

Justin Gosses
Houston, Texas

Office of The Chief Scientist
hq-publicaccess@mail.nasa.gov
NASA Headquarters
300 E Street SW
Washington DC 20024-3210

Response to Request for Information Regarding NASA 2023 Proposed Public Access Plan

This document has been written in response to the [request for information](#) regarding the [May 2023 NASA Public Access Plan](#). While these are my personal opinions, I should note that I previously worked as a NASA contractor supporting code.nasa.gov and currently work for Microsoft, which owns GitHub. My comments are in response to question 5 : *“Suggestions on sharing and archiving of software. Sites like GitHub and Zenodo offer ways to distribute and manage software. NASA is seeking suggestions on improving the archiving, sharing, and maintenance of software for reuse.”*

Plan should more explicitly define software archival versus maintenance

In several places the proposed NASA Public Access Plan would benefit from more explicitly describing the differences between archival of a fixed version of the software versus maintaining an active and changing public code repository over time. For example, in the line below from page 20 discussing requirements that Software Management Plans must address, “maintained” and “preserved” could be mistakenly read as the same thing.

```
Plans for archiving and preserving of the software, as appropriate (use of existing databases or public repositories will be strongly encouraged), including how long the software will be preserved or maintained
```

Archived software is an unchanging and fixed version. As stated in the plan, an archived version of the software is important for reproducibility. Without an archived version of the software, it is difficult for others to evaluate the validity of a study that depends on software. As such, it is reasonable to have code archival as a requirement. However, archived code should be described as the minimum and not implied to be the only or best way to share software.

NASA should also encourage, although likely not require, use of free, public facing version control systems (GitHub, GitLab, etc.) that allow others besides the original authors to easily access software in a manner such that the software can evolve over time. As NASA likely does not want to mention any particular product in policy, the term “free, public-facing software version control systems” could be used instead.

Making the code available to the public in these systems enables several behaviors that increase the value and impact of the software not possible with an archived version of the software alone. Placing it on these types of systems enables users to discover it from other public code that use it as a dependency or share a topic tag. More people are likely to discover it there as “free, public-facing software version control systems” are where developers spend a lot of their time, not NASA run systems that fewer know about such as [NASA Technical Report Server](#) or [code.nasa.gov](#). As these systems also allow the public to read software without downloading it, the software can be more quickly evaluated without having to download and open large files. This reduces friction and increases rates of reuse. Code that exists on a “free, public-facing software version control systems” is also more likely to improve over time. Users who are not the original authors can add an issue when a bug or security vulnerability is found or make pull requests to add new features. Code for one study can be generalized with the insights of the community and turned into a tool that is reusable. To summarize, software on “free, public-facing software version control systems” is more discoverable, reusable, secure, generalizable, and extendable. In contrast, software that only exists in archived form or as supplemental information to a publication is much less likely to be a part of these behaviors as it is by definition a static artifact. Suggest changing the previously mentioned text on page 20 to:

```
- Plans for preserving a fixed, archival version of the software that was used to produced the results in the publication, as appropriate (use of existing databases or archival systems will be strongly encouraged).  
- State whether the software will optionally also exist on a free, public-facing software version control platform where the software can evolve over time as the community reports bugs, submits new features, etc.  
- For software that has a community of users beyond the context of the study being funded, discuss plans for community health and governance.
```

Note that the changed words have been highlighted in yellow but the original text is above.

Please note that the term “repository” has been removed from the original text. The term “code repository” is commonly used in modern software development to mean a single project collection of code files. However, in the Public Access Plan it is referred to in a way that means a data system to hold static versions of many thousands of software projects, datasets, etc. To avoid confusion, suggest the term “repository” either be not used or be defined within the plan. The modern software development definition of the word is used in this document. To further clarify these points regarding archival, the following bullet point under requirements should be modified from:

```
All proposals or project plans submitted to NASA for scientific research funding will be required to include a Software Management Plan (SMP) that describes whether and how software generated through the course of the proposed research will be shared and preserved (including timeframe), or explains why software sharing and/or preservation are not possible or scientifically appropriate. At a minimum, SMPs must describe how software sharing and preservation will enable validation of published results, or how such results could be validated if software are not shared or preserved.
```

Into a version that separates archived software from non-archived software:

All proposals or project plans submitted to NASA for scientific research funding will be required to include a Software Management Plan (SMP) that describes whether and how software generated through the course of the proposed research will be shared and preserved (including timeframe), or explains why software sharing and/or preservation are not possible or scientifically appropriate. At a minimum, SMPs must describe how software preservation will enable validation of published results, or how such results could be validated if software are not shared or preserved. SMPs may optionally describe where a version of the software that will change and evolve post publication will be publicly accessed in addition to the fixed software archive used in the publication.

Additionally, on page 20 under “implementation” sub-heading there is the following statement:

Require all researchers to share their scientific software developed to support a scholarly publication at the time of publication. This includes the scientific software that are displayed in charts and figures or needed to validate the scientific conclusions of the publication. This requirement could be met by including the software as supplementary information to the published article, or through other means. The published article should indicate how the software can be accessed.

This statement is very similar to the fifth bullet point on page 19 under the “requirements” sub-heading. The statement puts too much emphasis on releasing code as supplementary information files where it is least likely to be discovered, accessed, reused, bug fixes submitted, etc. To maximize value of funded software creation, suggest changing both bullet points on to:

Require all researchers to share their scientific software developed to support a scholarly publication at the time of publication or prior. This includes the scientific software that are displayed in charts and figures or needed to validate the scientific conclusions of the publication. This requirement could be met by including the software as supplementary information to the published article, linking to archived versions of the software on a NASA-recognized archive service (such as NTRS or Zenodo), or through other means. The publication should indicate how the software can be accessed. Stating “upon request” is not a valid means of public release. If the software is available not just as an archive but also as a publicly accessible software repository that changes over time, both humans and machines should be able to discover it from the publication.

It is important that going from a publication to both the fixed, archived version of the software and the live, updating, or changing version of the software repository is as frictionless for users as possible to maximize the value of NASA’s funded software creation.

Educational Needs: Evolving Software, Archival Software, and DOIs

There is a strong need to educate scientific software developers on the benefits and methods of both archives and having a live software repository on a public-facing version control system

that can reflect changes over time. When asked how to archive both requirements several times in the past, I have recommended using both [Zenodo](#) and [GitHub](#). However, any options that enable the same behaviors, software evolution other time and DOIs tagged to fixed releases of said software, would be sufficient. Zenodo is foremost an archive service, but it has integrations with Github that makes it easy to package a fixed version of a software repository into a GitHub Release and then turning that release artifact into an archived version on Zenodo with a unique DOI. The process is relatively simple but the instructions to do so on Zenodo and GitHub are easy to miss and at times confusing as evidenced by the numerous [attempts to explain it elsewhere](#). NASA has a role to play in making more scientific software developers aware of these processes and establishing them as at least best practice if not required. There are a few minor changes that could be made to the public access plan to clarify when creating a software archive with a DOI has value. On page 20, it could be interpreted only software not related to a paper needs a DOI.

```
Software released independently from the peer-reviewed manuscript must be assigned a Unique Digital Object Identifier (DOI) to enable preservation, discovery, and citation of the software.
```

This wording needs modification as (1) people may hesitate to cite a paper DOI if the software is only a small part of the paper. (2) people may need to cite a different version of software than was released at publication. The plan should instead add the additional text as shown below:

```
Software released independently from the peer-reviewed manuscript must be assigned a Unique Digital Object Identifier (DOI) to enable preservation, discovery, and citation of the software. Software released at the same time as a publication can optionally have its own DOI. DOIs may optionally be created for any additional versions of the software, referred to as releases.
```

GitHub or GitLab releases are fixed versions of software, so they meet some of the needs for archival, but the fact that they do not automatically get a DOI limits their use as archives without linking them to Zenodo, which gives them a DOI.

Educational Needs: Code Citation

Software is sometimes treated as less valuable than publications in academia; partly as papers get cited whereas code is rarely cited. NASA should encourage the use of [citation files](#) in code repositories such as [CITATION.cff](#) and push for community standards around code citation to enable more effective programmatic tracking of code and publication usage between different systems. As an example, [Zenodo consumes CITATION.cff files](#) found in linked GitHub repositories. Suggest adding two bullet points to the list of requirements on page 18 and 19

```
- NASA employees, contractors, and grantees are encouraged to use community standard citation files in their software projects.  
- No matter where their code, data, and publications are stored NASA employees, contractors, and grantees are encouraged to fill out system metadata to ensure maximum linkages between them.
```