Citation File Format (CFF)

Specification - Version 1.0.0-beta

Stephan Druskat (mail@sdruskat.net)

19 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 other members of the group 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 has been published on the blog page of the Software Sustainability Institute.

Rationale

- Implement enablement for principles

Goals

Implement the principles from "Software Citation Principles".

Similar efforts

Terminology

Format

- ALSO CHECK AGAINST SOFT CIT IMPLEMENTATION WG GITHUB which is meant to support implementers
- Check TAGS (e.g., yaml.org/type/) - custom tag repository?
- how to reference custom schema?

File structure

Reference types

- software
- software-source-code
- software-executable
- software-container
- software (others)

Keys

CFF defines the following keys.

CFF Key	CFF Data Type	CFF Description
abbreviation	String	The abbreviation of the work
abstract	String	The abstract of a work
authors	Collection of **entities**	The author of a work
bibtex	String	A BibTeX version of the reference
collection-title	String	The title of a collection or proceedings
commit	String	The commit hash or 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-download	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
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
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

CFF Key	CFF Data Type	CFF Description
license-url	String (*URL*)	The URL of the license text under which a work is licensed
locfrom	Integer	The line of code in the file where the work starts
locto	Integer	The line of code in the file where the work ends
nihmsid	String	The NIHMSID of a 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-executable	String (*URL*)	The repository where the executable version of the work is stored
sender	Collection of **entities**	The sender of a personal communication
start	Integer	The start page of the work
thesis-type	String	The type of the thesis that is the work
translators	Collection of **entities**	The translator of a work
type	**Reference 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.

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}"`. When the entity does *not* represent a person, the value for `name` must not include a comma `,`.

Entity key	Entity Data Type	optional
name	String	

Entity key	Entity Data Type	optional
city	String	•
country	String	•
street	String	•
orcid	String	•
email	String	•
affiliation	String	•
tel	String	•
fax	String	•
website	String (*URL*)	•
datefrom	Date	•
dateto	Date	•
location	String	•
role	**Role string**	•

Table 2: Complete list of entity keys.

Roles

An entity representing a person can be assigned a role. The defined roles are:

Key
artist
assignee (e.g., of a patent)
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)

Key
software engineer (e.g., for a software)
technical writer (e.g., of a software documentation)
tester (e.g., of a software)
trainer

Table 3: Defined roles for entities.

Reference types

Software-specific keys

These keys aim to implement the basic and further requirements for the use cases of software citation presented in [1, p. 6].

- Unique identifier
- Software name
- Author(s)
- firstname
- middlename
- lastname
- email
- orcid
- contributor role? E.g., !author, !contributor, !tester, !benchmarker, !documenter, !evangelist, !engineer, !designer
  - author:
  - contributor!
  - tester?
  - benchmarker?
  - documenter?
  - evangelist?
  - engineer?
  - designer?
  - (patcher?)
  - (manager?)
  - trainer?
- Contributor role (under author?) - *what's this?*
- Version number
- Git commit hash (if no DOI)
- Subversion revision no. (if no DOI)
- Release date - **DATES: FORMAT?**
- Location/repo
- location (e.g., webservice, closed source)
- repository (defined as what?)

- Indexed citations - *what's this?*
- Software license
- Description
- Keywords
- Download date (if no version number and release date is available)
- contact name (person!) (this + email if no location/repo is available)
- doi (or use uid? create one key for all possibilities mentioned in principles: RRID, etc. – research them?), make them all "point" to `uid`
- uid:version
- uid[:general]
- uid:latest
- url (general use)
- bibtex (for bibtex-formatted entries)
- What about credit chains? Should thy play a role, i.e., should dependencies & "influences" (software that a software is derived from) be noted? How? Link to DOI! (not note anythin that it indirectly relates on, so just note dependencies, not dependencies of dependencies), make it possible to link to stuf like depsy for dependency trees

Non-software specific keys

CHECK IF THIS MAKES SENSE!

Key	Type	CodeMeta property	Description
vcs	URL	codeRepository	Link to the repository where the un-compiled, human readable code and related code is located (SVN, github, CodePlex).
uid	UID	-	- cf. Access to Software
version	number	-	In combination with `vcs` if no UID is available
commit	hash	-	In combination with `vcs` if no UID is available
landing-page	URL	-	According to software citation principles paper, what the UID should resolve to
contact	name	-	If software isn't publicly available
email	email address	-	Multi-purpose sub key for person, e.g., for the case that software isn't publicly available
release-date	date		
download-date	date		
authors	list		
contributors	list		
dependencies	???	???	???

Schema

- Define one! (PyKwalify?)

Examples

```
- message: "If you use this software, please cite the software itself and the journal paper describing its
- TYPE: "SOFTWARE (Use YAML explicit typing? !software)"
  authors:
    - firstname: Stephan
      lastname: Druskat
      orcid: 1234-5678-9012-3456
    - firstname: Neil
      lastname: "Chue Hong"
    - firstname: Radovan
      lastname: Bast
      orcid: 1234-5678-9012-3456
  csv: "git://github.com/sdruskat/cffp"
  doi: 10043/zenodo.1234
  title: "Citation File Format Parser"
  version: "1.1.2"
- TYPE: JOURNAL
  authors:
    - firstname: Stephan
      lastname: Druskat
      orcid: 1234-5678-9012-3456
    - firstname: Neil
      lastname: "Chue Hong"
    - firstname: Radovan
      lastname: Bast2
      orcid: 1234-5678-9012-3456
  day: 9
  doi: 10043/zenodo.12345
  frompage: 1
  issue: 11
  journal: "Journal for Open Research Software (JORS)"
  month: september
  subtitle: "Version 1.0"
  title: "The Citation File Format"
  topage: 34
  volume: 2017
  year: 2017
```

Infrastructure

Creating CFF `CITATION` files

Reading CFF `CITATION` files

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

Validating CFF `CITATION` files

Converting CFF `CITATION` files

Notes

- Virtual machines? UID?
- Containers (docker)? UID?
- Active instances?

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