Citation File Format (CFF)

Specification - Version 1.0.0-beta

Stephan Druskat (mail@sdruskat.net)

24 September 2017

Abstract

The *Citation File Format* (*CFF*) is a human- *and* machine-readable format for citation files, which provide references to (research and scientific) software to be used for citation and other types of reference. The format aims to support all use cases for software citation described in 1. CFF is serialized in YAML 1.2, and is therefore Unicode-based and cross-language (in terms of both natural language scripts and programming languages). This specification, together with the Unicode standard for characters, aims to provide all the information necessary to understand CFF, and to use (i.e., write) and re-use (i.e., read, validate, convert from) it. The specification is maintained openly at https://github.com/sdruskat/citation-file-format.

Contents

Introduction

Status of this document

This document reflects the first version of the *Citation File Format* (CFF). CFF has been developed in the context of the *Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE5.1)*, which was held on 6 September 2017 in Manchester, UK. More specifically, the constraints for CFF has been developed in the discusion and speed blogging group "Development and implementation of a standard format for CITATION files", whose members were Stephan Druskat (Humboldt-Universität zu Berlin, Germany), Neil Chue Hong (Software Sustainability Institute, University of Edinburgh, UK), Raniere Silva (Software Sustainability Institute, University of Manchester, UK), Radovan Bast (University of Tromsø, Norway), Andrew Rowley (University of Manchester, UK), and Alexander Konovalov (University of St. Andrews, UK).

CFF Version 1.0 has been developed by Stephan Druskat with contributions from the following.

- tba

CFF has been developed to provide the first iteration of a format for `CITATION` files which could be recommended to readers of the blog post which has been produced by the group during the workshop and shortly after, and which will be published on the blog page of the Software Sustainability Institute.

Rationale

> The rationale for a standardized, machine- and human-readable format for `CITATION` files is discussed in 2. CFF has been developed to support all use cases for the citation of software, as discussed in 1, and thus promote attribution and credit for software in general, and research software in particular.

In a blog post 3, Robin Wilson has introduced `CITATION` files as a means to make citation information for software easily accessible. This accessibility is important, because in order to receive deserved credit for research software in the academic system - where credit is still mainly measured based on citations -, the citation information for software must be made visible; Authors will only cite software if the citation information is readily available, as there is no standard, easily deducible way (yet) to cite software, such as there is for journals for example.

Some have followed the advice, and have uploaded `CITATION` (or `CITATION.md`, or `CITATION.txt`) files to the root of the source code repository holding their software. While this practice has made for a good start, plain text, unstandardized `CITATION` files are not machine-readable, and machine-readability is a precondition for re-use of the citation information in different contexts which could further support a fair distribution of credit for research software.

Goals

The goal of CFF is to provide an all-purpose citation format (similar to BibTeX or RIS), and specifically provide optimized means of citation for software via the provision of software-specific reference keys and types, e.g., a dedicated type for source code and one for executables, and a reference key for versions.

The ultimate goal of CFF as a project is comprehensive uptake and re-use of the format by Research Software Engineers and software developers as well as by vendors and services, such as software repositories, reference managers, etc., in order to boost the visibility of citation information for research software, and empower the fair distribution of credit for software development, maintenance, etc., in academia.

Concepts

For users of other reference formats, such as BibTeX or RIS, it is important to note that in CFF, all available keys can be used for all Work Types. CFF leaves reasonability of use with format users and providers of tooling, such as conversion software for CFF and other formats. In other words, the use of keys should follow common sense. If not, it will confuse the user of the `CITATION` file, and some of the information will probably be lost in re-use scenarios such as conversion or display. If you feel that CFF does not offer a solution for your specific use case, please consider contributing to the format as described in section Contributions.

Furthermore please note that if a section of a work is referenced, this is not supported by a dedicated Work Type. Instead, the `section` key in the parent type (i.e., `book` for a section of a book, etc.) should be used.

Format

CFF `CITATION` files must be named `CITATION.cff`.

CFF is implemented in YAML 1.2, as the language provides optimal human-readability and the required core data types.

File structure

CFF `CITATION` files are made up of

- exactly one message containing instructions on how to cite the software which the file is associated with;
- one or more references, containing at least `type`, `author`, and `title` information.

For full examples, please see section Examples.

Start the file with a message object:

```
- message: If you use this software, please cite the works below.
```

Add a reference object:

```
- message: If you use this software, please cite the works below.
- type: software-code
  authors:
    - name: Druskat::Stephan
      orcid: 0000-0003-4925-7248
  title: Stephan's Research Software
  doi: 10043/zenodo.1234
```

Complete the reference with the respective information, and perhaps add more references.

```
- message: If you use this software, please cite the works below.
- type: software-code
  authors:
    - name: Druskat::Stephan
      orcid: 0000-0003-4925-7248
  title: Stephan's Research Software
  version: 1.0.4
  languages:
    - Java
    - Python
    - Haskell
    - Rust
    - JSON
  doi: 10043/zenodo.1234
- type: article
  authors:
    - name: Druskat::Stephan
      orcid: 0000-0003-4925-7248
      role: main-author
    - name: McAuthor::Clodagh
      orcid: 0000-0001-1234-5678
      role: main-author
```

```
  - name: Nown::Unk
  - name: Stant::Studentass I.
    orcid: 0000-0001-4321-4083
    role: contributor
title: A fast implementation of McAuthor's algorithm
journal: Journal of Sound Research Software
volume: 42
issue: 1
month: 1
year: 2017
start: 138
end: 147
doi: 12345/josrs.9876543
```

Formatting

CFF is YAML 1.2, so it follows the formatting rules of YAML 1.2, of which one of the most important ones is that the colon (`:`) after a key should always be followed by a whitespace.

Keys

CFF defines the following keys.

CFF Key	**CFF Data Type**	**Description**
abbreviation	String	The abbreviation of the work
abstract	String	The abstract of a work
authors	Collection of **entities**	The author of a work
collection-title	String	The title of a collection or proceedings
collection-type	String	The type of a collection
commit	String	The (e.g., Git) commit hash or (e.g., Subversion) revision number of the work
conference	Entity	The conference where the work was presented
contact	Collection of **entities**	The contact person for a work
copyright	String	The copyright information pertaining to the work
data-type	String	The data type of a data set
database	String	The name of the database where a work was accessed/is stored
database-provider	Entity	The provider of the database where a work was accessed/is stored
date-accessed	Date	The date the work has been last accessed
date-downloaded	Date	The date the work has been downloaded
date-published	Date	The date the work has been published
date-released	Date	The date the work has been released
department	String	The department where a work has been produced
dependencies	String (*URI*)	A Uniform Resource Identifier pointing to a resource that makes the dependencies of a work accessible
doi	String	The DOI of the work

CFF Key	CFF Data Type	Description
edition	String	The edition of the work
editors	Collection of **entities**	The editors of a work
editors-series	Collection of **entities**	The editors of a series in which a work has been published
entry	String	An entry in the collection that constitutes the work
filename	String	The name of the electronic file containing the work
format	String	The format in which a work is represented
institution	Entity	The institution where a work has been produced or published
isbn	String	The ISBN of the work
issn	String	The ISSN of the work
issue	Integer	The issue of a periodical in which a work appeared
issue-date	String	The publication date of the issue of a periodical in which a work appeared
issue-title	String	The name of the issue of a periodical in which the work appeared
journal	String	The name of the journal/magazine/newspaper/periodical where the work was published
keywords	Collection of strings	Keywords pertaining to the work
languages	Collection of strings	The language of the work
license	String	The license under which a work is licensed
license-url	String (*URL*)	The URL of the license text under which a work is licensed
loc-start	Integer	The line of code in the file where the work starts
loc-end	Integer	The line of code in the file where the work ends
message	String	A message providing the user with instructions on how to cite the work the `CITATION` file is attached to
month	Integer	The month in which a work has been published
nihmsid	String	The NIHMSID of a work
notes	String	Notes pertaining to the work
number	String	The accession number for a work
number-volumes	Integer	The number of volumes making up the collection in which the work has been published
pages	Integer	The number of pages of the work
patent-states	String	The states for which a patent is granted
pmcid	String	The PMCID of a work
publisher	Entity	The name of the publisher who has published the work
recipients	Collection of **entities**	The recipient of a personal communication
repository	String (*URL*)	The repository where the work is stored
repository-code	String (*URL*)	The version control system where the source code of the work is stored
repository-artifact	String (*URL*)	The repository where the (executable/binary) artifact of the work is stored

CFF Key	CFF Data Type	Description
section	String	The section of a work that is referenced
sender	Collection of **entities**	The sender of a personal communication
status	**Status string**	The publication status of the work
start	Integer	The start page of the work
thesis-type	String	The type of the thesis that is the work
title	String	The title of the work
translators	Collection of **entities**	The translator of a work
type	**Work Type string**	The type of the work
url	String (*URL*)	The URL of the work
version	String	The version of the work
volume	Integer	The volume of the periodical in which a work appeared
volume-title	String	The title of the volume in which the work appeared
year	Integer	The year in which a work has been published
year-original	Integer	The year of the original publication

Table 1: Complete list of CFF keys.

Exemplary use cases

This section details exemplary use cases for some of the keys to avoid ambiguity/misuse.

abstract

- If the work is a journal paper or other academic work: The abstract of the work.
- If the work is a film, broadcast or similar: The synopsis of the work.

department

- If the work is a thesis: The academic department where the thesis has been produced.
- If the work is a government document: The governmental department which has issued the document.

dependencies

- If the work is a software: A URL of a metadata entry, e.g., `http://depsy.org/package/python/nltk`; the URI or a URL of a file listing the software's dependencies, e.g., `file:///NOTICE` (if the artifact of the software version available from `repository-artifact` includes a `NOTICE` file in the root folder), or `https://github.com/user/project/blob/master/NOTICE`.
- If the work is a virtual machine or a software container: A URI of a file, or a URL of a website, listing the software packages that are installed on the virtual machine or included in the container.

format

- If the work is a music file: The digital format in which a musical piece is saved, e.g., MP3.
- If the work is a data set: The digital format in which the data set is saved.
- If the work is a painting: The format of the painting, e.g., the width and height of the canvas.

institution

- If the work is a report: The institution where the report has been produced.
- If the work is a case: The court where a case has been held.
- If the work is a blog post: The institution responsible for running the blog.

- If the work is a patent, legal rule or similar: The issuing institution of the patent/rule.
- If the work is a grant: The funding agency sponsoring the grant.
- If the work is a thesis: The university where a thesis has been produced.
- If the work is a statute: The institution or geographical unit which the statute adheres to.
- If the work is a historical work, illuminated manuscript or similar: The library or archive where the work is held.
- If the work is a conference: The organisation which held the conference.

languages

- If the work is a book: The language in which the book is written.
- If the work is a software: The programming/markup languages in which the software is written.

month

- If the work is a conference: The month in which the conference has been held.
- If the work is a magazine article: The month in which the magazine issue containing the article has been published.

number

- If the work is a conference paper: E.g., the submission number of the paper
- If the work is a grant: The grant number provided by the funding agency.
- If the work is a work of art: E.g., the catalogue number provided by a museum holding the artwork.
- If the work is a report: The report number of a report.
- If the work is a patent: The patent number of the work.
- If the work is a historical work, illuminated manuscript or similar: The codex or folio number of a manuscript, or the library identifier for a manuscript.

term

- If the work is a dictionary or encyclopedia: The term in the dictionary or encyclopedia that is being referenced.

title

- If the work is a case: The name of the case (e.g., Name v. Name).

version

- If the work is a software: The version of the referenced software.

Entities

Entity objects can represent different types of entities, e.g., a person, publishing company, or conference. In CFF, they are realized as collections with a defined set of keys. Only the key `name` is mandatory. When the entity represents a person, the `name` key must be formatted following the pattern `"{last names} :: {first names} {middle names}"`. This pattern is used to parse names correctly, and implicitly disambiguate person entities from other entities. Therefore, if a non-person entity name follows this pattern, it must be given as `{first part of the name} \:: {second part of the name}`.

Note that the whitespaces preceding and following the separators (`::`, `\::`) are optional.

Entity key	Entity Data Type	optional
name	String	
city	String	•
country	String	•
street	String	•
orcid	String	•
email	String	•

Entity key	Entity Data Type	optional
affiliation	String	•
tel	String	•
fax	String	•
website	String (*URL*)	•
date-start	Date	•
date-end	Date	•
location	String	•
role	**Role string**	•

Table 2: Complete list of entity keys.

Roles

An entity representing a person can be assigned a role for the purposes of specifying authorship status, e.g., to distinguish main authors of a software from contributors who have provided a small patch. The defined roles are:

Key
artist
assignee (e.g., of a patent)
main-author
benchmarker (e.g., of a software)
cartographer
composer
contributor
creator
designer
director (e.g., of a movie)
editor (e.g., of an edited book/edition)
evangelist (e.g., for a software)
insitution (e.g., issuing a standard)
inventor
manager (e.g., of a software project)
programmer
reporter (e.g., of a court case)
reporter (e.g., of a software bug)
researcher (e.g., authoring a data set)
software engineer (e.g., for a software)
technical writer (e.g., of a software documentation)
tester (e.g., of a software)
trainer

Key

Table 3: Defined roles for entities.

Statuses

Works can have a different status of publication, e.g., journal papers. CFF provides the following defined statuses for works.

Status (String)	Description
in-preparation	A work in preparation, e.g., a manuscript
abstract	The abstract of a work
submitted	A work that has been submitted for publication
in-press	A work that has been accepted for publication but has not yet been published
advance-online	A work that has been published online in advance of publication in the target medium

Table 4: Defined statuses for works

Work Types

Work Type string	Description
art	A work of art, e.g., a painting
article	
audiovisual	
bill	A legal bill
blog	A blog post
book	A book or e-book
catalogue	
conference	
conference-paper	
data	A data set
database	An aggregated or online database
dictionary	
edited-work	An edited work, e.g., a book
encyclopedia	
film-broadcast	A film or broadcast
generic	The fallback type
government-document	
grant	A research or other grant
hearing	

Work Type string	Description
historical-work	A historical work, e.g., a medieval manuscript
legal-case	
legal-rule	
magazine-article	
manual	A manual
map	A geographical map
multimedia	A multimedia file
music	A music file or sheet music
newspaper-article	
pamphlet	
patent	
personal-communication	
proceedings	Conference proceedings
report	
serial	
slides	Slides, i.e., a published slide deck
software	Software
software-code	Software source code
software-container	A software container (e.g., a docker container)
software-executable	An executable software, i.e., a binary/artifact
software-virtual-machine	A virtual machine/vm image
sound-recording	
standard	
statute	
thesis	An academic thesis
unpublished	
video	A video recording
website	

Table 5: Complete list of CFF work types.

Schema

It is planned to provide a PyKwalify schema for the validation of CFF files. This is work in progress.

Examples

A software with a DOI

Note that [1, p. 12] recommends

> […] the use of DOIs as the unique identifier due to their common usage and acceptance, particularly as they are the standard for other digital products such as publications.

Furthermore, DOIs should point to a "unique, specific software version" [1, p. 12]. Also it is recommended [1, p. 13] that:

> the [DOI] should resolve to a persistent landing page that contains metadata and a link to the software itself, rather than directly to the source code files, repository, or executable.

Therefore, a minimal `CITATION.cff` file in such a case would look similar to the following.

```yaml
- message: If you use this software, please cite it as below.
- type: software
  authors:
    - name: Druskat::Stephan
      orcid: 0000-0003-4925-7248
  title: Stephan's Research Software
  version: 1.0.4
  doi: 10043/zenodo.1234
```

A more comprehensive version could look similar to the following.

```yaml
- message: If you use this software, please cite it as below.
- type: software
  authors:
    - name: Druskat::Stephan
      orcid: 0000-0003-4925-7248
      affiliation: Humboldt-Universität zu Berlin, Dept. of German Studies and Linguistics
      email: mail@sdruskat.net
      website: https://hu.berlin/sdruskat
  title: Stephan's Research Software
  version: 1.0.4
  doi: 10043/zenodo.1234
  commit: ab3d513
  repository-code: https://github.com/sdruskat/stephans-research-software
  repository-artifact: https://hu.berlin/nexus/srs
  date-published: 2017-09-23
  dependencies: https://github.com/sdruskat/stephans-research-software/blob/srs-1.0.4/NOTICE
  keywords:
    - "McAuthor's algorithm"
    - linguistics
    - nlp
    - parser
    - deep convolutional neural network
  languages:
    - Java 1.8
    - Python 3.3
    - C
    - Haskell
    - Turbo Pascal
    - Rust
  license: Apache License, Version 2.0
  license-url: http://www.apache.org/licenses/LICENSE-2.0
  url: https://sdruskat.github.io/stephans-research-software
```

A software without a DOI

For software without a DOI, it is recommended that "the metadata should still provide information on how to access the specific software, but this may be a company's product number or a link to a website that allows the software be purchased." [1, p. 13]. Furthermore, "if the version number and release date are not available, the download date can be used. Similarly, the contact name/email is an alternative to the location/repository." [1, p. 7]

Hence, for a closed source software without a DOI for which the version number and release date cannot be determined, a `CITATION.cff` file could look like this.

```yaml
- message: If you dare to use this commercial, closed-source, unversioned software in your research, pleas
- type: software
  title: Opaquity
  number: opq-1234-XZVF-ACME-RLY
  date-downloaded: 2017-02-31
  contact:
    - name: Vader::Darth
      affiliation: Dark Side Software
      location: DS-1 Orbital Battle Station, near Scarif
      email: father@imperial-empire.com
      tel: +850 (0)123-45-666
```

Infrastructure

It is planned to provide further infrastructure (e.g., software packages), to support the following use cases for CFF:

- Creating CFF `CITATION` files
- Reading CFF `CITATION` files
- Validating CFF `CITATION` files
- Converting CFF `CITATION` files

For some use cases in software, cf. https://www.software.ac.uk/blog/2014-07-30-oh-research-software-how-shalt-i-cite-thee

Contributions

Link to `CONTRIBUTING.md`, tba.

License

References

[1]A. M. Smith, D. S. Katz, K. E. Niemeyer, and FORCE11 Software Citation Working Group, "Software citation principles," *PeerJ Computer Science*, vol. 2, p. e86, Sep. 2016 [Online]. Available: https://doi.org/10.7717/peerj-cs.86

[2]S. Druskat, "Track 2 Lightning Talk: Should CITATION files be standardized?" in *Proceedings of the Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE5.1)*, 2017 [Online]. Available: https://doi.org/10.6084/m9.figshare.3827058

[3]R. Wilson, “Encouraging citation of software–introducing cITATION files.” 2013 [Online]. Available: https://www.software.ac.uk/blog/2013-09-02-encouraging-citation-software-introducing-citation-files