Integrating third party open source libraries, source code, and assets

Totara products are built by developing and using open source technology. We use numerous third party libraries, code sources, and assets within our products both for production and development processes.

When implementing solutions we frequently review coding solutions from third party solutions for their suitability to our task and wider product offering. For example, if we had a need to produce PDF documents for export from within our platform we would look for third party PHP libraries that had supporting communities behind them, were actively maintained, and open to contribution. If some situations the viability of a new feature or improvement may depend upon our ability to find a suitable library.

The following policy framework outlines what is involved in reviewing and choosing a third party library for inclusion within Totara products.

Assessing suitability

There are several criteria that must be met when selecting a suitable third party library.

License

The license MUST be compatible with GPLv3 or greater for distribution purposes.
If the license is not compatible for distribution With Totara products then under no circumstances can the library or source code be used.

The following licenses are known to be compatible and can be included in core:

- Apache 2.0
- BSD 2-clause
- BSD 3-clause
- LGPL 2.1+
- LGPL 3
- GPL 2.0+
- GPL 3.0+
- MIT
- PHP 2.0
- PHP 3.0
- ISC

In addition there are licenses that may be usable depending upon how the items they license are used:

- CC BY 4.0
- CC BY-SA 4.0
- CCO 1.0
- SIL OFL 1.1

Regardless of the license please seek advice from the Director of Product Development, the Product Manager, or the CEO.

Licenses of included works

It is common for third party libraries to include code and libraries from other sources, and it is not a given that they are compliant.
The code base of the third party library being included must be reviewed in its entirety and the licenses for all secondary third party solutions need to be confirmed and noted.

Suitability to task

We want to choose a library that does the job we require. This means looking at the primary requirements for such a solution as per the task.
However it is also very important that the general requirements of all features and improvements destined for Totara products are observed.
The following questions should be asked of every library, source code, or asset being considered:

- Does it meet all of our needs for the project in question? If not what are we compromising?
- Are there any other projects in the organisation that have needs to a library of this type, if so does this meet their requirements?
  We don't want multiple libraries doing similar things in our code base. It may be unavoidable in some situations, however where we can we want to minimise the number of libraries we include.
- Is the user experience this library produces accessible?

- Does the user experience this library produces meet our design standards?
- Can the user experience be altered to meet our design guidelines?
- Can the user experience be extended and customised by our partners and subscribers?

Quality of code

We support the software that we produce, along with third party libraries that we incorporate.
Quality and security are of paramount importance. It must be stable, maintainable, secure and perform to requirements.
The following questions should be asked of every library being considered:

- Is the code being actively maintained?
- Is there evidence of both consistent and recent maintenance?
  Is the project alive and well, does it have a good heartbeat.
- Are there multiple contributors to the source code?
  A single developer may produce great code, but if they are the only person involved on the project then there is no guarantee of future maintenance.
- Is there evidence of support either by the maintainers or the community?
- Is the third party project owned an umbrella or hub organisation with known standards and quality control, for example: Apache, OpenJS foundation, PHPLeague?
- Is there good documentation or reference material?
- Are other large projects using the library?
- Is the feedback from others good?
- Do they make stable releases or is it progressive only?
- Do they clearly and accurately separate the changes to core within their version control system?
- Is there a published contribution guidelines or coding standard?
- Is there a published security issue policy?
- Does the supported environment of the library or source code align with that of our product?

In addition to these a code review should be completed, with emphasis on the following:

- Is the code readable?
- Is the code secure?
- Will the code scale and perform?
- Is the library configurable and/or extendable?
- Are there unit tests?
- Are there acceptance tests?
- Is the structure of the code isolated such that integration is possible?
- Could we maintain the code going forward if we had to?

Integrating the library or source code

All third party libraries and source codes being included within our product need to be carefully integrated.
As well as following our standard development workflow the following conditions should be observed:

- The library or source code must be imported into new isolated directory either within the platform if usage will be shared, or directly within a plugin if usage is isolated.
  The location of this directory will depend upon the nature of the third party code. For example JavaScript will be imported and wrapped into an AMD module, back end libraries into the lib or classes directory depending upon their structure and organisation.
- The library must be wrapped and usage of the library should occur through the wrapper.
  This is to ensure that should we need to replace the library or handle for API changes outside of our usual deprecate policy then we can do so directly in our wrapper using maps and transmutation. The wrapper should be auto-loadable, and should include any further auto-loading definition required for the library.
- An entry for the library must be added to a thirdpartylibs.xml file. If the plugin is in core this is lib/thirdpartylibs.xml, otherwise it is thirdpartylibs.xml directly in the plugin directory.
  This entry must name the library, provide a link to the website for the library, record the version of the library that has been imported, and record the license of the library that has been imported.
- A readme_totara.md file should be added to the main directory of the plugin.
  It needs to describe the following:
  - The name of the library/source code
  - Where the source code can be found.
  - Any files we have added, including but not limited to the readme file itself, and any wrappers.
  - Any changes that we have made to the library, either to fix issues, or enhance the library.
    This should tracking numbers.
  - Instructions on importing and updating the library in the future.
- Unit tests for the wrapper and the functions of the library or source code that are exposed to our product.
- Automated acceptance tests for any user experience the library or source code exposes.

Once the implementation has been created, and in addition to the standard development workflow requirements, a further review from the director of product development or lead developer must be sought and an approval provided.

Updating and maintaining third party code

Any updates to libraries that are made must adhere to our strict change and deprecation guidelines.
Libraries for the next upcoming release can be upgraded any time before release, but must provide backwards compatibility and deprecation via the wrapper.

Once a library has been released in a stable release the API of that library can not change, nor can the requirements or behaviour of the library or source code.
As such as we must separate improvements and new features from bug fixes, the same way we do for our own product.

We do not upgrade libraries in stable branches unless there is a need, either new functionality or improvements we require, or security / bug fixes made to the library since we last imported.
Some libraries make this easy by producing security or stable releases akin to our own. If not, then working issue by issue, change by change may be required.

When updating or making changes to a library the following must be done:

- Review the license, and version of, to ensure that it is still compliant for use within the Totara product.
- Update the thirdpartylibs.xml file to represent the correct version, license, and license version.
- Update readme_totara.md noting the new version, additions, and changes made.

Contributions back to the maintaining community

While integrating and support third party libraries within Totara products it is not uncommon to encounter bugs that we must fix, or areas requiring code or feature improvements.
Where possible these should be shared back with the community maintaining the library or source code, for their consideration and hopefully inclusion into the source code.

This benefits us in two ways, 1) more eyes on the code we are adding or fixing, helping to ensure the quality of Totara products, and 2) if accepted means that we do not need to support the customisation into the future.

When contributing fixes the developer should fork the third party source code in their chosen version control system. Make the changes required, including adding automated tests if possible, and then either produce a patch or create a pull request to share the contribution.

At no time should Totara specific functionality be involved -, it is important that contributions are limited to just that community's own codebase.