

# Package ‘handlr’

May 8, 2026

**Title** Convert Among Citation Formats

**Description** Converts among many citation formats, including 'BibTeX', 'Citeproc', 'Codemeta', 'RDF XML', 'RIS', 'Schema.org', and 'Citation File Format'. A low level 'R6' class is provided, as well as stand-alone functions for each citation format for both read and write.

**Version** 0.3.1

**License** MIT + file LICENSE

**URL** <https://github.com/ropensci/handlr>,  
<https://docs.ropensci.org/handlr/>

**BugReports** <https://github.com/ropensci/handlr/issues>

**Encoding** UTF-8

**Language** en-US

**Imports** jsonlite, crul, xml2, urltools, mime, yaml

**Suggests** testthat, jsonld, data.table, bibtex

**RoxygenNote** 7.3.2

**X-schema.org-applicationCategory** Metadata

**X-schema.org-keywords** doi, metadata, citation, bibtex, Crossref, Crosscite, Codemeta, RIS, Citeproc, RDF, XML, JSON

**X-schema.org-isPartOf** <https://ropensci.org>

**NeedsCompilation** no

**Author** Scott Chamberlain [aut] (ORCID:  
<<https://orcid.org/0000-0003-1444-9135>>),  
Brenton M. Wiernik [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-9560-6336>>),  
Thierry Onkelinx [ctb] (ORCID: <<https://orcid.org/0000-0001-8804-4216>>)

**Maintainer** Brenton M. Wiernik <brenton@wiernik.org>

**Repository** CRAN

**Date/Publication** 2025-03-03 23:30:08 UTC

## Contents

bibtex_reader . . . . .	2
bibtex_writer . . . . .	3
c.handl . . . . .	4
cff_reader . . . . .	5
cff_reference_types . . . . .	6
cff_writer . . . . .	7
citeproc_reader . . . . .	8
citeproc_writer . . . . .	9
codemeta_reader . . . . .	10
codemeta_writer . . . . .	11
handl . . . . .	12
HandlClient . . . . .	13
handl_to_df . . . . .	18
rdf_xml_writer . . . . .	19
ris_reader . . . . .	20
ris_writer . . . . .	21
schema_org_writer . . . . .	22
<b>Index</b>	<b>24</b>

---

bibtex_reader	<i>bibtex reader</i>
---------------	----------------------

---

### Description

bibtex reader

### Usage

bibtex\_reader(x)

### Arguments

x (character) a file path or a bibtex string

### Value

an object of class handl; see [handl](#) for more

### Note

requires package bibtex, an optional package for handl

### See Also

Other readers: [cff\\_reader\(\)](#), [citeproc\\_reader\(\)](#), [codemeta\\_reader\(\)](#), [ris\\_reader\(\)](#)

Other bibtex: [bibtex\\_writer\(\)](#)

**Examples**

```

if (requireNamespace("bibtex", quietly=TRUE)) {
  (z <- system.file('extdata/crossref.bib', package = "handlr"))
  bibtex_reader(x = z)
  (z <- system.file('extdata/bibtex.bib', package = "handlr"))
  bibtex_reader(x = z)

  # many at once
  (z <- system.file('extdata/bib-many.bib', package = "handlr"))
  bibtex_reader(x = z)
}

```

---

bibtex_writer	<i>bibtex writer</i>
---------------	----------------------

---

**Description**

bibtex writer

**Usage**

```
bibtex_writer(z, key = NULL)
```

**Arguments**

z	an object of class <code>handlr</code> ; see <a href="#">handlr</a> for more
key	(character) optional bibtex key to use. if NULL we attempt try the following fields in order: <code>key</code> , <code>identifier</code> , <code>id</code> , <code>doi</code> . if you pass in output from <a href="#">bibtex_reader()</a> you're likely to have a <code>key</code> field, but otherwise probably not

**Value**

an object of class `BibEntry`

**See Also**

Other writers: [cff\\_writer\(\)](#), [citeproc\\_writer\(\)](#), [codemeta\\_writer\(\)](#), [rdf\\_xml\\_writer\(\)](#), [ris\\_writer\(\)](#), [schema\\_org\\_writer\(\)](#)

Other bibtex: [bibtex\\_reader\(\)](#)

**Examples**

```

(z <- system.file('extdata/citeproc.json', package = "handlr"))
(tmp <- citeproc_reader(z))
bibtex_writer(z = tmp)
cat(bibtex_writer(z = tmp), sep = "\n")

# give a bibtex key

```

```
cat(bibtex_writer(tmp, "foobar89"), sep = "\n")

# many at once
if (requireNamespace("bibtex", quietly=TRUE)) {
  (z <- system.file('extdata/bib-many.bib', package = "handlr"))
  out <- bibtex_reader(x = z)
  bibtex_writer(out)
}
```

---

c.handl

*combine many handl objects*

---

## Description

combine many handl objects

## Usage

```
## S3 method for class 'handl'
c(...)
```

## Arguments

... one or more objects of class handl; see [handl](#) for more. all inputs must be of class handl. if the first input is not of class handl, you will not get back an object of class handl

## Value

an object of class handl of length equal to number of handl objects passed in

## Examples

```
z <- system.file('extdata/crossref.ris', package = "handlr")
cr <- ris_reader(z)
z <- system.file('extdata/peerj.ris', package = "handlr")
prj <- ris_reader(z)
res <- c(cr, prj)
res
invisible(lapply(bibtex_writer(res), cat, sep = "\n\n"))
```

---

cffi_reader	<i>Citation File Format (cff) reader</i>
-------------	--

---

## Description

Citation File Format (cff) reader

## Usage

```
cffi_reader(x)
```

## Arguments

x (character) a file path or a yaml string

## Details

CFF only supports one citation, so many will always be FALSE.

Required fields:

- CFF **v1.1.0**: cff-version, version, message, date-released, title, authors.
- CFF **v1.2.0**: cff-version, message, title, authors.

We'll stop with error if any of these are missing.

You can though have many references in your CFF file associated with the citation. references is an optional component in cff files. If included, we check the following:

- each reference must have the 3 required fields: type, authors, title
- type must be in the allowed set, see [cff\\_reference\\_types](#)
- the elements within authors must each be an entity or person object <https://github.com/citation-file-format/citation-file-format#entity-objects> <https://github.com/citation-file-format/citation-file-format#person-objects>
- title must be a string

## Value

an object of class handl; see [handl](#) for more

## References

CFF format: <https://github.com/citation-file-format/citation-file-format>

## See Also

Other readers: [bibtex\\_reader\(\)](#), [citeproc\\_reader\(\)](#), [codemeta\\_reader\(\)](#), [ris\\_reader\(\)](#)

Other cff: [cff\\_writer\(\)](#)

**Examples**

```
(z <- system.file("extdata/citation.cff", package = "handlr"))
res <- cff_reader(x = z)
res
res$cff_version
res$software_version
res$message
res$id
res$doi
res$title
res$author
res$references

# no references
(z <- system.file("extdata/citation-norefs.cff", package = "handlr"))
out <- cff_reader(x = z)
out
out$references
```

---

cff\_reference\_types    *cff references types*

---

**Description**

cff references types

**Usage**

cff\_reference\_types

**Format**

An object of class character of length 47.

**Details**

cff citation format types for references

**References**

<https://bit.ly/2PRK1Vt>

---

cff\_writer                      *Citation File Format (cff) writer*

---

## Description

Citation File Format (cff) writer

## Usage

```
cff_writer(  
  z,  
  path = NULL,  
  message = "Please cite the following works when using this software."  
)
```

## Arguments

z	an object of class <code>handl</code> ; see <a href="#">handl</a> for more
path	a file path or connection; default: <code>stdout()</code>
message	a message to display. Defaults to "Please cite the following works when using this software."

## Details

uses `yaml::write_yaml` to write to yaml format that CFF uses

## Value

text if one cff citation or list of many

## Converting to CFF from other formats

CFF has required fields that can't be missing. This means that converting from other citation types to CFF will likely require adding the required CFF fields manually. Adding fields to a `handl` object is easy: it's really just an R list so add named elements to it. The required CFF fields are:

- CFF v1.1.0:
  - cff-version: add `cff_version`
  - message: add `message`
  - version: add `software_version`
  - title: add `title`
  - authors: add `author`
  - date-released: add `date_published`
- CFF v1.2.0:
  - Only fields `cff-version`, `message`, `title` and `authors` are required.

If `cff_version` is not provided, the value by default is "1.2.0".

## References

CFF format: <https://github.com/citation-file-format/citation-file-format>

## See Also

Other writers: [bibtex\\_writer\(\)](#), [citeproc\\_writer\(\)](#), [codemeta\\_writer\(\)](#), [rdf\\_xml\\_writer\(\)](#), [ris\\_writer\(\)](#), [schema\\_org\\_writer\(\)](#)

Other cff: [cff\\_reader\(\)](#)

## Examples

```
(z <- system.file('extdata/citation.cff', package = "handlr"))
res <- cff_reader(x = z)
res
unclass(res)
cff_writer(res)
cat(cff_writer(res))
f <- tempfile()
cff_writer(res, f)
readLines(f)
unlink(f)

# convert from a different citation format
## see "Converting to CFF from other formats" above
z <- system.file('extdata/citeproc.json', package = "handlr")
w <- citeproc_reader(x = z)
# cff_writer(w) # fails unless we add required fields
w$cff_version <- "1.1.0"
w$software_version <- "2.5"
w$title <- "A cool library"
w$date_published <- "2017-12-18"
cff_writer(w)
cat(cff_writer(w))
```

---

citeproc\_reader

*citeproc reader*

---

## Description

citeproc reader

## Usage

```
citeproc_reader(x)
```

## Arguments

x (character) a file path or string

**Value**

an object of class `handl`; see [handl](#) for more

**See Also**

Other readers: [bibtex\\_reader\(\)](#), [cff\\_reader\(\)](#), [codemeta\\_reader\(\)](#), [ris\\_reader\(\)](#)

Other citeproc: [citeproc\\_writer\(\)](#)

**Examples**

```
# single
z <- system.file('extdata/citeproc.json', package = "handlr")
citeproc_reader(x = z)
w <- system.file('extdata/citeproc2.json', package = "handlr")
citeproc_reader(x = w)

# many
z <- system.file('extdata/citeproc-many.json', package = "handlr")
citeproc_reader(x = z)
```

---

<code>citeproc_writer</code>	<i>citeproc writer</i>
------------------------------	------------------------

---

**Description**

citeproc writer

**Usage**

```
citeproc_writer(z, auto_unbox = TRUE, pretty = TRUE, ...)
```

**Arguments**

<code>z</code>	an object of class <code>handl</code> ; see <a href="#">handl</a> for more
<code>auto_unbox</code>	(logical) automatically "unbox" all atomic vectors of length 1 (default: TRUE). passed to <a href="#">jsonlite::toJSON()</a>
<code>pretty</code>	(logical) adds indentation whitespace to JSON output (default: TRUE), passed to <a href="#">jsonlite::toJSON()</a>
<code>...</code>	further params passed to <a href="#">jsonlite::toJSON()</a>

**Value**

citeproc as JSON

**See Also**

Other writers: [bibtex\\_writer\(\)](#), [cff\\_writer\(\)](#), [codemeta\\_writer\(\)](#), [rdf\\_xml\\_writer\(\)](#), [ris\\_writer\(\)](#), [schema\\_org\\_writer\(\)](#)

Other citeproc: [citeproc\\_reader\(\)](#)

**Examples**

```
z <- system.file('extdata/citeproc.json', package = "handlr")
(tmp <- citeproc_reader(z))
citeproc_writer(z = tmp)
citeproc_writer(z = tmp, pretty = FALSE)
cat(ris_writer(z = tmp))

# many
z <- system.file('extdata/citeproc-many.json', package = "handlr")
w <- citeproc_reader(x = z)
citeproc_writer(w)
```

---

codemeta\_reader

*codemeta reader*

---

**Description**

codemeta reader

**Usage**

```
codemeta_reader(x)
```

**Arguments**

x (character) a file path or string (character or json)

**Value**

an object of class `handlr`; see [handlr](#) for more

**See Also**

Other readers: [bibtex\\_reader\(\)](#), [cff\\_reader\(\)](#), [citeproc\\_reader\(\)](#), [ris\\_reader\(\)](#)

Other codemeta: [codemeta\\_writer\(\)](#)

**Examples**

```
# single
(z <- system.file('extdata/codemeta.json', package = "handlr"))
codemeta_reader(x = z)

# many
(z <- system.file('extdata/codemeta-many.json', package = "handlr"))
codemeta_reader(x = z)
```

---

codemeta_writer	<i>codemeta writer</i>
-----------------	------------------------

---

**Description**

codemeta writer

**Usage**

```
codemeta_writer(z, auto_unbox = TRUE, pretty = TRUE, ...)
```

**Arguments**

<code>z</code>	an object of class <code>handl</code> ; see <a href="#">handl</a> for more
<code>auto_unbox</code>	(logical) automatically "unbox" all atomic vectors of length 1 (default: TRUE). passed to <a href="#">jsonlite::toJSON()</a>
<code>pretty</code>	(logical) adds indentation whitespace to JSON output (default: TRUE), passed to <a href="#">jsonlite::toJSON()</a>
<code>...</code>	further params passed to <a href="#">jsonlite::toJSON()</a>

**Value**

an object of class `json`

**See Also**

Other writers: [bibtex\\_writer\(\)](#), [cff\\_writer\(\)](#), [citeproc\\_writer\(\)](#), [rdf\\_xml\\_writer\(\)](#), [ris\\_writer\(\)](#), [schema\\_org\\_writer\(\)](#)

Other codemeta: [codemeta\\_reader\(\)](#)

**Examples**

```
if (requireNamespace("bibtex", quietly=TRUE)) {
  (x <- system.file('extdata/crossref.bib', package = "handlr"))
  (z <- bibtex_reader(x))
  codemeta_writer(z)
}
```

```
# many citeproc to schema
z <- system.file('extdata/citeproc-many.json', package = "handlr")
w <- citeproc_reader(x = z)
codemeta_writer(w)
codemeta_writer(w, pretty = FALSE)
```

---

handl	<i>handl object</i>
-------	---------------------

---

## Description

handl object

## Details

A handl object is what's returned from the reader functions, and what is passed to the writer functions. The handl object is a list, but using the `print.handl` method makes it look something like:

```
<handl>
  from: codemeta
  many: TRUE
  count: 2
  first 10
    id/doi: https://doi.org/10.5063%2ff1m61h5x
    id/doi: https://doi.org/10.5063%2ff1m61h5x
```

You can always `unclass()` the object to get the list itself.

The handl object follows <https://github.com/datacite/bolognese>, which uses the Crosscite format as its internal representation. Note that we don't currently support writing to or reading from Crosscite.

Details on each entry are stored in the named attributes:

- `from`: the data type the citations come from
- `many`: is there more than 1 citation?
- `count`: number of citations
- finally, some details of the first 10 are printed

If you have a handl object with 1 citation, it is a named list that you can access with normal key indexing. If the result is `length > 1`, the data is an unnamed list of named lists; the top level list is unnamed, with each list within it being named.

Each named list should have the following components:

- `key`: (string) a key for the citation, e.g., in a bibtex file
- `id`: (string) an id for the work being referenced, often a DOI
- `type`: (string) type of work

- `bibtex_type`: (string) bibtex type
- `citeproc_type`: (string) citeproc type
- `ris_type`: (string) ris type
- `resource_type_general`
- `additional_type`: (string) additional type
- `doi`: (string) DOI
- `b_url`: (string) additional URL
- `title`: (string) the title of the work
- `author`: (list) authors, with each author a named list of
  - `type`: type, typically "Person"
  - `name`: full name
  - `givenName`: given (first) name
  - `familyName`: family (last) name
- `publisher`: (string) the publisher name
- `is_part_of`: (list) what the work is published in, or part of, a named list with:
  - `type`: (string) the type of work
  - `title`: (string) title of the work, often a journal or edited book
  - `issn`: (string) the ISSN
- `date_published`: (string)
- `volume`: (string) the volume, if applicable
- `first_page`: (string) the first page
- `last_page`: (string) the last page
- `description`: (string) description of the work, often an abstract
- `license`: (string) license of the work, a named list
- `state`: (string) the state of the list
- `software_version`: (string) software version

Citeproc formats may have extra fields that begin with `cs1_`

---

HandlrClient

*HandlrClient*

---

## Description

handlr client, read and write to and from all citation formats

**Details**

The various inputs to the `x` parameter are handled in different ways:

- `file`: contents read from file, we grab file extension, and we guess format based on combination of contents and file extension because file extensions may belie what's in the file
- `string`: string read in, and we guess format based on contents of the string
- `DOI`: we request citeproc-json format from the Crossref API
- `DOI url`: we request citeproc-json format from the Crossref API

**Public fields**

`path` (character) non-empty if file path passed to initialize

`string` (character) non-empty if string (non-file) passed to initialize

`parsed` after `read()` is run, the parsed content

`file` (logical) TRUE if a file passed to initialize, else FALSE

`ext` (character) the file extension

`format_guessed` (character) the guessed file format

`doi` (character) the DOI, if any found

**Methods****Public methods:**

- `HandlrClient#print()`
- `HandlrClient$new()`
- `HandlrClient$read()`
- `HandlrClient$write()`
- `HandlrClient$as_df()`
- `HandlrClient$clone()`

**Method** `print()`: print method for `HandlrClient` objects

*Usage:*

```
HandlrClient#print(x, ...)
```

*Arguments:*

`x` self

`...` ignored

**Method** `new()`: Create a new `HandlrClient` object

*Usage:*

```
HandlrClient$new(x, format = NULL, ...)
```

*Arguments:*

`x` (character) a file path (the file must exist), a string containing contents of the citation, a DOI, or a DOI as a URL. See Details.

format (character) one of citeproc, ris, bibtex, codemeta, cff, or NULL. If NULL, we attempt to guess the format, and error if we can not guess

... curl options passed on to [crul::verb-GET](#)

*Returns:* A new HandlrClient object

**Method** read(): read input

*Usage:*

```
HandlrClient$read(format = NULL, ...)
```

*Arguments:*

format (character) one of citeproc, ris, bibtex, codemeta, cff, or NULL. If NULL, we attempt to guess the format, and error if we can not guess

... further args to the writer fxn, if any

**Method** write(): write to std out or file

*Usage:*

```
HandlrClient$write(format, file = NULL, ...)
```

*Arguments:*

format (character) one of citeproc, ris, bibtex, schema\_org, rdxml, codemeta, or cff

file a file path, if NULL to stdout. for format=ris, number of files must equal number of ris citations

... further args to the writer fxn, if any

**Method** as\_df(): convert data to a data.frame using [handl\\_to\\_df\(\)](#)

*Usage:*

```
HandlrClient$as_df()
```

*Returns:* a data.frame

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

```
HandlrClient$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## Note

If \$parsed is NULL then it's likely \$read() has not been run - in which case we attempt to run \$read() to populate \$parsed

## Examples

```
# read() can be run with format specified or not
# if format not given, we attempt to guess the format and then read
z <- system.file('extdata/citeproc.json', package = "handlr")
(x <- HandlrClient$new(x = z))
x$read()
```

```

x$read("citeproc")
x$parsed

# you can run read() then write()
# or just run write(), and read() will be run for you if possible
z <- system.file('extdata/citeproc.json', package = "handlr")
(x <- HandlrClient$new(x = z))
cat(x$write("ris"))

# read from a DOI as a url
if (interactive()) {
  (x <- HandlrClient$new('https://doi.org/10.7554/elife.01567'))
  x$parsed
  x$read()
  x$parsed
  x$write('bibtex')
}

# read from a DOI
if (interactive()) {
  (x <- HandlrClient$new('10.7554/elife.01567'))
  x$parsed
  x$read()
  x$write('bibtex')
}

# read in citeproc, write out bibtex
z <- system.file('extdata/citeproc.json', package = "handlr")
(x <- HandlrClient$new(x = z))
x$path
x$ext
x$read("citeproc")
x$parsed
x$write("bibtex")
f <- tempfile(fileext = ".bib")
x$write("bibtex", file = f)
readLines(f)
unlink(f)

# read in ris, write out ris
z <- system.file('extdata/peerj.ris', package = "handlr")
(x <- HandlrClient$new(x = z))
x$path
x$format_guessed
x$read("ris")
x$parsed
x$write("ris")
cat(x$write("ris"))

# read in bibtex, write out ris
(z <- system.file('extdata/bibtex.bib', package = "handlr"))
(x <- HandlrClient$new(x = z))
x$path

```

```
x$format_guessed
if (requireNamespace("bibtex", quietly = TRUE)) {
x$read("bibtex")
x$parsed
x$write("ris")
cat(x$write("ris"))
}

# read in bibtex, write out RDF XML
if (requireNamespace("bibtex", quietly = TRUE) && interactive()) {
  (z <- system.file('extdata/bibtex.bib', package = "handlr"))
  (x <- HandlrClient$new(x = z))
  x$path
  x$format_guessed
  x$read("bibtex")
  x$parsed
  x$write("rdfxml")
  cat(x$write("rdfxml"))
}

# codemeta
(z <- system.file('extdata/codemeta.json', package = "handlr"))
(x <- HandlrClient$new(x = z))
x$path
x$format_guessed
x$read("codemeta")
x$parsed
x$write("codemeta")

# cff: Citation File Format
(z <- system.file('extdata/citation.cff', package = "handlr"))
(x <- HandlrClient$new(x = z))
x$path
x$format_guessed
x$read("cff")
x$parsed
x$write("codemeta")

# > 1 citation
z <- system.file('extdata/citeproc-many.json', package = "handlr")
(x <- HandlrClient$new(x = z))
x$parsed
x$read()
x$parsed
## schmea org
x$write("schema_org")
## bibtex
x$write("bibtex")
## bibtex to file
f <- tempfile(fileext=".bib")
x$write("bibtex", f)
readLines(f)
unlink(f)
```

```
## to RIS
x$write("ris")
### only one per file, so not combined
files <- replicate(2, tempfile(fileext=".ris"))
x$write("ris", files)
lapply(files, readLines)

# handle strings instead of files
z <- system.file('extdata/citeproc-crossref.json', package = "handlr")
(x <- HandlrClient$new(x = readLines(z)))
x$read("citeproc")
x$parsed
cat(x$write("bibtex"), sep = "\n")
```

---

handl\_to\_df

*handl to data.frame conversion*


---

## Description

handl to data.frame conversion

## Usage

```
handl_to_df(x)
```

## Arguments

x                    an object of class handl

## Value

data.frame with column following [handl](#), with as many rows as there are citations

## Note

requires the Suggested package `data.table`

## Examples

```
z <- system.file('extdata/crossref.ris', package = "handlr")
res <- ris_reader(z)
handl_to_df(res)

(x <- HandlrClient$new(x = z))
x$sas_df() # empty data.frame
x$read()
x$sas_df() # data.frame with citation data

if (requireNamespace("bibtex", quietly=TRUE)) {
  (z <- system.file('extdata/bib-many.bib', package = "handlr"))
```

```
res2 <- bibtex_reader(x = z)
handl_to_df(res2)
}
```

---

rdf_xml_writer	<i>RDF XML writer</i>
----------------	-----------------------

---

## Description

RDF XML writer

## Usage

```
rdf_xml_writer(z, ...)
```

## Arguments

`z` an object of class `handl`; see [handl](#) for more  
`...` further params passed to `jsonld::jsonld_to_rdf()`

## Details

package `jsonld` required for this writer

## Value

RDF XML

## See Also

Other writers: [bibtex\\_writer\(\)](#), [cff\\_writer\(\)](#), [citeproc\\_writer\(\)](#), [codemeta\\_writer\(\)](#), [ris\\_writer\(\)](#), [schema\\_org\\_writer\(\)](#)

## Examples

```
if (require("jsonld") && interactive()) {
  library("jsonld")
  z <- system.file('extdata/citeproc.json', package = "handlr")
  (tmp <- citeproc_reader(z))

  if (requireNamespace("bibtex", quietly=TRUE)) {
    (z <- system.file('extdata/bibtex.bib', package = "handlr"))
    (tmp <- bibtex_reader(z))
    rdf_xml_writer(z = tmp)
    cat(rdf_xml_writer(z = tmp))
  }
}
```

---

ris_reader	<i>ris reader (Research Information Systems)</i>
------------	--

---

**Description**

ris reader (Research Information Systems)

**Usage**

```
ris_reader(x)
```

**Arguments**

x (character) a file path or string

**Value**

an object of class `handl`; see [handl](#) for more

**References**

RIS tags [https://en.wikipedia.org/wiki/RIS\\_\(file\\_format\)](https://en.wikipedia.org/wiki/RIS_(file_format))

**See Also**

Other readers: [bibtex\\_reader\(\)](#), [cff\\_reader\(\)](#), [citeproc\\_reader\(\)](#), [codemeta\\_reader\(\)](#)

Other ris: [ris\\_writer\(\)](#)

**Examples**

```
z <- system.file('extdata/crossref.ris', package = "handlr")
ris_reader(z)

z <- system.file('extdata/peerj.ris', package = "handlr")
ris_reader(z)

z <- system.file('extdata/plos.ris', package = "handlr")
ris_reader(z)

# from a string
z <- system.file('extdata/crossref.ris', package = "handlr")
my_string <- ris_writer(ris_reader(z))
class(my_string)
ris_reader(my_string)

# many
z <- system.file('extdata/multiple-eg.ris', package = "handlr")
ris_reader(z)
```

---

ris_writer	<i>ris writer (Research Information Systems)</i>
------------	--

---

## Description

ris writer (Research Information Systems)

## Usage

```
ris_writer(z)
```

## Arguments

z                    an object of class handl; see [handl](#) for more

## Value

text if one RIS citation or list of many

## References

RIS tags [https://en.wikipedia.org/wiki/RIS\\_\(file\\_format\)](https://en.wikipedia.org/wiki/RIS_(file_format))

## See Also

Other writers: [bibtex\\_writer\(\)](#), [cff\\_writer\(\)](#), [citeproc\\_writer\(\)](#), [codemeta\\_writer\(\)](#), [rdf\\_xml\\_writer\(\)](#), [schema\\_org\\_writer\(\)](#)

Other ris: [ris\\_reader\(\)](#)

## Examples

```
# from a RIS file
z <- system.file('extdata/crossref.ris', package = "handlr")
tmp <- ris_reader(z)
cat(ris_writer(z = tmp))

# peerj
z <- system.file('extdata/peerj.ris', package = "handlr")
tmp <- ris_reader(z)
cat(ris_writer(z = tmp))

# plos
z <- system.file('extdata/plos.ris', package = "handlr")
tmp <- ris_reader(z)
cat(ris_writer(z = tmp))

# elsevier
z <- system.file('extdata/elsevier.ris', package = "handlr")
tmp <- ris_reader(z)
```

```

cat(ris_writer(z = tmp))

z <- system.file('extdata/citeproc.json', package = "handlr")
res <- citeproc_reader(z)
cat(ris_writer(z = res))

# many
## combine many RIS in a handl object
z <- system.file('extdata/crossref.ris', package = "handlr")
cr <- ris_reader(z)
z <- system.file('extdata/peerj.ris', package = "handlr")
prj <- ris_reader(z)
c(cr, prj)

# many bibtex to ris via c method
if (requireNamespace("bibtex", quietly=TRUE)) {
  a <- system.file('extdata/bibtex.bib', package = "handlr")
  b <- system.file('extdata/crossref.bib', package = "handlr")
  aa <- bibtex_reader(a)
  bb <- bibtex_reader(b)
  (res <- c(aa, bb))
  cat(ris_writer(res), sep = "\n\n")
}

## many Citeproc to RIS
z <- system.file('extdata/citeproc-many.json', package = "handlr")
w <- citeproc_reader(x = z)
ris_writer(w)
cat(ris_writer(w), sep = "\n")

```

---

schema\_org\_writer

*Schema org writer*

---

## Description

Schema org writer

## Usage

```
schema_org_writer(z, auto_unbox = TRUE, pretty = TRUE, ...)
```

## Arguments

z	an object of class <code>handl</code> ; see <a href="#">handl</a> for more
auto_unbox	(logical) automatically "unbox" all atomic vectors of length 1 (default: TRUE). passed to <code>jsonlite::toJSON()</code>
pretty	(logical) adds indentation whitespace to JSON output (default: TRUE), passed to <code>jsonlite::toJSON()</code>
...	further params passed to <code>jsonlite::toJSON()</code>

**Value**

an object of class json

**See Also**

Other writers: [bibtex\\_writer\(\)](#), [cff\\_writer\(\)](#), [citeproc\\_writer\(\)](#), [codemeta\\_writer\(\)](#), [rdf\\_xml\\_writer\(\)](#), [ris\\_writer\(\)](#)

**Examples**

```
if (requireNamespace("bibtex", quietly=TRUE)) {
  (z <- system.file('extdata/bibtex.bib', package = "handlr"))
  (tmp <- bibtex_reader(z))
  schema_org_writer(tmp)
  schema_org_writer(tmp, pretty = FALSE)
}

# many citeproc to schema
z <- system.file('extdata/citeproc-many.json', package = "handlr")
w <- citeproc_reader(x = z)
schema_org_writer(w)
schema_org_writer(w, pretty = FALSE)
```

# Index

- \* **bibtex**
    - bibtex\_reader, 2
    - bibtex\_writer, 3
  - \* **cff**
    - cff\_reader, 5
    - cff\_writer, 7
  - \* **citeproc**
    - citeproc\_reader, 8
    - citeproc\_writer, 9
  - \* **codemeta**
    - codemeta\_reader, 10
    - codemeta\_writer, 11
  - \* **datasets**
    - cff\_reference\_types, 6
  - \* **rdf-xml**
    - rdf\_xml\_writer, 19
  - \* **readers**
    - bibtex\_reader, 2
    - cff\_reader, 5
    - citeproc\_reader, 8
    - codemeta\_reader, 10
    - ris\_reader, 20
  - \* **ris**
    - ris\_reader, 20
    - ris\_writer, 21
  - \* **schema\_org**
    - schema\_org\_writer, 22
  - \* **writers**
    - bibtex\_writer, 3
    - cff\_writer, 7
    - citeproc\_writer, 9
    - codemeta\_writer, 11
    - rdf\_xml\_writer, 19
    - ris\_writer, 21
    - schema\_org\_writer, 22
- bibtex\_reader, 2, 3, 5, 9, 10, 20  
bibtex\_reader(), 3  
bibtex\_writer, 2, 3, 8, 10, 11, 19, 21, 23
- c.handl, 4  
cff\_reader, 2, 5, 8–10, 20  
cff\_reference\_types, 5, 6  
cff\_writer, 3, 5, 7, 10, 11, 19, 21, 23  
citeproc\_reader, 2, 5, 8, 10, 20  
citeproc\_writer, 3, 8, 9, 9, 11, 19, 21, 23  
codemeta\_reader, 2, 5, 9, 10, 11, 20  
codemeta\_writer, 3, 8, 10, 11, 19, 21, 23  
crul::verb-GET, 15
- handl, 2–5, 7, 9–11, 12, 18–22  
handl\_to\_df, 18  
handl\_to\_df(), 15  
HandlrClient, 13
- jsonld::jsonld\_to\_rdf(), 19  
jsonlite::toJSON(), 9, 11, 22
- rdf\_xml\_writer, 3, 8, 10, 11, 19, 21, 23  
ris\_reader, 2, 5, 9, 10, 20, 21  
ris\_writer, 3, 8, 10, 11, 19, 20, 21, 23
- schema\_org\_writer, 3, 8, 10, 11, 19, 21, 22