

# Package: cffr (via r-universe)

August 28, 2024

**Title** Generate Citation File Format ('cff') Metadata for R Packages

**Version** 1.1.1

**Description** The Citation File Format version 1.2.0

<doi:10.5281/zenodo.5171937> is a human and machine readable file format which provides citation metadata for software. This package provides core utilities to generate and validate this metadata.

**License** GPL (>= 3)

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

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

**Depends** R (>= 4.0.0)

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

**Suggests** bibtex (>= 0.5.0), knitr, lifecycle, rmarkdown, testthat (>= 3.0.0), usethis

**VignetteBuilder** knitr

**Config/Needs/website** devtools

**Config/testthat/edition** 3

**Config/testthat/parallel** true

**Encoding** UTF-8

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

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

**X-schema.org-keywords** attribution, citation, credit, citation-files,  
cff, metadata

**Repository** <https://ropensci.r-universe.dev>

**RemoteUrl** <https://github.com/ropensci/cffr>

**RemoteRef** main

**RemoteSha** 54d4f08107059daf742a22d3d1cf9f0eb0c47c18

## 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	20
cff_schema	21
cff_validate	22
cff_write	23
cff_write_bib	25
cran_to_spdx	27
<b>Index</b>	<b>29</b>

---

as_bibentry	Create <a href="#">bibentry</a> objects from several sources
-------------	--

---

## Description

This function creates [bibentry](#) objects from different metadata sources ([cff](#) objects, DESCRIPTION files, etc.). The inverse transformation (bibentry object to [cff\\_ref\\_lst](#)) can be done with the corresponding [as\\_cff.bibentry\(\)](#) method.

With [toBibtex\(\)](#) it is possible to 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 that would be used for generating the bibentry object via <b>cffr</b> . It could be: <ul style="list-style-type: none"> <li>• A missing value. That would retrieve the DESCRIPTION file on your in-development package.</li> <li>• An existing cff object created with <code>cff()</code>, <code>cff_create()</code> or <code>as_cff()</code>.</li> <li>• Path to a CITATION.cff file ("CITATION.cff"),</li> <li>• The name of an installed package ("jsonlite"), or</li> <li>• Path to a DESCRIPTION file ("DESCRIPTION").</li> </ul>
...	Additional arguments to be passed to or from methods.
what	Fields to extract from a full cff object. The value could be: <ul style="list-style-type: none"> <li>• preferred: This would create a single entry with the main citation info of the package (key preferred-citation).</li> <li>• references: Extract all the entries of references key.</li> <li>• all: A combination of the previous two options. This would extract both the preferred-citation and the references key.</li> </ul> See vignette("crosswalk", package = "cffr").

## Details

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

`as_bibtex()` tries to map the information of the source x into a cff] object and performs a mapping of the metadata to `vignette("bibtex_cff", "cfr")`.

## Value

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

## References

- Patashnik, Oren. "BIBTEXTING" February 1988. <https://osl.ugr.es/CTAN/biblio/bibtex/base/btxdoc.pdf>.
- Haines, R., & The Ruby Citation File Format Developers. (2021). *Ruby CFF Library (Version 0.9.0)* (Computer software). doi:10.5281/zenodo.1184077.
- Hernangomez D (2022). "BibTeX and CFF, a potential crosswalk." *The cffr package, Vignettes*. doi:10.21105/joss.03900, [https://docs.ropensci.org/cffr/articles/bibtex\\_cff.html](https://docs.ropensci.org/cffr/articles/bibtex_cff.html).

**See Also**

`utils::bibentry()` to understand more about the bibentry class.

- `vignette("crosswalk", package = "cfr")` provides details on how the metadata of a package is mapped to produce a cff object.
- `vignette("bibtex_cff", "cfr")` provides detailed information about the internal mapping performed between cff objects and BibTeX markup ( both cff to BibTeX and BibTeX to cff).

Other related functions:

- `utils::toBibtex()`.

Other functions for working with BibTeX format: `cff_read()`, `cff_read_bib_text()`, `cff_write_bib()`, `encoded_utf_to_latex()`

Coercing between **R** classes with **S3 Methods**: `as_cff()`, `as_cff_person()`, `cff_class`

**Examples**

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

cff_object

# bibentry object
bib <- as_bibentry(cff_object)

class(bib)

bib

# Print as bibtex
toBibtex(bib)

# Thanks to the S3 Method we can also do
toBibtex(cff_object)

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

path <- system.file("examples/CITATION_complete.cff", package = "cfr")
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 = "cfr")
desc_file <- as_bibentry(path2)

toBibtex(desc_file)
```

as\_cff

*Coerce lists, person and bibentry objects to cff*

## Description

as\_cff() turns an existing list-like **R** object into a so-called **cff**, a list with class **cff**, with the corresponding **sub-class** if 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 object as produced by `toBibtex()`.
- Default: Other inputs are first coerced with `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 could be coerced to a list.

**...**                Additional arguments to be passed on to other methods.

## Details

For `as_cff.bibentry()` / `as_cff.Bibtex()` see `vignette("bibtex_cff", "cfr")` to understand how the mapping is performed.

`as_cff_person()` is preferred over `as_cff.person()`, since it can handle character person such as "Davis, Jr., Sammy". For person objects both functions are similar.

## Value

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

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

## See Also

- `cff()`: Create a full `cff` object from scratch.
- `cff_modify()`: Modify a `cff` object.
- `cff_create()`: Create a `cff` object of a **R** package.
- `cff_read()`: Create a `cff` object from a external file.
- `as_cff_person()`: Recommended way for creating persons in CFF format.

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

Coercing between **R** classes with **S3 Methods**: `as_bibentry()`, `as_cff_person()`, `cff_class`

## Examples

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

class(cffobj)

# Nice display thanks to yaml package
cffobj

# bibentry method
a_cit <- citation("cfr")[[1]]

a_cit

as_cff(a_cit)
```

```
# Bibtex method
a_bib <- toBibtex(a_cit)

a_bib

as_cff(a_cit)
```

---

as_cff_person	<i>Coerce <b>R</b> objects to <a href="#">cff_pers_lst</a> objects (cff persons)</i>
---------------	--

---

## 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 by the [Citation File Format schema](#).

as\_cff\_person is an S3 generic, with methods for:

- person: objects created with [person\(\)](#).
- character: String with the definition of an author or several authors, using the standard BibTeX notation (see Markey, 2007) and others, like the output of [format\(\)](#) for person (see [format.person\(\)](#)).
- Default: Other inputs are first coerced with [as.character\(\)](#).

The inverse transformation (cff\_pers\_lst to person) can be done 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

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

## Details

as\_cff\_person() would recognize if the input should be converted using the CFF reference for definition.person or definition.entity.

as\_cff\_person() uses a custom algorithm that tries to break a name as explained in Section 11 of "Tame the BeaST" (Markey, 2007) (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 field given-names.
- von is mapped to the CFF field name-particle.
- Last is mapped to the CFF field family-names.
- Jr is mapped to the CFF field name-suffix.

In the case of entities, the whole character would be mapped to name. It is a good practice to "protect" entity's names with {}:

```
# Don't do
entity <- "Elephant and Castle"
as_cff_person(entity)
- name: Elephant
- name: Castle

# Do
entity_protect <- "{Elephant and Castle}"
as_cff_person(entity_protect)
- name: Elephant and Castle
```

as\_cff\_person() would try to add as many information as possible. On character string coming from `format.person()` the email and the ORCID would be retrieved as well.

## Value

as\_cff\_person() returns an object of classes `cff_pers_lst`, `cff` according to the definitions.person or definitions.entity specified in the [Citation File Format schema](#). Each element of the cff\_pers\_lst object would have classes `cff_pers`, `cff`.

## References

- Patashnik, Oren. "BIBTEXTING" February 1988. <https://osl.ugr.es/CTAN/biblio/bibtex/base/btxdoc.pdf>.
- Markey, Nicolas. "Tame the BeaST" *The B to X of BibTeX, Version 1.4* (October 2007). [https://osl.ugr.es/CTAN/info/bibtex/tamethebeast/ttb\\_en.pdf](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](https://maverick.inria.fr/~Xavier.Decoret/resources/xdkbibtex/bibtex_summary.html#names)

See **Examples** for more information.



## See Also

Examples in `vignette("cfr", "cfr")` and `utils::person()`.

Learn more about the classes `cff_pers_lst`, `cff_pers` classes in `cff_class`.

Coercing between **R** classes with **S3 Methods**: `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)

# Class cff_pers_lst / cff
class(cff_person)

# With each element with class cff_pers / 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> ",
  "(<https://orcid.org/0000-0001-8457-4658>)"
)
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

# Printed as Bibtex thanks to the method
```

```

toBibtex(a_cff)

# Or as person object
as.person(a_cff)

# Or you can use BibTeX style as input if you prefer

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"))

```

---

cff

---

Create **cff** objects from direct inputs

---

## Description

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

## Usage

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

## Arguments

path	<b>[Deprecated]</b> path is no longer supported, use <a href="#">cff_read_cff_citation()</a> instead.
...	Named arguments to be used for creating a cff object. If no arguments are supplied (the default behavior), a minimal valid cff object is created.

## Details

`cff()` would convert `_` in the name of the argument to `-`, e.g. `cff_version = "1.2.0"` would be converted to `cff-version = "1.2.0"`.

Valid parameters 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 [list](#) object with a special [print](#) method.

### See Also

Other core functions of [cfr](#): [cff\\_create\(\)](#), [cff\\_modify\(\)](#), [cff\\_validate\(\)](#)

### Examples

```
# Blank cff
cff()

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

# 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
```

```
# Would pass
cff_validate(new)
```

---

**cff\_create**
*Create a **cff** object from several sources*


---

## Description

Create a full and possibly valid **cff** object from a given source. This object can be written to a `*.cff` file with `cff_write()`, see **Examples**.

Most of the heavy lifting of **cfr** is done via this function.

## Usage

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

## Arguments

<code>x</code>	<p>The source that would be used for generating the <b>cff</b> object. It could be:</p> <ul style="list-style-type: none"> <li>• A missing value. That would retrieve the DESCRIPTION file on your in-development <b>R</b> package.</li> <li>• An existing <b>cff</b> object.</li> <li>• The name of an installed package (<code>"jsonlite"</code>).</li> <li>• Path to a DESCRIPTION file (<code>"./DESCRIPTION"</code>).</li> </ul>
<code>keys</code>	List of additional keys to add to the <b>cff</b> object. See <code>cff_modify()</code> .
<code>cff_version</code>	The Citation File Format schema version that the CITATION.cff file adheres to for providing the citation metadata.
<code>gh_keywords</code>	Logical TRUE/FALSE. If the package is hosted on GitHub, would you like to add the repo topics as keywords?
<code>dependencies</code>	Logical TRUE/FALSE. Would you like to add the of your package to the references CFF key?
<code>authors_roles</code>	Roles to be considered as authors of the package when generating the CITATION.cff file. See <b>Details</b> .

## Details

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

By default, only persons whose role in the DESCRIPTION file of the package is author ("aut") or maintainer ("cre") are considered to be authors of the package. The default setting can be controlled via the `authors_roles` parameter. See **Details** on `person()` to get additional insights on person roles.

## Value

A `cffi` object.

## See Also

**Guide to Citation File Format schema version 1.2.0.**

- `cffi_modify()` as the recommended way to modify a `cffi` object.
- `cffi_write()` for creating a CFF file.
- `vignette("cffi", "cffi")` shows an introduction on how manipulate `cffi` objects.
- `vignette("crosswalk", package = "cffi")` provides details on how the metadata of a package is mapped to produce a `cffi` object.

Other core functions of **cffi**: `cffi()`, `cffi_modify()`, `cffi_validate()`

## 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 fields",
  abstract = "New abstract",
  keywords = c("A", "new", "list", "of", "keywords"),
  authors = as_cffi_person("New author")
)

cffi_create(demo_file, keys = newkeys)

# Update a field on a list - i.e: authors, contacts, etc.
# We are adding a new contact here

old <- cffi_create(demo_file)

new_contact <- append(
```

```

    old$contact,
    as_cffi_person(person(
      given = "I am",
      family = "New Contact"
    ))
  )

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

```

---

cffi_gha_update	<i>Install a <a href="https://CRAN.R-project.org/package=cffrcffi">R package</a> <b>cffi</b> GitHub Action</i>
-----------------	--

---

## Description

This function would install a **GitHub Action** on your repo. The action will update your CITATION.cffi when any of these events occur:

- You publish a new release of the package.
- Your DESCRIPTION or inst/CITATION are modified.
- The action can be run also manually.

## Usage

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

## Arguments

path	Project root directory.
overwrite	Logical. If already present, do you want to overwrite your action?

## Details

Triggers on your action can be modified, see [Events that trigger workflows](#).

## Value

Invisible, this function is called by its side effects.

## See Also

Other Git/GitHub helpers provided by **cffi**: [cffi\\_git\\_hook](#)

### Examples

```
## Not run:
cffi_github_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 of `usethis::use_git_hook()`.

### Usage

```
cffi_git_hook_install()

cffi_git_hook_remove()
```

### Details

This function would install 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 would warn you if any of the following conditions are met:

- You included in a commit your DESCRIPTION or inst/CITATION file, you are not including your CITATION.cff and the CITATION.cff file is "older" than any of your DESCRIPTION or inst/CITATION file.
- You have updated your CITATION.cff but you are not including it on your commit.

### Value

Invisible. This function is called for its side effects.

### A word of caution

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

This is typically the case when you have updated your DESCRIPTION or inst/CITATION files but those changes doesn't make a change on your CITATION.cff file (i.e. you are including new dependencies).

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

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

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

### Removing the git pre-commit hook

You can remove the pre-commit hook by running `cff_git_hook_remove()`.

### See Also

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

Other Git/GitHub helpers provided by **cffr**: `cff_gha_update()`

### Examples

```
## Not run:
cff_git_hook_install()

## End(Not run)
```

---

cff\_modify

*Modify a cff object*

---

### Description

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

### Usage

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

### Arguments

`x`                      A `cff` object.

`...`                    Named arguments to be used for modifying `x`. See also `...` argument in `cff()`.

### Details

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

It is possible to add additional keys not detected by `cff_create()` using the `keys` argument. A list of valid keys can be retrieved with `cff_schema_keys()`. Please refer to **Guide to Citation File Format schema version 1.2.0**. for additional details.

### Value

A `cff` object.



**See Also**

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

See `cff()` for creating `cff` objects from scratch.

Other core functions of **cffr**: `cff()`, `cff_create()`, `cff_validate()`

**Examples**

```
x <- cff()
x

cff_validate(x)

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

x_mod

cff_validate(x_mod)
```

---

cff\_read

Read an external file as a `cff` object

---

**Description**

Read files and convert them to `cff` objects. Files supported are:

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

`cff_read()` would try to guess the type of file provided in path. However we provide a series of alias for each specific type of file:

- `cff_read_cff_citation()`, that uses `yaml::read_yaml()`.
- `cff_read_description()`, using `desc::desc()`.
- `cff_read_citation()` uses `utils::readCitationFile()`.
- `cff_read_bib()` requires **bibtex** (>= 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	Path to a file.
...	Arguments to be passed to other functions (i.e. to <a href="#">yaml::read_yaml()</a> , <a href="#">bibtex::read.bib()</a> , etc.).
cff_version	The Citation File Format schema version that the CITATION.cff file adheres to for providing the citation metadata.
gh_keywords	Logical TRUE/FALSE. If the package is hosted on GitHub, would you like to add the repo topics as keywords?
authors_roles	Roles to be considered as authors of the package when generating the CITATION.cff file. See <b>Details</b> .
meta	A list of package metadata as obtained by <a href="#">utils::packageDescription()</a> or NULL (the default). See <b>Details</b> .
encoding	Encoding to be assumed for path. See <a href="#">readLines()</a> .

**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 2023) specifies how to create dynamic CITATION files using meta object, hence the meta argument in [cff\\_read\\_citation\(\)](#) may be needed for reading some files correctly.

**Value**

- `cff_read_cff_citation()` and `cff_read_description()` returns a object with class `cff`.
- `cff_read_citation()` and `cff_read_bib()` returns an object of classes `cff_ref_lst`, `cff` according to the definitions.references specified in the [Citation File Format schema](#).

Learn more about the **cffr** class system in [cff\\_class](#).

## References

- R Core Team (2023). *Writing R Extensions*. <https://cran.r-project.org/doc/manuals/r-release/R-exts.html>
- Hernangomez D (2022). "BibTeX and CFF, a potential crosswalk." *The cfr package, Vignettes*. doi:10.21105/joss.03900, [https://docs.ropensci.org/cfr/articles/bibtex\\_cff.html](https://docs.ropensci.org/cfr/articles/bibtex_cff.html).

## See Also

The underlying functions used for reading external files:

- `yml::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).

Other functions for reading external files: `cfr_read_bib_text()`

Other functions for working with BibTeX format: `as_bibentry()`, `cfr_read_bib_text()`, `cfr_write_bib()`, `encoded_utf_to_latex()`

## Examples

```
# Create cfr object from cfr file

from_cfr_file <- cfr_read(system.file("examples/CITATION_basic.cfr",
  package = "cfr")
))

head(from_cfr_file, 7)

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

from_desc

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

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

---

cff_read_bib_text	<i>Read BibTeX markup as a <a href="#">cff_ref_lst</a> object</i>
-------------------	---

---

## Description

Convert a [character](#) representing a BibTeX entry to a [cff\\_ref\\_lst](#) object.

## Usage

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

## Arguments

x	A vector of character objects with the full BibTeX string.
encoding	Encoding to be assumed for x, see <a href="#">readLines()</a> .
...	Arguments passed on to <a href="#">cff_read_bib()</a> .

## Details

This is a helper function that writes x to a \*.bib file and reads it with [cff\\_read\\_bib\(\)](#).

This function requires **bibtex** ( $\geq 0.5.0$ ) and uses [bibtex::read.bib\(\)](#).

## Value

An object of classes [cff\\_ref\\_lst](#), [cff](#) according to the definitions.references specified in the [Citation File Format schema](#). Each element of the cff\_ref\_lst object would have classes [cff\\_ref](#), [cff](#).

## See Also

[cff\\_read\\_bib\(\)](#) for reading \*.bib files.

Other functions for working with BibTeX format: [as\\_bibentry\(\)](#), [cff\\_read\(\)](#), [cff\\_write\\_bib\(\)](#), [encoded\\_utf\\_to\\_latex\(\)](#)

Other functions for reading external files: [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)
}
```

---

cff\_schema

Schema utils

---

## Description

Helper functions with the valid values of different fields, according to the [Citation File Format schema version 1.2.0](#).

- `cff_schema_keys()` provides the valid high-level keys of the Citation File Format.
- `cff_schema_keys_license()` provides the valid [SPDX license identifier\(s\)](#) to be used on the CITATION.cff file.
- `cff_schema_definitions_person()` and `cff_schema_definitions_entity()` returns the valid fields to be included when defining a person or entity.
- `cff_schema_definitions_refs()` provides the valid keys to be used on the preferred-citation and references keys.

## Usage

```
cff_schema_keys(sorted = FALSE)
```

```
cff_schema_keys_license()
```

```
cff_schema_definitions_person()
```

```
cff_schema_definitions_entity()
```

```
cff_schema_definitions_refs()
```

### Arguments

`sorted` Logical TRUE/FALSE. Should the keys be arranged alphabetically?

### Value

A vector of characters with the names of the valid keys to be used on a Citation File Format version 1.2.0

### Source

[Guide to Citation File Format schema version 1.2.0.](#)

### Examples

```
cff_schema_keys(sorted = TRUE)

# Valid Licenses keys
head(cff_schema_keys_license(), 20)

cff_schema_definitions_person()

cff_schema_definitions_entity()

cff_schema_definitions_refs()
```

---

<code>cff_validate</code>	<i>Validate a CITATION.cff file or a <a href="#">cff</a> 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

<code>x</code>	This is expected to be either a full cff object created with <a href="#">cff_create()</a> or the path to a CITATION.cff file to be validated. In the case of a *cff file it would read with <a href="#">cff_read_cff_citation()</a> .
<code>verbose</code>	Logical TRUE/FALSE. On TRUE the function would display informative messages.

**Value**

A message indicating the result of the validation and an invisible value TRUE/FALSE. On error, the results would have an attribute "errors" containing the error summary (see **Examples** and [attr\(\)](#)).

**See Also**

[Guide to Citation File Format schema version 1.2.0.](#)

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

Other core functions of **cffi**: [cffi\(\)](#), [cffi\\_create\(\)](#), [cffi\\_modify\(\)](#)

**Examples**

```
# Full .cffi example
cffi_validate(system.file("examples/CITATION_complete.cffi", package = "cffi"))

# Validate a cffi object
cffi <- cffi_create("jsonlite")
class(cffi)
cffi_validate(cffi)

# .cffi with errors
err_f <- system.file("examples/CITATION_error.cffi", package = "cffi")
# Can manipulate the errors as data frame
res <- try(cffi_validate(err_f))

isTRUE(res)
isFALSE(res)

attr(res, "errors")

# If a CITATION file (note that is not .cffi) it throws an error
try(cffi_validate(system.file("CITATION", package = "cffi")))
```

---

cffi\_write

---

Write a CITATION.cffi file

---

**Description**

**This is the core function of the package and likely to be the only one you would need when developing a package.**

This function writes out a CITATION.cffi file for a given package. This function is basically a wrapper around [cffi\\_create\(\)](#) to both create the [cffi](#) object and write it out 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,
  dependencies = TRUE,
  validate = TRUE,
  verbose = TRUE,
  authors_roles = c("aut", "cre"),
  encoding = "UTF-8"
)
```

## Arguments

<code>x</code>	The source that would be used for generating the <code>cff</code> object. It could be: <ul style="list-style-type: none"> <li>• A missing value. That would retrieve the DESCRIPTION file on your in-development <b>R</b> package.</li> <li>• An existing <code>cff</code> object.</li> <li>• The name of an installed package (<code>"jsonlite"</code>).</li> <li>• Path to a DESCRIPTION file (<code>"./DESCRIPTION"</code>).</li> </ul>
<code>outfile</code>	The name and path of the CITATION.cff to be created.
<code>keys</code>	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 that the CITATION.cff file adheres to for providing the citation metadata.
<code>gh_keywords</code>	Logical TRUE/FALSE. If the package is hosted on GitHub, would you like to add the repo topics as keywords?
<code>dependencies</code>	Logical TRUE/FALSE. Would you like to add the of your package to the references CFF key?
<code>validate</code>	validate Logical TRUE/FALSE. Should the new file be validated using <code>cff_validate()</code> ?
<code>verbose</code>	Logical TRUE/FALSE. On TRUE the function would display informative messages.
<code>authors_roles</code>	Roles to be considered as authors of the package when generating the CITATION.cff file. See <b>Details</b> .
<code>encoding</code>	The name of the encoding to be assumed. Default is <code>"UTF-8"</code> , but it can be any other value as accepted by <code>iconv()</code> , such as <code>"ASCII//TRANSLIT"</code> .

## Details

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

When creating and writing a CITATION.cff for the first time, the function adds the pattern `"^CITATION\\.cff$"` to your `.Rbuildignore` file to avoid NOTES and WARNINGS in R CMD CHECK.



**Value**

A CITATION.cff file and an (invisible) cff object.

**See Also**

[Guide to Citation File Format schema version 1.2.0](#). This function unifies the workflow `cff_create()` + `cff_validate()` + write a file.

Other functions for creating external files: `cff_write_bib()`

**Examples**

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

cff_obj

# Force clean-up
file.remove(tmpfile)
```

---

`cff_write_bib`*Export R objects to different file types*

---

**Description**

Export **R** objects representing citations to specific file types:

- `cff_write_bib()` creates a .bib file.
- `cff_write_citation()` creates a **R** citation file as explained in Section 1.9 CITATION files of *Writing R Extensions* (R Core Team 2023).

**Usage**

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

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

## Arguments

x	A <a href="#">bibentry</a> or a <a href="#">cff</a> object.
file	Name of the file to be created. If NULL it would display the lines to be written.
append	Whether to append the entries to an existing file or not.
verbose	Display informative messages
ascii	Whether to write the entries using ASCII characters only or not.
...	Arguments passed on to <a href="#">as_bibentry.cff</a> , <a href="#">as_bibentry.cff_ref</a> , <a href="#">as_bibentry.cff_ref_lst</a>
what	Fields to extract from a full cff object. The value could be: <ul style="list-style-type: none"> <li>• preferred: This would create a single entry with the main citation info of the package (key preferred-citation).</li> <li>• references: Extract all the entries of references key.</li> <li>• all: A combination of the previous two options. This would extract both the preferred-citation and the references key.</li> </ul> See <code>vignette("crosswalk", package = "cfr")</code> .

## Details

When x is a cff object it would be converted to Bibtex using [toBibtex.cff\(\)](#).

For security reasons, if the file already exists the function would create a backup copy on the same directory.

## Value

Writes the corresponding file specified on the file parameter.

## References

- R Core Team (2023). *Writing R Extensions*. <https://cran.r-project.org/doc/manuals/r-release/R-exts.html>

## See Also

`vignette("bibtex_cff", "cfr")`, [knitr::write\\_bib\(\)](#) and the following packages:

- [bibtex](#).
- [RefManageR](#)
- [rbibutils](#)

Other functions for working with BibTeX format: [as\\_bibentry\(\)](#), [cff\\_read\(\)](#), [cff\\_read\\_bib\\_text\(\)](#), [encoded\\_utf\\_to\\_latex\(\)](#)

Other functions for creating external 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 a 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, use 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 on CRAN packages and its (approximate) match on the [SPDX License List](#).

**Usage**

```
cran_to_spdx
```

**Format**

A data frame with 94 rows and 2 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, 20
  - cff\_write\_bib, 25
- \* **core**
  - cff, 10
  - cff\_create, 12
  - cff\_modify, 16
  - cff\_validate, 22
- \* **datasets**
  - cran\_to\_spdx, 27
- \* **git**
  - cff\_gha\_update, 14
  - cff\_git\_hook, 15
- \* **reading**
  - cff\_read, 17
  - cff\_read\_bib\_text, 20
- \* **s3method**
  - as\_bibentry, 2
  - as\_cff, 5
  - as\_cff\_person, 7
- \* **schemas**
  - cff\_schema, 21
- \* **writing**
  - cff\_write, 23
  - cff\_write\_bib, 25

as.cff(as\_cff), 5

as.character(), 7

as.list(), 5

as.person.cff\_pers(), 7

as.person.cff\_pers\_lst(), 7

as\_bibentry, 2, 6, 9, 19, 20, 26

as\_bibentry.cff, 26

as\_bibentry.cff\_ref, 26

as\_bibentry.cff\_ref\_lst, 26

as\_cff, 4, 5, 9

as\_cff(), 3

as\_cff.bibentry(), 2

as\_cff\_person, 4, 6, 7

as\_cff\_person(), 6

attr(), 23

bibentry, 2, 26

bibtex::read.bib(), 17–20

cff, 2, 5, 10, 10, 11–13, 16, 17, 22–24, 26

cff(), 3, 6, 16, 17

cff\_class, 4, 6, 9, 10, 18

cff\_create, 11, 12, 17, 23

cff\_create(), 3, 6, 16, 18, 22–25

cff\_gha\_update, 14, 16

cff\_git\_hook, 14, 15

cff\_git\_hook\_install(cff\_git\_hook), 15

cff\_git\_hook\_remove(cff\_git\_hook), 15

cff\_modify, 11, 13, 16, 23

cff\_modify(), 6, 12, 13, 24

cff\_pers\_lst, 7

cff\_read, 4, 17, 20, 26

cff\_read(), 6, 17

cff\_read\_bib(cff\_read), 17

cff\_read\_bib(), 17, 20

cff\_read\_bib\_text, 4, 19, 20, 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, 20

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`, 11, 13, 17, 22  
`cff_validate()`, 25  
`cff_write`, 23, 26  
`cff_write()`, 12, 13  
`cff_write_bib`, 4, 19, 20, 25, 25  
`cff_write_bib()`, 25  
`cff_write_citation (cff_write_bib)`, 25  
`cff_write_citation()`, 25  
`character`, 20  
`cran_to_spdx`, 27  
  
`desc::desc()`, 17, 19  
  
`encoded_utf_to_latex`, 4, 19, 20, 26  
  
`format()`, 7  
`format.person()`, 7, 8  
  
`iconv()`, 24  
  
`jsonvalidate::json_validate()`, 23  
  
`knitr::write_bib()`, 26  
  
`list`, 11  
  
`person()`, 7, 13  
`print`, 11  
  
`readLines()`, 18, 20  
  
`sub-class`, 5  
  
`toBibtex()`, 2, 3, 5  
`toBibtex.cff()`, 26  
  
`usethis::use_git()`, 16  
`usethis::use_git_hook()`, 15, 16  
`utils::bibentry()`, 4, 5  
`utils::modifyList()`, 17  
`utils::packageDescription()`, 18  
`utils::person()`, 5, 9  
`utils::readCitationFile()`, 17, 19  
`utils::toBibtex()`, 4  
  
`yaml::read_yaml()`, 17–19