Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 21, 2018

B. Claise
R. Barnes
J. Clarke
Cisco

January 17, 2018

# Semantic Versioning and Structure for IETF Specifications
# draft-claise-semver-02

## Abstract

In the Internet engineering ecosystem, there is increasingly a need for specifications that evolve over time, and which are encoded directly in structured formats (e.g., YANG models). Internet-Drafts are a poor fit for working groups that want to produce structured specifications, and publishing versions of an evolving specification as RFC makes it difficult to track the specification over time. This document outlines recommendations for how working groups can provide consistent, meaningful versions for specifications over time and work directly on structured documents while still fitting within established IETF processes.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 21, 2018.

Table of Contents



## 1. Introduction

The pace at which the software that drives the Internet is developed and deployed today is much faster than it was early in the Internet's history. In this environment, maintaining interoperability can be more challenging.

Two of the major mechanisms that have been developed for driving interoperability in a more dynamic ecosystem are structured specifications and semantic versioning. Structured specifications allow much of the work of implementing a specification to be automated, so that developers can focus on the parts of a specification that really need human involvement. Semantic versioning helps operators know what versions can be deployed without breaking running systems, so that they can safely deploy updated versions of a specification more quickly.

The traditional practices of the IETF interact poorly with these mechanisms. Each document presented to the IETF for last call and IESG approval must be formatted as an Internet-Draft, i.e., as unstructured text. All RFCs across the IETF share a single, common numbering space, so that RFC numbers have no useful semantic with respect to versioning. Nonetheless, there is still a need to be able to capture the consensus of the IETF at critical points in the life-cycle of a specification.

This document describes recommendations for how a working group (WG) that wishes to produce structured specifications with semantic version numbers can interact best with IETF processes.

## 2. Managing Semantic Versions

While some of this process might apply to completely new work (such as a YANG module), the process in this document applies to WG adopted work items or to modification of an existing IETF-approved work item (typically a bis document).

We start from the premise that a working group controls a version-controlled repository (e.g., Git, SVN, etc.) for a structured specification (not formatted as an Working Group Internet-Draft), and can "tag" commits in the repository as having certain version numbers. We assume that there is one repository per specification, so that version tags don't need to state the specification to which they refer.

The recommended structure for semantic versions follows the widely-used three-part convention, with an additional field for use in working group processes:

```
MAJOR.MINOR.PATCH
```

The fields in such a structured version have the following semantics (cf. semver.org):

- MAJOR is incremented when the new version of the specification is incompatible with previous versions.
- MINOR is incremented when new functionality is added in a manner that is backward-compatible with previous versions.
- PATCH is incremented when bug fixes are made in a backward-compatible manner.

In IETF terms, this versioning scheme provides functionality analogous to several parts of the traditional IETF process.

- MAJOR is analogous to an "obsoletes" relation between RFCs
- MINOR is analogous to an "updates" relation between RFCs
- PATCH is analogous to use of the RFC errata process

The more major a change to the specification, the more consensus is required. When the WG wants to make a MAJOR change to a structured specification, the specification MUST be converted into Internet-Draft format and run through the typical IETF consensus process. Every commit that is tagged with a MAJOR version change MUST also

have a tag of the form "RFC-XXXX" indicating the number of the RFC describing the change.

MINOR changes follow the same rules, except that the Area Director for the WG MAY approve the issuance of a minor version with only WG consensus, not full IETF consensus. Any change to the structured specification that significantly changes the security considerations for the protocol or requires additional IANA actions MUST be converted into Internet-Draft format and submitted for IETF consensus. With the approval of the AD for the WG, changes without such impacts MAY be approved by consensus of the working group.

PATCH-level changes MAY be made by the editors, with the consent of the WG chairs.

When a working group starts up work on a new version of the specification, regardless of whether it's a minor update or a complete rewrite, they SHOULD create a dedicated branch of the repository for the new version, where changes related to the new version will be committed. The semantic version, i.e. MAJOR and MINOR is assigned when this branch is merged back to the main specification. Once there is consensus to update the main specification to that version, the branch should be merged, and the merge commit tagged with the new version number.

When working on modification to an existing work item (typically a bis document), the process is to fork a git repo, branch, make a proposal, then push the branch back over to the Working Group when/if the Working Group adopts it.

## 2.1. Versioning for Work in Progress

It can be useful to mark certain versions of a work in progress as checkpoints, e.g., for reference at a hackathon. These checkpoints should follow their own version sequence, much like Internet drafts:

LABEL-VERSION

- LABEL is a string that identifies the feature branch

- VERSION is incremented whenever a new revision is tagged

These tags are analogous to Internet-Draft names. Much like an Internet-Draft name, the choice of LABEL values is up to the editors and WG chairs. For cases expected to result in a given version, they may choose to use that as a label value. For example, if the WG has agreed to embark on a major revision to the protocol, then they might

use the label "v2.0.0-beta", so that the working revisions would be "v2.0.0-beta-0", "v2.0.0-beta-1", etc.

It's important to note that not every commit needs a version. Much like working groups using Github to manage Internet-Drafts today only periodically submit them to the IETF, a WG can do work in a repository and only tag versions when they are useful to the WG. Beta versions should be tagged at key points in the development process:

- Before an IETF meeting or WG interim meeting
- Before a hackathon or interop event
- Before a working group or IETF last call

Work on a new version SHOULD be conducted on a dedicated branch. Once there is consensus to update the main specification to that version, the branch should be merged, and the merge commit tagged with the new version number.

## 3. IETF Consensus for Structured Specifications

While working groups may use any format for specifications under development, the Internet Standards process requires that a document proposed as an RFC must be submitted in the RFC format, i.e., as unstructured text. Proposed RFCs are also required to contain explanatory text not typically contained in structured specifications, most notably Security Considerations and IANA Considerations. Thus, a working group that is focused mainly on a structured specification will have to convert the structured specification to the RFC format and add the additional explanatory text.

In order to keep the repository as the primary home for the specification, the working group should keep any required explanatory text in the repository alongside the structured specification, and use automated tools to generate an RFC-formatted document from the artifacts in the repository. As the outputs are reviewed in the IETF last call and IESG processes, the editors should reflect their responses in the repository, generating updated versions of the RFC-formatted document as necessary.

Whenever an Internet-Draft is generated from the repository, the corresponding commit in the repository should be tagged with the full name and version of the Internet-Draft. Additionally, a reference to the repository (e.g., a URL) should exist in the text of the

resulting Internet-Draft. These steps enable the evolution of the draft to easily be tied back to the evolution of the repository.

## 4. Example history

The below sequence of commits and tags shows the progress of a structured specification through several stages of its life-cycle. (Time flows up from the bottom, as is common in version control logs; most recent is on top.)

An initial version of a protocol is proposed for a Birds of a Feather (BoF) and a WG is formed. The WG develops version 1.0.0 of the specification. Along the way, they tag betas when they need an easy way to refer to a version, e.g., before working group last call (WGLC).

Once the WG has reached consensus, an Internet-Draft is created from the repository (draft-ietf-wg-proto-00) and submitted for the IETF consensus process, resulting in an RFC (RFC XXX1) that describes the first version of the protocol. In this example, there is never a need to publish an individual (author-named) internet-draft, because the WG worked directly on the structured specification and obtained consensus on it.

Comments from the IETF last call (LC) and the IESG are incorporated in the repository, and new versions of the Internet-Draft are generated for IESG review and submission to the RFC editor.

```
...
|
* e3091df (tag:v1.0.0, tag:draft-ietf-wg-proto-02, tag:RFCXXX1)
|         Responses to IESG comments
|
* 7494725 (tag:draft-ietf-wg-proto-01) Responses to IETF LC comments
|
* 8e2be54 (tag:proto-2, tag:draft-ietf-wg-proto-00)
|         Responses to WGLC comments
|
* 9703a60 (tag:proto-1) Responses to comments at IETF meeting
|
* 2b83977 Responses to J. Smith comments
|
* 8b75e1e (tag:proto-0) Responses to BoF comments
|
* 1991498 Initial submission
```

The WG adds two features to the specification. The first feature is major enough that the chairs decide it needs IETF consensus,

resulting in a second Internet-Draft going through the IETF consensus process (draft-ietf-wg-proto-feature-00) to become an RFC (RFC XXX2). The second feature is minor enough that it can be approved by WG consensus.

```
...
|
*   a5f3214 (tag:v1.2.0) Merge branch 'v1.2'
|\
| * 8fb9cb6 Responses to WGLC comments on feature Y
| |
| * 39322e9 (tag:featureY-1) Responses to WG comments; ready for WGLC
| |
| * 39322e9 Add feature Y
|/
*   d1d201d (tag:v1.1.1) Fix validation errors
|
*   6571483 (tag:v1.1.0, tag:draft-ietf-wg-proto-feature-04,
|           tag:RFCXXX2) Merge branch 'v1.1'
|\
| * abc3f5e (tag:draft-ietf-wg-proto-feature-03) Resolution of DISCUSSes
| |         from Security AD
| |
| * 3ab54f3 (tag:draft-ietf-wg-proto-feature-02) Resolution of DISCUSSes
| |         from Internet and Transport ADs
| |
| * cabb1f6 (tag:draft-ietf-wg-proto-feature-01) Responses to WGLC
| |         and IETF LC comments on feature X
| |
| * fbfaa6b (tag:featureX-0, tag:draft-ietf-wg-proto-feature-00)
| |         Responses to comments on feature X
| |
| * 0630638 Add feature X
|/
* e3091df (tag:v1.0.0, tag:draft-ietf-wg-proto-02, tag:RFCXXX1)
|         Responses to IESG comments
|
...
```

The WG develops a major revision of the protocol, resulting in a third Internet-Draft (draft-ietf-wg-protobis-00) going through the IETF consensus process, resulting in RFC XXX3.

```
...
|
* a9d7d29 (tag:v2.0.0, tag:draft-ietf-wg-protobis-01 tag:RFCXXX3)
|         Merge branch 'v2'
|\
| * df5d437 (tag:draft-ietf-wg-protobis-00) Responses to
| |         WGLC and IETF LC comments
| |
| * 986ebb6 (tag:v2.0.0-beta-1) Checkpoint for hackathon
| |
| * d86986e (tag:v2.0.0-beta-0) Some more v2 features
| |
| * ca02154 Restructure for v2
|/
*   a5f3214 (tag:v1.2.0) Merge branch 'v1.2'
|
...
```

This example history is greatly simplified. In a real WG, there will be far more commits without versions, as the WG incorporates proposals, edits, explanatory text, etc. But this example highlights the key moments in the life-cycle of a specification.

## 5. Creating Internet-Drafts from a Repository

As noted above, WGs should have one repository per specification. Over the lifetime of the specification, it will be necessary for automated tools to build Internet-Drafts from this repository. A standardized repository layout can help automate this process.

The root level of the repository should have a file named "index.xml" or "index.md", depending on whether XML [1] or Markdown [2] is being used. This file should itself be a valid source for an Internet-Draft, including information about the title to be used, authors' / editors' names, security considerations, etc. It will act as a template into which the structured specification will be included when an Internet-Draft is created.

The template file should include files that comprise the structured specification as figures at appropriate points in the draft. The Markdown syntax provided by "mmark" provides a syntax for including code fragments [3] from external files, including the ability to selectively include parts of a file.

## 6. References

### 6.1. URIs

[1] https://tools.ietf.org/html/rfc7991

[2] https://github.com/miekg/mmark/wiki/Syntax

[3] https://github.com/miekg/mmark/wiki/Syntax#including-code-fragments

Authors' Addresses

Benoit Claise
Cisco

Email: bclaise@cisco.com

Richard Barnes
Cisco

Email: rlb@ipv.sx

Joe Clarke
Cisco

Email: jclarke@cisco.com