,name}:\n  1,Ada\n  2,Bob")

# Comment lines are skipped
from_toon("# settings\nkey: value")
```

---

json_to_toon *Convert JSON to TOON*

Description

Convert JSON to TOON

Usage

```
json_to_toon(
  json,
  pretty = TRUE,
  strict = TRUE,
  allow_comments = TRUE,
  delimiter = ","
)
```

Arguments

`json`	Character scalar containing JSON.
`pretty`	Deprecated and ignored (see `to_toon()`).
`strict`	Deprecated and ignored (see `to_toon()`).
`allow_comments`	Logical. Currently has no effect: JSON input cannot contain comments and the encoder never writes them.
`delimiter`	Document delimiter: `","` (default), `"\t"` or `"|"`.

Details

Requires the jsonlite package. If not installed, an error is thrown.

Value

Character scalar with class "toonlite_toon".

Examples

```
## Not run:
toon <- json_to_toon('{"name": "Alice", "age": 30}')

## End(Not run)
```

`parquet_to_toon` *Convert Parquet to TOON*

Description

Convert Parquet to TOON

Usage

```
parquet_to_toon(
  path_parquet,
  path_toon,
  tabular = TRUE,
  strict = TRUE,
  delimiter = ","
)
```

Arguments

path_parquet	Character scalar. Path to input Parquet file.
path_toon	Character scalar. Path to output TOON file.
tabular	Deprecated and ignored. Data frames are always written as a tabular TOON array; passing anything other than TRUE gives a warning.
strict	Deprecated and ignored (see to_toon()).
delimiter	Document delimiter: ",", "\t" or "|".

Details

Requires the arrow package. If not installed, an error is thrown.

Value

Invisibly returns NULL.

Examples

```
## Not run:
parquet_to_toon("data.parquet", "data.toon")

## End(Not run)
```

---

print.toonlite_toon *Print method for TOON strings*

Description

Print method for TOON strings

Usage

```
## S3 method for class 'toonlite_toon'
print(x, ...)
```

Arguments

x	TOON string object
...	Additional arguments (ignored)

Value

The input x, invisibly.

Examples

```
x <- to_toon(list(a = 1, b = 2))
print(x)
```

`read_toon` *Read TOON from file*

Description

Read TOON from file

Usage

```
read_toon(
  file,
  strict = TRUE,
  simplify = TRUE,
  allow_comments = TRUE,
  allow_duplicate_keys = TRUE,
  encoding = "UTF-8",
  indent = 2L
)
```

Arguments

`file`	Character scalar. Path to TOON file.
`strict`	Logical. If TRUE (default), apply the spec's strict decoding rules: declared lengths must match, indentation must be a multiple of `indent` with no tabs, blank lines inside arrays and duplicate keys are errors. If FALSE, the spec's five non-strict recoveries apply instead (count mismatches accepted, duplicate keys resolve last-write-wins, tab/odd indentation, blank lines and depth jumps tolerated).
`simplify`	Logical. If TRUE (default), arrays whose elements are all primitives of one type become atomic vectors (`null` becomes `NA`; integers mixed with doubles give a double vector).
`allow_comments`	Deprecated and ignored. Full-line # comments are part of TOON and are always skipped; passing anything other than TRUE warns.
`allow_duplicate_keys`	Deprecated and ignored. Duplicate keys are an error when `strict = TRUE` and resolve last-write-wins when `strict = FALSE`, as the spec requires. Supplying it warns.
`encoding`	Character. File encoding (default "UTF-8").
`indent`	Integer. Spaces per indentation level (the spec's `indentSize`, default 2).

Details

With `strict = FALSE`, a declared length that does not match the items found is accepted with a warning, as in `read_toon_df()`.

Value

R object representing the parsed TOON data; see `from_toon()`.

Examples

```
f <- tempfile(fileext = ".toon")
write_toon(list(a = 1, b = c("x", "y")), f)
read_toon(f)
```

---

read_toon_df *Read tabular TOON to data.frame*

Description

Reads a tabular array (`[N]{fields}:` followed by one row per line) into a data.frame, without building the intermediate list that `read_toon()` returns. Column types are inferred from the values: integers mixed with doubles give a double column, and anything else mixed (strings with numbers, booleans with numbers) a character column holding the values' canonical TOON text (`"true"`, `"1.5"`). Booleans mixed with numbers also warn. Nested field groups (`{id,customer{name}}`) become columns named by their dotted path (`customer.name`).

Usage

```
read_toon_df(
  file,
  key = NULL,
  strict = TRUE,
  allow_comments = TRUE,
  allow_duplicate_keys = TRUE,
  warn = TRUE,
  col_types = NULL,
  ragged_rows = c("error", "expand_warn"),
  n_mismatch = c("error", "warn"),
  max_extra_cols = Inf,
  indent = 2L
)
```

Arguments

file	Character scalar. Path to TOON file.
key	Character scalar or NULL. If NULL, the document root must be a tabular array. Otherwise the root must be an object and the tabular array is read from the field key (key[N]{fields}:).
strict	Logical. If TRUE (default), apply the spec's strict decoding rules: declared row counts must match, indentation must be a multiple of indent, tabs in indentation and blank lines between rows are errors. If FALSE, the spec's non-strict recoveries apply (a count mismatch warns).
allow_comments	Deprecated and ignored. Full-line # comments are part of TOON and are always skipped; passing anything other than TRUE warns.
allow_duplicate_keys	Deprecated and ignored; it has no effect on tabular reads. Passing anything other than TRUE gives a warning.
warn	Logical. If TRUE (default), emit warnings for recoveries such as a row count mismatch accepted under n_mismatch = "warn", and for columns that mix booleans and numbers.
col_types	Named character vector specifying column types: "logical", "integer", "double", or "character". A "character" column keeps unquoted values exactly as written.
ragged_rows	Character. How to handle rows whose width differs from the header: • "error" (default, as the spec requires). • "expand_warn": toonlite extension, not TOON conformant. Fill missing fields with NA, add columns V<n> for extra fields and warn.
n_mismatch	Character. How to handle a declared row count [N] that differs from the rows found, in strict mode: • "error" (default, as the spec requires). • "warn": toonlite extension, not TOON conformant. Return the rows found and warn.
max_extra_cols	Numeric. Maximum number of columns ragged_rows = "expand_warn" may add. Default Inf (no limit).
indent	Integer. Spaces per indentation level (the spec's indentSize, default 2).

Value

A base data.frame.

Examples

```
f <- tempfile(fileext = ".toon")
writeLines(c("[2]{id,name}:", "  1,Alice", "  2,Bob"), f)
read_toon_df(f)

writeLines(c("users[2]{id,name}:", "  1,Alice", "  2,Bob"), f)
read_toon_df(f, key = "users")
```

streaming *Streaming TOON I/O*

Description

Functions for streaming TOON data in batches.

tabular-io *Tabular TOON I/O (data.frame)*

Description

Functions for reading and writing tabular TOON data as data.frames.

to_toon *Serialize R object to TOON*

Description

Encodes an R value as a TOON document following spec v4.4: canonical numbers, minimal quoting, inline primitive arrays, tabular arrays for uniform objects (with nested field groups) and keyed tables for objects of uniform objects. The output has no trailing newline.

Usage

```
to_toon(
  x,
  pretty = TRUE,
  indent = 2L,
  strict = TRUE,
  allow_comments = FALSE,
  delimiter = ","
)
```

Arguments

x	R object to serialize.
pretty	Deprecated and ignored: output is always multi-line. `pretty = FALSE` used to produce invalid TOON; passing anything other than TRUE warns.
indent	Integer. Spaces per indentation level (the spec's `indentSize`, default 2).
strict	Deprecated and ignored: NaN and Inf are always written as `null`, as the spec requires. Supplying it warns.
allow_comments	Deprecated and ignored: encoders never emit comments. Passing TRUE warns.
delimiter	Document delimiter for inline arrays and tabular rows: `","` (default), `"\t"` or `"|"`; `"comma"`, `"tab"` and `"pipe"` also work.

Details

Nesting deeper than 1000 levels is an error, as it is when reading, so everything written can be read back.

R values map to the JSON data model as follows:

- `NULL`, `NA`, `NaN`, `Inf` -> `null`
- a length-1 atomic vector -> a primitive; wrap it in `I()` to write a one-element array
- other atomic vectors -> arrays (names are dropped)
- a list with names -> an object (names must be unique); a list without names -> an array
- a data.frame -> an array of row objects, written in tabular form when every column is atomic (data.frame columns become nested field groups)
- factor -> its level; Date -> `"YYYY-MM-DD"`; POSIXct -> `"YYYY-MM-DDTHH:MM:SSZ"` in UTC (whole seconds); `bit64::integer64` -> integer
- doubles use the shortest text that reads back to the same value (`1` not `1.0`, `0.1`, `1e-7`)
- other types (functions, environments, raw, complex) -> `null`

Value

Character scalar with class "toonlite_toon" containing the TOON document.

Examples

```
to_toon(list(name = "Alice", age = 30, tags = c("a", "b")))

to_toon(data.frame(id = 1:2, name = c("Ada", "Bob")))

to_toon(list(x = I(1)), delimiter = "|")
```

---

toon_info *Get TOON file info*

Description

Decodes the file and counts its arrays and objects.

Usage

```
toon_info(file, allow_comments = TRUE, strict = TRUE, indent = 2L)
```

Arguments

file	Character scalar. Path to TOON file.
allow_comments	Deprecated and ignored. Comment lines are always skipped; passing anything other than TRUE warns.
strict	Logical. If TRUE (default), apply the spec's strict decoding rules: declared lengths must match, indentation must be a multiple of indent with no tabs, blank lines inside arrays and duplicate keys are errors. If FALSE, the spec's five non-strict recoveries apply instead (count mismatches accepted, duplicate keys resolve last-write-wins, tab/odd indentation, blank lines and depth jumps tolerated).
indent	Integer. Spaces per indentation level (the spec's indentSize, default 2).

Value

A list with components:

- array_count: Integer. Number of arrays in the document.
- object_count: Integer. Number of objects in the document (including the rows of tabular arrays).
- has_tabular: Logical. Whether the document contains a tabular array.
- declared_rows: Integer or NA. Rows of the first tabular array.

Examples

```
f <- tempfile(fileext = ".toon")
write_toon(list(users = data.frame(id = 1:3)), f)
toon_info(f)
```

---

toon_peek *Peek at TOON file structure*

Description

Looks at the first n lines only, without decoding or validating the document.

Usage

```
toon_peek(file, n = 50L, allow_comments = TRUE)
```

Arguments

file	Character scalar. Path to TOON file.
n	Integer. Number of lines to preview (default 50).
allow_comments	Deprecated and ignored. Comment lines are always skipped; passing anything other than TRUE warns.

Value

A list with components:

- type: Character. The root form: "object", "array", "tabular_array", "primitive" or "unknown" (no content in the first n lines, or not a valid root).
- first_keys: Character vector. Up to five keys of a root object (entry keys for a keyed table).
- preview: Character vector. The first n lines as written.

Examples

```
f <- tempfile(fileext = ".toon")
write_toon(list(name = "Ada", tags = c("x", "y")), f)
toon_peek(f)
```

---

toon_stream_items *Stream array items*

Description

Decodes the items of an array one batch at a time and hands each batch to `callback`, so arrays larger than memory can be processed. Items can be any TOON value; rows of a tabular array arrive as row objects (named lists). For a data.frame per batch use `toon_stream_rows()`.

Usage

```
toon_stream_items(
  file,
  key = NULL,
  callback,
  batch_size = 1000L,
  strict = TRUE,
  allow_comments = TRUE,
  allow_duplicate_keys = TRUE,
  warn = TRUE,
  simplify = TRUE,
  indent = 2L
)
```

Arguments

`file`	Character scalar. Path to TOON file.
`key`	Character scalar or NULL. If NULL, the document root must be an array. Otherwise the root must be an object and the array is read from its field `key`.
`callback`	Function called with each batch: a list of up to `batch_size` items, each decoded as by `read_toon()`.

batch_size	Integer. Number of items per batch (default 1000).
strict	Logical. If TRUE (default), apply the spec's strict decoding rules: declared lengths must match, indentation must be a multiple of indent with no tabs, blank lines inside arrays and duplicate keys are errors. If FALSE, the spec's five non-strict recoveries apply instead (count mismatches accepted, duplicate keys resolve last-write-wins, tab/odd indentation, blank lines and depth jumps tolerated).
allow_comments	Deprecated and ignored. Full-line # comments are part of TOON and are always skipped; passing anything other than TRUE warns.
allow_duplicate_keys	Deprecated and ignored. Duplicate keys are an error when strict = TRUE and resolve last-write-wins when strict = FALSE, as the spec requires. Supplying it warns.
warn	Logical. If TRUE (default), warn when strict = FALSE accepts a declared length that does not match the items found, as read_toon() does. Warnings are given after the last batch.
simplify	Logical. If TRUE (default), arrays whose elements are all primitives of one type become atomic vectors (null becomes NA; integers mixed with doubles give a double vector).
indent	Integer. Spaces per indentation level (the spec's indentSize, default 2).

Details

The rest of the document is decoded and validated as well, but only the streamed items are handed out. An error found after some batches were delivered (a declared count that does not match in strict mode, or invalid content later in the file) is raised after those callbacks ran.

Value

Invisibly returns NULL.

Examples

```
f <- tempfile(fileext = ".toon")
write_toon(list(items = list(1, "two", list(x = 3))), f)
toon_stream_items(f, key = "items", batch_size = 2, callback = function(batch) {
  str(batch)
})
```

---

toon_stream_rows *Stream tabular TOON rows*

Description

Reads a tabular array in batches of rows, calling callback with each batch as a data.frame. Memory use is bounded by the batch size, not the file size. Rows are typed exactly as in read_toon_df(), per batch.

Usage

```
toon_stream_rows(
  file,
  key = NULL,
  callback,
  batch_size = 10000L,
  strict = TRUE,
  allow_comments = TRUE,
  allow_duplicate_keys = TRUE,
  warn = TRUE,
  col_types = NULL,
  ragged_rows = c("error", "expand_warn"),
  n_mismatch = c("error", "warn"),
  max_extra_cols = Inf,
  indent = 2L
)
```

Arguments

file	Character scalar. Path to TOON file.
key	Character scalar or NULL. If NULL, the document root must be a tabular array. Otherwise the root must be an object and the tabular array is read from the field key (key[N]{fields}:).
callback	Function. Called with each batch as a data.frame. Return value is ignored; errors propagate and stop the read.
batch_size	Integer. Number of rows per batch (default 10000).
strict	Logical. If TRUE (default), apply the spec's strict decoding rules: declared row counts must match, indentation must be a multiple of indent, tabs in indentation and blank lines between rows are errors. If FALSE, the spec's non-strict recoveries apply (a count mismatch warns).
allow_comments	Deprecated and ignored. Full-line # comments are part of TOON and are always skipped; passing anything other than TRUE warns.
allow_duplicate_keys	Deprecated and ignored; it has no effect on tabular reads. Passing anything other than TRUE gives a warning.
warn	Logical. If TRUE (default), emit warnings for recoveries such as a row count mismatch accepted under n_mismatch = "warn", and for columns that mix booleans and numbers.
col_types	Named character vector specifying column types: "logical", "integer", "double", or "character". A "character" column keeps unquoted values exactly as written.
ragged_rows	Character. How to handle rows whose width differs from the header: • "error" (default, as the spec requires). • "expand_warn": toonlite extension, not TOON conformant. Fill missing fields with NA, add columns V<n> for extra fields and warn.

n_mismatch	Character. How to handle a declared row count [N] that differs from the rows found, in strict mode: • "error" (default, as the spec requires). • "warn": toonlite extension, not TOON conformant. Return the rows found and warn.
max_extra_cols	Numeric. Maximum number of columns ragged_rows = "expand_warn" may add. Default Inf (no limit).
indent	Integer. Spaces per indentation level (the spec's indentSize, default 2).

Details

The whole table is checked as it is read; a declared row count that does not match is only known at the end, so it is reported after the last batch. Column order is stable within a stream; with ragged_rows = "expand_warn", later batches may have extra columns.

Value

Invisibly returns NULL.

Examples

```
f <- tempfile(fileext = ".toon")
writeLines(c("[3]{id,name}:", "  1,Alice", "  2,Bob", "  3,Carol"), f)
toon_stream_rows(f, callback = function(batch) print(nrow(batch)),
                 batch_size = 2)
```

---

toon_stream_write_rows

Stream write tabular rows

Description

Stream write tabular rows

Usage

```
toon_stream_write_rows(
  file,
  schema,
  row_source,
  batch_size = 10000L,
  indent = 2L,
  delimiter = ","
)
```

Arguments

file	Character scalar. Path to output file.
schema	Character vector of column names.
row_source	Function called with no arguments that returns the next batch as a data.frame, or NULL to end. It decides how many rows each batch has.
batch_size	Deprecated and ignored (it was never passed to row_source). Supplying it gives a warning.
indent	Integer. Spaces per indentation level (default 2).
delimiter	Document delimiter: ",", "\t" or "|".

Details

Writes a tabular TOON array without holding all rows in memory. Cells are formatted exactly as write_toon_df() formats them, so the file reads back the same way; batch columns must be atomic vectors. The row count goes in the header, so rows are first written to a temporary file and file is only written once row_source() returns NULL. If row_source() fails, file is left untouched. With no rows, file holds the empty array [].

Value

Invisibly returns the number of rows written.

Examples

```
f <- tempfile(fileext = ".toon")
i <- 0
row_source <- function() {
  i <<- i + 1
  if (i > 3) return(NULL)
  data.frame(x = i, y = i * 2)
}
toon_stream_write_rows(f, schema = c("x", "y"), row_source = row_source)
readLines(f, warn = FALSE)
```

---

toon_to_csv *Convert TOON to CSV*

Description

Convert TOON to CSV

Usage

```
toon_to_csv(
  path_toon,
  path_csv,
  key = NULL,
  strict = TRUE,
  allow_comments = TRUE,
  warn = TRUE
)
```

Arguments

path_toon	Character scalar. Path to input TOON file.
path_csv	Character scalar. Path to output CSV file.
key	Character scalar or NULL. If non-NULL, extract tabular array at root[key].
strict	Logical. If TRUE (default), apply the spec's strict decoding rules; FALSE applies its non-strict recoveries (see from_toon()).
allow_comments	Deprecated and ignored: comment lines are always skipped (see read_toon_df()).
warn	Logical. If TRUE (default), emit warnings.

Value

Invisibly returns NULL.

Examples

```
## Not run:
toon_to_csv("data.toon", "data.csv")

## End(Not run)
```

---

toon_to_feather *Convert TOON to Feather*

Description

Convert TOON to Feather

Usage

```
toon_to_feather(
  path_toon,
  path_feather,
  key = NULL,
  strict = TRUE,
```

```
  allow_comments = TRUE,
  warn = TRUE
)
```

Arguments

path_toon	Character scalar. Path to input TOON file.
path_feather	Character scalar. Path to output Feather file.
key	Character scalar or NULL. If non-NULL, extract tabular array.
strict	Logical. If TRUE (default), apply the spec's strict decoding rules; FALSE applies its non-strict recoveries (see from_toon()).
allow_comments	Deprecated and ignored: comment lines are always skipped (see read_toon_df()).
warn	Logical. If TRUE (default), emit warnings.

Details

Requires the arrow package. If not installed, an error is thrown.

Value

Invisibly returns NULL.

Examples

```
## Not run:
toon_to_feather("data.toon", "data.feather")

## End(Not run)
```

---

toon_to_json *Convert TOON to JSON*

Description

Convert TOON to JSON

Usage

```
toon_to_json(toon, pretty = FALSE, strict = TRUE, allow_comments = TRUE)
```

Arguments

toon	Character scalar containing TOON.
pretty	Logical. If TRUE, use multi-line JSON formatting (default FALSE).
strict	Logical. If TRUE (default), apply the spec's strict decoding rules; FALSE applies its non-strict recoveries (see from_toon()).
allow_comments	Deprecated and ignored: comment lines are always skipped (see from_toon()).

Details

Requires the jsonlite package. If not installed, an error is thrown.

Value

Character scalar containing JSON.

Examples

```
## Not run:
json <- toon_to_json('name: "Alice"\nage: 30')

## End(Not run)
```

`toon_to_parquet` *Convert TOON to Parquet*

Description

Convert TOON to Parquet

Usage

```
toon_to_parquet(
  path_toon,
  path_parquet,
  key = NULL,
  strict = TRUE,
  allow_comments = TRUE,
  warn = TRUE
)
```

Arguments

`path_toon`	Character scalar. Path to input TOON file.
`path_parquet`	Character scalar. Path to output Parquet file.
`key`	Character scalar or NULL. If non-NULL, extract tabular array.
`strict`	Logical. If TRUE (default), apply the spec's strict decoding rules; FALSE applies its non-strict recoveries (see `from_toon()`).
`allow_comments`	Deprecated and ignored: comment lines are always skipped (see `read_toon_df()`).
`warn`	Logical. If TRUE (default), emit warnings.

Details

Requires the arrow package. If not installed, an error is thrown.

Value

Invisibly returns NULL.

Examples

```
## Not run:
toon_to_parquet("data.toon", "data.parquet")

## End(Not run)
```

`validate_toon` *Validate TOON*

Description

Checks a document against the TOON spec (v4.4) by decoding it.

Usage

```
validate_toon(
  x,
  is_file = FALSE,
  strict = TRUE,
  allow_comments = TRUE,
  allow_duplicate_keys = TRUE,
  indent = 2L
)
```

Arguments

`x`	Character scalar containing TOON text, or file path if is_file=TRUE.
`is_file`	Logical. If TRUE, x is treated as a file path.
`strict`	Logical. If TRUE (default), apply the spec's strict decoding rules: declared lengths must match, indentation must be a multiple of `indent` with no tabs, blank lines inside arrays and duplicate keys are errors. If FALSE, the spec's five non-strict recoveries apply instead (count mismatches accepted, duplicate keys resolve last-write-wins, tab/odd indentation, blank lines and depth jumps tolerated).
`allow_comments`	Deprecated and ignored. Full-line # comments are part of TOON and are always skipped; passing anything other than TRUE warns.
`allow_duplicate_keys`	Deprecated and ignored. Duplicate keys are an error when `strict = TRUE` and resolve last-write-wins when `strict = FALSE`, as the spec requires. Supplying it warns.
`indent`	Integer. Spaces per indentation level (the spec's `indentSize`, default 2).

Details

Never throws unless there's an internal error (e.g., file unreadable).

Value

Logical scalar.

- TRUE if valid.
- FALSE if invalid, with attribute `attr(result, "error")` containing a structured error object with components: type, message, line, column, snippet, file.

Examples

```
validate_toon("key: value")

# Invalid TOON (returns FALSE with error attribute)
result <- validate_toon("items[3]: a,b")
if (!result) print(attr(result, "error")$message)
```

---

validation *TOON Validation and Formatting*

Description

Functions for validating and formatting TOON data.

---

write_toon *Write R object to TOON file*

Description

Encodes `x` exactly like `to_toon()` and writes it to `file` (UTF-8, no trailing newline) without building the whole text in memory.

Usage

```
write_toon(x, file, pretty = TRUE, indent = 2L, strict = TRUE, delimiter = ",")
```

Arguments

x	R object to serialize.
file	Character scalar. Path to output file.
pretty	Deprecated and ignored: output is always multi-line. pretty = FALSE used to produce invalid TOON; passing anything other than TRUE warns.
indent	Integer. Spaces per indentation level (the spec's indentSize, default 2).
strict	Deprecated and ignored: NaN and Inf are always written as null, as the spec requires. Supplying it warns.
delimiter	Document delimiter for inline arrays and tabular rows: "," (default), "\t" or "|"; "comma", "tab" and "pipe" also work.

Value

Invisibly returns NULL.

Examples

```
f <- tempfile(fileext = ".toon")
write_toon(list(x = 1, y = 2), f)
readLines(f, warn = FALSE)
```

---

write_toon_df *Write data.frame to tabular TOON*

Description

Writes df as a root tabular array ([N]{fields}: and one row per line), exactly as write_toon() writes a data.frame. A data.frame with no rows is written as the empty array [].

Usage

```
write_toon_df(
  df,
  file,
  tabular = TRUE,
  pretty = TRUE,
  indent = 2L,
  strict = TRUE,
  delimiter = ","
)
```

Arguments

df	A data.frame to write.
file	Character scalar. Path to output file.
tabular	Deprecated and ignored. Data frames are always written as a tabular TOON array; passing anything other than TRUE gives a warning.
pretty	Deprecated and ignored: output is always multi-line. `pretty = FALSE` used to produce invalid TOON; passing anything other than TRUE warns.
indent	Integer. Spaces per indentation level (the spec's `indentSize`, default 2).
strict	Deprecated and ignored: NaN and Inf are always written as `null`, as the spec requires. Supplying it warns.
delimiter	Document delimiter for inline arrays and tabular rows: `","` (default), `"\t"` or `"|"`; `"comma"`, `"tab"` and `"pipe"` also work.

Value

Invisibly returns NULL.

Examples

```
f <- tempfile(fileext = ".toon")
write_toon_df(head(mtcars[, 1:4], 3), f)
readLines(f, warn = FALSE)
```

Index

as_tabular_toon, 2
assert_toon, 3

conditions, 4
conversions, 5
core-io, 5
csv_to_toon, 5

feather_to_toon, 6
format_toon, 7
from_toon, 8
from_toon(), *13*, *23–25*

I(), *16*

json_to_toon, 9

parquet_to_toon, 10
print.toonlite_toon, 11

read_toon, 12
read_toon(), *13*, *18*, *19*
read_toon_df, 13
read_toon_df(), *13*, *19*, *23–25*

streaming, 15

tabular-io, 15
to_toon, 15
to_toon(), *5–7*, *10*, *11*, *27*
toon_info, 16
toon_peek, 17
toon_stream_items, 18
toon_stream_rows, 19
toon_stream_rows(), *18*
toon_stream_write_rows, 21
toon_to_csv, 22
toon_to_feather, 23
toon_to_json, 24
toon_to_parquet, 25

validate_toon, 26
validation, 27

write_toon, 27
write_toon(), *28*
write_toon_df, 28
write_toon_df(), *22*