

Package ‘cfr’

August 24, 2026

Title Generate Citation File Format ('CFF') Metadata

Version 1.4.2

Description Citation File Format ('CFF') version 1.2.0
<[doi:10.5281/zenodo.5171937](https://doi.org/10.5281/zenodo.5171937)> is a human- and machine-readable file
format for software citation metadata. Core utilities generate, read,
write and validate Citation File Format metadata for 'R' packages.

License GPL (>= 3)

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

BugReports <https://github.com/ropensci/cfr/issues>

Depends R (>= 4.1.0)

Imports cli (>= 2.0.0), desc (>= 1.3.0), jsonlite (>= 1.7.2),
jsonvalidate (>= 1.1.0), tools, utils, yaml (>= 2.2.1)

Suggests bibtex (>= 0.5.0), knitr, lifecycle, quarto, rmarkdown,
testthat (>= 3.3.0), usethis, withr

VignetteBuilder quarto

Config/Needs/website devtools

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

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

X-schema.org-keywords attribution, citation, credit, citation-files,
cfr, metadata, citation-file-format, cran, r, r-package,
ropensci, rstats, r-cran

NeedsCompilation no

Author Diego Hernangómez [aut, cre, cph] (ORCID: [<https://orcid.org/0000-0001-8457-4658>](https://orcid.org/0000-0001-8457-4658)),
João Martins [rev] (ORCID: [<https://orcid.org/0000-0001-7961-4280>](https://orcid.org/0000-0001-7961-4280)),
Scott Chamberlain [rev] (ORCID: [<https://orcid.org/0000-0003-1444-9135>](https://orcid.org/0000-0003-1444-9135))
Maintainer Diego Hernangómez <diego.hernangomezherrero@gmail.com>
Repository CRAN
Date/Publication 2026-08-24 12:10:02 UTC

Contents

as_bibentry	2
as_cff	5
as_cff_person	7
cff	10
cff_create	12
cff_gha_update	14
cff_git_hook	15
cff_modify	16
cff_read	17
cff_read_bib_text	19
cff_schema	21
cff_validate	22
cff_write	23
cff_write_bib	25
cran_to_spdx	27
Index	28

as_bibentry	Create bibentry objects from multiple sources
-------------	--

Description

This function creates **bibentry** objects from multiple metadata sources (**cff** objects, DESCRIPTION files and other sources). The inverse transformation from a **bibentry** object to **cff_ref_lst** can be performed with the corresponding **as_cff.bibentry()** method.

With **toBibtex()**, you can convert **cff** objects to BibTeX markup on the fly. See **Examples**.

Usage

```
as_bibentry(x, ...)  
  
## Default S3 method:  
as_bibentry(x, ...)
```

```
## S3 method for class 'character'
as_bibentry(x, ..., what = c("preferred", "references", "all"))

## S3 method for class '`NULL`'
as_bibentry(x, ...)

## S3 method for class 'list'
as_bibentry(x, ...)

## S3 method for class 'cff'
as_bibentry(x, ..., what = c("preferred", "references", "all"))

## S3 method for class 'cff_ref_lst'
as_bibentry(x, ...)

## S3 method for class 'cff_ref'
as_bibentry(x, ...)
```

Arguments

x	The source used to generate the bibentry object with cffr. It can be: • A missing value, which retrieves the DESCRIPTION file from your in-development package. • An existing cff object created with <code>cff()</code>, <code>cff_create()</code> or <code>as_cff()</code>. • A path to a CITATION.cff file ("CITATION.cff"). • The name of an installed package ("jsonlite"). • A path to a DESCRIPTION file ("DESCRIPTION").
...	Additional arguments passed to or from methods.
what	Fields to extract from a full cff object. It can be: • preferred: Create a single entry with the main citation information of the package (key preferred-citation). • references: Extract all entries of the references key. • all: Extract both the preferred-citation and references keys. See <code>vignette("r-cff", package = "cffr")</code>.

Details

An R bibentry object is the representation of a BibTeX entry. These objects can be converted to BibTeX markup with `utils::toBibtex()`, which creates an object of class Bibtex that can be printed and exported as a valid BibTeX entry.

`as_bibentry()` tries to map the information of the source x into a `cff` object and maps the metadata to BibTeX as described in `vignette("bibtex-cff", package = "cffr")`.

Value

`as_bibentry()` returns a bibentry object with one or more entries.

References

- Patashnik O (1988). "BIBTEXing." <https://osl.ugr.es/CTAN/biblio/bibtex/base/btxdoc.pdf>.
- Haines R, The Ruby Citation File Format Developers (2022). "Ruby CFF Library." doi:10.5281/zenodo.7294987. <https://github.com/citation-file-format/ruby-cff>.
- Hernangómez D (2022). "BibTeX and CFF, a potential crosswalk." **cffr** vignette. <https://docs.ropensci.org/cffr/articles/bibtex-cff.html>.

See Also

- `utils::bibentry()` documents the bibentry class.
- `vignette("r-cff", package = "cffr")` explains how the metadata of a package is mapped to produce a cff object.
- `vignette("bibtex-cff", package = "cffr")` provides details about the internal mapping performed between cff objects and BibTeX markup, both cff to BibTeX and BibTeX to cff.
- `utils::toBibtex()` converts bibentry objects to BibTeX markup.

Work with BibTeX metadata: `cff_read()`, `cff_read_bib_text()`, `cff_write_bib()`, `encoded_utf_to_latex()`

Convert between R classes: `as_cff()`, `as_cff_person()`, `cff_class`

Examples

```
# From a `cff` object ----
cff_object <- cff()

cff_object

# A `bibentry` object.
bib <- as_bibentry(cff_object)

class(bib)

bib

# Print as BibTeX.
toBibtex(bib)

# The S3 method also supports this.
toBibtex(cff_object)

# Other sources ----
# From a CITATION.cff.

path <- system.file("examples/CITATION_complete.cff", package = "cffr")
cff_file <- as_bibentry(path)

cff_file

# For an installed package with options.
```

```

installed_package <- as_bibentry("jsonvalidate", what = "all")

installed_package

# Use a DESCRIPTION file.
path2 <- system.file("examples/DESCRIPTION_gitlab", package = "cffr")
desc_file <- as_bibentry(path2)

toBibtex(desc_file)

```

as_cff

Coerce lists and citation objects to [cff](#)

Description

as_cff() turns an existing list-like R object into a [cff](#) object, a list with class cff and the corresponding [subclass](#) when applicable.

as_cff() is an S3 generic, with methods for:

- person objects as produced by [utils::person\(\)](#).
- bibentry objects as produced by [utils::bibentry\(\)](#).
- Bibtex objects as produced by [utils::toBibtex\(\)](#).
- Default: Other inputs are first coerced with [base::as.list\(\)](#).

Usage

```

as_cff(x, ...)

## Default S3 method:
as_cff(x, ...)

## S3 method for class 'list'
as_cff(x, ...)

## S3 method for class 'person'
as_cff(x, ...)

## S3 method for class 'bibentry'
as_cff(x, ...)

## S3 method for class 'Bibtex'
as_cff(x, ...)

```

Arguments

- | | |
|-----|---|
| x | A person, bibentry or other object that can be coerced to a list. |
| ... | Additional arguments passed on to other methods. |

Details

For `as_cff.bibentry()` and `as_cff.Bibtex()`, see `vignette("bibtex-cff", package = "cffr")` to understand how the mapping is performed.

`as_cff_person()` is preferred over `as_cff.person()` because it can handle character inputs such as "Davis, Jr., Sammy". For person objects both functions behave similarly.

Value

- `as_cff.person()` returns an object with classes `cff_pers_lst`, `cff`.
- `as_cff.bibentry()` and `as_cff.Bibtex()` return an object with classes `cff_ref_lst`, `cff`.
- The remaining methods return an object of class `cff`. However, if `x` has a structure compatible with `definitions.person`, `definitions.entity` or `definitions.reference`, the object has the corresponding subclass.

Learn more about the **cffr** class system in `cff_class`.

See Also

- `cff()` creates a full `cff` object from scratch.
- `cff_modify()` modifies a `cff` object.
- `cff_create()` creates a `cff` object for an R package.
- `cff_read()` creates a `cff` object from an external file.

Convert between R classes: `as_bibentry()`, `as_cff_person()`, `cff_class`

Examples

```
# Convert a list to a `cff` object.
cffobj <- as_cff(list(
  "cff-version" = "1.2.0",
  title = "Manipulating files"
))

class(cffobj)

# Display the YAML representation.
cffobj

# `bibentry` method.
a_cit <- citation("cffr")[[1]]

a_cit

as_cff(a_cit)
```

```
# BibTeX method.
a_bib <- toBibtex(a_cit)

a_bib

as_cff(a_cit)
```

as_cff_person

Coerce R objects to [cff_pers_lst](#) objects

Description

`as_cff_person()` turns an existing list-like R object into a [cff_pers_lst](#) object representing a list of `definitions.person` or `definitions.entity`, as defined in the following guide: [Citation File Format schema guide](#).

`as_cff_person()` is an S3 generic, with methods for:

- `person`: Objects created with [utils::person\(\)](#).
- `character`: Strings with the definition for one or more authors, using the standard BibTeX notation (see Markey, 2009) and related formats, such as the output of [base::format\(\)](#) for `person` objects (see [format.person\(\)](#)).
- `Default`: Other inputs are first coerced with [base::as.character\(\)](#).

The inverse transformation (`cff_pers_lst` to `person`) can be performed with the methods [as.person.cff_pers\(\)](#) and [as.person.cff_pers_lst\(\)](#).

Usage

```
as_cff_person(x, ...)
```

Default S3 method:

```
as_cff_person(x, ...)
```

S3 method for class 'person'

```
as_cff_person(x, ...)
```

S3 method for class 'character'

```
as_cff_person(x, ...)
```

Arguments

<code>x</code>	Any R object.
<code>...</code>	Ignored by this method.

Details

as_cff_person() recognizes whether the input should be converted with the CFF reference for definitions.person or definitions.entity.

as_cff_person() uses a custom algorithm that parses names as explained in Section 11 of "Tame the BeaST" (Markey, 2009) (see also Decoret, 2007):

- First von Last.
- von Last, First.
- von Last, Jr, First.

Mapping is performed as follows:

- First is mapped to the CFF key given-names.
- von is mapped to the CFF key name-particle.
- Last is mapped to the CFF key family-names.
- Jr is mapped to the CFF key name-suffix.

For entities, the entire character is mapped to name. We recommend "protecting" entity names with {}:

```
# Avoid unprotected entity names.
entity <- "Elephant and Castle"
as_cff_person(entity)
- name: Elephant
- name: Castle

# Protect entity names with braces.
entity_protect <- "{Elephant and Castle}"
as_cff_person(entity_protect)
- name: Elephant and Castle
```

as_cff_person() attempts to extract as much information as possible. For character strings from [format.person\(\)](#), the email and ORCID are also extracted.

Value

as_cff_person() returns an object of classes [cff_pers_lst](#), [cff](#) as defined by definitions.person or definitions.entity specified in the following guide: [Citation File Format schema guide](#). Each element of the cff_pers_lst object has classes [cff_pers](#), [cff](#).

References

- Patashnik O (1988). "BIBTEXing." <https://osl.ugr.es/CTAN/biblio/bibtex/base/btxdoc.pdf>.
- Markey N (2009). *Tame the BeaST: The B to X of BibTeX*. https://osl.ugr.es/CTAN/info/bibtex/tamethebeast/ttb_en.pdf.
- Decoret X (2007). "A summary of BibTeX." https://maverick.inria.fr/~Xavier.Decoret/resources/xdkbibtex/bibtex_summary.html#names.

See Also

Examples in `vignette("cfr", package = "cfr")` and `utils::person()`. Learn more about the `cff_pers_lst` and `cff_pers` classes in `cff_class`.

Convert between R classes: `as_bibentry()`, `as_cff()`, `cff_class`

Examples

```
# Create a person object.
a_person <- person(
  given = "First", family = "Author",
  role = c("aut", "cre"),
  email = "first.last@example.com", comment = c(
    ORCID = "0000-0001-8457-4658",
    affiliation = "An affiliation"
  )
)

a_person

cff_person <- as_cff_person(a_person)

# Classes `cff_pers_lst` and `cff`.
class(cff_person)

# Each element has classes `cff_pers` and `cff`.
class(cff_person[[1]])

# Print.
cff_person

# Back to person object with S3 method.
as.person(cff_person)

# Coerce a string.
a_str <- paste0(
  "Julio Iglesias <fake@email.com> ",
  "(city: Miami, region: California, country: US)"
)
as_cff_person(a_str)

# Several persons.
persons <- c(
  person("Clark", "Kent", comment = c(affiliation = "Daily Planet")),
  person("Lois", "Lane"), person("Oscorp Inc.")
)

a_cff <- as_cff_person(persons)

a_cff

# Print as BibTeX with the method.
toBibtex(a_cff)
```

```
# Or as a `person` object.
as.person(a_cff)

# Or use BibTeX style as input.

x <- "Frank Sinatra and Dean Martin and Davis, Jr., Sammy and Joey Bishop"

as_cff_person(x)

as_cff_person("Herbert von Karajan")

toBibtex(as_cff_person("Herbert von Karajan"))
```

cffi

Create cff objects from direct inputs

Description

A class and utility methods for reading, creating and storing CFF information. See [cff_class](#) to learn more about cff objects.

Usage

```
cff(path, ...)
```

Arguments

path	[Deprecated] path is no longer supported, use cff_read_cff_citation() instead.
...	Named arguments used to create a cff object. If no arguments are supplied (the default behavior), a minimal valid cff object is created.

Details

`cff()` converts `_` in the argument name to `-`. For example, `cff_version = "1.2.0"` is converted to `cff-version = "1.2.0"`.

Valid arguments are those specified on [cff_schema_keys\(\)](#):

- cff-version
- message
- type
- license
- title
- version
- doi

- identifiers
- abstract
- authors
- preferred-citation
- repository
- repository-artifact
- repository-code
- commit
- url
- date-released
- contact
- keywords
- references
- license-url

Value

A **cff** object. Under the hood, a cff object is a regular `base::list()` object with a special `print` method.

See Also

Core **cffr** workflow: `cff_create()`, `cff_modify()`, `cff_validate()`, `cff_write()`

Examples

```
# Blank `cff` object.
cff()

# Use custom parameters.
test <- cff(
  title = "Manipulating files",
  keywords = c("A", "new", "list", "of", "keywords"),
  authors = as_cff_person("New author")
)
test

# This would fail.
cff_validate(test)

# Modify with cff_create().
new <- cff_create(test, keys = list(
  "cff_version" = "1.2.0",
  message = "A blank file"
))
new
```

```
# This would pass.
cff_validate(new)
```

cff_create

Create a [cff](#) object from multiple sources

Description

Create a full, possibly valid [cff](#) object from a given source. This object can be written to a CITATION.cff file with [cff_write\(\)](#). See **Examples**.

This function performs most metadata extraction and conversion in [cfr](#).

Usage

```
cff_create(
  x,
  keys = list(),
  cff_version = "1.2.0",
  gh_keywords = TRUE,
  dependencies = TRUE,
  authors_roles = c("aut", "cre")
)
```

Arguments

x	The source used to generate the cff object. It can be: • A missing value, which retrieves the DESCRIPTION file from your in-development R package. • An existing cff object. • The name of an installed package ("jsonlite"). • A path to a DESCRIPTION file ("./DESCRIPTION").
keys	A list of additional keys to add to the cff object. See cff_modify() .
cff_version	The Citation File Format schema version used for the generated metadata.
gh_keywords	A logical value. If TRUE and the package is hosted on GitHub, add the repository topics as keywords.
dependencies	A logical value. If TRUE, add the package dependencies to the references CFF key.
authors_roles	Roles to be considered as authors of the package when generating the CITATION.cff file. See Details .

Details

If `x` is a path to a DESCRIPTION file or if `inst/CITATION` is not present in your package, **cffi** auto-generates a preferred-citation key using the information provided in that file.

By default, only persons whose role in the DESCRIPTION file of the package is author ("aut") or maintainer ("cre") are considered package authors. The default setting can be controlled with the `authors_roles` argument. See **Details** on `utils::person()` for additional information about person roles.

Value

A `cffi` object.

See Also

[Citation File Format schema guide](#).

- `vignette("cffi", package = "cffi")` shows an introduction to manipulating `cffi` objects.
- `vignette("r-cffi", package = "cffi")` provides details about how the metadata of a package is mapped to produce a `cffi` object.

Core **cffi** workflow: `cffi()`, `cffi_modify()`, `cffi_validate()`, `cffi_write()`

Examples

```
# Installed package.
cffi_create("jsonlite")

# Demo file.
demo_file <- system.file("examples/DESCRIPTION_basic", package = "cffi")
cffi_create(demo_file)

# Add additional keys.

newkeys <- list(
  message = "This overwrites keys",
  abstract = "New abstract",
  keywords = c("A", "new", "list", "of", "keywords"),
  authors = as_cffi_person("New author")
)

cffi_create(demo_file, keys = newkeys)

# Update a key in a list, such as authors or contacts.
# Add a new contact.

old <- cffi_create(demo_file)

new_contact <- append(
  old$contact,
  as_cffi_person(person(
    given = "I am",
```

```

    family = "New Contact"
  ))
)

cff_create(demo_file, keys = list("contact" = new_contact))

```

cff_gha_update	<i>Install a R</i> <i>hrefhttps://CRAN.R-project.org/package=cffr</i> cffr <i>GitHub Actions workflow</i>
----------------	--

Description

This function installs a **GitHub Actions** workflow in your repository. The workflow updates your CITATION.cff when any of these events occur:

- You publish a new release of the package.
- Your DESCRIPTION or inst/CITATION file is modified.
- The workflow can be run manually.

Usage

```
cff_gha_update(path = ".", overwrite = FALSE)
```

Arguments

path	Project root directory.
overwrite	A logical value. If TRUE, overwrite an existing workflow.

Details

Workflow triggers can be modified. See [Events that trigger workflows](#).

Value

Invisible. This function is called for its side effects.

See Also

Keep ‘CITATION.cff’ up to date: [cff_git_hook](#)

Examples

```

## Not run:
cff_gha_update()

## End(Not run)

```

cffi_git_hook*Use a Git pre-commit hook* **[Experimental]**

Description

Install a **pre-commit hook** that reminds you to update your CITATION.cff file. This is a wrapper around `usethis::use_git_hook()`.

Usage

```
cffi_git_hook_install()
```

```
cffi_git_hook_remove()
```

Details

This function installs a pre-commit hook using `usethis::use_git_hook()`.

A pre-commit hook is a script that identifies simple issues before submission to code review. This pre-commit hook warns you if any of the following conditions are met:

- You included your DESCRIPTION or inst/CITATION file in a commit but did not include your CITATION.cff and the CITATION.cff file is "older" than your DESCRIPTION or inst/CITATION file.
- You updated your CITATION.cff but did not include it in your commit.

Value

Invisible. This function is called for its side effects.

A word of caution

The pre-commit hook may prevent you from committing if you are not updating your CITATION.cff. However, the detection mechanism is not perfect and may be triggered even if you have attempted to update your CITATION.cff file.

This typically occurs when you have updated your DESCRIPTION or inst/CITATION files, but those changes do not affect your CITATION.cff file, for example, when you add new dependencies.

In those cases, you can override the check by running `git commit --no-verify` in the terminal.

If you are using **RStudio**, you can also run this command from an R script by selecting that line and sending it to the terminal with:

- Windows & Linux: Ctrl+Alt+Enter.
- Mac: Cmd+Option+Return.

Removing the Git pre-commit hook

You can remove the pre-commit hook using `cffi_git_hook_remove()`.

See Also

- `usethis::use_git_hook()`, which is the underlying function used by `cff_git_hook_install()`.
- `usethis::use_git()` and related functions of **usethis** for using Git with R packages.

Keep ‘CITATION.cff’ up to date: `cff_gha_update()`

Examples

```
## Not run:
cff_git_hook_install()

## End(Not run)
```

`cff_modify`

Modify a `cff` object

Description

Add new keys to a `cff` object or modify existing ones.

Usage

```
cff_modify(x, ...)
```

Arguments

<code>x</code>	A <code>cff</code> object.
<code>...</code>	Named arguments used to modify <code>x</code> . See also the <code>...</code> argument in <code>cff()</code> .

Details

Keys provided in `...` override the corresponding key in `x`.

You can add additional keys not detected by `cff_create()` using the `keys` argument. A list of valid keys can be retrieved with `cff_schema_keys()`. See the following guide for additional details: [Citation File Format schema guide](#).

Value

A `cff` object.

See Also

This function is a wrapper of `utils::modifyList()`.

Core **cfr** workflow: `cff()`, `cff_create()`, `cff_validate()`, `cff_write()`

Examples

```

x <- cff()
x

cff_validate(x)

x_mod <- cff_modify(x,
  contact = as_cff_person("A contact"),
  message = "This overwrites keys",
  title = "New Title",
  abstract = "New abstract",
  doi = "10.21105/joss.03900"
)

x_mod

cff_validate(x_mod)

```

cff_read	<i>Read an external file as a cff object</i>
----------	---

Description

Read files and convert them to **cff** objects. Supported files are:

- CITATION.cff files.
- DESCRIPTION files.
- R citation files (usually located in inst/CITATION).
- BibTeX files (with extension *.bib).

cff_read() attempts to guess the type of file provided in path. However, we provide aliases for each specific file type:

- **cff_read_cff_citation()**, which uses `yaml::read_yaml()`.
- **cff_read_description()**, which uses `desc::desc()`.
- **cff_read_citation()**, which uses `utils::readCitationFile()`.
- **cff_read_bib()**, which requires **bibtex** ($\geq 0.5.0$) and uses `bibtex::read.bib()`.

Usage

```

cff_read(path, ...)

cff_read_cff_citation(path, ...)

cff_read_description(
  path,

```

```

  cff_version = "1.2.0",
  gh_keywords = TRUE,
  authors_roles = c("aut", "cre"),
  ...
)

cff_read_citation(path, meta = NULL, ...)

cff_read_bib(path, encoding = "UTF-8", ...)

```

Arguments

path	A path to a file.
...	Arguments passed to other functions, for example to yaml::read_yaml() or bibtex::read.bib() .
cff_version	The Citation File Format schema version used for the generated metadata.
gh_keywords	A logical value. If TRUE and the package is hosted on GitHub, add the repository topics as keywords.
authors_roles	Roles to be considered as authors of the package when generating the CITATION.cff file. See Details .
meta	A list of package metadata as obtained by utils::packageDescription() or NULL (the default). See Details .
encoding	Encoding to be assumed for path. See base::readLines() .

Details

For details of `cff_read_description()`, see [cff_create\(\)](#).

The meta object:

Section 1.9 CITATION files of *Writing R Extensions* (R Core Team 2026) specifies how to create dynamic CITATION files using a meta object. Therefore, the meta argument in [cff_read_citation\(\)](#) may be needed to read some files correctly.

Value

- `cff_read_cff_citation()` and `cff_read_description()` return an object with class `cff`.
- `cff_read_citation()` and `cff_read_bib()` return an object of classes `cff_ref_lst`, `cff` as defined by the `definitions.reference` specified in the following guide: [Citation File Format schema guide](#).

Learn more about the **cffr** class system in [cff_class](#).

References

- R Core Team (2026). *Writing R Extensions*. <https://cran.r-project.org/doc/manuals/r-release/R-exts.html>.
- Hernangómez D (2022). "BibTeX and CFF, a potential crosswalk." **cffr** vignette. <https://docs.ropensci.org/cffr/articles/bibtex-cff.html>.

See Also

The underlying functions used for reading external files:

- `yaml::read_yaml()` for CITATION.cff files.
- `desc::desc()` for DESCRIPTION files.
- `utils::readCitationFile()` for R citation files.
- `bibtex::read.bib()` for BibTeX files (extension *.bib).

Read external citation metadata: `cfr_read_bib_text()`

Work with BibTeX metadata: `as_bibentry()`, `cfr_read_bib_text()`, `cfr_write_bib()`, `encoded_utf_to_latex()`

Examples

```
# Create a `cfr` object from a `CITATION.cff` file.
from_cfr_file <- cfr_read(system.file("examples/CITATION_basic.cfr",
  package = "cfr")
))

head(from_cfr_file, 7)

# Create a `cfr` object from DESCRIPTION.
from_desc <- cfr_read(system.file("examples/DESCRIPTION_basic",
  package = "cfr")
))

from_desc

# Create a `cfr` object from BibTeX.
if (requireNamespace("bibtex", quietly = TRUE)) {
  from_bib <- cfr_read(system.file("examples/example.bib",
    package = "cfr")
  ))

  # First item only.
  from_bib[[1]]
}

# Create a `cfr` object from CITATION.
from_citation <- cfr_read(system.file("CITATION", package = "cfr"))

# First item only.
from_citation[[1]]
```

cfr_read_bib_text

Read BibTeX markup as a `cfr_ref_lst` object

Description

Convert one or more complete BibTeX entries supplied as a character vector into a `cfr_ref_lst` object.

Usage

```
cff_read_bib_text(x, encoding = "UTF-8", ...)
```

Arguments

x	A character vector with one or more complete BibTeX entries.
encoding	Encoding to be assumed for x. See <code>base::readLines()</code> .
...	Arguments passed to <code>cff_read_bib()</code> .

Details

This function writes x to a temporary *.bib file and reads it using `cff_read_bib()`.

This function requires **bibtex** (>= 0.5.0) and uses `bibtex::read.bib()` for parsing.

Value

An object of classes `cff_ref_lst`, `cff` as defined by the definitions.reference specified in the following guide: [Citation File Format schema guide](#). Each element of the `cff_ref_lst` object has classes `cff_ref`, `cff`.

See Also

`cff_read_bib()` for reading *.bib files.

Work with BibTeX metadata: `as_bibentry()`, `cff_read()`, `cff_write_bib()`, `encoded_utf_to_latex()`

Read external citation metadata: `cff_read()`

Examples

```
if (requireNamespace("bibtex", quietly = TRUE)) {
  x <- c(
    "@book{einstein1921,
      title      = {Relativity: The Special and the General Theory},
      author     = {Einstein, Albert},
      year       = 1920,
      publisher  = {Henry Holt and Company},
      address    = {London, United Kingdom},
      isbn       = 9781587340925
    }",
    "@misc{misc-full,
      title      = {Handing out random pamphlets in airports},
      author     = {Joe-Bob Missilany},
      year       = 1984,
      month      = oct,
      note       = {This is a full MISC entry},
      howpublished = {Handed out at O'Hare}
    }"
  )

  cff_read_bib_text(x)
}
```

cffi_schema*Inspect CFF schema values*

Description

Helper functions with valid values for different keys and definition fields. See the following guide: [Citation File Format schema guide](#).

- `cffi_schema_keys()` provides the valid high-level keys of the Citation File Format.
- `cffi_schema_keys_license()` provides valid **SPDX license identifier(s)** for the CITATION.cff file.
- `cffi_schema_definitions_person()` and `cffi_schema_definitions_entity()` return the valid fields to include when defining a person or entity.
- `cffi_schema_definitions_refs()` provides the valid keys to use in the preferred-citation and references keys.

Usage

```
cffi_schema_keys(sorted = FALSE)
```

```
cffi_schema_keys_license()
```

```
cffi_schema_definitions_person()
```

```
cffi_schema_definitions_entity()
```

```
cffi_schema_definitions_refs()
```

Arguments

`sorted` A logical value. If TRUE, arrange the keys alphabetically.

Value

A character vector with the names of valid keys for Citation File Format version 1.2.0.

Source

[Citation File Format schema guide](#).

Examples

```
cffi_schema_keys(sorted = TRUE)
# Valid license keys.
head(cffi_schema_keys_license(), 20)
cffi_schema_definitions_person()
cffi_schema_definitions_entity()
cffi_schema_definitions_refs()
```

cff_validate	<i>Validate a CITATION.cff file or a cff object</i>
--------------	---

Description

Validate a CITATION.cff file or a [cff](#) object using the corresponding [validation schema](#).

Usage

```
cff_validate(x = "CITATION.cff", verbose = TRUE)
```

Arguments

x	A full cff object created with cff_create() or the path to a CITATION.cff file to be validated. A *.cff file is read with cff_read_cff_citation() .
verbose	A logical value. If TRUE, the function displays informative messages.

Value

Invisibly returns a logical value indicating the validation result. On error, the result has an "errors" attribute with the error summary. If verbose is TRUE, the function also displays the result. See [Examples](#) and [base::attr\(\)](#).

See Also

[jsonvalidate::json_validate\(\)](#), which is the function that performs the validation.

[Citation File Format schema guide](#).

Core **cfr** workflow: [cff\(\)](#), [cff_create\(\)](#), [cff_modify\(\)](#), [cff_write\(\)](#)

Examples

```
# Full `.cff` example.
cff_validate(system.file("examples/CITATION_complete.cff", package = "cfr"))

# Validate a `cff` object.
cfr <- cff_create("jsonlite")
class(cfr)
cff_validate(cfr)

# `.cff` with errors.
err_f <- system.file("examples/CITATION_error.cff", package = "cfr")
# You can manipulate the errors as a data frame.
res <- try(cff_validate(err_f))

isTRUE(res)
isFALSE(res)
```

```
# Detailed results.
df_errors <- attr(res, "errors")

df_errors[, c("field", "message")]

# An R citation file is not a `.cff` file, so it throws an error.
try(cff_validate(system.file("CITATION", package = "cffr")))
```

cff_write

Write a CITATION.cff file

Description

`cff_write()` is the primary workflow for package development.

This function writes a `CITATION.cff` file for a given package. It wraps `cff_create()` to create the `cff` object, then writes it to a YAML-formatted file in one command.

Usage

```
cff_write(
  x,
  outfile = "CITATION.cff",
  keys = list(),
  cff_version = "1.2.0",
  gh_keywords = TRUE,
  r_citation = FALSE,
  dependencies = TRUE,
  validate = TRUE,
  verbose = TRUE,
  authors_roles = c("aut", "cre"),
  encoding = "UTF-8"
)
```

Arguments

<code>x</code>	The source used to generate the <code>cff</code> object. It can be: • A missing value, which retrieves the DESCRIPTION file from your in-development R package. • An existing <code>cff</code> object. • The name of an installed package (<code>"jsonlite"</code>). • A path to a DESCRIPTION file (<code>"./DESCRIPTION"</code>).
<code>outfile</code>	The name and path of the <code>CITATION.cff</code> to be created. Relative paths are resolved from the current working directory, independently of the source specified in <code>x</code>.
<code>keys</code>	A list of additional keys to add to the <code>cff</code> object. See <code>cff_modify()</code>.
<code>cff_version</code>	The Citation File Format schema version used for the generated metadata.

gh_keywords	A logical value. If TRUE and the package is hosted on GitHub, add the repository topics as keywords.
r_citation	A logical value. If TRUE, the R package citation (inst/CITATION) is created or updated relative to the current working directory. No backup copy is created. For more control, use <code>cff_write_citation()</code> .
dependencies	A logical value. If TRUE, add the package dependencies to the references CFF key.
validate	A logical value. If TRUE, validate the new file with <code>cff_validate()</code> .
verbose	A logical value. If TRUE, the function displays informative messages.
authors_roles	Roles to be considered as authors of the package when generating the CITATION.cff file. See Details .
encoding	The name of the encoding to be assumed. Default is "UTF-8", but it can be any other value accepted by <code>base::iconv()</code> , such as "ASCII//TRANSLIT".

Details

For details of `authors_roles`, see `cff_create()`.

The `x` argument identifies the metadata source. It does not determine the output directory. This allows you to create a CITATION.cff from a package or file located outside the current working directory.

When creating and writing a CITATION.cff for a package in the current working directory, this function adds the pattern "`^CITATION\\.cff$`" to the local `.Rbuildignore` file.

Value

Invisibly returns the generated `cff` object. This function is called primarily for its side effect of writing a CITATION.cff file.

See Also

[Citation File Format schema guide](#).

Core **cffr** workflow: `cff()`, `cff_create()`, `cff_modify()`, `cff_validate()`

Write citation metadata files: `cff_write_bib()`

Examples

```
tmpfile <- tempfile(fileext = ".cff")
cff_obj <- cff_write("jsonlite", outfile = tmpfile)

cff_obj

# Force cleanup.
file.remove(tmpfile)
```

cffi_write_bib

*Export R objects to multiple file types***Description**

Export R objects representing citations to specific file formats:

- `cffi_write_bib()` creates a .bib file.
- `cffi_write_citation()` creates an R citation file as described in Section 1.9 of *Writing R Extensions* (R Core Team 2026).

Usage

```
cffi_write_bib(
  x,
  file = tempfile(fileext = ".bib"),
  append = FALSE,
  verbose = TRUE,
  ascii = FALSE,
  ...
)

cffi_write_citation(
  x,
  file = tempfile("CITATION_"),
  append = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

x	A bibentry or a cffi object.
file	Name of the file to be created. If NULL, the lines are displayed instead.
append	A logical value. If TRUE, append entries to an existing file.
verbose	A logical value. If TRUE, display informative messages.
ascii	A logical value. If TRUE, write entries using ASCII characters only.
...	Arguments passed on to as_bibentry.cffi , as_bibentry.cffi_ref , as_bibentry.cffi_ref_lst

what Fields to extract from a full cffi object. It can be:

- preferred: Create a single entry with the main citation information of the package (key preferred-citation).
- references: Extract all entries of the references key.
- all: Extract both the preferred-citation and references keys.

See `vignette("r-cffi", package = "cffi")`.

Details

When `x` is a `cff` object, it is converted to BibTeX using `toBibtex.cff()`.

For security reasons, if the file already exists, the function creates a backup copy in the same directory.

Value

Invisibly returns `NULL`. This function is called for its side effect of writing a file or displaying its contents.

References

R Core Team (2026). *Writing R Extensions*. <https://cran.r-project.org/doc/manuals/r-release/R-exts.html>.

See Also

`vignette("bibtex-cff", package = "cfr")`, `knitr::write_bib()` and the following packages:

- **bibtex**.
- **RefManageR**.
- **rbibutils**.

Work with BibTeX metadata: `as_bibentry()`, `cff_read()`, `cff_read_bib_text()`, `encoded_utf_to_latex()`

Write citation metadata files: `cff_write()`

Examples

```
bib <- bibentry("Misc",
  title = "My title",
  author = "Fran Pérez"
)

my_temp_bib <- tempfile(fileext = ".bib")

cff_write_bib(bib, file = my_temp_bib)

cat(readLines(my_temp_bib), sep = "\n")

cff_write_bib(bib, file = my_temp_bib, ascii = TRUE, append = TRUE)

cat(readLines(my_temp_bib), sep = "\n")
# Create an R citation file.
# Use a system file.
f <- system.file("examples/preferred-citation-book.cff", package = "cfr")
a_cff <- cff_read(f)

out <- file.path(tempdir(), "CITATION")
cff_write_citation(a_cff, file = out)
```

```
# Check by reading with a meta object.
meta <- packageDescription("cfr")
meta$Encoding <- "UTF-8"

utils::readCitationFile(out, meta)
```

cran_to_spdx

Mapping between License fields and SPDX

Description

A dataset containing the mapping between the License strings observed in CRAN packages and their approximate match in the [SPDX License List](#).

Format

A data frame with 86 rows and two variables:

LICENSE A valid License string on CRAN.

SPDX A valid SPDX license identifier.

Source

<https://spdx.org/licenses/>.

See Also

Writing R Extensions, [Licensing section](#).

Examples

```
data("cran_to_spdx")

head(cran_to_spdx, 20)
```

Index

- * **bibtex**
 - as_bibentry, 2
 - cff_read, 17
 - cff_read_bib_text, 19
 - cff_write_bib, 25
- * **conversions**
 - as_bibentry, 2
 - as_cff, 5
 - as_cff_person, 7
- * **core**
 - cff, 10
 - cff_create, 12
 - cff_modify, 16
 - cff_validate, 22
 - cff_write, 23
- * **datasets**
 - cran_to_spdx, 27
- * **git**
 - cff_gha_update, 14
 - cff_git_hook, 15
- * **reading**
 - cff_read, 17
 - cff_read_bib_text, 19
- * **schemas**
 - cff_schema, 21
- * **writing**
 - cff_write, 23
 - cff_write_bib, 25

as.cff(as_cff), 5

as.person.cff_pers(), 7

as.person.cff_pers_lst(), 7

as_bibentry, 2

as_bibentry(), 6, 9, 19, 20, 26

as_bibentry.cff, 25

as_bibentry.cff_ref, 25

as_bibentry.cff_ref_lst, 25

as_cff, 5

as_cff(), 3, 4, 9

as_cff.bibentry(), 2

as_cff_person, 7

as_cff_person(), 4, 6

base::as.character(), 7

base::as.list(), 5

base::attr(), 22

base::format(), 7

base::iconv(), 24

base::list(), 11

base::readLines(), 18, 20

bibentry, 2, 25

bibtex::read.bib(), 17–20

cff, 2, 3, 5, 10, 10, 11–13, 16, 17, 22–25

cff(), 3, 6, 13, 16, 22, 24

cff_class, 4, 6, 9, 10, 18

cff_create, 12

cff_create(), 3, 6, 11, 16, 18, 22–24

cff_gha_update, 14

cff_gha_update(), 16

cff_git_hook, 14, 15

cff_git_hook_install(cff_git_hook), 15

cff_git_hook_remove(cff_git_hook), 15

cff_modify, 16

cff_modify(), 6, 11–13, 22–24

cff_pers_lst, 7

cff_read, 17

cff_read(), 4, 6, 17, 20, 26

cff_read_bib(cff_read), 17

cff_read_bib(), 17, 20

cff_read_bib_text, 19

cff_read_bib_text(), 4, 19, 26

cff_read_cff_citation(cff_read), 17

cff_read_cff_citation(), 10, 17, 22

cff_read_citation(cff_read), 17

cff_read_citation(), 17, 18

cff_read_description(cff_read), 17

cff_read_description(), 17

cff_ref_lst, 2, 19

cff_schema, 21

cff_schema_definitions_entity
 (cff_schema), 21
cff_schema_definitions_entity(), 21
cff_schema_definitions_person
 (cff_schema), 21
cff_schema_definitions_person(), 21
cff_schema_definitions_refs
 (cff_schema), 21
cff_schema_definitions_refs(), 21
cff_schema_keys (cff_schema), 21
cff_schema_keys(), 10, 16, 21
cff_schema_keys_license (cff_schema), 21
cff_schema_keys_license(), 21
cff_validate, 22
cff_validate(), 11, 13, 16, 24
cff_write, 23
cff_write(), 11–13, 16, 22, 26
cff_write_bib, 25
cff_write_bib(), 4, 19, 20, 24, 25
cff_write_citation (cff_write_bib), 25
cff_write_citation(), 24, 25
cran_to_spdx, 27

desc::desc(), 17, 19

encoded_utf_to_latex(), 4, 19, 20, 26

format.person(), 7, 8

jsonvalidate::json_validate(), 22

knitr::write_bib(), 26

print, 11

subclass, 5

toBibtex(), 2
toBibtex.cff(), 26

usethis::use_git(), 16
usethis::use_git_hook(), 15, 16
utils::bibentry(), 4, 5
utils::modifyList(), 16
utils::packageDescription(), 18
utils::person(), 5, 7, 9, 13
utils::readCitationFile(), 17, 19
utils::toBibtex(), 3–5

yaml::read_yaml(), 17–19