Guideline to GitHub use in the eArchiving Building Block

A proper front page will be created for the publication occurring after implementation of review comments.

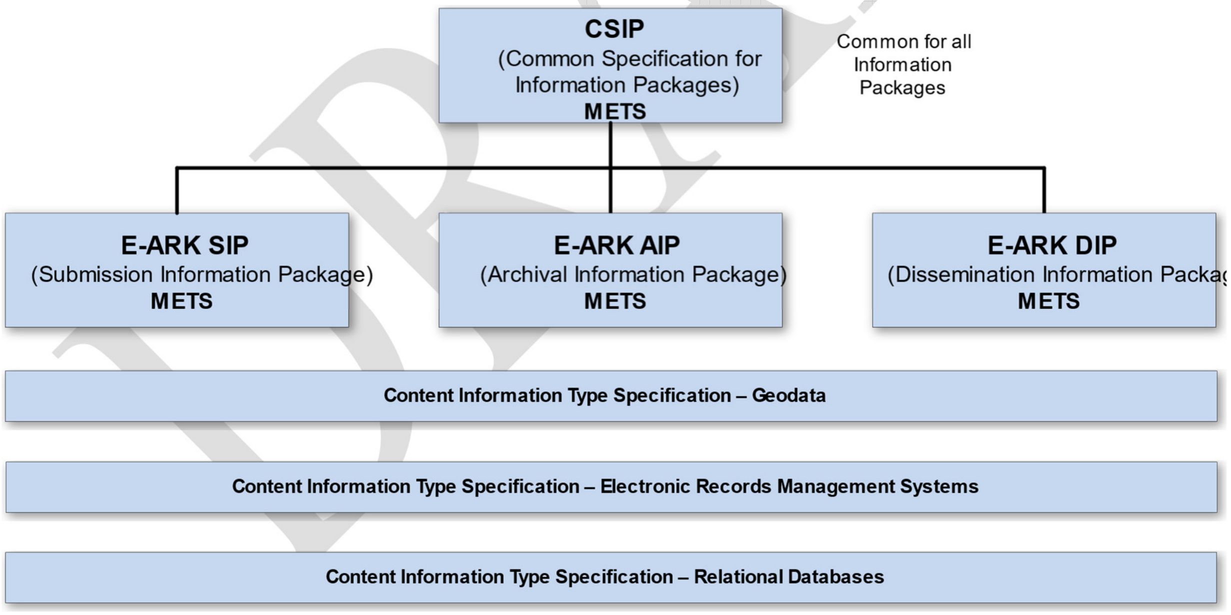
1 Preface

The correct preface will be inserted for the publication occurring after implementation of review comments.

1.1 Aim of the specification

This E-ARK specification is part of a family of specifications that provide a common set of requirements for packaging digital information. These specifications are based on common, international standards for transmitting, describing and preserving digital data. They have been produced to help data creators, software developers and digital archives tackle the challenge of short-, medium- and long-term data management and reuse in a sustainable, authentic, cost-efficient, manageable and interoperable way.

The foundation for these specifications is the Reference Model for an Open Archival Information System (OAIS) which has Information Packages at its core. Familiarity with the core functional entities of OAIS is a prerequisite for understanding the specifications. A visualisation of the current specification network can be seen here:


CSIP
(Common Specification for Information Packages)
METS
Common for all Information Packages
E-ARK SIP
(Submission Information Package)
METS
E-ARK AIP
(Archival Information Package)
METS
E-ARK DIP
(Dissemination Information Packag
METS
Content Information Type Specification – Geodata
Content Information Type Specification – Electronic Records Management Systems
Content Information Type Specification – Relational Databases

The E-ARK specification dependency hierarchy

Specification	Aim and Goals
Common Specification for	This document introduces the concept of a Common Specification for Information Packages (CSIP). Its three main purposes are to:

Specification	Aim and Goals
Information Packages	• Establish a common understanding of the requirements which need to be met in order to achieve interoperability of Information Packages. • Establish a common base for the development of more specific Information Package definitions and tools within the digital preservation community. • Propose the details of an XML-based implementation of the requirements using, to the largest possible extent, standards which are widely used in international digital preservation. Ultimately the goal of the Common Specification is to reach a level of interoperability between all Information Packages so that tools implementing the Common Specification can be adopted by institutions without the need for further modifications or adaptations.
E-ARK SIP	The main aims of this specification are to: • Define a general structure for a Submission Information Package format suitable for a wide variety of archival scenarios, e.g. document and image collections, databases or geographical data. • Enhance interoperability between Producers and Archives. • Recommend best practices regarding metadata, content and structure of Submission Information Packages.
E-ARK AIP	The main aims of this specification are to: • Define a generic structure of the AIP format suitable for a wide variety of data types, such as document and image collections, archival records, databases or geographical data. • Recommend a set of metadata related to the structural and the preservation aspects of the AIP as implemented by the reference implementation eArchiving ToolBox (formerly earkweb). • Ensure the format is suitable to store large quantities of data.
E-ARK DIP	The main aims of this specification are to: • Define a generic structure of the DIP format suitable for a wide variety of archival records, such as document and image collections, databases or geographical data. • Recommend a set of metadata related to the structural and access aspects of the DIP.
Content Information Type Specifications	The main aim and goal of a Content Information Type Specification is to: • Define, in technical terms, how data and metadata must be

Specification	Aim and Goals
	formatted and placed within a CSIP Information Package in order to achieve interoperability in exchanging specific Content Information. The number of possible Content Information Type Specifications is unlimited.

1.2 Organisational support

This specification is maintained by the Digital Information LifeCycle Interoperability Standards Board (DILCIS Board). The DILCIS Board (http://dilcis.eu/) was created to enhance and maintain the draft specifications developed in the European Archival Records and Knowledge Preservation Project (E-ARK project) which concluded in January 2017 (http://eark-project.com/). The Board consists of eight members, but there is no restriction on the number of participants in the work. All Board documents and specifications are stored in GitHub (https://github.com/DILCISBoard) while published versions are made available on the Board webpage. Since 2018 the DILCIS Board has been responsible for the core specifications in the Connecting Europe Facility eArchiving Building Block (https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eArchiving).

1.3 Authors

A full list of contributors to this specification, as well as the revision history can be found in Appendix 1.

TABLE OF CONTENT

LIST OF FIGURES

1 Context

The document will be extended for the publication occurring after implementation of review comments.

1.1 Purpose

The purpose of this guideline is to further explain and describe the use of GitHub with the specifications within the DILCIS Board and the eArchiving Building Block.

1.2 Scope

This Guideline will describe the main concepts and ways of interacting with the specifications through the use of GitHub.

2 Getting started

A primer on Git concepts and workflow is given below. If you are comfortable with Git and GitHub you can move directly to the section about contributing. This document provides an introduction for those unfamiliar with Git and GitHub's support for branching and versioning. It uses the development done by the DILCIS Board of the Common Specification for Information Packages for illustration purposes.

3 GitHub tools and references

This is intended as a guide to GitHub. In addition, there are many useful online resources (e.g. https://guides.github.com/).

4 Git/GitHub concepts

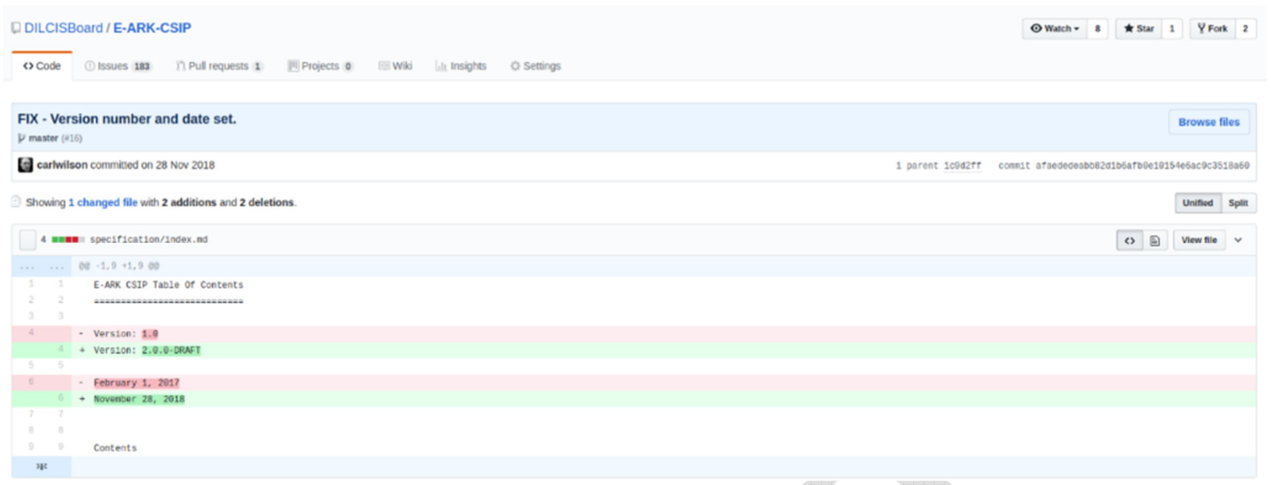
There are four Git concepts that need to be understood. These are: commits, tags, branches/branching, and pull requests.

4.1 Commits

Commits are checked in units of work on a document. This is a simple model where an author makes changes to a file or files in the repository and then checks them into the repository. This check-in is a commit, and it records:

- the email and id of the author who made the change with the time and date of the check-in;
- a comment by the commit author describing the changes;
- a record of the changes made;
- a unique SHA1 identifier for the check-in itself.

Here is an illustration of a small commit to the CSIP repository, it simply changes the version number and publication date:



FIX - Version number and date set.

master (#16)

carlwilson committed on 28 Nov 2018

1 parent 1c9d2ff   commit afaededeabb82d1b6afb0e10154e6ac9c3518a60

Showing 1 changed file with 2 additions and 2 deletions.

```
4 specification/index.md
@@ -1,9 +1,9 @@
  E-ARK CSIP Table Of Contents
  ============================

- Version: 1.0
+ Version: 2.0.0-DRAFT

- February 1, 2017
+ November 28, 2018


  Contents
```

Figure 1: A commit to the CSIP repository

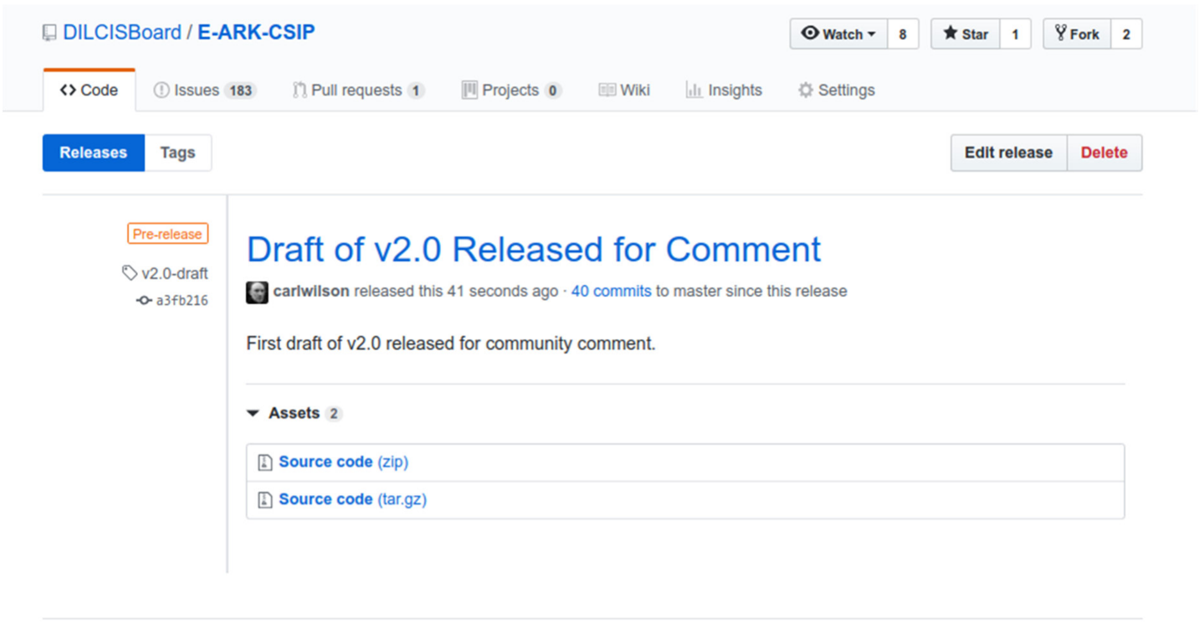
The original can be found at https://github.com/DILCISBoard/E-ARK-CSIP/commit/afaededeabb82d1b6afb0e10154e6ac9c3518a60 the long last part of the URL is the SHA1 id of the commit itself. Work is built up as a chain of commits. Ideally, an individual commit should be small as it makes tracking changes and rolling back work easier. The name commit is derived from the act of committing a change to the permanent record by checking the work into a repository.

4.2 Tags

A tag (https://git-scm.com/book/en/v2/Git-Basics-Tagging) is effectively a bookmark to an individual commit and is used to record a significant state in the project repository. Released versions are given a tag, and typical tag names tend to reflect this (e.g. v2.0 for the CSIP). There are two types of tags:

- Lightweight tags are used by authors to bookmark particular states of work.
- Official release tags tend to be annotated tags store more information (e.g. author, comment, dates etc. to inform users that the state is an approved version).

The v2.0 draft of the CSIP is shown in the figure below.


DILCISBoard / E-ARK-CSIP

Pre-release
v2.0-draft
a3fb216

Draft of v2.0 Released for Comment

carlwilson released this 41 seconds ago · 40 commits to master since this release

First draft of v2.0 released for community comment.

Figure 2: The bookmark for CSIP release v2.0

4.3 Branches and versioning

Git branches (https://git-scm.com/book/en/v1/Git-Branching-What-a-Branch-Is) are a way of organising different strands of work within a repository. It is the use of branches that allow multiple authors to work on a single document simultaneously. In effect, branches are nothing more than mobile tags that record a particular state of work. To illustrate this, consider the current work on the CSIP which might be organised into the following branches:

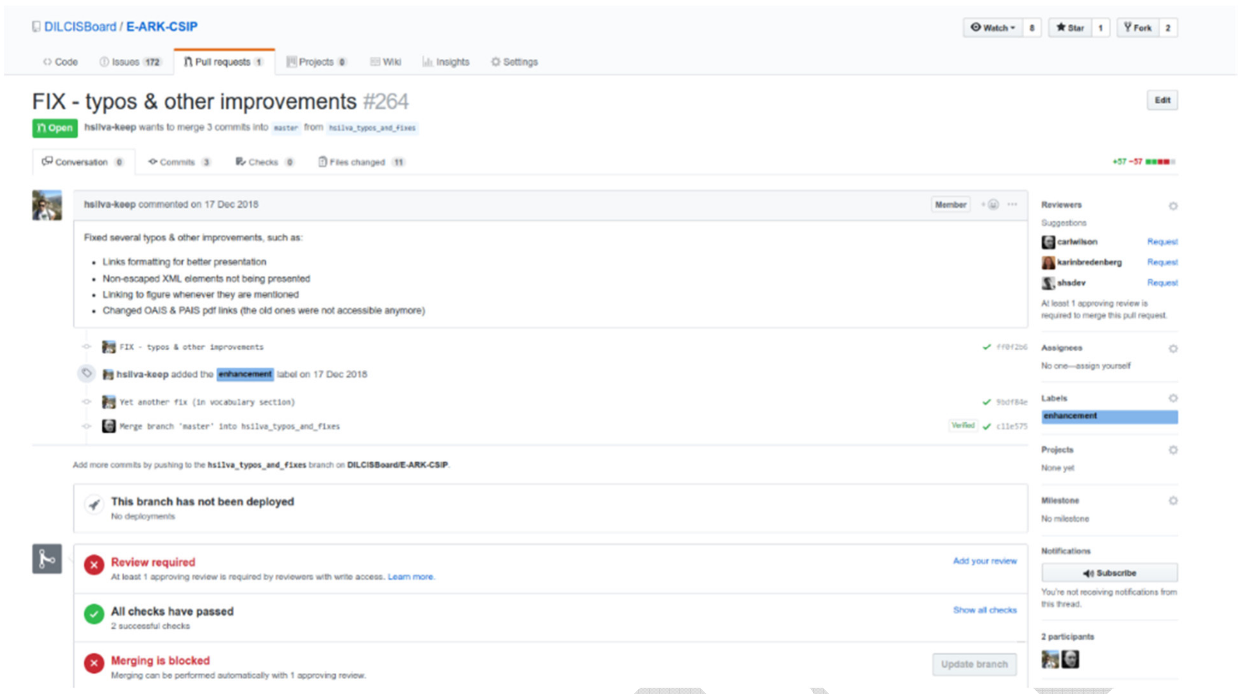
- Master is the latest officially released version of the specification. This is the designated main branch for the repository making it the default branch viewed on GitHub. The information in this branch is authoritative because it has been reviewed and approved by the DILCIS Board. If you are looking for a version of the specification to use in your organisation or are unsure as to which branch to use it is advisable to start here.
- Integration holds the work that has been done on the upcoming version of the specification. While the work here has been reviewed by the DILCIS Board it may be subject to change as the new specification version evolves. If you are interested in reviewing or contributing to future versions of the specification you should start here.

Each repository also contains specific release branches named rel/<version-no>. These are used to:

- allow the board and users to easily view previous versions of the specification; and
- make editorial changes to a specific version, usually correcting spelling issues or fixing typos.

4.4 Pull Requests (PR)

Pull requests (https://help.github.com/en/articles/about-pull-requests) are GitHub's mechanism for sharing and organising work done in other branches. Once a pull request is open its contents can be reviewed by other project members and follow-on commits can be added if necessary. A pull request can be initiated by comparing any two branches for changes. Once a pull request is made it usually, depending on the repository policy, requires a review before it can be merged. The figure below shows a PR made against the CSIP repository that awaits review.



FIX - typos & other improvements #264

Open hsilva-keep wants to merge 3 commits into `master` from `hsilva_typos_and_fixes`

hsilva-keep commented on 17 Dec 2018

Fixed several typos & other improvements, such as:

- Links formatting for better presentation
- Non-escaped XML elements not being presented
- Linking to figure whenever they are mentioned
- Changed OAIS & PAIS pdf links (the old ones were not accessible anymore)

```
FIX - typos & other improvements                               ✓ ff0f2b6
```
hsilva-keep added the enhancement label on 17 Dec 2018
```
Yet another fix (in vocabulary section)                        ✓ 9bdf84e
Merge branch 'master' into hsilva_typos_and_fixes              ✓ c11e575
```

Add more commits by pushing to the **hsilva_typos_and_fixes** branch on **DILCISBoard/E-ARK-CSIP**.

This branch has not been deployed
No deployments

Review required
At least 1 approving review is required by reviewers with write access. Learn more.

All checks have passed
2 successful checks

Merging is blocked
Merging can be performed automatically with 1 approving review.

At least 1 approving review is required to merge this pull request.

No one—assign yourself

None yet

No milestone

You're not receiving notifications from this thread.

Figure 3: A PR awaiting review

GitHub provides a pull request view for each project. Here are the open pull requests for the Common Specification project: https://github.com/DILCISBoard/E-ARK-CSIP/pulls. This view shows pull requests that have been closed: https://github.com/DILCISBoard/E-ARK-CSIP/pulls?q=is%3Apr+is%3Aclosed.

5 The GitHub workflow

The different repositories in Git and GitHub management follow this branching principle.

- master which is the current released state of a repository project;
- rel/v2.0 where ongoing corrections to the v2.0 draft are made before publication via master;
- feat/segmented_ips (or other name depending on what the feature is handling) where development of the feature documentation can continue for release in a future version; and
- integration which is the current working copy of the repository project to ensure that different strands of work can be merged together before publication.

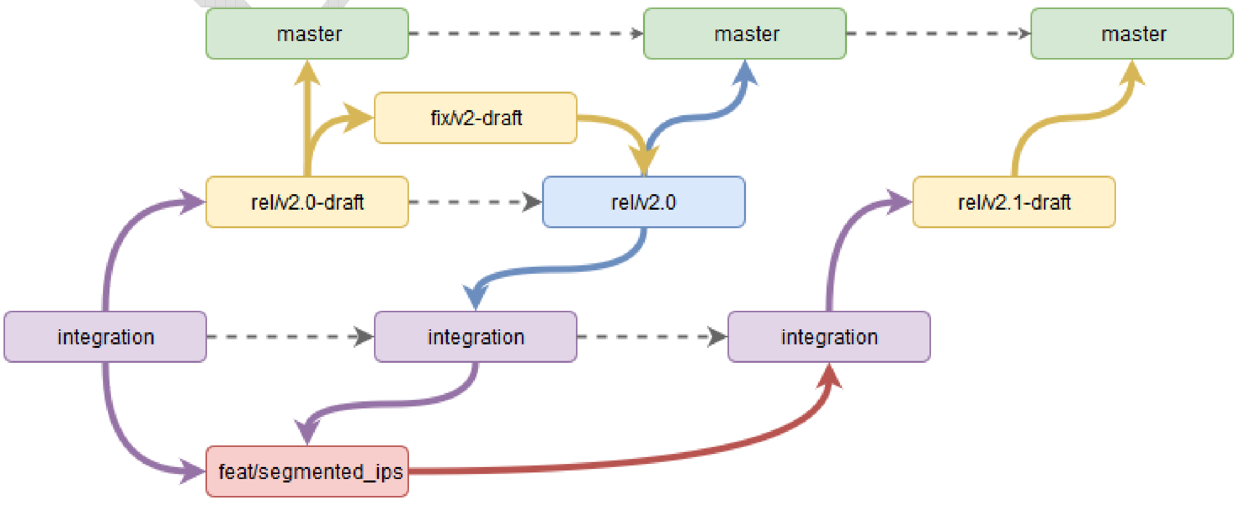


Figure 4: The GitHub workflow

In Figure 4 it can be seen that integration (in purple) and master (in green) are the two consistent branches. Master always shows the latest official release and is updated from release branches, not from integration. Following the orange boxes for work on draft releases, it can be seen that the draft release rel/v2.0-draft is created from the integration branch. The official draft release is also pushed to master. Revisions to the draft are made in the fix/v2-draft which is merged with the rel/v2.0-draft branch to create the corrected rel/v2.0 branch. In reality, this work would take place in several branches as separate strands of work. The diagram excludes these for clarity.

Once version 2.0 is ready in rel/v2.0, it can be merged to master and to integration because the content also needs to be in the working version. At the same time work can continue in the red feat/segmented_ips branch. The author must merge the latest work from integration here also so that other work (e.g. typos fixed in v2.0 are retained). Once work has finished on the segmented IP branch, it can be merged to master for publication in a future specification version (e.g. v2.1).

6 Contributing to the specification process

We use GitHub (https://github.com/) to organise, assign and review the work with the specifications. This means that to participate, you will need to sign up to GitHub and get a GitHub id (https://github.com/join).

6.1 Writing on GitHub

Most of your interaction with GitHub itself will be through its online forms, which are generally simple. Most forms provide a free text box for comments and/or a description. While plain text is fine, it lacks even rudimentary formatting. Most places where you can enter free text on GitHub support Markdown (https://daringfireball.net/projects/markdown/), specifically GitHub flavoured Markdown (https://guides.github.com/features/mastering-markdown/).

7 Providing feedback

Feedback can be provided using GitHub issues or as a Pull Request.

7.1 Providing feedback using GitHub issues

The simplest way of contributing is to provide feedback on our work using GitHub Issues (https://guides.github.com/features/issues/). Each project has its own issue tracker which can be found using the Issues tab on the GitHub project page or appending/issues to the projects GitHub URL. As an example, the GitHub issues page for CSIP is available here: https://github.com/DILCISBoard/E-ARK-CSIP/issues. The GitHub instructions for creating an issue can be found here (https://help.github.com/en/articles/creating-an-issue).

7.2 Providing a Contribution as a Pull Request

The branch arrangement allows changes to be made to a specific version of a project. These will then be reviewed by a member of the DILCIS Board before publication. Changes are submitted as GitHub pull requests (https://help.github.com/en/articles/about-pull-requests). A pull request is simply a set of edits made to code. Each pull request is made against a specific branch. It can be tricky working out which branch to use when submitting a pull request. Here are a few guidelines:

- pull requests should rarely if ever, be made against the master branch. This is the officially released branch and does not contain the latest changes;
- pull requests should usually be created using the branch that represents the version you wish to amend, for instance, changes to version 2.0 of the APP should be made against the rel/2.0 branch; and
- if you submit a pull request against the wrong branch do not worry, there is no harm done and a member of the review team will get in touch and help.

7.2.1 Pull Request Style

When providing contributions as pull requests remember that small is beautiful. If you are planning to provide a number of changes please break them down into multiple pull requests rather than bundle them into a single contribution. This is because:

- small pull requests are easier to review;
- review responsibilities can be split across multiple contributions; and
- if there is a review issue with a single part of a large pull request, then none of the work contributed can be merged.

Be sure to provide a detailed description of the changes you are submitting using the pull request form. This makes it easier for reviewers to evaluate your contribution.

7.2.2 Cloning a Repository

In order to create a pull request, it is usually best to create your own copy of the repository on GitHub (https://help.github.com/en/articles/cloning-a-repository), known as a fork or a clone. You will need a GitHub account to fork a repository. If you have one you simply need to press the fork button shown in the top right of the repository's home screen:

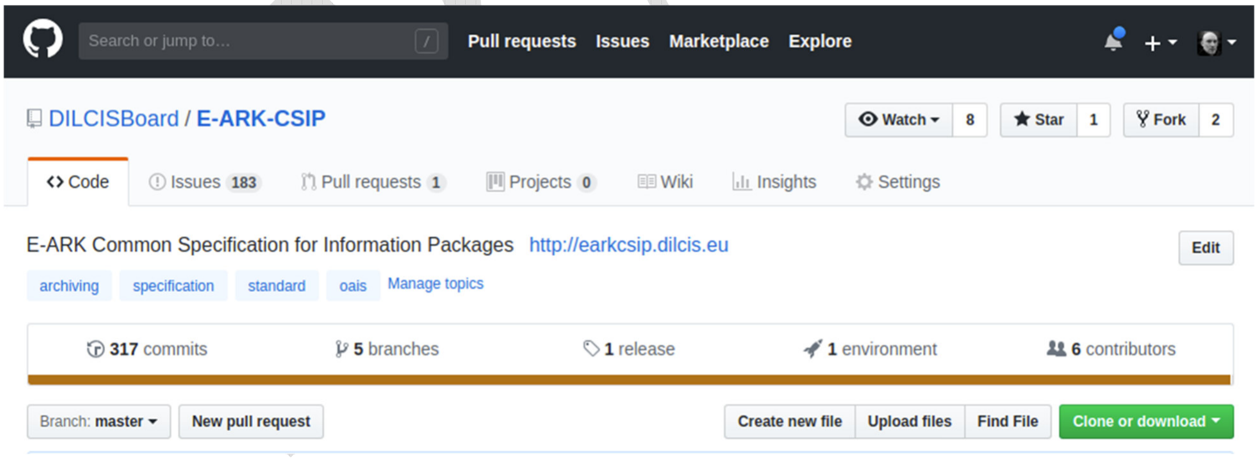


Figure 5: Buttons in GitHub

You can then make the changes you want, either on a working copy of the repository on your own machine, or using GitHub's web editor (https://help.github.com/en/articles/editing-files-in-your-repository). Once your changes are ready, you can submit them as a pull request (https://help.github.com/en/articles/creating-a-pull-request), be sure to select the appropriate branch. Again if you make a mistake, someone will get in touch to help.

7.2.3 Creating a new Branch

Fill these parts. Keep all for now, there might be parts removed.

To create a new branch on GitHub follo

1. From the repo home page ensure that the branch you wish to copy, in this case, master is selected.
2. Hit the pull down button and type the new branch name. In the image below, we are creating the rel/2.0-draft branch.
3. Click the "Create branch: rel/2.0-draft" panel. The name will be that of the branch you are creating. Check the "from 'master'" tag to ensure you are cloning the branch you intend, in this case master.
4. Do your work in this new branch!

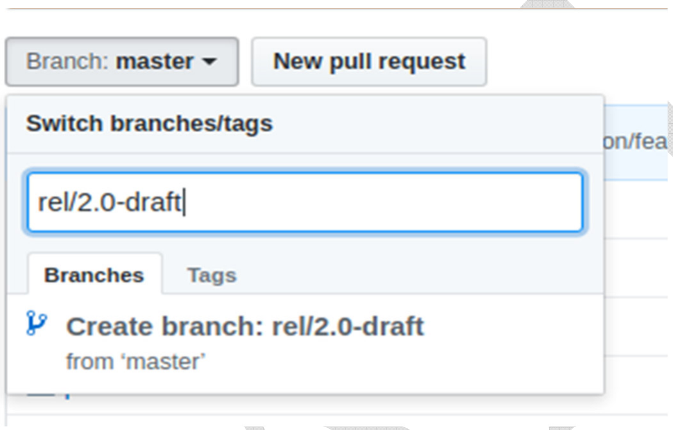


Figure 6: Naming a branch in GitHub

8 Postface

AUTHOR(S)	
Name(s)	Organisation(s)
Carl Wilson	OPF
Karin Bredenberg	Kommunalförbundet Sydarkivera
Jaime Kaminski	Highbury R&D Ltd

REVIEWER(S)	
Name(s)	Organisation(s)

[Name]	[Affiliation]
[Name]	[Affiliation]
[Name]	[Affiliation]

Project co-funded by the European Commission within the ICT Policy Support Programme **Dissemination Level**		
P	**Public**	x
C	**Confidential, only for members of the Consortium and the Commission Services**	

REVISION HISTORY AND STATEMENT OF ORIGINALITY

Submitted Revisions History

Revision No.	Date	Authors(s)	Organisation	Description
[Version]	[Date]	[Who]	[Affiliation]	[What]

Statement of originality:
This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.