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.

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.

Table of Contents

Introduction

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

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

Key	Type	CodeMeta property	Description
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