# hyper Documentation

***Release 1.0.0***

**Cory Benfield and Contributors**

**Jan 14, 2018**

# Contents

Hyper is a set of related projects that provide HTTP/2 functionality to Python projects. The aim is to provide a complete HTTP and HTTP/2 toolbox, ranging from complete off-the-shelf solutions to projects that solve individual specific problems. Developers should be able to compose these solutions together to fit their specific use-cases, using the general-purpose code where appropriate and writing code to well-defined interfaces where they have specific requirements.

The hyper project is comprised of the following sub-projects:

- hyper-h2, a complete HTTP/2 protocol stack that does not perform any I/O, intended to be independent of framework.
- hyperframe, a HTTP/2 framing layer.
- hpack, a HPACK implementation in pure-Python.
- brotlipy, a CFFI-based library for Brotli compression.
- priority, a Python implementation of the HTTP/2 Priority tree.
- wsproto, an implementation of the WebSocket protocol that, like hyper-h2, does not perform any I/O.

Please follow the links above for more details on each of those sub-projects. The rest of this documentation applies to the project as a whole.

CHAPTER 1

# Contents

## 1.1 Contributing

If you're reading this, you're probably interested in contributing to one of the Hyper projects! First, we'd like to say thank you! Projects like this one live-and-die based on the support they receive from others, and the fact that you're even considering supporting the Hyper project is incredibly generous of you.

This document lays out general guidelines and advice for contributing to any of the Hyper projects. If you're thinking of helping, start by reading this thoroughly and getting a feel for how the project works. If you've still got questions after reading this, please ask us in IRC: we'd be happy to clarify any questions you may have.

We've organized this guide into sections:

- our code of conduct, which applies to all contributions of any kind
- general guidelines for all contributors
- specific information based on the type of contribution you're thinking of making
  - *Code Contributions*
  - *Documentation Contributions*
  - *Bug Reports*
  - *Feature Requests*

### 1.1.1 Code Of Conduct

The following code of conduct applies to all projects that are under the umbrella of the hyper project.

#### Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body

size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

### Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

### Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

### Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

### Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team by email. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

#### Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available here.

### 1.1.2 All Contributions

#### Get Early Feedback

If you are contributing, do not feel the need to sit on your contribution until it is perfectly polished and complete. It helps everyone involved for you to seek feedback as early as you possibly can. Submitting an early, unfinished version of your contribution for feedback in no way prejudices your chances of getting that contribution accepted, and can save you from putting a lot of work into a contribution that is not suitable for the project.

#### Contribution Suitability

The maintainers of a sub-project have the last word on whether or not a contribution is suitable for that project. We consider all contributions, but from time to time contributions will be rejected because they do not suit the project.

If necessary, a contributor should feel free to ask maintainers of other hyper projects to adjudicate, especially if they feel their contribution is being unfairly discriminated against.

If your contribution is rejected, don't despair! So long as you followed these guidelines, you'll have a much better chance of getting your next contribution accepted.

### 1.1.3 Code Contributions

When contributing code, you'll want to follow this checklist:

1. Fork the repository for the project in question on GitHub.
2. Run the tests to confirm they all pass on your system. If they don't, you'll need to investigate why they fail. If you're unable to diagnose this yourself, raise it as a bug report by following the guidelines in this document: *Bug Reports*.
3. Write tests that demonstrate your bug or feature. *Ensure that they fail*.
4. Make your change.
5. Run the entire test suite again, confirming that all tests pass * including the ones you just added*.
6. Send a GitHub Pull Request containing your changes. GitHub Pull Requests are the expected method of code collaboration on this project. If you object to the GitHub workflow, you may mail a patch to one of the project maintainers.

The following sub-sections go into more detail on some of the points above.

#### Tests & Code Coverage

All Hyper projects have a substantial suite of tests, both unit tests and integration tests, and have 100% code and branch coverage. Whenever you contribute, you must write tests that exercise your contributed code, and you must not regress the code coverage.

To run the tests, you need to install tox. Once you have, you can run the tests against all supported platforms by simply executing `tox`.

If you're having trouble running the tests, please consider raising a bug report using the guidelines in the *Bug Reports* section.

If you've done this but want to get contributing right away, you can take advantage of the fact that all Hyper projects use a continuous integration system. This system will automatically run the tests against any pull request raised against any Hyper repository. The continuous integration system treats a regression in code coverage as a failure of the test suite.

Before a contribution is merged, it must have a green run through the CI system.

#### Code Review

Contributions will not be merged until they've been code reviewed. You should implement any code review feedback unless you strongly object to it. If you object to the code review feedback, you should make your case clearly and calmly. If, after doing so, the feedback is judged to apply still, you must either incorporate the feedback or withdraw your contribution.

#### New Contributors

If you are new or relatively new to Open Source, welcome! The Hyper project aims to be a gentle introduction to the world of Open Source. If you're concerned about how best to contribute, please consider mailing a maintainer or asking for help on IRC.

Please also check the *Get Early Feedback* section.

#### Security

We have a security policy we take very seriously. Please read *Security* for more details.

### 1.1.4 Documentation Contributions

Documentation improvements are always welcome! The documentation files for individual projects live in the `docs/` directory of the codebase for that project, and the general documentation for the project as a whole live in the documentation repository. They're written in reStructuredText, and use Sphinx to generate the full suite of documentation.

When contributing documentation, please attempt to follow the style of the documentation files. Use a soft-limit of 79 characters wide in your text files and a semi-formal prose style.

### 1.1.5 Bug Reports

Bug reports are hugely important! Before you raise one, though, please check through the GitHub issues for that project, **both open and closed**, to confirm that the bug hasn't been reported before. Duplicate bug reports are a huge drain on the time of other contributors, and should be avoided as much as possible.

### 1.1.6 Feature Requests

Feature requests are always welcome, but please note that all the general guidelines for contribution apply. Also note that the importance of a feature request *without* an associated Pull Request is always lower than the importance of one *with* an associated Pull Request: code is more valuable than ideas.

## 1.2 One Of The Team

The Hyper project is interested in fostering as much community as possible, to encourage people interested in HTTP and HTTP/2 to get involved and get their hands dirty.

For that reason, the Hyper project has a policy: if you have made a commit to any of the Hyper sub-projects, you are automatically entitled to be made a member of the GitHub organisation: specifically, the contributors team. This team is empowered to create branches, merge pull requests, close bug reports, and generally to get involved with the ongoing maintenance of all of the hyper projects.

Membership of this team is *not mandatory*: if you're not interested in being a part of the project, that's totally ok! Additionally, if you become a member but later decide you aren't interested, that's fine too: you don't owe us any of your time.

However, if you want a corner of the open source ecosystem to feel at home, we want you to feel that way here. You should get as involved as you feel comfortable with.

### 1.2.1 Responsibility

Becoming a contributor grants you quite a lot of power: in particular, you can merge pull requests and close bug reports. These can potentially give mischievous or malicious individuals the ability to cause a great deal of annoyance for everyone else.

That's not a reason not to share these powers: we believe our community is full of reasonable people who care about working well together. However, it does mean we need to provide *some* guidance about how to work together.

Here is what we expect of our contributors:

- Please don't merge pull requests unless a maintainer or administrator has OK'd the change. We want to work as a team, but the maintainers are responsible for the 'spirit' of the project, and often have a better understanding of its needs than new contributors.
- Please don't merge your own pull requests. Code review is important, and skipping it is bad.
- You need to follow our code of conduct, as outlined in our *Contributing* policy.

Contributors who repeatedly fail to meet these three points will have their contribution privileges removed. The goal here is to build a team of people who work together to a common end: people who aren't interested in that will need to find a home elsewhere.

### 1.2.2 Maintainers

If you establish a record of valuable contribution and an understanding of the workflow involved with the Hyper project, then you may be made a *sub-project maintainer*. These members have additional authority over their sub-projects: specifically, they are able to push directly to the master branch and to manage the permissions of other contributors to their repository.

Maintainers have more authority to go along with their record of contribution, and also have more responsibility for the continued good health of their project. Good maintainership is a vital part of the ongoing health of these sub-projects: it's the most important role in any open source project.

Maintainers are expected to be familiar with our *Contributing* policy and our *Security* policy, as they'll be responsible for enforcing those policies. They're also responsible for ensuring that as many of the contributors as possible feel like they're making important contributions. Maintainers are stewards of culture, as much as they are of code.

## 1.3 Security

### 1.3.1 Our Goals

When it comes to security issues, the Hyper project has one goal: to be tough and competent.

Security issues have real effects on real users, and it is the duty of library authors to protect users from those problems as much as possible. As a result, security is a paramount focus for our project, because if we fail and users are exploited or made vulnerable, we are to blame.

This attitude leads to the following guidelines for contributors and maintainers:

- The default configuration of a Hyper project **must** be as secure as possible.
- Any change that weakens the security of the default configuration of a Hyper project is to be resisted.
- We will react swiftly to being notified of security vulnerabilities.
- When notified of security vulnerabilities we will treat them all as serious incidents until otherwise determined: *there is no such thing as a low severity security bug*.
- We will work with downstream maintainers to backport patches to all versions of the software that are affected by a vulnerability, regardless of whether that version is still actively supported by the core team.
- It is better to break functioning-but-insecure code than it is to leave unaware users vulnerable.
- If and when we break functioning-but-insecure code, we will provide error messages that adequately explain the problem.
- Any insecure or less secure options made available in a Hyper project must cause warnings to be issued when used.

### 1.3.2 Known Vulnerabilities

This section of the page contains all known vulnerabilities in all libraries that make up the Python Hyper project. These vulnerabilities have all been reported to us via our *vulnerability disclosure policy*.

| # | Vulnerability | Date Announced | Library | First Version | Last Version | CVE |
|---|---|---|---|---|---|---|
| 2 | HPACK Bomb | 2016-08-04 | HPACK | 1.0.0 | 2.2.0 | CVE-2016-6581 |
| 1 | DoS via unlimited stream insertion | 2016-08-04 | Priority | 1.0.0 | 1.1.1 | CVE-2016-6580 |

### 1.3.3 Vulnerability Disclosure

If you think you have found a potential security vulnerability in a Hyper project, please email Lukasa directly. **Do not file a public issue.**

Lukasa's PGP Key fingerprint is:

- 90DC AE40 FEA7 4B14 9B70 662D F25F 2144 EEC1 373D

If English is not your first language, please try to describe the problem and its impact to the best of your ability. For greater detail, please use your native language and we will try our best to translate it using online services.

Please also include the code you used to find the problem and the shortest amount of code necessary to reproduce it.

Please do not disclose this to anyone else. We will retrieve a CVE identifier if necessary and give you full credit under whatever name or alias you provide. We will only request an identifier when we have a fix and can publish it in a release.

We will respect your privacy and will only publicize your involvement if you grant us permission.

### Process

This following information discusses the process the Hyper project follows in response to vulnerability disclosures. If you are disclosing a vulnerability, this section of the documentation lets you know how we will respond to your disclosure.

### Timeline

When you report an issue, one of the project members will respond to you within two days at the outside. In most cases responses will be faster, usually within 12 hours. This initial response will at the very least confirm receipt of the report.

If we were able to rapidly reproduce the issue, the initial response will also contain confirmation of the issue. If we are not, we will often ask for more information about the reproduction scenario.

Our goal is to have a fix for any vulnerability released within two weeks of the initial disclosure. This may potentially involve shipping an interim release that simply disables function while a more mature fix can be prepared, but will in the vast majority of cases mean shipping a complete release as soon as possible.

Throughout the fix process we will keep you up to speed with how the fix is progressing. Once the fix is prepared, we will notify you that we believe we have a fix. Often we will ask you to confirm the fix resolves the problem in your environment, especially if we are not confident of our reproduction scenario.

At this point, we will prepare for the release. We will obtain a CVE number if one is required, providing you with full credit for the discovery. We will also decide on a planned release date, and let you know when it is. This release date will *always* be on a weekday.

At this point we will reach out to our major downstream packagers to notify them of an impending security-related patch so they can make arrangements. In addition, these packagers will be provided with the intended patch ahead of time, to ensure that they are able to promptly release their downstream packages. Currently the list of people we actively contact *ahead of a public release* is empty. We will notify these individuals at least a week ahead of our planned release date to ensure that they have sufficient time to prepare. If you believe you should be on this list, please let one of the maintainers know at one of the email addresses at the top of this article.

On release day, we will push the patch to our public repository, along with an updated changelog that describes the issue and credits you. We will then issue a PyPI release containing the patch.

At this point, we will publicise the release. This will involve mails to mailing lists, Tweets, and all other communication mechanisms available to the core team.

We will also explicitly mention which commits contain the fix to make it easier for other distributors and users to easily patch their own versions of the project if upgrading is not an option. We will also work with downstream distributors to patch their software.