PHEP: 1
Title: PHEP Purpose and Guidelines
Author: Jonathan T. Niehof <jtniehof@gmail.com> <https://orcid.org/0000-0001-6286-5809>
Discussions-To: https://github.com/heliophysicsPy/standards/pull/22
Revision: 1
Status: Final
Type: Process
Content-Type: text/markdown; charset=UTF-8; variant=CommonMark
Created: 20-Oct-2023
Post-History: 24-Oct-2023, 31-Oct-2023, 09-Nov-2023, 27-Nov-2023, 06-Dec-2023, 14-Dec-2023,
07-Feb-2024, 28-Feb-2024, 13-Mar-2024, 20-May-2024
Resolution: https://docs.google.com/document/d/1htbibknl1Npv1hQ1O4L22RKO_s8uc9dZT1iTJ49nMJg/edit
#heading=h.a0b498f5euue,
https://docs.google.com/document/d/1RNGKHSf1S562WmDhCdYAAy3wG2tl6LT3-KBFYmRXTnA/edit

What is a PHEP?

PHEP stands for PyHC Enhancement Proposal. A PHEP provides information to the Python in Heliophysics Community (PyHC), describes its processes, or establishes standards for its packages.

PHEPs are the primary mechanism for proposing standards, collecting community input, and documenting the design decisions underlying PyHC standards and recommendations. The PHEP process develops and documents consensus within the community. The PHEP author is responsible for building this consensus and documenting dissenting opinions.

Because PHEPs are maintained as text files in a versioned repository, their commit history is the historical record of the feature proposal. This record is available by the normal git commands for retrieving older commits and can be browsed on GitHub.

PHEP Motivation

The PyHC standards have not been updated since their initial approval in 2018, and there is no established process for updating them. PyHC discussions often involve what packages "should" do, or "should be required" to do. For standards to be effective, they need to be seen as legitimate by those they may be imposed on.

A mechanism is needed to capture the sense of the community and allow it to progress while recognizing that the community is broad, diffuse, and diverse, and not all stakeholders are present for all conversations.

PHEP Audience

The typical audience for PHEPs is the developers and user communities of PyHC packages. Other parts of the PyHC community (hereafter "community") may also use the process (particularly for Informational PHEPs) to document conventions and to manage complex design coordination problems that require collaboration across multiple projects.

This PHEP refers to "PyHC leadership": individuals or groups with statutory authority over the PyHC project. This may be the Principal Investigator or other individuals or groups appropriately designated in the future, such as a steering committee.

PHEP Types

There are three types of PHEP:

1. A **Standards** Track PHEP describes a new or modified standard for PyHC packages. These standards are requirements that PyHC packages are expected to follow, similar to the original PyHC standards. Examples include testing requirements, OS support, and dependency handling.
2. An **Informational** PHEP describes a design issue, or provides general guidelines or information to the community, but does not propose a standard. Informational PHEPs do not necessarily represent a community recommendation, so users and implementers are free to ignore Informational PHEPs or follow their advice.
3. A **Process** PHEP describes a process surrounding PyHC, or proposes a change to a process. Process PHEPs are like Standards Track PHEPs but apply to areas other than PyHC packages. Unlike Informational PHEPs, they are more than recommendations, and users are typically not free to ignore them. Examples include procedures, guidelines, and changes to the decision-making process.

PHEP Applicability

Standards Track PHEPs are recommended standards for PyHC packages. Most packages should make a good-faith effort to comply. PyHC core packages are strongly recommended to comply and generally must do so, absent a compelling reason not to. A PHEP which core packages cannot agree to implement likely does not have sufficient community support for acceptance. Standards track PHEP compliance must be considered as a factor in evaluating current and potential PyHC packages.

Informational PHEPs are for the benefit of PyHC developers and users and may represent "best practices". They are not binding.

Process PHEPs determine the process by which the PyHC community makes formal decisions and as such bind all participants in PyHC.

A PHEP may provide further guidance on its applicability and implementation (e.g. timelines for compliance). PHEPs may define the applicability of other PHEPs; these are usually Process PHEPs.

For all three types, key words in PHEPs must be interpreted according to RFC2119; the use of `must`, `must not`, `should`, and `should not` are preferred.

PHEP Workflow

All PHEP workflow is conducted via the GitHub PyHC standards repository.

PHEP Editors

PHEP editors are responsible for managing the administrative and editorial aspects of the PHEP workflow (e.g. assigning PHEP numbers and changing their status). See PHEP Editor Responsibilities & Workflow for details.

PHEP editorship is by nomination of the current editors, subject to approval by PyHC leadership. In the event there are fewer than two active editors, PyHC leadership may appoint editors from active participants in the community, sufficient to bring the total to two.

The editors can be contacted by mentioning `@heliophysicsPy/phep-editors` on GitHub.

Start with an idea

The PHEP process begins with a new idea for PyHC. It is highly recommended that a single PHEP contain a single key proposal or new idea; the more focused the PHEP, the more successful it tends to be. If in doubt, split your PHEP into several well-focused ones.

Each PHEP must have an author: someone who writes the PHEP using the style and format described below, shepherds the discussion, and attempts to build community consensus around the idea. The PHEP author

should first attempt to ascertain whether the idea is PHEP-able. Suggested venues for this discussion include a PyHC telecon, other established PyHC discussion channels including the email list or Matrix/Slack, and an issue in the standards repository (which should be linked to the PR when the PHEP is submitted). The author may choose the approach that seems best for their topic.

Vetting an idea publicly before writing a PHEP is meant to save the author time. Asking the community first helps prevent too much time being spent on something that may be rejected based on prior discussions (searching the Internet does not always do the trick). It also helps to make sure the idea is applicable to the entire community.

Early vetting allows PHEP authors to collect community feedback while avoiding long-winded and open-ended discussions. Strategies such as soliciting private or narrowly-tailored feedback in the early design phase, collaborating with other community members with expertise in the PHEP's subject matter, and picking an appropriately-specialized discussion for the PHEP's topic should be considered.

Once the author has asked the community whether an idea has any chance of acceptance, a draft PHEP should be submitted as described below. This gives the author a chance to flesh out the draft PHEP to make it properly formatted, of high quality, and to address initial concerns about the proposal.

Submitting a PHEP

The proposal must be submitted as a draft PHEP via a GitHub pull request. The draft must be written in PHEP style as described below. The editors may correct minor errors.

The standard PHEP workflow is:

- You, the PHEP author, fork the PyHC standards repository, and create a file named `phep-9999.md` that contains your new PHEP. Use "9999" as your draft PHEP number.
- In the "Type:" header field, enter "Standards Track", "Informational", or "Process" as appropriate, and for the "Status:" field enter "Draft". For full details, see PHEP Header Preamble.
- Push this to a branch (not `main`) of your GitHub fork and submit a draft pull request. Each PR must contain a single PHEP; open multiple PRs (from separate branches) if drafting multiple PHEPs.
- The PHEP editors review your PR for structure, formatting, and other errors. Approval criteria are:
  - It is sound and complete. The ideas must make technical sense. The editors do not consider whether they seem likely to be accepted.
  - It is of reasonable scope, not too broad or unfocused.
  - The title accurately describes the content.
  - The PHEP's language (spelling, grammar, sentence structure, etc.) should be correct and conformant. Markdown (MD) markup and other formatting should be clear and reasonable.

  Editors are generally quite lenient about this initial review, expecting that problems will be corrected by the reviewing process. Correctness is the responsibility of authors and reviewers, not the editors.

  If the PHEP isn't ready for approval, an editor will send it back to the author with specific instructions.
- Once approved, they will assign your PHEP a number. All PHEPs are sequentially-numbered at time of assignment. "Special" numbers (X, 0, etc.) are not used. Upon approval, the author must mark the PHEP PR ready for review.

The PHEP editors will not unreasonably deny approval of a PHEP. Reasons for denying PHEP status include duplication of effort, being technically unsound, not providing proper motivation or addressing backwards compatibility, or so incompatible with PyHC practice as to be infeasible.

Standards Track PHEPs may consist of two parts, a design document and a reference implementation. It is generally recommended that at least a prototype implementation in an existing or new PyHC package be co-developed with the PHEP, as ideas that sound good in principle sometimes turn out to be impractical when implemented.

Discussing a PHEP

Once the draft PHEP is submitted in an open PR and a PHEP number has been assigned, the PR discussion thread provides the central place to discuss and review its contents. The PHEP must be updated so that the `Discussions-To` header links to the PR. The discussion must also be announced by the PHEP authors via reasonable means; this includes (but is not necessarily limited to) the PyHC email list and the PyHC Matrix/Slack.

Informal conversation may happen elsewhere, but the discussion of record is the public thread on the pull request. Substantive input into the PHEP or questions about it belong in the PR comments and side conversations should result in comments on the PR. This ensures everyone can follow and contribute, avoids fragmenting the discussion, and makes sure input is fully considered as part of the PHEP review process. Feedback on the PR thread is a critical part of what the community will consider when reviewing the PHEP. Individual comments or the GitHub "review" process may both be used. Commenters may use the GitHub "approve" feedback to indicate they support the PHEP; however, this does not change the PHEP status to "Final".

PHEP authors should incorporate community feedback by making edits and additional pushes to their branch. They must add a new `Post-History` entry with each push. It is recommended to address as much actionable feedback as practicable with each push; however, each push may contain multiple commits for ease of review. Once a PHEP PR has been marked ready for review, PHEP authors must not rewrite history after a push, i.e. no force-pushing.

If applicable, the reference implementation should be developed and tested as part of the discussion process. If changes are made based on implementation experience and user feedback, they should be noted in the PHEP.

PHEP authors are responsible for managing the progress of the discussion of their PHEP, for instance by updating the PR description to highlight topics of particular interest and to provide the dates of votes on the PHEP resolution.

If a PHEP undergoes a significant re-write or other major changes, the author(s) and editor may discuss closing the existing PR and opening a new one, starting from the changed version. If this occurs, the `Discussions-To` link must be updated, and the new PR discussion should link to the old for reference.

PHEP discussions are subject to the PyHC Code of Conduct.

PHEP Review & Resolution

Once the authors have completed a PHEP, they may request a review for style and consistency from the PHEP editors. However, content review and acceptance of the PHEP is the responsibility of the community. It should be clear from the discussion when the community is beginning to agree: concerns have been addressed and new ones are not being raised, PR reviews with "approve" status are appearing, discussion is quieting down. At this point the PHEP author should seek formal acceptance.

A PHEP is accepted by consensus of attendees (virtual or in-person) at both a PyHC regular telecon and a regular PyHC meeting, in either order, with no edits to the PHEP occurring in between. Potential acceptance of a PHEP must be placed on the agenda at least a month before the meeting, including a link to the open PR with the proposed-final language of the PHEP. The formal announcement of a vote must take place *after* the previous vote, effectively requiring a month between votes. Persons not in attendance at the meetings may object via comment on the PR. The PHEP may be edited up until the initial vote, for instance to incorporate discussion at the meeting. However, any edits after a vote invalidate that acceptance, and two further votes (one at a meeting, one on a telecon) must be taken.

A "regular PyHC meeting" is any meeting (virtual, physical, or hybrid) organized for the purpose of discussing PyHC business, of at least a half-day duration, where a substantial portion of the community is expected to participate, and announced as such well in advance. This includes the semiannual meetings but may also include side gatherings at other meetings which are of interest to the community and designated as PyHC meetings.

"Consensus" here means no concrete objections have been raised: an accepted PHEP must be one that the entire community can at least live with. The PHEP authors must make the case for the PHEP; objectors must provide meaningful concerns that could, in principle, be addressed by the authors. All are expected to discuss in good faith, while recognizing that good faith does not guarantee consensus can be reached. The author of a PHEP should be present for the vote on the PHEP so they can contribute to any discussion and answer questions. If they are not able to make the meeting, they should designate an alternate who can speak effectively for them.

For a PHEP to be accepted it must meet certain minimum criteria. It must be a clear and complete description of the proposed change. The change must represent a net improvement. The proposed implementation, if applicable, must be solid and must not complicate PyHC projects unduly. If applicable, the reference implementation must be completed and incorporated into a PyHC project's source code repository. Finally, a proposed enhancement must be "pythonic" in order to be accepted by the community. ("Pythonic" is an imprecise term; it should be understood in the context of the broad Python ecosystem.)

Once a PHEP has been accepted, its status will be changed to "Final".

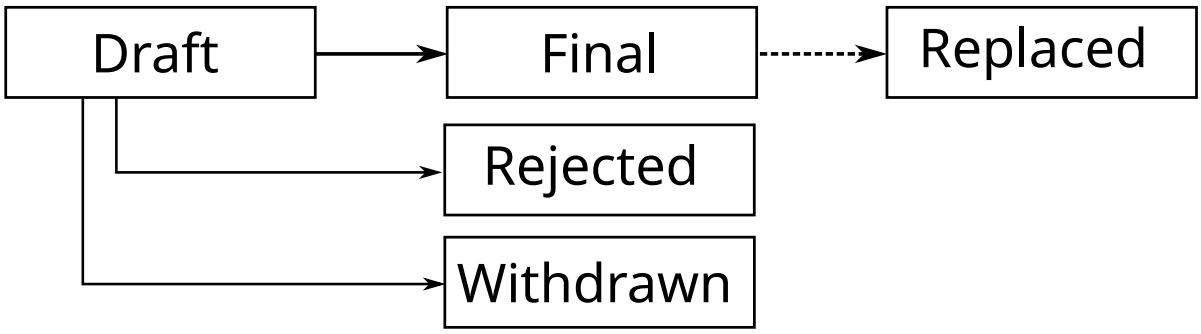
If no progress is being made on the PHEP, the editor should consult the author. The author may resume work on the PHEP, transfer it to another author, or withdraw it. If the author takes none of these actions, the editor may choose to bring the PHEP to the community for acceptance or rejection in its current form.

If a PHEP fails to find consensus, the author may choose to modify it for reconsideration. The PHEP may be "Withdrawn" if the PHEP author themself decides that the PHEP is actually a bad idea, or has accepted that a competing proposal is a better alternative. A PHEP can also "Rejected" if it fails to find consensus and the author is not willing or able to modify it. Perhaps after all is said and done it was not a good idea. It is still important to have a record of this fact.

When a PHEP changes state, the PHEP preamble must be updated accordingly. In addition to updating the Status field, the Resolution header must be added with a direct link to the minutes of the telecons and meetings where a decision on the PHEP was made. The Revision section must be updated with the date of acceptance, as appropriate. The editors must post announcements of PHEP resolution to the PyHC mailing list.

Final PHEPs can also be replaced by a different PHEP, rendering the original obsolete.

The possible paths of the status of PHEPs are as follows:

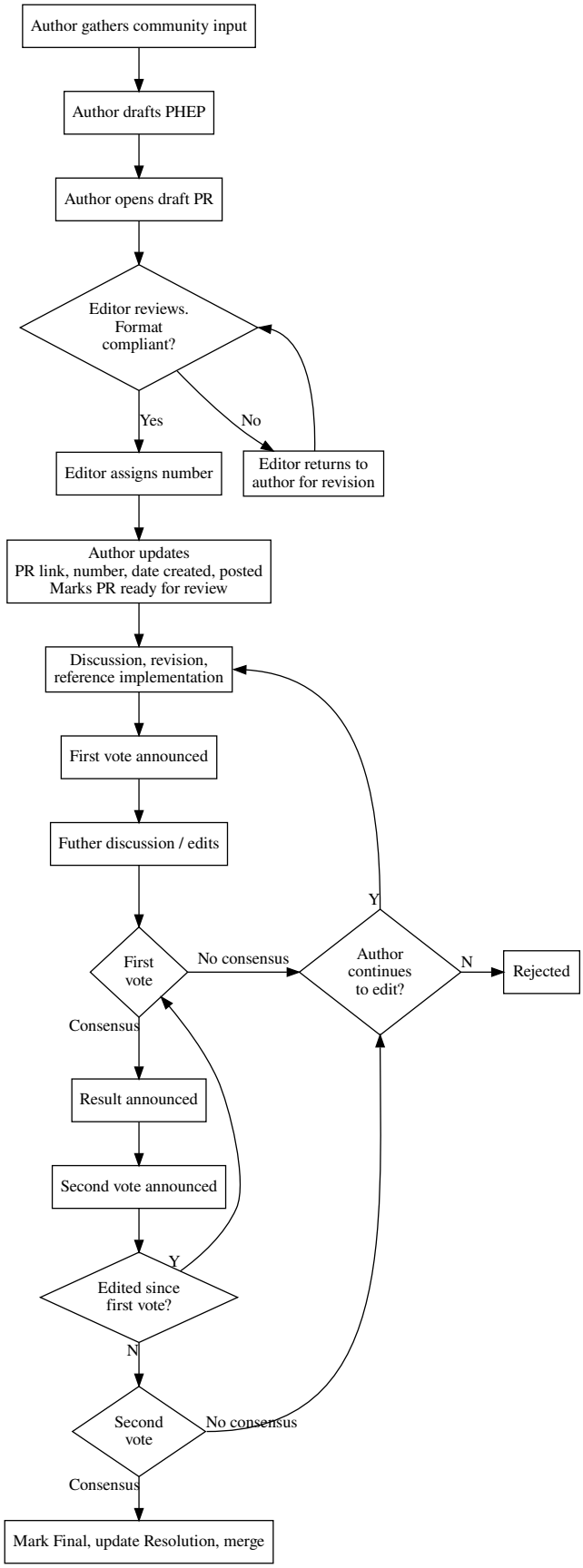


Upon reaching "Final", "Withdrawn", or "Rejected" status, the editors or author push a commit to the PR branch which:

1. Updates the Status appropriately.
2. Puts the current date in the Post-History header.
3. Updates the Revisions section with the date accepted or rejected, as appropriate.
4. Updates the copyright date as necessary.
5. Updates the DOI in the citation with a new DOI that the editors have reserved.

The editors merge the PR into `main` (maintaining the commit history). They tag the resulting commit and mint the reserved DOI. If the new PHEP replaces an existing one, the new DOI should be marked as a new version of the previous, and the Replaced PHEP must have the `Replaced-By` field in its header updated (see PHEP Maintenance).

The complete PHEP approval process is:


Author gathers community input
Author drafts PHEP
Author opens draft PR
Editor reviews.
Format
compliant?
Yes
No
Editor assigns number
Editor returns to
author for revision
Author updates
PR link, number, date created, posted
Marks PR ready for review
Discussion, revision,
reference implementation
First vote announced
Futher discussion / edits
First
vote
No consensus
Author
continues
to edit?
Y
N
Rejected
Consensus
Result announced
Second vote announced
Edited since
first vote?
Y
N
Second
vote
No consensus
Consensus
Mark Final, update Resolution, merge

The author may withdraw the PHEP at any time, so this is not shown. Replacement of a PHEP is not shown as it results from the approval of a new, replacement PHEP using the same process.

PHEP Maintenance

PHEPs are not substantially modified after they have left the Draft state. Once resolution is reached, a PHEP is considered a historical document rather than a living specification and the PHEP number is a permanent identifier for the document.

Rejected and Withdrawn PHEPs must not be resurrected but may be used as the basis for a new PHEP.

Final PHEPs are largely immutable and must be replaced by new PHEPs if substantive changes are needed, rather than updated. The scope of a PHEP should be carefully considered in this light. A PHEP should also be written to be appropriately flexible to changes in technical details that do not affect the core purpose of the PHEP, e.g. specific URI's. The division of details between the PHEP and the reference implementation should be chosen carefully.

A Final PHEP can be revised to accommodate changes that do not change its substance. The permissible scope can be described as changes that would reflect a "patch" or "subminor" version change in semantic versioning schemes; examples include:

1. A PHEP is Replaced due to acceptance of a new PHEP, requiring an update to the header.
2. A clear typo, misspelling, or other error not affecting the meaning of a PHEP is found.
3. A URI or similar external reference is moved / updated.

A revision can be made by the PHEP editors or by the author; if the editors intend to make a revision, they should consult the original author.

To revise a PHEP, a PR must be opened with the following changes to the PHEP:

1. The Revision header is incremented by one.
2. The Status is updated to Replaced and the Replaced-By header updated if a PHEP has been replaced.
3. The Post-History header is updated with the date of the new PR.
4. A new entry is added in the Revisions section, briefly explaining the changes and the need for them.
5. The copyright date is updated as necessary.
6. A new DOI is reserved and the DOI updated in the citation.

These changes must be made in a single commit, which the editors can merge after minor discussion. This PR should not be opened until the path forward is clear: this process is for incremental, non-controversial revisions. Once merged, the commit is tagged. A DOI is minted and marked as new version of the previous DOI.

What belongs in a successful PHEP?

Each PHEP should have the following elements, where they are applicable. It is suggested that each element be addressed in a dedicated section of the PHEP.

1. **Preamble** - RFC 2822 style headers containing metadata about the PHEP, described below.
2. **Abstract** - a short (~200 word) description of the issue being addressed.
3. **Motivation** - The motivation must clearly explain why the existing standards are inadequate to address the problem that the PHEP solves. This can include collecting documented support for the PHEP from important projects in the PyHC ecosystem. PHEP submissions without sufficient motivation may be rejected.
4. **Rationale** - The rationale fleshes out the specification by describing why particular design decisions were made. It should describe alternate designs that were considered and related work.

The rationale should provide evidence of consensus within the community and discuss important objections or concerns raised during discussion.

5. **Specification** - The technical specification should completely describe the proposed standard. The specification should be detailed enough to allow competing, interoperable implementations, where applicable.
6. **Backwards Compatibility** - All PHEPs that introduce backwards incompatibilities must describe these incompatibilities and their severity. The PHEP must explain how the author proposes to deal with these incompatibilities.
7. **Security Implications** - Any security concerns should be explicitly written out.
8. **How to Teach This** - This section may include key points and recommended documentation changes that would help users, new and experienced, apply the PHEP to their work. This section should also document any changes the PHEP makes relative to a PHEP it replaces or some other widely-used standard or reference. This allows an implementation compliant with a previous PHEP (or other standard) to be easily updated for compliance with the PHEP.
9. **Reference Implementation** - The reference implementation must be completed before any PHEP is given status "Final", but it need not be completed before the PHEP is accepted. While there is merit to the approach of reaching consensus on the specification and rationale before writing code, the principle of "rough consensus and running code" is still useful when it comes to resolving many discussions of API details.

   The final implementation must include test code and documentation appropriate for the relevant PyHC project(s).
10. **Rejected Ideas** - Throughout the discussion of a PHEP, various ideas will be proposed which are not incorporated. Those rejected ideas should be recorded along with the reasoning as to why they were rejected. This helps record the thought process behind the final version of the PHEP and prevents people from bringing up the same rejected idea again in subsequent discussions.
11. **Open Issues** - While a PHEP is in draft, ideas can come up which warrant further discussion. Those ideas should be recorded so people know that they are being thought about but do not have a concrete resolution. This helps make sure all issues required for the PHEP to be ready for consideration are complete.
12. **Footnotes** - A collection of footnotes cited in the PHEP, and a place to list non-inline hyperlink targets.
13. **Revisions** - Brief summary of changes by revision number, including the date of approval.
14. **Copyright/license** - Each new PHEP must be placed under a dual license of public domain and CC0-1.0-Universal and include citation information (see this PHEP for an example). The citation must include the DOI for the specific revision of the PHEP, although a DOI for all revisions of the PHEP may also be minted.

PHEP Formats and Templates

PHEPs are UTF-8 encoded text files using the Markdown format. A future PHEP may include a template. PHEP Markdown must be compatible with the most recent version of the CommonMark specification at the time the PHEP is approved. PHEPs must not use GitHub extensions.

The editors are responsible for enforcing PHEP format and consistency of style within a PHEP. They may use automated tools (e.g. pre-commit hooks) to assist.

PHEP Header Preamble

Each PHEP must begin with an RFC 2822 style header preamble, marked off with a code fence. The headers must appear in the following order. Headers marked with "*" are optional. All other headers are required.

```
  ```
  PHEP: <phep number>
  Title: <phep title>
  Author: <list of authors' real names; optionally, email addrs and/or ORCIDs>
  Discussions-To: <URL of current PR>
  Revision: <phep version number>
  Status: <Draft | Rejected | Withdrawn | Final | Replaced>
  Type: <Standards Track | Informational | Process>
  Content-Type: text/markdown; charset=UTF-8; variant=CommonMark
* Requires: <phep numbers>
  Created: <date created on, in dd-mmm-yyyy format>
  Post-History: <dates, in dd-mmm-yyyy format>
* Replaces: <phep number>
* Replaced-By: <phep number>
* Resolution: <url>
  ```
```

The Author header lists the names and optionally the email addresses and/or ORCIDs of all the authors of the PHEP. The format of the Author header values must be:

`Random J. User <random@example.com> <https://orcid.org/0000-0000-0000-0000>`

if the email address and ORCID are included, and just:

`Random J. User`

for just the name (similarly for ORCID without email address and vice versa).

If there are multiple authors, each should be on a separate line following RFC 2822 continuation line conventions.

The Discussions-To header provides the URL to the current PR discussion thread for the PHEP.

The Revision header is an integer, starting at 1, indicating the revision of the PHEP. This is only incremented when a Final PHEP is revised; see PHEP maintenance for details.

The Type header specifies the type of PHEP: Standards Track, Informational, or Process.

The format of a PHEP is specified with a Content-Type header. All PHEPs must use Markdown, and have a value of `text/markdown; charset=UTF-8; variant=CommonMark`.

The Created header records the date that the PHEP was assigned a number, while Post-History is used to record the dates of pushes to the pull request for the PHEP. Both sets of dates should be in `dd-mmm-yyyy` format, e.g. `14-Aug-2001`.

PHEPs may have a Requires header, indicating the PHEP numbers that this PHEP depends on.

PHEPs may also have a Replaced-By header indicating that a PHEP has been rendered obsolete by a later document; the value is the number of the PHEP that replaces the current document. The newer PHEP must have a Replaces header containing the number of the PHEP that it rendered obsolete.

Auxiliary Files

PHEPs may include auxiliary files such as diagrams. Such files must be placed in a subdirectory called `phep-XXXX`, where "XXXX" is the PHEP number. There are no constraints on the names used in files.

Transferring PHEP Ownership

It occasionally becomes necessary to transfer ownership of PHEPs to a new author. It is preferable to retain the original author as a co-author of the transferred PHEP, at their discretion. A good reason to transfer ownership is because the original author no longer has the time or interest in updating it or following through with the PHEP process, or is unreachable. A bad reason to transfer ownership is because the author doesn't agree with the direction of the PHEP. One aim of the PHEP process is to try to build consensus around a PHEP, but if that's not possible, an author can always submit a competing PHEP.

If you are interested in assuming ownership of a PHEP, mention the original author and `@heliophysicsPy/phep-editors` in a comment on the PHEP's pull request. If the original author doesn't respond in a timely manner, the PHEP editors will make a unilateral decision, which can be reversed.

PHEP Editor Responsibilities & Workflow

A PHEP editor must be added to the `@heliophysicsPy/phep-editors` group on GitHub and must watch the `standards` repository.

For each new PHEP that comes in, an editor does the following:

- Read the PHEP to check if it is ready: sound and complete. The ideas must make technical sense, even if they don't seem likely to be accepted.
- The title should accurately describe the content.
- The PHEP's type is correctly labeled ("Standards Track", "Informational", or "Process"), and its status is "Draft".
- The file name extension is correct (i.e. `.md`).
- Skim the PHEP for obvious defects in language (spelling, grammar, sentence structure, etc.) or formatting. Editors may correct problems themselves, but are not required to do so.
- If a project is portrayed as benefiting from or supporting the PHEP, make sure there is some direct indication from the project included to make the support clear.

If the PHEP isn't ready, an editor will send it back to the author for updates, with specific instructions.

Once the PHEP is ready for discussion, a PHEP editor will:

- Assign a PHEP number (almost always the next available number).
- Ask the author to update the PHEP with the PR link, mark the PR ready for review, and post an announcement.

PHEP editors do not make judgments on the content or merits of PHEPs. They merely do the administrative & editorial part. They may participate in the community discussion of a PHEP but have no obligation to be advocates.

Once an author seeks acceptance of a PHEP, the editors should carefully review the existing wording to make sure it is clear, grammatically reasonable, and free of misspellings or other typos.

Disagreements with editors must be mediated by PyHC leadership (as documented in PHEP audience) and they may impose sanctions, including removal of the editor role.

PHEP 1 Acceptance

This PHEP is subject to the same review and acceptance process that it proposes.

PyHC leadership must ensure that discussion of the PHEP process is on the agenda of at least one regular PyHC meeting annually.

Rationale

PHEP 1 is based on PEP 1, which is a process that has credibility in the Python community and has proven appropriate for a fairly large group, the PyHC community being (by definition) larger than the community for any one PyHC package. From this basis, changes were made to make the process relevant to PyHC and to streamline where practicable.

Rejected Ideas

Acceptance without reference implementation

In contrast to PEP 1, any reference implementation must be complete before a PHEP is brought to a vote: there is no "Accepted" state between Draft and Final. This simplifies flow and avoids defining a process for moving from Accepted to Final, particularly if the implementation results in changes.

Effort may go into a reference implementation for a standard that then gets voted down, or requires major changes to be accepted. This can be avoided by getting substantial input before starting the reference implementation and allowing the standard and implementation to evolve together. Implementation is also likely to be smaller effort (relative to the standard) for PyHC than for the Python language.

ASCII encoding

As originally proposed PHEPs were limited to ASCII where practicable. This was deemed overly restrictive; authors should have some freedom of format and style, and it's up to the authors and editors to ensure this is both reasonable and consistent within a PHEP. CommonMark specifies Unicode without an encoding.

Immutability of PHEPs

As currently written, PHEPs are laregly immutable once Final. Substantive changes require replacement rather than modification. This maintains clarity of reference, "PHEP-x" rather than "PHEP-x version y", and avoids confusion arising from rolling updates. It encourages getting the PHEP correct before first acceptance. Finally, it removes the need to define and document an update process.

A PHEP can be updated by introducing a new PHEP that starts with and builds on the original text. Everything is then open for discussion. This encourages complete re-evaluation if the community reaches the point where something is important enough to revisit.

On the downside, immutability discourages substantive but still small and potentially useful changes. It could make more work by requiring submission of a new PHEP.

From the immutability discussion, a streamlined process was introduced for making very small changes, such as fixing typos. This process also accommodates previously implicit changes such as marking a PHEP as replaced or withdrawn after final.

The possible approaches that emerged from discussion roughly span the range:

1. Pure immutability, a PHEP cannot be changed after it is Final. This has conceptually the simplest process (fewest possible steps, headers etc.) but is inflexible.
2. Allowing non-substantive changes to be undertaken without community review; PHEP-1 is written to this standard.
3. Allowing some level of substantive change with potentially more streamlined review, without drafting a new PHEP. This is perhaps the most flexible approach.

One of the reasons for minimizing streamlined changes is the lack of a central decision-making authority; there is, at this writing, no PyHC steering committee or similar, that can assume responsibility for approvals in the absence of community consensus. A future PHEP can be introduced replacing PHEP-1 to provide for a broader revision process.

Quorum requirement

The resolution process does not require a quorum for a vote. Suggested quorum requirements included:

- Requiring "substantial community representation" at a vote.
- Requiring representation from at least one core package at a vote.

In theory any properly-announced meeting could "vote" to accept a PHEP with a very small number of people present and potentially ram through a standard without community consensus. Safeguards preventing this are:

- Objections may be raised in the PR, so nobody need be present to break consensus for acceptance.
- The minimum one-month notice before the meeting allows objections to be raised.
- If edits are made at the meeting before the vote, the the period between the two votes does not permit edits and allows for further objections.

As consensus is required, any of these "nays" carry the vote.

The recommendation that the author of the PHEP be present for the vote came out of quorum discussions.

Recording names at acceptance

There was consideration of recording the names of people present when a PHEP is accepted, or who specifically want to "sign on". This was decided to be out-of-scope for the PHEP process; the notes of the meeting should capture relevant information.

Source formatting

Certain ideas regarding the format of the Markdown source were removed as more complicated than their benefit: making the source the version of record over the rendered, notes on specifics of rendering, and trying to get pretty diffs of Markdown.

Open Issues

There are no remaining open issues.

Revisions

Revision 1 (15-Apr-2024): Initial approval.

Copyright

 It should be cited as:

```
@techreport(phep1,
  author = {Jonathan T. Niehof},
  title  = {PHEP Purpose and Guidelines},
    year = {2023--2024},
    type = {PHEP},
  number = {1},
     doi = {10.5281/zenodo.10988007}
)
```

This document draws from:

- https://peps.python.org/pep-0001/

- https://numpy.org/neps/nep-0000.html
- https://github.com/sunpy/sunpy-SEP/blob/master/SEP-0001.md