Totara Learning Solutions
Level 3, 150 Willis Street
PO Box 11-630
Wellington 6011

t: +64 4 803 2410
e: info@totaralms.com
w: www.totaralms.com

Contributing Code to Totara LMS

A Guide for Totara Partners

Introduction

As an open source codebase, Totara LMS's many advantages for Partners and Subscribers include the freedom to customise the platform. Within the Totara core team, we intentionally design and develop the platform to make it as flexible and adaptable as possible, and we encourage you to customise the platform to truly meet your customers' needs.

We also recognise that many organisations have common needs for functional improvements, and that the Totara team cannot always develop all requested improvements in the desired timeframes.

When appropriate, we look to collaborate with our community of Partners and Subscribers to help fast-track enhancements whenever possible, for the benefit of all in the Totara community. In fact, many new features, extensions, and improvements in Totara LMS originate from code contributions from Partners and Subscribers.

In most cases, Partners solely undertake the development reflecting the needs of one or more of their Totara customers. In some instances, Subscribers are able to tackle the development on their own. Either way, when the enhancements are complex and/or affect core architecture, the Totara team is usually directly involved in the process. The purpose of this document is to provide guidance to Partners undertaking development intended for contribution to the core product.

Benefits of Collaboration

The benefits of collaborating with Totara's R&D team are many;

- **Financial Rebates:** Totara Learning Solutions has an incentive mechanism for rewarding code contributions through providing credits agreed with the contributing party.
- **Easier upgrade path for your customers:** Contributions that are incorporated into the Totara product results avoids costly code conflicts when upgrading to future versions.
- **Supported code:** Once merged into core, your code is supported by the Totara R&D team. Any subsequent bug fixes will be provided as part of Totara's standard support agreement.
- **Code ported to future Totara versions:** Ongoing support includes porting your code to new Totara releases. When a new version of Totara is released, your contribution will be updated to work with the new version by the Totara team.
- **Code quality checks:** The Totara R&D team has a robust review and testing process. All code is reviewed to ensure security, stability, scalability, and cross-platform functionality.
- **Ongoing improvements:** Existing features are regularly extended or improved, either by the Totara team or via Partners and Subscribers. This means your feature will likely gain further enhancements for free.

- **Language translations:** Totara Learning Solutions has a team of translators who provide language packs in a wide range of languages. As part of the core product, your contribution will be automatically translated into all supported languages.
- **Everyone benefits:** Your contribution improves Totara LMS for all Subscribers. Similarly contributions from others benefit your use of Totara. Together, we can accelerate the development of Totara to the benefit of all users.

There are some aspects to effective collaboration you need to consider when planning to contribute your code to the core product:

- **Engagement with the Totara development team:** Rather than just developing the feature on your own timeframes, it is necessary to allow time in your project schedule for feedback from the Totara team at several key milestones, and for fixing any issues raised.
- **Design must be general purpose and configurable:** Since the feature will be used by a wide range of organisations, it needs functional flexibility and configuration options not often required by a single organisation (e.g., to support a wide range of use cases for any given capability). This can lead to more design/development work than is initially apparent.
- **Code must adhere to Totara's coding guidelines:** This ensures that the code will work across all Totara's supported platforms (e.g. all supported browsers, server operating systems, databases, languages including right to left languages, large-scale environments etc). Reviewing Totara's coding guidelines before development starts can help keep this overhead to a minimum.

Code contribution process

By engaging early and regularly with the team at Totara Learning you can smooth the process and minimise the need for changes to the specification and code. The code contribution process is managed by the Totara team over a number of checkpoints, allowing review of plans and progress, and to provide assistance along the way. The checkpoints and the key steps leading up to them are outlined below. Exact timelines for the process will vary from project to project.

Step 1: Engage the community and core development team early on

When first considering work on a new piece of functionality, post a new discussion in the Roadmap forum in the Totara Community website's Partner Zone:

http://community.totaralms.com/mod/forum/view.php?id=817

In the discussion, describe what you're planning to implement. If possible, include a draft list of detailed requirements ("Ability to..." statements), mock-ups, use cases, and conceptual diagrams to explain the functionality. By doing so, discussion with the Totara development team can begin early. Direct discussions with the Totara development team can be facilitated via emails, private forums and skype calls as required.

Discussion in the Totara Community forum allows the wider Totara community to collaborate on the idea and helps the Totara team understand the level of demand for the feature. Questions answered by the discussions may include:

- Has anyone already developed this functionality?
- Does anyone have a workaround to achieve this kind of functionality already?
- Is there interest from others in this feature?
- What features would this functionality need?
- What configuration options would this functionality need?

Use this information to fine-tune your idea and come up with a specification. Keep the community informed about progress through the remaining steps by updating the original forum discussion.

As the process moves forward we may schedule a direct Totara developer meeting to discuss your idea in more detail and answer any questions.

CHECKPOINT: Totara Learning decides whether the feature is to be included in core.

Step 2: Write a specification

The Totara team uses agile development methodologies including quality documentation prior to the development stage. Specifications should provide sufficient detail about the feature and how it will be developed. The level of detail required will depend on the complexity and size of the new feature. For a minor feature a couple of mockups and a paragraph or two describing the functionality is likely to be sufficient. For more complex features there needs to be enough detail to allow the core team to understand exactly how the functionality will behave or for a developer to build the feature based on the specification.

Your specification should include:

- Use cases and/or user stories
- Detailed requirements, explaining complex features in more detail
- UI specification including wireframes and descriptions of the UI design elements
- Technical specifications (core code changes, database schema, new capabilities, code structure, events, cron commands, classes, etc)

For an example of the level of detail required for a significant feature, see the open badges development page and accompanying specification here:

- http://docs.moodle.org/dev/openbadges
- https://github.com/totara/openbadges-docs

The Totara team can assist to ensure the specification work meets the necessary requirements for feature development.

CHECKPOINT: Totara core team reviews the specification. Allow time to revise specification based on feedback.

Step 3: Review Totara coding guidelines

Ensure that you know about the coding standards required to have code accepted into Totara. These are designed to make Totara features accessible and usable to Totara's wide audience, and include the following:

- Correct internationalisation: languages, language direction, locale settings such as date formats, unicode support
- Cross database support
- Cross operating system support
- Cross browser support
- Meeting Totara's coding standards and guidelines (including scalability, security)
- UI design consistent with existing interfaces, and compatible with existing themes
- Accessibility: WAG, 508 support, non-javascript support for learner facing code
- Tests to accompany code

Full details of Totara's coding guidelines and development practices are available on request.

Step 4: Development with Totara core input

Prior to development commencing we strongly recommend arranging for regular feedback milestones with the Totara team at each stage of development. This helps ensure you are on the right track and minimises wasted effort if any problems can be caught early in the process. Code reviews should be scheduled and if the timeframes shift please try to let the Totara team know in advance so that the code review can be rescheduled.

To facilitate code sharing, the Totara team will set up a private GIT repository shared between you and Totara. Major features are developed on a feature branch, but minor features can be pushed directly into Totara Learning's online code review system (see below).

Step 5: Code review by Totara core

Once initial development is complete, your code must pass through a full code review before being merged into the core codebase. We use an online code review system called "Gerrit" which can be found at https://review.totaralms.com/ (ask us for an account to access it).

Instructions on configuring GIT to push to the review system can be found on the Totara development wiki (ask us for an account to access it).

Note: Once code has been pushed to the review system, it may be accessible to other partners or subscribers with code sharing access (i.e. it will no longer be private between Totara and your organisation).

Once a feature has been pushed to review it must pass the following checks:

- **Verification:** The code will be run through our automated test suite. This will check that the code installs correctly, all unit tests pass, and other automated checks (e.g., common coding errors) pass. The automated process is normally completed within 10-20 minutes. Details of the process (including reasons for failure if the patch fails) can be found at https://test.totaralms.com/. If the patch works as intended the patch will be marked as "Verified".

- **Code read:** A Totara developer will read the code and make sure everything makes sense and follows the coding standards. This is test is designed to catch a range of issues including logic errors, platform-dependent code, security and scalability issues and coding style issues. When the code has been read and accepted the patch will be given a status of '+1'.

- **UX tests:** Another Totara core developer will checkout the patch and test the functionality. This test is designed to check for a different set of issues including finding bugs, ensuring the UI is clear and cross browser testing. When the code has been tested and works as intended the patch will be given a status of '+2'.

Full details of what is checked during the review process can be found in the Totara development wiki. If you don't already have an account on the wiki, ask us for an account.

CHECKPOINT: Full code review by the Totara core development team via code review system. Allow time to fix issues found during review.

Step 6: Fix any remaining issues

The last step is to resolve any issues found by the Totara team during review. Usually the best way to do this is to resubmit new patch-sets on top of the original one. This allows the Totara team to easily see what has changed and comment on the updated version. It is common for a feature to go through multiple revisions in review before it is merged.

Once code has been accepted it will be merged into a stable branch. At this point, the Totara team takes over ongoing maintenance. Any bug fixes provided after this point are appreciated but not required.

Note: There are occasions when a feature has been developed but the project has finished so there is no time or budget for the partner or subscriber to fix any issues found during review stage. While we prefer the original development team to fix any issues found, we appreciate this isn't always

possible. The Totara team may get involved in the development, as work priorities allow - e.g. in particular when we see a highly-demanded feature contributed. Please discuss with the Totara team if this situation arises.

The diagram on the next page captures the overall code contribution process for Totara Partners.