Citation File Format - Core Module (CFF-Core)

1.0.0

Stephan Druskat (mail@sdruskat.net)

11 December 2017

Abstract

The *Citation File Format* (*CFF*) is a human- *and* machine-readable format for CITATION files. These files provide citation metadata for (research and scientific) software. The format aims to support citation-specific use cases for software citation described in [1]. CFF is serialized in YAML 1.2, and is therefore Unicode-based and cross-language. 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. These specifications are maintained openly at https://github.com/citation-file-format/citation-file-format. CFF is a source format for CodeMeta JSON files.

Contents

Known bugs: citation-file-format#41

Introduction

Status of this document

This document reflects version 1.0.0 of the *Citation File Format* (CFF)'s *Core* branch (cf. Status of the format). 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 have been developed in the discussion 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).

This version is based partly on the changes introduced during the FORCE2017 hackathon of the FORCE11 Software Citation Implementation Working Group in order to make CFF crosswalkable in CodeMeta. It mainly introduces format modularization, and the final specification of CFF-Core as the module for providing citation metadata.

Future versions will additionally introduce at least a metadata module to capture the maximum amount of metadata that can be represented in a CodeMeta JSON file.

CFF Version 1.0.0 has been developed by Stephan Druskat with contributions from the following.

- Morane Gruenpeter (@moranegg) helped prepare CFF 1.0-RC1 for the CodeMeta crosswalk, and tested the crosswalk.
- Neil Chue Hong (@npch) consulted and helped prepare CFF 1.0-RC1 for the CodeMeta crosswalk.

Reporters

- Radovan Bast (@bast)
- Raniere Silva (@rgaiacs)
- Michael R. Crusoe (@mr-c)

CFF has been developed to provide 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 more detail in [2]. CFF has been developed to support all citation-specific 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, free form `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.

Status of the format

CFF-Core has been branched out to address issue citation-file-format/citation-file-format#23. While an ideal format for software citation would arguably implement transitive credit [4], there is no concrete implementation yet that is available for practical use by software creators. The most concrete suggestion for an implementation is [5], the application of which, however, seems impractical from a usability point of view in terms of human-writability. Especially regarding the current state of the practice of providing citation metadata for software [6], it seems that at this point in time, a lower-threshold approach - in terms of technical complexity - could potentially drive higher participation in the provision of software citation metadata.

On the other end of the complexity spectrum, free form `CITATION` files as suggested by Robin Wilson [3] provide an easy-to-access way of providing citation metadata for software. However, as these files are not reliably formatted, and thus not machine-readable, their potential for re-use is rather low. Re-use of software citation metadata, however, is a key factor in the promotion of software citation along the software citation principles [1], and ultimately the fair distribution of credit for software in science.

CFF aims to provide a compromise between the ideal state of software citation, i.e., transitive credit, and the state of the practice, i.e., free form `CITATION` files.

CFF-Core, in this context, provides an implementation for basic software citation metadata, so that creators of research software can supply relevant metadata for citation easily. CFF-Core aims at covering use cases *1 (Use software for a paper)*, *2 (Use software in/with new software)*, and *15 (Store software entry)* as defined in the software citation principles paper [1].

CFF-Core can be used as a source format for CodeMeta JSON, which is emerging as the standard format for software metadata.

Further development of CFF will include a CFF-Meta module which will provide a set of keys to provide more general metadata for research software. CFF-Meta will implement the complete set of CodeMeta keys, and will be usable as an extension to CFF-Core.

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, cf. Reference types.

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. This can partly be achieved through CFF's compatibility with CodeMeta.

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 reference types. For now, 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 reference type. Instead, the `section` key should be used within a specific "parent" (i.e., `book` for a section of a book, etc.).

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. For details, see the YAML 1.2 Specifications [7].

Formatting

CFF 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.

Structure is determined by indentation, i.e., lines containing nested elements must be indented by at least one whitespace character, although using at least two whitespaces as a standard for indentation preserves readability.

Value strings can (and sometimes should) be double-quoted, e.g. `"string"`, especially when they contain YAML special characters, or special characters in general. These include:

```
: { } [ ] , & * # ? | - < > = ! % @ \
```

File structure

CFF `CITATION` files represent YAML 1.2 dictionaries ("maps") with the following keys listed in the table below. These primary keys are used to specify

- the version of CFF in use (`cff-version`);
- a message which should be conveyed to the user of the software, along the lines of "If you use this software, please cite it as follows" (`message`);
- the citation metadata for the software version itself, according to [1], i.e., metadata that can be picked up in a CodeMeta JSON file;
- optionally, a list of references which should be cited in different use cases or scopes, e.g., a software paper describing the abstract concepts of the software (`references`).

`cff-version`

`cff-version` must specify the exact version of the Citation File Format that is used for the file.

```
cff-version: 1.0.0
```

`message`

`message` must specify instructions to users on how to cite the software the CITATION.cff file is associated with.

```
message: "Please cite the following works when using this software."
```

Software citation metadata

The primary citation metadata provided to users that wish to cite the software version which the CFF file is for. This metadata can be provided via a subset of the keys from the table below. The keys follow the basic requirements for software citation metadata for the three basic use cases, as detailed in [1, p. 31] and reproduced below for convenience.

Table 1: Basic requirements for citation use cases, reproduced from [1, p. 31]. Bullet points (•) indicate that the use case depends on that metadata, plus signs (+) indicate that the use case would benefit from that metadata if available.

Use case:	Use software for a paper	Use software in/with new software	Store software entry
Unique identifier	•	•	•
Software name	•	•	•
Author(s)	•	•	•
Version number	•	•	•
Release date	•	•	•
Location/repository	•	•	•
Software license	+	+	
Description	+	+	+
Keywords			+

Provision of the metadata should follow the best practices detailed in [1], i.e.:

- **Give credit** where credit is due: Make sure that you include every person in the authors list who deserves to be listed as an author. This may include people that have not contributed lines of code, but have contributed as testers, designers, reporters, managers, etc.
- Provide a **unique identifier**: Publish your software version via services that provide it with a DOI, e.g., Zenodo or figshare. This also ensures **accessibility** of the software, as in these cases the unique identifier points to a landing page rather than an actual software product.
- Be **specific** by providing a version number, and citation metadata pertaining to that version. If possible, let your unique identifier point to a software version rather than the "software as such" (all versions of the software). If you want to provide the user with the possibility to cite the software as such in addition to a specific version, create a reference to the software landing page, a software paper, or similar in the CFF file.
  If, for some reason, you cannot provide a DOI for your software version, provide the version number or a commit reference (e.g., a Git hash or a Subversion revision number), and a URL to the source code or build artifact repository.
  If your software is closed source and you cannot provide a DOI, provide the version number or other version identifier, contact details, and a URL which points to the software landing page, homepage, or similar.

Finally, following the software citation principle of *Persistence*, make sure that the citation metadata persists, even when the software's lifespan is over.

CFF-Core provides the following keys for software citation metadata.

Table 2: CFF-Core keys and accepted data types for the provision of citation metadata.

CFF key	CFF data type	Description
`abstract`	String	A description of the software (version)
`authors`	Collection of entity or person objects	The author(s) of the software
`commit`	String	The commit hash or revision number of the software version
`contact`	Collection of entity or person objects	The contact person, group, company, etc. for the software version
`date-released`	Date	The release date of the software version
`doi`	String	The DOI of the work (not the resolver URL, i.e., *10.5281/zenodo.1003150*, not *http://doi.org/10.5281/zenodo.1003150*)
`keywords`	Collection of strings	Keywords pertaining to the software version
`license`	SPDX License List Identifier string (or name string for non-standard licenses)	The license the software version is licensed under
`license-url`	String (URL)	The URL of the license text under which the software version is licensed (only for non-standard licenses not included in the SPDX License List)
`repository`	String (URL)	The URL to the software version in a repository (when the repository is neither a source code repository or a build artifact repository)
`repository-code`	String (URL)	The URL to the software version in a source code repository
`repository-artifact`	String (URL)	The URL to the software version in a build artifact/binary/release repository
`title`	String	The name of the software (may include a specific name for the software version)
`url`	String (URL)	The URL to a landing page/website for the software version
`version`	String	The version of the software

`references`

Provides an optional list of references pertaining to the software version, or the software itself, e.g., a software paper describing the abstract concepts of the software, a paper describing an algorithm that has been implemented in the software version, etc.

A reference item, i.e., an item in the list under `references`, must at least specify values for the following mandatory keys: `type`, `authors`, `title`.

`type` must specify the type of the referenced work. For a list of available values, cf. reference types.

`authors` must specify a list of person objects.

`title` must specify the title of the referenced work.

Example:

```yaml
cff-version: 1.0.0
message: "Please cite the following works when using this software."
...
references:
  - type: book
```

```
  authors:
    - ...
  title: The science of citation
- type: software
  authors:
    - ...
  title: Software Citation Tool
```

Additionally, it can contain any further reference keys. In version 1.0.0, CFF does not specify a strict schema where specific reference types can only contain specific reference keys, although this may be implemented in future versions.

Reference keys

CFF-Core defines the following reference keys.

Table 3: Complete list of CFF-Core reference keys.

CFF Key	CFF Data Type	Description
`abbreviation`	String	The abbreviation of the work
`abstract`	String	The abstract of a work
`authors`	Collection of *entity* or *person objects*	The author of a work
`collection-doi`	String	The DOI of a collection conttaining the 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 object*	The conference where the work was presented
`contact`	Collection of *entity* or *person objects*	The contact person, group, company, etc. 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 object*	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
`doi`	String	The DOI of the work
`edition`	String	The edition of the work
`editors`	Collection of *entity* or *person objects*	The editors of a work
`editors-series`	Collection of *entity* or *person objects*	The editors of a series in which a work has been published
`end`	Integer	The end page of the work
`entry`	String	An entry in the collection that constitutes the work

CFF Key	CFF Data Type	Description
`filename`	String	The name of the electronic file containing the work
`format`	String	The format in which a work is represented
`institution`	*Entity object*	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 ISO 639 *language strings*	The language of the work
`license`	*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
`location`	*Entity object*	The location of the work
`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
`medium`	String	The medium of the work
`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 object*	The publisher who has published the work
`recipients`	Collection of *entity* or *person objects*	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
`scope`	String	The scope of the reference, e.g., the section of the work it adheres to
`section`	String	The section of a work that is referenced
`senders`	Collection of *person objects*	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

CFF Key	CFF Data Type	Description
`title`	String	The title of the work
`translators`	Collection of *entity* or *person objects*	The translator of a work
`type`	*Reference types 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

Notable reference keys

`conference`, **`database-provider`**, **`institution`**, **`publisher`**

These keys take an entity object as value. Entity objects reference named entities and provide a fixed set of keys, such as `name` and contact information.

Example:

```yaml
references:
  - type: book
    publisher:
      - name: PeerJ
        city: London
        country: GB
        website: https://peerj.com/
```

`authors`, **`contact`**, **`editors`**, **`editors-series`**, **`recipients`**, **`senders`**, **`translators`**

These keys take a collection of person objects as value. Person objects provide a fixed set of keys to reference individuals, including a detailed set for specifiying personal names, an affiliation, etc.

Example:

```yaml
references:
  - type: software
    authors:
      - family-names: Druskat
        given-names: Stephan
        orcid: 0000-0003-4925-7248
        affiliation: "Humboldt-Universität zu Berlin"
        email: "mail@sdruskat.net"
        website: http://sdruskat.net
      - family-names: Beethoven
        name-particle: van
        given-names: Ludwig
      - family-names: Fernández de Córdoba
        given-names: Gonzalo
        name-suffix: Jr.
    ...
```

`type`, **`languages`**, **`status`**

These keys only take values from a defined set, cf. the respective sections:

- Reference types

- Language strings
- Status strings

`license-url`, **`repository`**, **`repository-code`**, **`repository-artifact`**, **`url`**

These keys take URL strings as values. URLs will be validated by a regular expression, such as the one provided in a GitHub Gist by Diego Perini.

`keywords`

This key takes a collection of strings.

Example:

```
references:
  - type: software
    keywords:
      - linguistics
      - "multi-layer annotation"
      - web service
    ...
```

`scope`

A reference item can specify a more detailed scope for the reference, via the reference key `scope`. This key can be useful if certain references should only be cited under specific circumstances, e.g., only when a specific package of the software is used. In such a case, the package would ideally have its own CFF file, but if this is not possible for whatever reason, the `scope` key may come in handy.

For a discussion of this key, cf. issue citation-file-format/citation-file-format#15.

Example:

```
references:
  - scope: "Cite this paper when you run software X with flag --xy"
    type: article
    ...
```

Exemplary uses

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.

`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 conference: The organisation which held the conference.

`languages`

- If the work is a book: The language in which the book is written.

`location`

- If the work is an artwork: E.g., the museum holding the work.
- If the work is a historical work, illuminated manuscript or similar: The library or archive where the work is held.

`medium`

- If the work is an artwork: The medium of the artwork, e.g., "photograph", "painting", "oil on canvas", etc.
- If the work is a book or similar: Whether it is a printed book or an ebook.

`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.

Reference types

Table 4: Complete list of CFF reference types.

Reference 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`	

Reference type string	Description
`film-broadcast`	A film or broadcast
`generic`	The fallback type
`government-document`	
`grant`	A research or other grant
`hearing`	
`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`	

Objects

Entity objects

Entity objects can represent different types of entities, e.g., a publishing company, or conference. In CFF, they are realized as collections with a defined set of keys. Only the key `name` is mandatory.

Table 5: Complete list of keys for entity objects.

Entity key	Entity Data Type	optional
`name`	String	
`address`	String	•
`city`	String	•
`region`	String	•
`post-code`	String	•
`country`	String	•
`orcid`	String	•
`email`	String	•
`tel`	String	•

Entity key	Entity Data Type	optional
`fax`	String	•
`website`	String (*URL*)	•
`date-start`	Date	•
`date-end`	Date	•
`location`	String	•

Exemplary uses

`address`

- To be used for street names and house numbers, etc.

`region`

- To be used for, e.g., states (as in US states or German federal states).

`post-code`

- The post code or zip code of an address.

`country`

- The ISO 3166-1 alpha-2 country code for a country. A list of ISO 3166-1 alpha-2 codes can be found at Wikipedia:ISO 3166-1.

Example:

```
references:
  - type: book
    publisher:
      - name: PeerJ
        city: London
        country: GB
```

`date-start` and `date-end`

- The start and end date of, e.g., a conference. This must be formatted according to ISO 8601, e.g., `YYYY-MM-DD`, or `2017-10-04T16:20:57+00:00`.

Person objects

A person object represents a person. In CFF, person objects are realized as collections with a defined set of keys, of which only `family-names` and `given-names` are mandatory.

Table 6: Complete list of keys for person objects.

Entity key	Entity Data Type	optional
`family-names`	String	
`given-names`	String	
`name-particle`	String	•
`name-suffix`	String	•
`affiliation`	String	•
`address`	String	•
`city`	String	•
`region`	String	•
`post-code`	String	•
`country`	String	•
`orcid`	String	•
`email`	String	•

Entity key	Entity Data Type	optional
`tel`	String	•
`fax`	String	•
`website`	String (*URL*)	•

Exemplary uses

Name keys

CFF aims to implement a culturally neutral model for personal names, according to the suggestions on splitting personal names by the W3C and the implementation of personal name splitting in BibTeX [8].

To this end, CFF provides four generic keys to specify personal names:

1. Values for `family-names` specify family names, including combinations of given and patronymic forms, such as *Guðmundsdóttir* or *bin Osman*; double names with or without hyphen, such as *Leutheusser-Schnarrenberger* or *Sánchez Vicario*. It can potentially also specify names that include prepositions or (nobiliary) particles, especially if they occur in between family names such as in Spanish- or Portuguese-origin names, such as *Fernández de Córdoba*.
2. Values for `given-names` specify given and any other names.
3. Values for `name-particle` specify nobiliary particles and prepositions, such as in Ludwig *van* Beethoven or Rafael *van der* Vaart.
4. Values for `name-suffix` specify suffixes such as *Jr.* or *III* (as in Frank Edwin Wright *III*).

Note that these keys may still not be optimal for, e.g., Icelandic names which do not have the concept of family names, or Chinese generation names, but the alternative is highly localized customization, which would be counterintuitive as to CFF's goal to be easily accessible. Thus, it is ultimately the task of CFF file authors to find the optimal name split in any given case.

`affiliation`

- To specify the affiliation of a person, e.g., a university, research centre, etc.

Address keys

- Cf. Entity objects for details.

`orcid`

- To specify an ORCID identifier in the format `dddd-dddd-dddd-dddd`, e.g., `0000-0003-4925-7248`.

Specified value strings

The keys `status`, `license`, and `languages` can only take values from a fixed set of strings. These are specified below.

Status strings

Works can have a different status of publication, e.g., journal papers. CFF specifies the following value strings for the key `status`.

Table 7: Defined statuses for works.

Status (String)	Description
`in-preparation`	A work in preparation, e.g., a manuscript (covers drafts)
`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

Status (String)	Description
`advance-online`	A work that has been published online in advance of publication in the target medium
`preprint`	A work that has been published as a preprint before peer review

License strings

License strings must conform with the SPDX Licenses list, i.e., a license must be specified via the short identifier from the list. If a license is not included in the SPDX Licenses list, the `license-url` should be provided as a fallback.

Example:

```
references:
  - type: software
    authors:
      - ...
    title: My Research Tool
    license: Apache-2.0
  - type: software
    authors:
      - ...
    title: Obscure Research Tool
    license-url: http://r3s34archs0ft.com/eula
```

Language strings

Natural languages as a value for the key `languages` are specified via their respective 3-character ISO 639-3 code. A list of ISO 639-3 codes in maintained at Wikipedia:List of ISO 639-3 codes. Alternatively, a language's 2-character ISO 639-1 code may be used. A list of ISO 639-1 codes is maintained at Wikipedia:List of ISO 639-1 codes.

Example for a work in both English and Daakaka:

```
references:
  - type: book
    ...
    languages:
      - en
      - bpa
```

Schema

Work is still in progress to provide a schema for CFF, against which CFF files can be validated.

Examples

Software examples

The main focus of CFF-Core is to comprehensively cover the provision of citation metadata for software. To this end, it should always be used based on the Software Citation Principles [1]! Please make sure you follow the best practices wherever possible. Two typical scenarios for software citation metadata include the existence and respectively lack of a DOI for the software for which citation metadata is provided, for both of which examples follow.

A software with a DOI

Note that [1, p. 12] recommend

> […] 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
cff-version: 1.0.0
message: If you use this software, please cite it as below.
references:
  - type: software
    authors:
      - family-names: Druskat
        given-names: Stephan
        orcid: 0000-0003-4925-7248
    title: My Research Tool
    version: 1.0.4
    doi: 10043/zenodo.1234
```

A more comprehensive version could look similar to the following.

```yaml
cff-version: 1.0.0
message: If you use this software, please cite it as below.
references:
  - type: software
    authors:
      - family-names: Druskat
        given-names: 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: My Research Tool
    version: 1.0.4
    doi: 10043/zenodo.1234
    repository-code: https://github.com/sdruskat/my-research-tool
    repository-artifact: https://hu.berlin/nexus/mrt
    date-published: 2017-09-23
    keywords:
      - "McAuthor's algorithm"
      - linguistics
      - nlp
      - parser
      - deep convolutional neural network
    license: Apache-2.0
    url: https://sdruskat.github.io/my-research-tool
```

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
cff-version: 1.0.0
message: "If you dare to use this commercial, closed-source, unversioned software
in your research, please at least cite it as below."
references:
  - type: software
    title: Opaquity
    number: opq-1234-XZVF-ACME-RLY
    date-downloaded: 2017-02-31
    contact:
      - family-names: Vader
        given-names: Darth
        affiliation: Dark Side Software
        location: DS-1 Orbital Battle Station, near Scarif
        email: father@imperial-empire.com
        tel: +850 (0)123-45-666
```

`software` (with two references)

```yaml
cff-version: 1.0.0
message: "If you use My Research Tool, please cite both the software and the
outline paper."
references:
  - type: software
    authors:
      - family-names: Doe
        given-names: Jane
      - family-names: Bielefeld
        name-particle: von
        given-names: Arthur
      - family-names: McAuthor
        given-names: Juniper
        name-suffix: Jr.
    title: My Research Tool
    doi: 10043/zenodo.1234
  - type: article
    authors:
      - family-names: Doe
        given-names: Jane
      - family-names: Bielefeld
        name-particle: von
        given-names: Arthur
    title: "My Research Tool: A 100% accuracy syntax parser for all languages"
    year: 2099
    journal: Journal of Hard Science Fiction
    volume: 42
    issue: 13
    doi: 10.9999/hardscifi-lang.42132
```

`software-code` (without a DOI: code repository + commit)

> We recognize that there are certain situations where it may not be possible to follow the recommended best-practice. For example, if (1) the software authors did not register a DOI and/or release a specific version, or (2) the version of the software used does not match what is available to cite. In those cases, falling back on a combination of the repository URL and version number/commit hash would be an appropriate way to cite the software used. [1, p. 12]

```yaml
cff-version: 1.0.0
message: "If you use this MRT alpha snapshot version, please cite."
references:
  - type: software-code
    authors:
      - family-names: Doe
        given-names: Jane
    title: My Research Tool Prototype
    version: 0.0.1-alpha1-build1507284872
    repository-code: https://github.com/doe/mrt
    commit: 160d54f9e935c914df38c1ffda752112b5c979a8
```

`software-container`

```yaml
cff-version: 1.0.0
message: "If you use the MRT Docker container, please cite the following."
references:
  - type: software-container
    authors:
      - name: "Humboldt-Universität zu Berlin"
        website: https://www.linguistik.hu-berlin.de/
      - family-names: Doe
        given-names: Jane
    title: mrt-iain-m-banks
    version: 1.0.4 (Iain M. Banks)
    url: https://github.com/doe/docker-brew-mrt-core/blob/160d54f9e935/iain/Dockerfile
    repository: https://hub.docker.hu-berlin.de/_/mrt-iain-m-banks/
```

`software-executable`

```yaml
cff-version: 1.0.0
message: "If you use MRT, please cite the following."
references:
  - type: software-executable
    authors:
      - family-names: Doe
        given-names: Jane
    title: My Research Tool Kickstarter
    version: 2.0.0
    doi: 10043/zenodo.1234
    repository-artifact: https://hu.berlin/nexus/mrt-kickstarter
    filename: mrt2-kickstarter.exe
```

Other examples

art

```yaml
cff-version: 1.0.0
message: "If you use this software, please cite the following."
references:
  - type: art
    authors:
      - family-names: Picasso
        given-names: Pablo
    title: Guernica
    year: 1937
    medium: Oil on canvas
    format: 349.3cm x 776.6cm
    location:
      - name: Museo Reina Sofia
        city: Madrid
        country: ES
```

article

```yaml
cff-version: 1.0.0
message: "If you use this software, please cite the following paper."
references:
  - type: article
    authors:
      - family-names: Smith
        given-names: Arfon M.
      - family-names: Katz
        given-names: Daniel S.
        affiliation: "National Center for Supercomputing Applications &
        Electrical and Computer Engineering Department & School of Information
        Sciences, University of Illinois at Urbana-Champaign, Urbana, Illinois,
        United States"
        orcid: 0000-0001-5934-7525
      - family-names: Niemeyer
        given-names: Kyle E.
      - name: "FORCE11 Software Citation Working Group"
        website: https://www.force11.org/group/software-citation-working-group
    title: "Software citation principles"
    year: 2016
    journal: PeerJ Computer Science
    volume: 2
    issue: e86
    doi: 10.7717/peerj-cs.86
    url: https://doi.org/10.7717/peerj-cs.86
```

blog

```yaml
cff-version: 1.0.0
message: "If you use MRT in your research, please cite the following blog article."
references:
  - type: blog
    authors:
      - family-names: Doe
```

```
      given-names: Jane
  title: "Implement a 100% accuracy syntax parser for all languages? No probs!"
  date-published: 2017-09-23
  url: https://hu-berlin.de/blogs/jdoe/2017/09/23/if-only
  institution:
    - name: "Humboldt-Universität zu Berlin"
      city: Berlin
      country: DE
```

book

```
cff-version: 1.0.0
message: "If you use MRT for your research, please cite the following book."
references:
  - type: book
    authors:
      - family-names: Doe
        given-names: Jane
    title: "The future of syntax parsing"
    year: 2017
    publisher:
      - name: Far Out Publications
        city: Bielefeld
    medium: print
```

conference-paper

```
cff-version: 1.0.0
message: "If you use MRT for your research, please cite the following."
references:
  - type: conference-paper
    authors:
      - family-names: Doe
        given-names: Jane
    title: "Ultimate-accuracy syntax parsing with My Research Tool"
    year: 2017
    collection-title: "Proceedings of the 1st Conference on Wishful Thinking"
    collection-doi: 10043.zenodo.4321
    editors:
      - family-names: Kirk
        given-names: James T.
    conference:
      - name: 1st Conference on Wishful Thinking
        location: Spock's Inn Hotel and Bar
        address: 123 Main St
        city: Bielefeld
        region: Jarvis Island
        post-code: 12345
        country: UM
        date-start: 2017-04-01
        date-end: 2017-04-01
    start: 42
    end: 45
    doi: 10043/zenodo.1234
```

edited-work

Note that the editors of the edited work must be specified under the `authors` key. Specific citation styles may or may not attach a suffix to the authors, such as ", eds." or similar.

```yaml
cff-version: 1.0.0
message: "If you use MRT, please cite the following."
references:
  - type: edited-work
    authors:
      - family-names: Doe
        given-names: Jane
    title: "Ultimate-accuracy parsing in practice"
    year: 2017
    publisher:
      - name: Far Out Publications
        city: Bielefeld
        country: DE
```

report

```yaml
cff-version: 1.0.0
message: "If you use MRT in your research, please cite the following."
references:
  - type: report
    authors:
      - name: Fictional Parsing Interest Group, ACME Inc.
    title: "100% accuracy syntax parsing at ACME"
    url: http://www.acme.com/sigs/fp/reports/hpsp.pdf
    year: 2017
    date-accessed: 2017-09-23
```

thesis

```yaml
cff-version: 1.0.0
message: "If you use MRT in your research, please cite the following."
references:
  - type: thesis
    authors:
      - family-names: Doe
        given-names: Jane
    title: "A high accuracy syntax parser in Visual Basic"
    thesis-type: PhD
    year: 2017
    department: Dept. of Universal Language Philosophy
    institution:
      - name: "Humboldt-Universität zu Berlin"
        city: Berlin
        country: DE
    database: Thesiserver
    date-accessed: 2017-09-23
    date-published: 2017-03-21
    url: http://thesiserver.hu-berlin.de/2017/march/phd/doe-12345
```

Infrastructure

The roadmap for CFF plans for the provision of further infrastructure (e.g., software packages and web services), to support the following use cases for CFF:

- Creating CFF files
- Reading CFF files
- Validating CFF files
- Converting CFF files

Contributions

Contributions to the format specifications are welcome! For details on how to contribute, please refer to the GitHub repository for CFF at http://github.com/sdruskat/citation-file-format.

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

[4] D. S. Katz, "Transitive credit as a means to address social and technological concerns stemming from citation and attribution of digital products," *Journal of Open Research Software*, vol. 2, no. 1, p. e20, 2014.

[5] D. S. Katz and A. M. Smith, "Implementing transitive credit with JSON-LD," *Journal of Open Research Software*, vol. 3, no. e7, 2015 [Online]. Available: https://doi.org/10.5334/jors.by

[6] J. Howison and J. Bullard, "Software in the scientific literature: Problems with seeing, finding, and using software mentioned in the biology literature," *Journal of the Association for Information Science and Technology*, vol. 67, no. 9, pp. 2137–2155, 2016 [Online]. Available: http://dx.doi.org/10.1002/asi.23538

[7] O. Ben-Kiki, C. Evans, and I. döt Net, "YAML Ain't Markup Language (YAML™) Version 1.2. 3rd Edition, Patched at 2009-10-01." 2009 [Online]. Available: http://yaml.org/spec/1.2/spec.html

[8] J.-M. Hufflen, "Names in bibtex and mlBibTeX," *TUGboat*, vol. 27, no. 2, pp. 243–253, Nov. 2006 [Online]. Available: https://www.tug.org/TUGboat/tb27-2/tb87hufflen.pdf