Stream:	Internet Engineering Task Force (IETF)
RFC:	8874
Category:	Informational
Published:	August 2020
ISSN:	2070-1721
Authors:	M. Thomson	B. Stark
---	---	---
	Mozilla	*AT&T*

RFC 8874
Working Group GitHub Usage Guidance

Abstract

This document provides a set of guidelines for working groups that choose to use GitHub for their work.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8874.

Table of Contents

1. Introduction
   1.1. Distributed Version Control Systems
   1.2. GitHub
   1.3. Other Services
   1.4. Document Goals
   1.5. Notational Conventions
2. Administrative Policies
   2.1. Organizations
   2.2. Communicating Policies
3. Deciding to Use GitHub
   3.1. What to Use GitHub For
   3.2. Repositories
   3.3. Editors and Contributors
   3.4. Document Formats
4. Contribution Methods
   4.1. Issue Tracker
      4.1.1. Issue Labels
      4.1.2. Closing Issues
      4.1.3. Reopening Issues
   4.2. Pull Requests
      4.2.1. Discussion on Pull Requests
      4.2.2. Merging Pull Requests
   4.3. Monitoring Activity
5. Typical Working Group Policies
   5.1. Document Management Mode
   5.2. Issue Tracking Mode

5.3. Issue Discussion Mode
5.3.1. Early Design Phases
5.3.2. Managing Mature Documents
5.4. Issue Labeling Schemes
5.4.1. Editorial/Design Labeling
5.4.2. Decision Labeling
5.4.3. Component Labeling
5.4.4. Other Labels
6. Internet-Draft Publication
7. Assessing Consensus
8. Continuous Integration
9. Advice to Editors
10. Security Considerations
11. IANA Considerations
12. References
12.1. Normative References
12.2. Informative References
Acknowledgments
Authors' Addresses

1. Introduction

The IETF has an open and transparent process for developing standards. The use of GitHub or similar tools, when used as part of this process, can have several objectives. GitHub provides tools that can be helpful in editing documents. Use of this service has been found to reduce the time that a working group needs to produce documents and to improve the quality of the final result.

The use of version control improves the traceability and visibility of changes. Issue tracking can be used to manage open issues and provide a record of their resolution. Pull requests allow for better engagement on technical and editorial changes, and encourage contributions from a larger set of contributors. Using GitHub can also broaden the community of contributors for a specification.

The main purpose of this document is to provide guidelines for how a working group might integrate the capabilities provided by GitHub into their processes for developing Internet-Drafts. Whether to use GitHub and whether to adopt these practices is left to the discretion of the working group.

This document is meant as a supplement to existing working group practices. It provides guidance to working group chairs and participants on how they can best use GitHub within the framework established by RFC 2418 [RFC2418]. This document aims to establish norms that reduce the variation in usage patterns between different working groups and to help avoid issues that have been encountered in the past.

A companion document, [RFC8875], describes administrative processes that support the practices described in this document.

Although the operation of IRTF research groups can be similar in function to working groups, this document only directly addresses the needs of working groups. However, other groups may draw inspiration for GitHub use from the contents herein.

1.1. Distributed Version Control Systems

Version control systems are a critical component of software engineering and are also quite useful for document editing.

Git is a distributed version control system that can operate without a central service. Each instance of a repository contains a number of revisions. Each revision stores the complete state of a set of files. Users are able to create new revisions in their copy of a repository and share revisions between copies of repositories.

1.2. GitHub

GitHub is a service operated at <https://github.com/>. GitHub provides centralized storage for Git repositories. GitHub is freely accessible on the open Internet.

GitHub provides a simplified and integrated interface to Git and also provides basic user management, an issue tracker, associated wikis, project hosting, and other features.

There are a large number of projects at GitHub and a very large community of contributors. One way in which some IETF working groups have benefited from use of the service is through increased numbers of reviews of the document and associated issues, along with other improvements that come from facilitating participation by a broader community.

1.3. Other Services

Git is not the only version control system available, nor is GitHub the only possible choice for hosting. There are other services that host revision control repositories and provide similar additional features as GitHub. For instance, BitBucket and GitLab provide similar feature sets. In addition to a hosted service, software for custom installations exists.

This document concentrates primarily on GitHub as it has a large and active community of contributors. As a result, some content might not be applicable to other similar services. A working group that decides to adopt an alternative tool or service can still benefit from the general guidance in this document.

1.4. Document Goals

This document aims to describe how a working group might best apply GitHub to their work. The intent is to allow each working group considerable flexibility in how they use GitHub.

This document requires that policies for use of GitHub are agreed upon and clearly communicated within the working group (see Section 2). The remainder of the document contains guidelines and advice on how to construct a workable policy.

The requirements here apply to the case where a working group decides to use GitHub as a primary means of interaction. Individuals can set their own policies when using GitHub for managing their own drafts or for managing drafts that they edit on behalf of a working group that has not explicitly adopted GitHub.

For both sets of users, this document aims to provide some amount of advice on practices that have been effective.

This document only aims to address use of GitHub in developing documents. A working group could choose to use the tool to aid in managing their charter or session materials such as agendas, minutes, and presentations. Though the advice here might apply more broadly, using GitHub to manage other material is out of scope for this document.

1.5. Notational Conventions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses a lot of terms related to Git and GitHub; see [GLOSSARY] for information on these terms.

2. Administrative Policies

The following administrative rules provide the necessary oversight and transparency.

2.1. Organizations

Organizations are a way of forming groups of contributors on GitHub. The working group **SHOULD** create a new organization for its work. A working group organization **SHOULD** be named consistently so that it can be found. For instance, the name could be ietf-wg-<wgname>, as recommended in [RFC8875].

A single organization **SHOULD NOT** be used for all IETF activity or all activity within an area. Large organizations create too much overhead for general management tasks.

GitHub requires that each organization have at least one owner. The owners for a working group repository **MUST** include responsible area directors and the IETF Secretariat. Working group chairs **SHOULD** also be included as owners. Area directors **MAY** also designate a delegate that becomes an owner, such as another area director from the same area. An organization **MUST** have at least two owners.

Within an organization, members can be grouped into teams. A team with "Admin" access to repositories **SHOULD** be created for the working group chairs and any working group secretary.

Details about creating organizations adhering to these guidelines can be found in [RFC8875].

2.2. Communicating Policies

Each working group **MAY** set its own policy as to whether and how it uses GitHub. It is important that occasional participants in the working group and others accustomed to IETF tools be able to determine this and easily find the policy and GitHub organization.

A simple example of how to do this is to include a link to the GitHub organization on the working group charter page in the datatracker. Similarly, if there are additional resources, such as mailing lists, links to those resources could also be added.

Repositories **MUST** include a copy of or reference to the policy that applies to managing any documents they contain. Updating the README or CONTRIBUTING file in the repository with details of the process ensures that the process is recorded in a stable location other than the mailing list archive. This also makes working group policies available to casual contributors who might only interact with the GitHub repository.

GitHub prominently links to the CONTRIBUTING file on certain pages. This file **SHOULD** be used in preference to the README for information that new contributors need. The README **SHOULD** contain a link to the CONTRIBUTING file.

In addition to working group policies, notices on repositories **MUST** include citations for the IETF Note Well.

3. Deciding to Use GitHub

Working group chairs are responsible for determining how to best accomplish the charter objectives in an open and transparent fashion. The working group chairs are responsible for determining if there is interest in using GitHub and for making a consensus call about whether the proposed policy and use is acceptable.

Chairs **SHOULD** involve area directors in any decision to use GitHub, especially where substantive discussion of issues is permitted as described in Section 5.3.

3.1. What to Use GitHub For

Working group chairs decide what GitHub features the working group will rely upon. Section 4 contains a more thorough discussion on the different features that can be used.

Working group chairs who decide to use GitHub **MUST** inform the working group of their decision on the working group mailing list. An email detailing how the working group intends to use GitHub is sufficient, though it might be helpful to occasionally remind new contributors of these guidelines.

Working group chairs are responsible for ensuring that any policy they adopt is enforced and maintained.

The set of GitHub features (Section 4) that the working group relies upon need to be clearly documented in policies. This document provides some guidance on potential policies and how those might be applied.

Features that the working group does not rely upon can be made available to document editors. Editors are then able to use these features for their own purposes. For example, though the working group might not formally use issues to track items that require further discussion in order to reach consensus, keeping the issue tracker available to editors can be valuable.

Working group policies need to be set with the goal of improving transparency, participation, and ultimately the quality of documents. At times, it might be appropriate to impose some limitations on what document editors are able to do in order to serve these goals. Chairs are encouraged to periodically consult with document editors to ensure that policies are effective.

A document editor can still use GitHub independently for documents that they edit, even if the working group does not expressly choose to use GitHub. Any such public repository **MUST** follow the IETF Note Well and bear notices; see Section 2.2. This recognizes that editors have traditionally chosen their own methods for managing the documents they edit but preserves the need for contributors to understand their obligations with respect to IETF processes.

Work done in GitHub has no special status. The output of any activity using GitHub needs to be taken to the working group and is subject to approval, rejection, or modification by the working group as with any other input.

3.2. Repositories

New repositories can be created within the working group organization at the discretion of the chairs. Chairs could decide to only create new repositories for adopted working group items, or they might create repositories for individual documents on request.

Maintaining private repositories for working group products is not recommended without specific cause. For instance, a document that details a security vulnerability might be kept private prior to its initial publication as an Internet-Draft. Once an Internet-Draft is published, repositories for working group documents **MUST** be made public.

The adoption status of any document **MUST** be clear from the contents of the repository. This can be achieved by having the name of the document reflect status (that is, draft-ietf-<wgname>-... indicates that the document was adopted) or through a prominent notice (such as in the README).

Experience has shown that maintaining separate repositories for independent documents is most manageable. This allows the work in that repository to be focused on a single item.

Closely related documents, such as those that together address a single milestone, might be placed in a single repository. This allows editors to more easily manage changes and issues that affect multiple documents.

Maintaining multiple documents in the same repository can add overhead that negatively affects individual documents. For instance, issues might require additional markings to identify the document that they affect. Also, because editors all have write access to the repository, managing the set of people with write access to a larger repository is more difficult (Section 3.3).

3.3. Editors and Contributors

Working group chairs **MUST** give document editors write access to document repositories. This can be done by creating teams with write access and allocating editors to those teams or by making editors collaborators on the repository.

Working group chairs **MAY** also grant other individuals write access for other reasons such as maintaining supporting code or build configurations. Working group chairs, as administrators or owners of the organization, might also have write access to repositories. Users other than document editors, including chairs, **SHOULD NOT** make changes to working group documents without prior coordination with document editors.

A working group **MAY** create a team for regular contributors that is only given read access to a repository. This does not confer additional privileges on these contributors; it instead allows for issues and pull requests to be assigned to those people. This can be used to manage the assignment of editorial or review tasks to individuals outside of the editor team.

3.4. Document Formats

In addition to the canonical XML format [RFC7991], document editors might choose to use a different input form for editing documents, such as Markdown. Markdown-based formats are more accessible for new contributors, though ultimately, decisions about format are left to document editors.

Formats that are not text-based **SHOULD NOT** be used, as these are ill-disposed to the sorts of interaction that revision control enables.

4. Contribution Methods

Contributions to documents come in many forms. GitHub provides a range of options in addition to email. Input on GitHub can take the form of new issues and pull requests, comments on issues and pull requests, and comments on commits.

4.1. Issue Tracker

The GitHub issue tracker can be an effective way of managing the set of open issues on a document. Issues, both open and closed, can be a useful way of recording decisions made by a working group.

Issues can be given arbitrary labels, assigned to contributors, and assembled into milestones. The issue tracker is integrated into the repository; an issue can be closed using a special marker in a commit message.

When deciding to use GitHub, working group chairs **MUST** decide how the GitHub issue tracker is used. Use of the issue tracker could be limited to recording the existence of issues, or it might be used as the venue for substantial technical discussion between contributors.

A working group policy **MAY** require that all substantive changes be tracked using issues. Suggested policies for the use of the GitHub issue tracker are the primary subject of Section 5.

4.1.1. Issue Labels

A system of labeling issues can be effective in managing issues. For instance, marking substantive issues separately from editorial can be helpful at guiding discussion. Using labels can also be helpful in identifying issues for which consensus has been achieved but that require editors to integrate the changes into a document.

Labels can be used to identify particular categories of issues or to mark specific issues for discussion at an upcoming session.

Chairs communicate any process that specifically relates to the use of labels to the working group. This includes the semantics of labels, and who can apply and remove these labels. Section 5.4 describes some basic strategies that might be adopted to manage decision-making processes.

4.1.2. Closing Issues

Editors have write access to repositories, which also allows them to close issues. The user that opens an issue is also able to close the issue. Chairs **MUST** provide guidance on who is permitted to close an issue and under what conditions.

Restrictions on who can close an issue and under what circumstances are generally not advisable until a document has reached a certain degree of maturity.

4.1.3. Reopening Issues

Issues that have reached a resolution that has working group consensus **MUST NOT** be reopened unless new information is presented.

For long-running work items, new contributors often raise issues that have already been resolved. Moreover, there could be temptation to reopen contentious issues resolved with rough consensus. Determining whether arguments presented in favor of reopening an issue represents new information might require some discussion in the working group.

Chairs are empowered to exercise discretion in determining whether or not to reopen issues. For more difficult matters, the chairs **MAY** insist that the working group reach consensus on whether an issue should be reopened. Note, however, that any product of this process still needs to have the support of rough consensus in the working group, which could justify reopening issues.

4.2. Pull Requests

A pull request is a GitHub feature that allows a user to request a change to a repository. A user does not need to have write access to a repository to create a pull request. A user can create a "fork", or copy, of any public repository. The user has write access to their own fork, allowing them to make local changes. A pull request asks the owner of a repository to merge a specific set of changes from a fork (or any branch) into their copy.

Editors are encouraged to make pull requests for all substantial changes rather than committing directly to the "primary" branch of the repository. See Section 5.3.2 for discussion on what constitutes a substantial change. A pull request creates an artifact that records the reasons for changes and provides other contributors with an opportunity to review the change. Ideally, pull requests that address substantive issues mention the issue they address in the opening comment. A working group policy could require that pull requests be used in this fashion.

> Note: This document assumes that there is a unified effort on a document, all concentrated on a single Git branch. More advanced usage of Git is not in the scope of this document.

Pull requests have many of the same properties as issues, including the ability to host discussion and bear labels. Critically, using pull requests creates a record of actions taken.

For significant changes, leaving a pull request open until discussion of the issue within the working group concludes allows the pull request to track the discussion and properly capture the outcome of discussions. Pull requests can be updated as discussions continue, or in response to feedback.

Groups of editors could adopt a practice of having one editor create a pull request and another merge it. This ensures that changes are reviewed by editors. Editors are given discretion in how they manage changes amongst themselves.

4.2.1. Discussion on Pull Requests

In addition to the features that pull requests share with issues, users can also review the changes in a pull request. This is a valuable feature, but it presents some challenges.

Comments in a review other than a summary are attached to specific lines of the proposed change. Such comments can be hard or impossible to find if changes are subsequently made to the pull request. This is problematic for contributors who do not track discussions closely.

For this reason, working group chairs **SHOULD** discourage the use of inline comments for substantial technical discussion of issues.

4.2.2. Merging Pull Requests

A working group **MUST** determine who is permitted to merge pull requests. Document editors **SHOULD** be permitted to merge pull requests at their discretion. This requires that editors exercise some judgment. Working group chairs **MAY** occasionally identify a pull request and request that editors withhold merging until working group consensus has been assessed.

Note that the copy of a document that is maintained on GitHub does not need to be a perfect reflection of working group consensus at every point in time. Document editors need some flexibility in how they manage a document.

4.3. Monitoring Activity

GitHub produces individualized email notifications of activity that each user can adjust to their preferences. In addition to these, some working groups have created read-only mailing lists that receive notifications about activity on working group repositories. The volume of information on these lists can be too high to monitor actively, but access to an archive of actions can be useful.

An alternative is to rely on periodic email summaries of activity, such as those produced by a notification tool like github-notify-ml. This tool has been used effectively in several working groups, though it requires server infrastructure.

Additionally, clear reporting about the changes that were included in each revision of an Internet-Draft helps ensure that contributors can follow activity. This might be achieved by requesting that editors provide a change log that captures substantive changes to the document in each revision.

5. Typical Working Group Policies

Current experience with use of GitHub suggests a few different approaches to greater use of the tool in working groups.

This section describes some basic modes for interacting with GitHub, each progressively more involved. This starts with a very lightweight interaction where document management is the only feature that is formally used; then, progressively more intensive use of the GitHub issue

tracking capabilities is described. These approaches differ primarily in how discussion of substantive matters is managed. Most of the advice in this document applies equally to all models.

Working groups can adjust these policies to suit their needs but are advised to avoid gratuitous changes for the sake of consistency across the IETF as a whole. It is possible to use different processes for different documents in the working group.

Working group chairs are responsible for confirming that the working group has consensus to adopt any process. In particular, the introduction of a more tightly controlled process can have the effect of privileging positions already captured in documents, which might disadvantage alternative viewpoints.

5.1. Document Management Mode

In this mode of interaction, GitHub repositories are used to manage changes to documents, but the bulk of the work is conducted using email, face-to-face meetings, and other more traditional interactions. The intent of this policy is to enable document and issue management using GitHub while minimizing the complexity of the process.

In the version of this mode with the least interaction with GitHub, a repository is created for the purposes of document management by editors. Editors might maintain issues and pull requests for their own benefit, but these have no formal standing in the working group process.

5.2. Issue Tracking Mode

In addition to managing documents, the working group might choose to use GitHub for tracking outstanding issues. In this mode of interaction, a record of the existence of substantive technical discussions is tracked using issues in the issue tracker. However, discussion of any substantial matters is always conducted on mailing lists.

Under this mode, issues and pull requests can be opened by anyone, but anything deemed substantive **MUST** be resolved exclusively on the mailing list. Discussion on GitHub is limited to recording the state of issues. Only editorial matters can be resolved using the issue tracker.

Chairs and editors are given discretion in determining what issues are substantive. As documents mature, it is generally prudent to prefer consulting the mailing list where there is doubt. As with other working group decisions, chairs are the arbiters in case of dispute.

A recurrent problem with this mode of interaction is the tendency for discussions to spontaneously develop in the issue tracker. This requires a degree of discipline from chairs and editors to ensure that any substantive matters are taken to the mailing list.

Retaining mailing lists as the primary venue for discussion of substantive matters ensures that this mode, along with the document management mode, is most compatible with existing work practices for working groups. Participants in a working group that operates under either model can reasonably be expected to receive all relevant communication about the work of the group from the working group mailing list.

Though the mailing list is used for making decisions, the issue tracker can still be a useful record of the state of issues. It is often useful if chairs or editors record details of decisions in issue comments when closing issues as resolved.

5.3. Issue Discussion Mode

This GitHub interaction mode differs from the other modes in that discussion relating to substantive technical matters is allowed to occur on GitHub issues. Though decisions are always subject to confirmation on the mailing list, participants are permitted to conduct substantive discussions on the issue tracker. In some cases, this can include making some decisions without involving the working group mailing list.

A working group mailing list remains a critical venue for decision making, even where issue discussion occurs elsewhere. Working group mailing lists generally include a wider audience than those who follow issue discussion, so difficult issues always benefit from list discussion.

Decisions about working group consensus **MUST** always be confirmed using the working group mailing list. However, depending on the maturity of documents, this might be a more lightweight interaction such as sending an email confirmation for an initial set of resolutions arising from discussions on the issue tracker.

Using the mailing list to resolve difficult or controversial issues is strongly encouraged. In those cases, the issue tracker might be used to more fully develop an understanding of problems before initiating a discussion on the mailing list, along lines similar to the design team process (see Section 6.5 of [RFC2418]).

As a more involved process, adopting this mode can require changes in policies as documents become more mature.

5.3.1. Early Design Phases

During early phases of the design of a protocol, chairs **MAY** allow editors to manage all aspects of issues. Editors are permitted to make decisions about how to both identify and resolve technical issues, including making any changes that editors feel necessary.

The primary reason to grant editors more discretionary power is to improve the speed with which changes can be made. In many cases, documents that are adopted by a working group are already sufficiently mature, and a looser process is not beneficial. A looser process increases the risk of missing issues that need working group consensus and integrating substantive changes based on decisions that don't reflect the consensus of the working group.

Changes made by editors under this process do not lack options for identifying and correcting problems. GitHub and Git provide tools for ensuring that changes are tracked and can be audited. Within the usual working group process, it is expected that Internet-Drafts will receive regular review. Also, process checkpoints like Working Group Last Call (WGLC; Section 7.4 of [RFC2418]) provide additional safeguards against abuse.

Working groups are advised against allowing editors this degree of flexibility for the entirety of a document life cycle. Once a document is more stable and mature, it could be useful to move to a more tightly controlled process.

5.3.2. Managing Mature Documents

As a document matures, it becomes more important to understand not just that the document as a whole retains the support of the working group, but that changes are not made without wider consultation.

Chairs **MAY** choose to manage the process of deciding which issues are substantive. For instance, chairs might reserve the ability to use the `design` label for new issues (see Section 5.4.1) and to close issues marked as `design`. Chairs **SHOULD** always allow document editors to identify and address editorial issues as they see fit.

As documents mature further, explicit confirmation of technical decisions with the working group mailing list becomes more important.

Chairs can declare working group consensus regarding the resolution of issues in the abstract, allowing editors discretion on how to capture the decisions in documents.

More mature documents require not only consensus, but consensus about specific text. Ideally, substantive changes to documents that have passed WGLC are proposed as pull requests and **MUST** be discussed on the mailing list. Having chairs explicitly confirm consensus on changes ensures that previous consensus decisions are not overturned without cause. Chairs **MAY** institute this stricter process prior to WGLC.

> Note: It is generally sufficient to trust editors to manage adherence with these policies, aided by the transparency provided by the version control system. There are tools that can be used to more tightly control access to repositories, but they can be overly constraining.

5.4. Issue Labeling Schemes

Several schemes for use of issue labels in managing issues have been used successfully. This section outlines these strategies and how they might be applied.

A design/editorial split (see Section 5.4.1) is useful in all cases in which the issue tracking capability is used. A working group that only uses GitHub for issue tracking might find that distinction sufficient for their needs.

Working groups or editors might use additional labels as they choose. Any label that is used as part of a process requires that the process be documented and announced by working group chairs. Editors **SHOULD** be permitted to use labels to manage issues without any formal process significance being attached to those issues.

5.4.1. Editorial/Design Labeling

The most important distinction about an issue is whether it is substantive. The labels of `editorial` and `design` are used to represent this distinction.

An issue labeled as `editorial` has no substantive effect on a document except to the extent that addressing the issue might make understanding the specification easier. Resolution of `editorial` issues can be left to the discretion of editors.

An issue labeled as `design` has or might have a substantive effect on a document. For protocol specifications, a `design` issue is one that might affect implementations or interoperability requirements. Addressing a `design` issue ultimately requires working group consensus, even if the resolution is to make no change.

This distinction can be applied to all types of documents. For instance, a `design` issue for an Informational document might be raised to discuss possible changes to important concepts in the document.

5.4.2. Decision Labeling

Labels can be used to manage processes. As documents mature and issues become more numerous, labels can be used to clearly mark the status of issues. In particular, the labeling of issues can be used to help manage working group decisions.

For documents that are less mature, issues with resolutions but no specific proposals for changes to text might be marked `editor-ready` as a way of signaling that there is consensus on an approach, but no specific proposal. Chairs might use this to signal that discussion is complete and that editors are to be given discretion in the construction of text.

In contrast, if specific text is a prerequisite for resolving issues, as might be the case for more mature documents, a `proposal-ready` label might be used by editors to mark issues that they believe to have acceptable resolutions.

For resolved issues, a `has-consensus` label might be used by chairs to mark issues for which formal working group decisions have been made (Section 6.1 of [RFC2418]).

A `future` or `next-version` label might be used to mark and thereby save issues for a future version of, or extension to, a protocol, particularly where a resolution is made to take no action.

5.4.3. Component Labeling

Repositories with multiple interrelated documents or a complex document with multiple logical components might benefit from labels that identify different aspects of the work. The choice of appropriate labels for components will depend on the structure of specific documents.

5.4.4. Other Labels

Other labels can be used depending on the needs of editors and working group processes. For example,

- An `invalid` label might be used for issues that were raised in error.
- A `blocked` label might indicate an issue is awaiting resolution of an external process or related issue.
- A `parked` label might be used to indicate issues that do not require immediate working group attention.

6. Internet-Draft Publication

During the development of a document, individual revisions of the document can be built and formally submitted as an Internet-Draft. This creates a stable snapshot and makes the content of the in-progress document available to a wider audience. Documents submitted as Internet-Drafts are not expected to address all open issues or merge outstanding pull requests.

Section 7.1 of [RFC2418] recommends that editors create a new Internet-Draft submission two weeks prior to every session, which includes IETF meetings, other in-person meetings, and telephone or video conferences. Though discussion could use the current version of a document from version control, participants in a session cannot be expected to monitor changes to documents in real time; a published Internet-Draft ensures that there is a common, stable state that is known to all participants.

Internet-Drafts that use a GitHub repository **SHOULD** include a notice that includes a reference to the repository. This notice might also include information about where to discuss the draft.

Revisions used to generate documents that are submitted as Internet-Drafts **SHOULD** be tagged in repositories to provide a record of submissions.

Working group chairs **MAY** request a revision of an Internet-Draft being managed on GitHub at any time, in consultation with document editors.

7. Assessing Consensus

The work that occurs on GitHub could be part of the consensus process, but the ultimate decision on consensus regarding a document is made by the chairs [RFC2026].

GitHub facilitates more involved interactions, which can result in a much higher level of activity than a typical working group mailing list. Participants who wish to limit their time commitment might follow GitHub activity selectively, either by following only specific issues or by occasionally reviewing the state of the document. Other participants might not use GitHub at all. Chairs are reminded that assessing consensus based on GitHub content alone cannot be assumed to reach all interested participants.

As described in [RFC2418], chairs consider input from all discussion venues when assessing consensus. These include mailing lists, IETF meetings, and interim meetings in addition to discussion on GitHub. Each venue has different selection biases that might need to be considered.

A working group chair **MUST** consult the working group mailing list for any issue that is potentially contentious. Relying on input provided through GitHub alone might result in gaining input from a narrower set of participants. This includes important milestones like Working Group Last Call, where review from the widest possible audience ensures a higher quality document.

If permitted, GitHub will be used for technical discussion and decisions, especially during early stages of development of a document. Any decisions are confirmed through review within the working group and, ultimately, through Working Group Last Call; see Section 7.4 of [RFC2418].

The use of issues and labels has been effective in managing contentious issues. Explicitly labeling closed issues to identify those with formal consensus means that there is no confusion about the status of issues.

8. Continuous Integration

Various third-party services offer the ability to run tests and other work when changes are made to a repository.

One common practice is to use these continuous integration services to build a text or HTML version of a document. This is then published to GitHub Pages, which allows users to view a version of the most recent revision of a document. Including a prominent link to this version of the document (such as in the README) makes it easier for new contributors to find a readable copy of the most recent version of a draft. In addition, including links to differences between this generated version and any published document helps contributors identify recent changes.

Continuous integration can also validate pull requests and other changes for errors. The most basic check is whether the source file can be transformed successfully into a valid Internet-Draft. For example, this might include checking that the XML source is syntactically correct.

For a document that uses formal languages as part of the specification, such as schema or source code, a continuous integration system might also be used to validate any formal language that the document contains. Tests for any source code that the document contains might be run, or examples might be checked for correctness.

9. Advice to Editors

Document editors are primarily responsible for maintaining documents. Taking on a few additional tasks can greatly improve the process for the working group.

Using GitHub means that it is more likely that a contribution is made by users who are not very familiar with the work. Pull requests from new contributors can contain errors or omissions. Duplicate issues are commonplace. Proposed changes might have grammatical errors or they

might diverge from existing style. If a change is generally sound, rather than rejecting the pull request or requesting changes, editors could instead accept the change and then make any necessary corrections.

Editors **SHOULD NOT** close a pull request or issue without first understanding why the item was created. Editors and chairs **SHOULD** try to explain every action clearly and concisely. Even if a contributor seems rude, being courteous in response is always best.

If a contributor makes a comment that raises a new issue, editors can create an issue or, if there is an obvious solution, a pull request. It does not matter what venue the issue was raised in (e.g., email, issue discussion, a pull request review); capturing issues quickly ensures that problems become visible and can be tracked.

This takes a little more effort, but these simple steps can help encourage contributions, which will ultimately improve the quality of documents.

10. Security Considerations

Continuity of operations is always a consideration when taking a dependency on an external service. If GitHub were to fail in some way, anyone relying upon its services would be seriously affected.

Widespread use of Git reduces the exposure to a system failure because the primary repository is replicated in multiple locations. This includes hosted web pages; the content of web pages is maintained as a branch in the main repository.

However, other information maintained on GitHub is more vulnerable to loss. This includes issues and discussion on those issues, discussion and reviews of commits and pull requests, and any content hosted on the wiki. Tools exist for extracting this information for backup.

As specified in [RFC8875], backup copies of repositories and other important data **SHOULD** be maintained.

The potential for malicious actions by compromised or malcontent editors, chairs, and area directors is relevant in maintaining the integrity of the content that GitHub hosts. Backups allow for recovery of content, and regular submissions as Internet-Drafts ensure that work is not lost completely.

A compromise of GitHub does not pose a significant threat to working group operations as it is expected that most data, aside from individual credentials, is made public.

A compromise of credentials could mean loss of control for repositories an organizations. All contributors, especially those with commit or admin privileges **SHOULD** use current best practices for protection of credentials, such as multi-factor authentication.

11. IANA Considerations

This document has no IANA actions.

12. References

12.1. Normative References

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, DOI 10.17487/RFC2026, October 1996, <https://www.rfc-editor.org/info/rfc2026>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2418] Bradner, S., "IETF Working Group Guidelines and Procedures", BCP 25, RFC 2418, DOI 10.17487/RFC2418, September 1998, <https://www.rfc-editor.org/info/rfc2418>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

12.2. Informative References

[GLOSSARY] GitHub, "GitHub glossary", March 2020, <https://help.github.com/en/github/getting-started-with-github/github-glossary>.

[RFC7991] Hoffman, P., "The "xml2rfc" Version 3 Vocabulary", RFC 7991, DOI 10.17487/RFC7991, December 2016, <https://www.rfc-editor.org/info/rfc7991>.

[RFC8875] Cooper, A. and P. Hoffman, "Working Group GitHub Administration", RFC 8875, DOI 10.17487/RFC8875, August 2020, <https://www.rfc-editor.org/info/rfc8875>.

Acknowledgments

This work would not have been possible without the hard work of those people who have trialed the use of GitHub at the IETF. Alia Atlas contributed significant text to an earlier draft version of this document. Tommy Pauly, Rich Salz, and Christopher Wood all provided significant input.

Authors' Addresses

Martin Thomson
Mozilla
Email: mt@lowentropy.net

Barbara Stark
AT&T
Email: barbara.stark@att.com