How to Contribute

Contents

- Contents
- Reporting an Issue
- Privately Reporting a Security Vulnerability
- Contributing Code
    - When contributing a bug fix
    - When contributing a new feature
    - Best practices when creating Pull Requests
    - Doing a Code Review
- Collaborating in Jira
- Becoming a Maintainer
- Continuous Integration / Continuous Delivery
- DCO

Reporting an Issue

Backlog items for Hyperledger Indy projects are tracked in the Hyperledger Jira instance. To create, edit, or comment, you need a Linux Foundation ID.

A good bug report is easy for others to understand. If they cannot reproduce your issue, they are unlikely to be able to address it.

1. Before submitting an issue, search to see if someone has already reported something similar. If you find an unresolved duplicate issue, add to it any additional details you think can help.
2. Your bug report should include sections such as:
    a. *Environment*: The environment where you saw the issue.
        i. The operating system, where you obtained your software, how you installed it, etc
        ii. Set the Jira field "Affects Version" to the version of the software you were running when you saw the issue. (Leave the "Fix Version" field alone until the code is done and assigned to a release.)
    b. *Steps to Reproduce*: step-by-step explanation of how to cause the problem from a clean installation of the software.
    c. *Expected Behavior*: The desired behavior that you expected to see.
    d. *Observed Behavior*: The problematic behavior that you saw.
    e. *Notes*: Any notes that you think would help the person investigating the issue.

Important notes:

- You may need to be patient waiting for someone to investigate your issue. Contributors to Hyperledger Indy have their own priorities. Unless you have paid someone to assist you, remember that anyone looking at your issue is a volunteer and you should be grateful for their assistance.
- If someone seriously tries to reproduce your issue and cannot, they will close it with the resolution "Cannot Reproduce". This indicates that the problem is likely specific to your system. You can find the solution yourself, contract with someone for assistance, or reopen the issue and add further details. Please do not reopen an issue if you don't have new information that can help the investigation.
- If you are seeing a problem that appears to be a duplicate of a fixed issue, it is generally better to create a new issue and link to the old one instead of reopening the old issue.

Privately Reporting a Security Vulnerability

If you identify an exploitable security vulnerability in an Indy project, please responsibly disclose it in private so that we have the opportunity to address it before public disclosure.

Please report using HackerOne or by sending an email to the team at security@sovrin.org.

Contributing Code

When contributing a bug fix

1. It is preferable to start with a Jira issue, as other might contribute guidance on the best way to diagnose and fix the problem.
2. Then submit a Pull Request in GitHub with your suggested changes.
    a. Don't forget to include a unit test which reproduces the problem and is fixed by the PR.
    b. Reference the associated Jira issue in the PR title.
    c. Sign your commit to provide an attestation of the How to Contribute#DCO.
3. Wait
    a. CI will be run on your PR.
    b. Once CI passes, a maintainer will review your PR.
    c. If the maintainer asks for changes, you will need to respond.
    d. Once the maintainer is happy, the PR will be accepted and included in the next release.

When contributing a new feature

1. It is preferable for you to start with a Jira issue before putting in significant effort.
    a. Get feedback on whether the community is interested in the feature.
    b. Solicit ideas on the best way to approach the implementation.
2. Consider submitting a HIPE
    a. If the change impacts more than one project, then a Hyperledger Indy Project Enhancement needs to be discussed.
3. Submit a Pull Request in GitHub with your architecture plan to the design folder of the repository you are changing.
    a. The architects and maintainers will give you feedback and suggestions when accepting the design.
    b. The design then becomes part of the developer documentation.
4. Submit a Pull Request in GitHub with your changes.
    a. Sign your commit to provide an attestation of the How to Contribute#DCO.
    b. Don't forget to update all relevant tests, changes to CI / CD, documentation, translations, etc.
        i. API changes need to be reflected in all layers of the system within the repository you are updating. For example, LibIndy API improvements have to be reflected in the CLI and language bindings.
5. Wait
    a. CI will be run on your PR.
    b. Once CI passes, a maintainer will review your PR.
    c. If the maintainer asks for changes, you will need to respond.
    d. Once the maintainer is happy, the PR will be accepted and included in the next release.

Best practices when creating Pull Requests

1. Make a private fork of the repository you want to contribute to.
2. Assign the related Jira ticket to yourself and set it to "In Progress" so that others know you are working on it.
3. Write good code:
    a. We prefer Test Driven Development.
    b. Unit tests are required. Integration tests are encouraged.
    c. Pay attention to the code standards documented in the repository where you are contributing.
        i. Observe best practices and style guidelines in the repository's "Coding-Conventions.md" file.
        ii. Write idiomatic code for the language you're using (e.g., PEP8 for python)
        iii. Respect other stylistic and design choices until you build enough credibility to influence the project as a whole.
    d. Understand the Apache 2 license: don't contribute anything encumbered by copyright or patent issues.
    e. Don't add new dependencies without consulting a maintainer.
    f. The CI pipeline is configured to fail on compiler warnings.
4. Confirm that all tests pass.
5. Don't mix unrelated improvements in the same Pull Requset.
6. Submit a Pull Request to the master branch of the upstream repository.
7. Move the Jira ticket to "Code Review" status.
8. Announce your PR on #indy-pr-review at chat.hyperledger.com (give the PR URL and Jira reference).
    a. The reviewer should assign the ticket to themselves.
    b. Especially engage the maintainers if the change is big, significant, or risky.
9. Monitor the pull request for comments from the reviewer.
10. Be patient.
    a. Our goal is to not let pull requests go for a week without some type of follow up from the maintainers.

Other best practices:

- Communicate proactively and pragmatically.
- Be informed before you ask questions (wiki, chat, design docs, jira).
- Contribute a lot, and recruit your friends!
- Feel free to tout your contributions on LinkedIn, resumes, and so forth.

Doing a Code Review

When doing a code review, ensure to check:

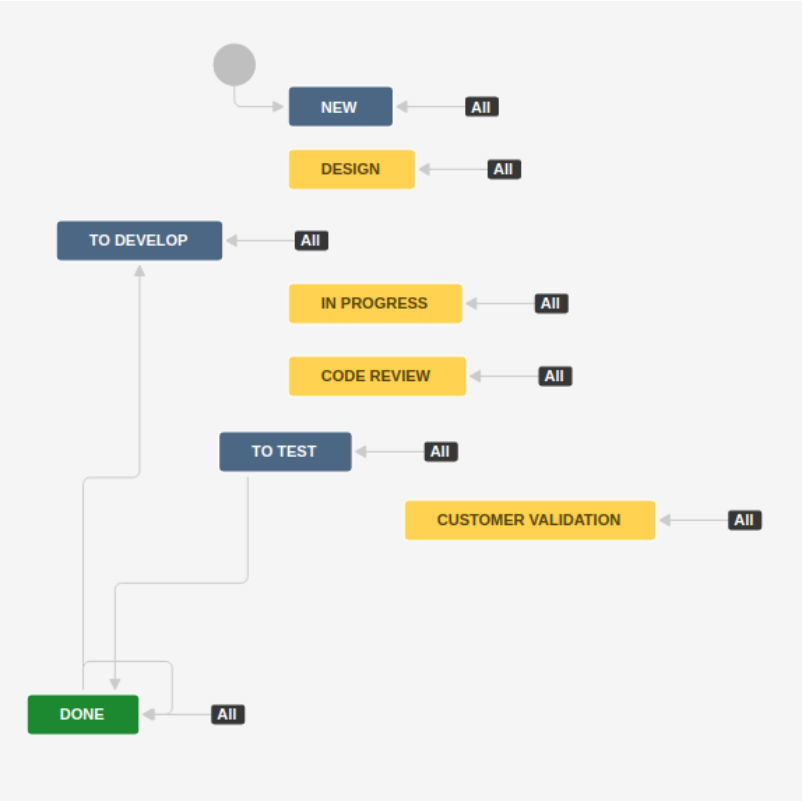
- Functionality: Does the new code work without breaking anything? Is it backwards compatible?
- Tests: Do the new tests pass? Are there enough tests for the right things?
- Quality: Is the code clean, well encapsulated, simple, and conformant to the style guidelines for the repository? Does it make wise choices about refactoring?
- Safety: Does the code avoid any new dependencies or IP encumbrances?

1. Use GitHub to give feedback to the submitter. Be grateful, encouraging, and helpful, but also don't be afraid to ask for improvements and enforce a high standard.
2. If your review is favorable, assign the Jira ticket to a maintainer with a note that you think it is ready to merge.
3. Optionally, ping the maintainer and submitter on chat with a status update.

When in doubt, escalate your question to maintainers on chat, on the mailing list, or on a working group call. Maintainers can then coordinate the necessary conversations to resolve the issue.

Collaborating in Jira

We use Jira to collaborate on the backlog for various Jira projects. Each Indy project shares a common workflow:



In issue moves through the following statuses:

- Issues are created as New.
- There is an optional Design step if we want to spend some time on the UX and architecture before development.
- When someone decides the issue is valuable, clear, and ready for a developer to start work, it is transitioned to the status "To Develop".
- When development work starts, the issue should be marked as "In Progress".
- When development work is done, and the PR is ready for review, it should be moved to "Code Review".
- When the PR has been merged, the issue should be moved to "To Test" where a different developer should pick it up and confirm that it works.
- There is an optional "Customer Validation" step that can be used to ensure the requirements or solution meets the customer's needs.
- When the issue is complete, it should be transitioned to "Done".

Important notes on Transitions:

- Any state can be transitioned to any other state.
- When transitioning an issue to Done, a resolution must be set indicating why the issue is considered "Done". It could be "Done", "Duplicate", "Won't Do", "Invalid", etc.
- When transitioning from "To Test" to "Done", use the transition "Done as fixed" to automatically set the resolution to "Done".
- An issue that is "Done" can be moved back to "To Develop" using the transition "Reopen" which will clear the resolution.
- The transition "Fix Resolution" is used to change the resolution of a completed issue without changing its status away from "Done".

Best practices:

- It is best practice to include a comment every time you touch an issue indicating to others what you did and why.
- Each team working in Jira should create a "sprint" where they claim tickets that they have in progress.
- Each time might also create an additional sprint containing a handful of tickets they want to work on next.
- The label "help-wanted" should be applied to tickets that are good for other people assist with.

Becoming a Maintainer

A maintainer can merge pull requests in GitHub. We need more maintainers of Hyperledger Indy!

- Maintainers are expected to be well informed, unselfish, accessible, and good (proactive) communicators.
- Multiple maintainers are encouraged for timezone convenience and to avoid risks during staff transitions.
- The project's maintainers and other key contributors are identified a file called Maintainers.md.
- Any folder can have a Maintainers.md specific to that folder.
- Recognize that many parts of a project have shared scope and collaboration will be necessary among all maintainers to keep it in good shape.

To become a maintainer:

1. Select a code base that you are interested in helping to improve.
2. Learn the code base.

3. Attend the relevant working group sessions, read the mailing list threads, and collaborate with other developers at chat.hyperledger.com.
4. Regularly submit pull requests, and work with the existing maintainers enough to have them accepted.
5. Ask to become a maintainer.

A maintainer has to contribute to more than their own priorities. To remain a maintainer, one must:

- Demonstrate that one is collaborating with the other maintainers by attending the relevant working group sessions, reading the mailing list threads, and helping other developers at chat.hyperledger.com.
- Review the pull requests of others, provide feedback, and accept them when they are at a high enough quality.
- Propose architecture improvements as designs and HIPES, and help improve the proposals of others.
- Proactively reach out to other maintainers to ensure a common unified vision for the project.
- Contribute to documentation, release testing, integration testing, maintenance of CI / CD, triage of community submitted Jira issues, and other release management tasks.
- Smoothly transition out of maintainership when necessary by informing the other maintainers of the project.

Continuous Integration / Continuous Delivery

- A build is “broken” if any automated processes (compile, package, test, etc) fail.
- Anybody can institute automated build processes, but only ones endorsed by maintainers are normative.
- A broken build in master is an urgent team emergency, because it halts all productive work by others. Whoever discovers the broken build should report to @all on #indy-build immediately, and claim ownership or find an owner.
- Maintainers fix breaks in their scope; the default fix is to revert a problematic change.
- Submitters should plan to be available to maintainers soon after a PR is merged, so they can help with troubleshooting. They should proactively debug and reach out to maintainers if they notice an issue.
- Build logs will be made available.

DCO

All commits to Hyperledger Projects must attest to the Developer Certificate of Origin. Enforcement is done by the DCO App.