Citation File Format (CFF)

Stephan Druskat (mail@sdruskat.net)

04 October 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. These specifications are maintained openly at https://github.com/sdruskat/citation-file-format.

Contents

Introduction

Status of this document

This document reflects the version {{ page.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 {{ page.version }} has been developed by Stephan Druskat with contributions from the following.

- Radovan Bast (@bast): Reporter
- Raniere Silva (@rgaiacs): Reporter

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 more detail 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, 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.

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. 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 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. For details, see the YAML 1.2 Specifications **???**.

File structure

CFF `CITATION` files are YAML 1.2 dictionaries ("maps") with three mandatory keys: `cff-version`, `message`, `references`.

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

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

`references` must specify a list of references.

Example:

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

Reference structure

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

Additionally, it can contain any further reference keys. In version {{ page.version }}, 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.

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.

Reference keys

CFF defines the following reference keys.

Table 1: Complete list of CFF keys.

CFF Key	CFF Data Type	Description
abbreviation	String	The abbreviation of the work

CFF Key	**CFF Data Type**	**Description**
abstract	String	The abstract of a work
authors	Collection of **entities**	The author of a work
cff-version	String	The version of Citation File Format this file is formatted in
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
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 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 **Language 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

CFF Key	**CFF Data Type**	**Description**
pmcid	String	The PMCID of a work
programming-languages	Collection of **Programming language strings**	The programming language of the 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
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	**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

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.

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.

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.

Table 2: Complete list of entity keys.

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

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:

Table 3: Defined roles for entities.

Key
administrator (e.g., of a software system)
artist
assignee (e.g., of a patent)
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
main-author
maintainer (of a software project)
manager (e.g., of a software project)
programmer
reporter (e.g., of a court case/a software bug)
researcher (e.g., authoring a data set/informing a software implementation)
engineer (e.g., for a software)
technical-writer (e.g., of a software documentation)
tester (e.g., of a software)
trainer

Statuses

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

Table 4: 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

Reference types

Table 5: 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	
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	

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:

```
languages:
  - en
  - bpa
```

Programming language strings

CFF knows the following programming language strings. If a language is not included, please use the string `other` with a lower-case, hyphenated string argument, and do not include the version of the programming language used, e.g., for *My Fancy Language v4.2.1*, use `other=my-fancy-language`. Additionally, please create an issue on the GitHub repository for CFF, asking to include the programming language in the list.

Table 6: List of programming language names available in CFF. Table based on the languages available on GitHub (via https://github.com/github/linguist/blob/master/lib/linguist/languages.yml, MIT license, Copyright (c) 2017 GitHub, Inc.).

CFF key	Language name	Language type
1c-enterprise	1C Enterprise	programming
abap	ABAP	programming
abnf	ABNF	data
actionscript	ActionScript	programming
ada	Ada	programming
adobe-font-metrics	Adobe Font Metrics	data
agda	Agda	programming
ags-script	AGS Script	programming
alloy	Alloy	programming
alpine-abuild	Alpine Abuild	programming
ampl	AMPL	programming
ant-build-system	Ant Build System	data
antlr	ANTLR	programming
apacheconf	ApacheConf	data
apex	Apex	programming
api-blueprint	API Blueprint	markup
apl	APL	programming
apollo-guidance-computer	Apollo Guidance Computer	programming
applescript	AppleScript	programming
arc	Arc	programming
arduino	Arduino	programming
asciidoc	AsciiDoc	prose
asn.1	ASN.1	data
asp	ASP	programming
aspectj	AspectJ	programming
assembly	Assembly	programming
ats	ATS	programming
augeas	Augeas	programming
autohotkey	AutoHotkey	programming
autoit	AutoIt	programming
awk	Awk	programming
ballerina	Ballerina	programming
batchfile	Batchfile	programming
befunge	Befunge	programming
bison	Bison	programming

CFF key	Language name	Language type
bitbake	BitBake	programming
blade	Blade	markup
blitzbasic	BlitzBasic	programming
blitzmax	BlitzMax	programming
bluespec	Bluespec	programming
boo	Boo	programming
brainfuck	Brainfuck	programming
brightscript	Brightscript	programming
bro	Bro	programming
c#	C#	programming
c++	C++	programming
c	C	programming
c-objdump	C-ObjDump	data
c2hs-haskell	C2hs Haskell	programming
cap'n-proto	Cap'n Proto	programming
cartocss	CartoCSS	programming
ceylon	Ceylon	programming
chapel	Chapel	programming
charity	Charity	programming
chuck	ChucK	programming
cirru	Cirru	programming
clarion	Clarion	programming
clean	Clean	programming
click	Click	programming
clips	CLIPS	programming
clojure	Clojure	programming
closure-templates	Closure Templates	markup
cmake	CMake	programming
cobol	COBOL	programming
coffeescript	CoffeeScript	programming
coldfusion	ColdFusion	programming
coldfusion-cfc	ColdFusion CFC	programming
collada	COLLADA	data
common-lisp	Common Lisp	programming
component-pascal	Component Pascal	programming
cool	Cool	programming
coq	Coq	programming
cpp-objdump	Cpp-ObjDump	data
creole	Creole	prose
crystal	Crystal	programming
cson	CSON	data
csound	Csound	programming
csound-document	Csound Document	programming
csound-score	Csound Score	programming
css	CSS	markup
csv	CSV	data
cuda	Cuda	programming
cweb	CWeb	programming
cycript	Cycript	programming
cython	Cython	programming
d	D	programming
d-objdump	D-ObjDump	data
darcs-patch	Darcs Patch	data
dart	Dart	programming
dataweave	DataWeave	programming

CFF key	Language name	Language type
desktop	desktop	data
diff	Diff	data
digital-command-language	DIGITAL Command Language	programming
dm	DM	programming
dns-zone	DNS Zone	data
dockerfile	Dockerfile	data
dogescript	Dogescript	programming
dtrace	DTrace	programming
dylan	Dylan	programming
e	E	programming
eagle	Eagle	data
easybuild	Easybuild	data
ebnf	EBNF	data
ec	eC	programming
ecere-projects	Ecere Projects	data
ecl	ECL	programming
eclipse	ECLiPSe	programming
edn	edn	data
eiffel	Eiffel	programming
ejs	EJS	markup
elixir	Elixir	programming
elm	Elm	programming
emacs-lisp	Emacs Lisp	programming
emberscript	EmberScript	programming
eq	EQ	programming
erlang	Erlang	programming
f#	F#	programming
factor	Factor	programming
fancy	Fancy	programming
fantom	Fantom	programming
filebench-wml	Filebench WML	programming
filterscript	Filterscript	programming
fish	fish	programming
flux	FLUX	programming
formatted	Formatted	data
forth	Forth	programming
fortran	Fortran	programming
freemarker	FreeMarker	programming
frege	Frege	programming
g-code	G-code	data
game-maker-language	Game Maker Language	programming
gams	GAMS	programming
gap	GAP	programming
gcc-machine-description	GCC Machine Description	programming
gdb	GDB	programming
gdscript	GDScript	programming
genie	Genie	programming
genshi	Genshi	programming
gentoo-ebuild	Gentoo Ebuild	programming
gentoo-eclass	Gentoo Eclass	programming
gerber-image	Gerber Image	data
gettext-catalog	Gettext Catalog	prose
gherkin	Gherkin	programming
glsl	GLSL	programming
glyph	Glyph	programming

CFF key	Language name	Language type
gn	GN	data
gnuplot	Gnuplot	programming
go	Go	programming
golo	Golo	programming
gosu	Gosu	programming
grace	Grace	programming
gradle	Gradle	data
grammatical-framework	Grammatical Framework	programming
graph-modeling-language	Graph Modeling Language	data
graphql	GraphQL	data
graphviz-(dot)	Graphviz (DOT)	data
groovy	Groovy	programming
groovy-server-pages	Groovy Server Pages	programming
hack	Hack	programming
haml	Haml	markup
handlebars	Handlebars	markup
harbour	Harbour	programming
haskell	Haskell	programming
haxe	Haxe	programming
hcl	HCL	programming
hlsl	HLSL	programming
html+django	HTML+Django	markup
html+ecr	HTML+ECR	markup
html+eex	HTML+EEX	markup
html+erb	HTML+ERB	markup
html+php	HTML+PHP	markup
html	HTML	markup
http	HTTP	data
hy	Hy	programming
hyphy	HyPhy	programming
idl	IDL	programming
idris	Idris	programming
igor-pro	IGOR Pro	programming
inform-7	Inform 7	programming
ini	INI	data
inno-setup	Inno Setup	programming
io	Io	programming
ioke	Ioke	programming
irc-log	IRC log	data
isabelle	Isabelle	programming
isabelle-root	Isabelle ROOT	programming
j	J	programming
jasmin	Jasmin	programming
java	Java	programming
java-server-pages	Java Server Pages	programming
javascript	JavaScript	programming
jflex	JFlex	programming
jison	Jison	programming
jison-lex	Jison Lex	programming
jolie	Jolie	programming
json	JSON	data
json5	JSON5	data
jsoniq	JSONiq	programming
jsonld	JSONLD	data
jsx	JSX	programming

CFF key	Language name	Language type
julia	Julia	programming
jupyter-notebook	Jupyter Notebook	markup
kicad-layout	KiCad Layout	data
kicad-legacy-layout	KiCad Legacy Layout	data
kicad-schematic	KiCad Schematic	data
kit	Kit	markup
kotlin	Kotlin	programming
krl	KRL	programming
labview	LabVIEW	programming
lasso	Lasso	programming
latte	Latte	markup
lean	Lean	programming
less	Less	markup
lex	Lex	programming
lfe	LFE	programming
lilypond	LilyPond	programming
limbo	Limbo	programming
linker-script	Linker Script	data
linux-kernel-module	Linux Kernel Module	data
liquid	Liquid	markup
literate-agda	Literate Agda	programming
literate-coffeescript	Literate CoffeeScript	programming
literate-haskell	Literate Haskell	programming
livescript	LiveScript	programming
llvm	LLVM	programming
logos	Logos	programming
logtalk	Logtalk	programming
lolcode	LOLCODE	programming
lookml	LookML	programming
loomscript	LoomScript	programming
lsl	LSL	programming
lua	Lua	programming
m	M	programming
m4	M4	programming
m4sugar	M4Sugar	programming
makefile	Makefile	programming
mako	Mako	programming
markdown	Markdown	prose
marko	Marko	markup
mask	Mask	markup
mathematica	Mathematica	programming
matlab	Matlab	programming
maven-pom	Maven POM	data
max	Max	programming
maxscript	MAXScript	programming
mediawiki	MediaWiki	prose
mercury	Mercury	programming
meson	Meson	programming
metal	Metal	programming
minid	MiniD	programming
mirah	Mirah	programming
modelica	Modelica	programming
modula-2	Modula-2	programming
module-management-system	Module Management System	programming
monkey	Monkey	programming

CFF key	Language name	Language type
moocode	Moocode	programming
moonscript	MoonScript	programming
mql4	MQL4	programming
mql5	MQL5	programming
mtml	MTML	markup
muf	MUF	programming
mupad	mupad	programming
myghty	Myghty	programming
ncl	NCL	programming
nearley	Nearley	programming
nemerle	Nemerle	programming
nesc	nesC	programming
netlinx+erb	NetLinx+ERB	programming
netlinx	NetLinx	programming
netlogo	NetLogo	programming
newlisp	NewLisp	programming
nginx	Nginx	data
nim	Nim	programming
ninja	Ninja	data
nit	Nit	programming
nix	Nix	programming
nl	NL	data
nsis	NSIS	programming
nu	Nu	programming
numpy	NumPy	programming
objdump	ObjDump	data
objective-c++	Objective-C++	programming
objective-c	Objective-C	programming
objective-j	Objective-J	programming
ocaml	OCaml	programming
omgrofl	Omgrofl	programming
ooc	ooc	programming
opa	Opa	programming
opal	Opal	programming
opencl	OpenCL	programming
openedge-abl	OpenEdge ABL	programming
openrc-runscript	OpenRC runscript	programming
openscad	OpenSCAD	programming
opentype-feature-file	OpenType Feature File	data
org	Org	prose
other		
ox	Ox	programming
oxygene	Oxygene	programming
oz	Oz	programming
p4	P4	programming
pan	Pan	programming
papyrus	Papyrus	programming
parrot	Parrot	programming
parrot-assembly	Parrot Assembly	programming
parrot-internal-representation	Parrot Internal Representation	programming
pascal	Pascal	programming
pawn	PAWN	programming
pep8	Pep8	programming
perl	Perl	programming
perl-6	Perl 6	programming

CFF key	Language name	Language type
php	PHP	programming
pic	Pic	markup
pickle	Pickle	data
picolisp	PicoLisp	programming
piglatin	PigLatin	programming
pike	Pike	programming
plpgsql	PLpgSQL	programming
plsql	PLSQL	programming
pod	Pod	prose
pogoscript	PogoScript	programming
pony	Pony	programming
postscript	PostScript	markup
pov-ray-sdl	POV-Ray SDL	programming
powerbuilder	PowerBuilder	programming
powershell	PowerShell	programming
processing	Processing	programming
prolog	Prolog	programming
propeller-spin	Propeller Spin	programming
protocol-buffer	Protocol Buffer	data
public-key	Public Key	data
pug	Pug	markup
puppet	Puppet	programming
pure-data	Pure Data	data
purebasic	PureBasic	programming
purescript	PureScript	programming
python	Python	programming
python-console	Python console	programming
python-traceback	Python traceback	data
qmake	QMake	programming
qml	QML	programming
r	R	programming
racket	Racket	programming
ragel	Ragel	programming
raml	RAML	markup
rascal	Rascal	programming
raw-token-data	Raw token data	data
rdoc	RDoc	prose
realbasic	REALbasic	programming
reason	Reason	programming
rebol	Rebol	programming
red	Red	programming
redcode	Redcode	programming
regular-expression	Regular Expression	data
ren’py	Ren’Py	programming
renderscript	RenderScript	programming
restructuredtext	reStructuredText	prose
rexx	REXX	programming
rhtml	RHTML	markup
ring	Ring	programming
rmarkdown	RMarkdown	prose
robotframework	RobotFramework	programming
roff	Roff	markup
rouge	Rouge	programming
rpm-spec	RPM Spec	data
ruby	Ruby	programming

CFF key	Language name	Language type
runoff	RUNOFF	markup
rust	Rust	programming
sage	Sage	programming
saltstack	SaltStack	programming
sas	SAS	programming
sass	Sass	markup
scala	Scala	programming
scaml	Scaml	markup
scheme	Scheme	programming
scilab	Scilab	programming
scss	SCSS	markup
self	Self	programming
shaderlab	ShaderLab	programming
shell	Shell	programming
shellsession	ShellSession	programming
shen	Shen	programming
slash	Slash	programming
slim	Slim	markup
smali	Smali	programming
smalltalk	Smalltalk	programming
smarty	Smarty	programming
smt	SMT	programming
sourcepawn	SourcePawn	programming
sparql	SPARQL	data
spline-font-database	Spline Font Database	data
sqf	SQF	programming
sql	SQL	data
sqlpl	SQLPL	programming
squirrel	Squirrel	programming
srecode-template	SRecode Template	markup
stan	Stan	programming
standard-ml	Standard ML	programming
stata	Stata	programming
ston	STON	data
stylus	Stylus	markup
sublime-text-config	Sublime Text Config	data
subrip-text	SubRip Text	data
supercollider	SuperCollider	programming
svg	SVG	data
swift	Swift	programming
systemverilog	SystemVerilog	programming
tcl	Tcl	programming
tcsh	Tcsh	programming
tea	Tea	markup
terra	Terra	programming
tex	TeX	markup
text	Text	prose
textile	Textile	prose
thrift	Thrift	programming
ti-program	TI Program	programming
tla	TLA	programming
toml	TOML	data
turing	Turing	programming
turtle	Turtle	data
twig	Twig	markup

CFF key	Language name	Language type
txl	TXL	programming
type-language	Type Language	data
typescript	TypeScript	programming
unified-parallel-c	Unified Parallel C	programming
unity3d-asset	Unity3D Asset	data
unix-assembly	Unix Assembly	programming
uno	Uno	programming
unrealscript	UnrealScript	programming
urweb	UrWeb	programming
vala	Vala	programming
vcl	VCL	programming
verilog	Verilog	programming
vhdl	VHDL	programming
vim-script	Vim script	programming
visual-basic	Visual Basic	programming
volt	Volt	programming
vue	Vue	markup
wavefront-material	Wavefront Material	data
wavefront-object	Wavefront Object	data
web-ontology-language	Web Ontology Language	data
webassembly	WebAssembly	programming
webidl	WebIDL	programming
wisp	wisp	programming
world-of-warcraft-addon-data	World of Warcraft Addon Data	data
x10	X10	programming
xbase	xBase	programming
xc	XC	programming
xcompose	XCompose	data
xml	XML	data
xojo	Xojo	programming
xpages	XPages	data
xpm	XPM	data
xproc	XProc	programming
xquery	XQuery	programming
xs	XS	programming
xslt	XSLT	programming
xtend	Xtend	programming
yacc	Yacc	programming
yaml	YAML	data
yang	YANG	data
zephir	Zephir	programming
zimpl	Zimpl	programming

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” {% cite principles, 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.

```
- 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.

```
- 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
  programming-languages:
    - java
    - python
    - c
    - haskell
    - 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.” {% cite principles, 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.

```
- 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