Upstreaming

Contents

Upstreaming changes made to a piece of Open Source software provides distinct advantages for the author of the changes and the ecosystem of users of software. The author can expect to see:

- Reduced overhead in on-going maintenance of their code base: With the changes available in the upstream version, the author will no longer need to port changes when upgrading to a newer version of the software.
- Reduced risk of incompatible changes and/or features being added to the upstream code base: When making local modifications to a code base, there is a risk that at some future point any local changes will fail to apply without significant changes or a comparable feature will be implemented with different semantics requiring either conversion to this feature to be carried out or continuing to carry the local change with much reduced advantages.
- Greater review of the proposed changes: Open source projects tend to review changes before they are merged and whilst not perfect, such reviews tend to be carried out by developers with a good working knowledge of the code base, which results in a better review than would be achieved in many settings.
- Potentially increased testing of added features: Other users of the software either evaluating or using the upstreamed features may uncover bugs or security holes in the added features that might otherwise go missed. This results in increased robustness for your users.
- Potential for further complementary features being added: The addition of a feature may prompt other developers to add complimentary features that prove useful to either you or your users.

The upstream project obviously benefits from the addition of features as this makes the software appealing to a wider audience and thus is likely to increase adoption. Having an active community may also help to increase adoption in its own right.

Whilst there is obviously an advantages for upstream projects to accept new features, it is also important that they thoroughly review and consider such changes. Bad changes could lead to instability of the application, negatively impacting all users of the project. Additionally the maintainers[1] are taking on the task of providing some maintenance for the added features. It is thus important that they ensure such changes comply with the coding conventions and other polices to ensure ease of maintenance and continued good health of the project.

[1] https://jwd.pages.apertis.org/apertis-website/policies/contributions/#the-role-of-maintainers

Where to upstream

As mentioned elsewhere, Apertis is a derivative of Debian, which it's self packages software projects from many sources outside of Debian its self. Depending on the changes being made, this may present 3 options as to where to upstream the changes:

- Apertis
- Debian
- Main project of the software component

The effort required to get changes accepted by each of these places and the associated delay in seeing your changes reach a released version of Apertis are going to differ, often quite drastically and is frequently inversely proportional to the on-going maintenance costs.

As a user of Apertis, it is likely that submitting changes to Apertis may offer the lowest barrier to entry and fastest route to see the changes reflected in an Apertis release. There may be instances where Apertis offers the only real option, for example where Apertis is maintaining an older version of the project for licensing reasons.

It is likely that Debian will only accept very limited types of change. Some changes such as security and bug fixes may be more viable for upstreaming to Debian, as a general rule feature additions may be less likely to be accepted, though this will depend on how "alive" the upstream project is, what the feature is and how the maintainer feels about it (after all, the maintainer will be taking on the burden of maintaining the patches). Any patches that are accepted by Debian may take longer to be picked up by Apertis (depending on exactly where the changes landed).

The last option is to submit changes directly to the upstream project. Clearly packaging related changes can't be submitted here as these are not generally handled by the upstream project. It is also possible that the version used in the current Apertis is behind upstream development branch because Apertis prioritizes stability over new features. However the upstream development branch is where changes would need to be submitted, even when using this branch incurs additional effort to port and test. The advantage to submitting to the upstream project is that the changes will require no further ongoing porting to newer versions as it will be in the main code base.

In order to alleviate the significant delay that a user of Apertis is likely to see between upstreaming to either Debian or upstream projects and the changes appearing in Apertis, it is very likely that Apertis would be willing to accept backported upstream changes to the version currently in use. This provides the user with the advantage of being able to immediately use the functionality without needing to carry local changes, whilst the Apertis developers can expect to only need to carry the changes in the short to medium term until the changes filter through.

What can be upstreamed

Most systems are comprised of parts that exist to provide a relatively standard environment that are likely to be (or could be) common to many devices using similar hardware or requiring similar functionality and parts that provide some kind of unique experience or custom logic specific to the device in question. It is the parts that form the standard environment where the advantages of upstreaming are most commonly exploited as these are the parts most likely to benefit others and which benefit the most from others usage.

Examples of items that would be prime candidates for upstreaming include:

- Drivers for publicly available devices, including peripheral devices and architectures/SoCs previously not supported
- Previously unsupported functionality in provided by supported devices
- Extending functionality for widely usable use cases in user space libraries and applications
- Bug fixes and security fixes for any upstream component

Ideally, such additions would be submitted to the main project in the first instance (with a backport submitted to Apertis once accepted upstream). Should upstream submission fail such patches would be considered on a case-by-case basis for addition into Apertis.

Note: Upstreaming is generally a process best considered from the outset. If upstreaming is planned at an early stage consider actively working with the community[2] during development, as this may streamline and simplify the development and upstreaming process.

Modifications and functionality that are not suitable for upstreaming will be considered on a case by case basis[3]. Whether they will be considered suitable for integration into the main Apertis repositories will depend in part on how broad the usefulness of the changes will be with the Apertis user base. At a minimum it would be necessary for such changes to comply with the coding standards of the relevant package, not impact the operation of Apertis for other users and be suitably licensed.

[2] https://jwd.pages.apertis.org/apertis-website/policies/contributions/#upstream-early-upstream-often

[3] https://jwd.pages.apertis.org/apertis-website/policies/contributions/#adding-components-to-apertis