Sakai Jira Guidelines

Sakai JIRA Guidelines

Sakai uses Atlassian's Jira software for issue management. It's used to track a variety of issues, including bug reports, suggestions for new functionality, tasked work, and community contributions. Outlined below are the general practices, procedures, and definitions adopted by the Sakai community for using Jira.

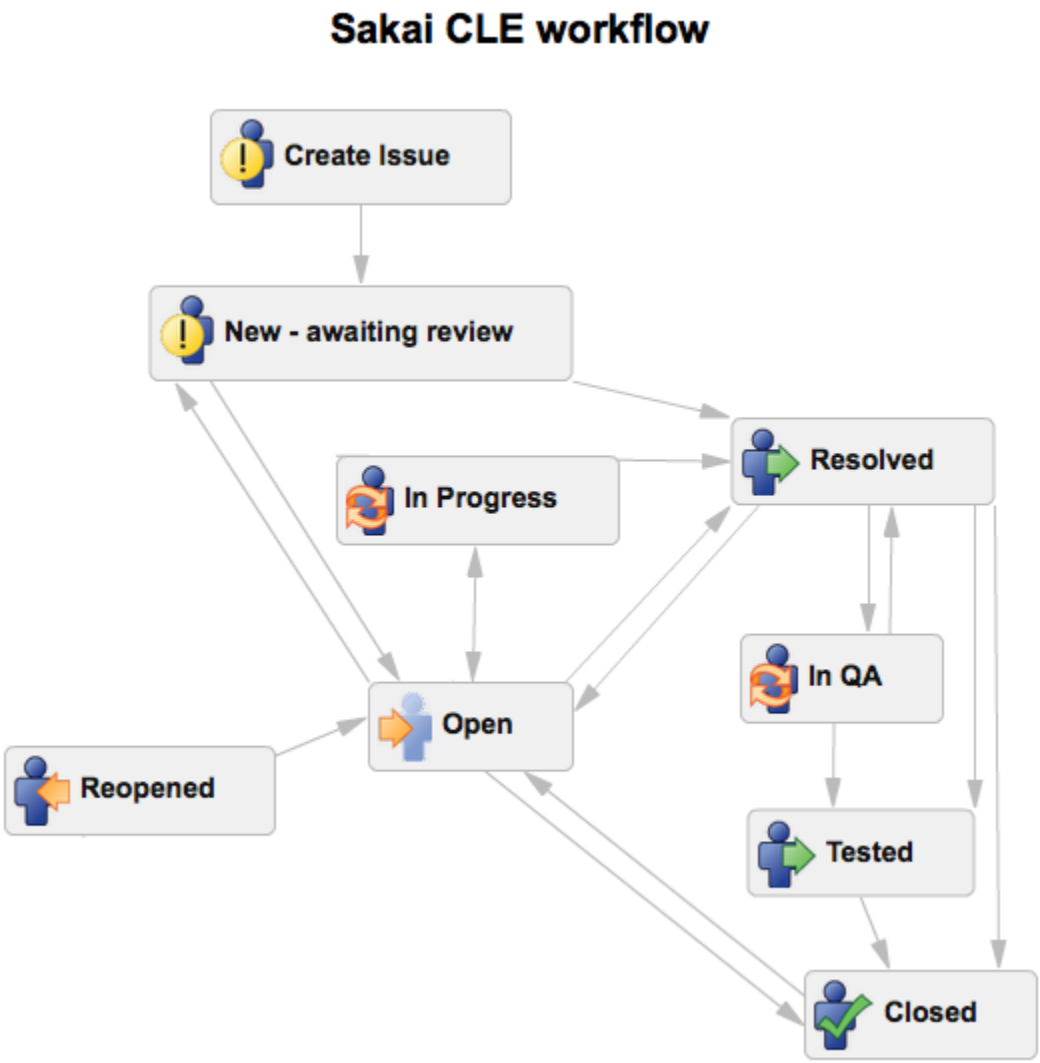
- You can search and view content in Jira without an account, but are required to have one to comment or create issues. Jira accounts are available through self-registration (Jira and Confluence share accounts).
- People who want to submit issues, after creating their account, should review the Workflow.
- If you have questions regarding this tracking software (Jira), please send them to jira-admins@collab.sakaiproject.org.
- For Developers: Maintenance Branch Guidelines | Feature Branches | Jira Branches
- Sakai Community Servers are accessed from here (you'll test on them before creating a community Jira: http://nightly2.sakaiproject.org/).

Summary

Basic Workflow

1. Anyone with a JIRA account creates an Issue (please see #Create Issue Detailed Instructions first)
   - Issue is automatically assigned to a **Awaiting Review** status
2. A Sakai Core team member, and/or anyone who wants to participate in "JIRA Triage" weekly calls, verifies the issue for accuracy (i.e. is it an actual issue) and completeness of details.
   - Complete and valid issues are set to status **Open** and *assigned* to a user or group (e.g. Samigo Team or Sakai Core Team) who can resolve it.
   - The team member may resolve the issue as *Duplicate* , *Incomplete* , or *Cannot Reproduce* if appropriate (see #Resolution).
   - It may also be set to **Awaiting Information** if it doesn't contain enough information to **Open** for work.
3. Once Open, developers ideally pick up the work and assign themselves to it as "Assignee." When they begin work, they selects **Start Progress** and begin to work on the issue (as time and priorities allow)
   - The Assignee selects **Stop Progress** if they are not going to work on the issue anymore for a few days and it is not resolved
4. The Assignee sets the issue to **Resolved** with the relevant #Resolution when work is completed
   - For tasks, if there is nothing explicit for the QA team to test, then the issue is **Closed** without testing.
5. The QA team (potentially you!) selects **Start QA** and begins verifying the issue
   - QA team member selects **Stop QA** if they are not going to complete the testing on the issue within a short time frame
6. The QA team verifies resolution of the issue and adds a comment with details of the testing results and process.
   - If verification fails, then it is **Reopened** for further work (automatically re-assigned to the user who resolved it)
   - If verification succeeds then QA marks the status as **Verified** (indicating it has been **Tested**)
7. The Release Manager merges the issue (if it is a bug) to previous supported and affected releases
   - The associated version merge status is set to **Resolved** (see #MergeStatus)
   - The issue is **Closed** by the Release Manager when the last merge has been completed

Diagram



(Exported from JIRA Sakai Workflow on 31 March 2012 - AZ)

Details

Create Issue Detailed Instructions

The Sakai community is encouraged to report issues and post feature requests in Jira. The Community maintains a set of servers for testing. If you find an issue on your own institution's Sakai servers, please don't report that to the community. Rather, each organization running Sakai is expected to have its own issues list. However, should an issue on your institutions' Sakai be found, AND that issue also exists on one of the community's servers, it should be reported to the community. Here's how!

1. You search JIRA to see if the issue already exists (Sakai Jira Search)
    - Search strategies include searching for a particular component, sorting by update date, and exporting as an MS Excel file (in Views)
2. You might also check the sakai mailing lists archives.
    - Searching sakai-dev archives may help if you get too many results
3. If you're not sure if the problem is a bug, then send a note to the sakai-dev mailing list first to ask
4. You then create an issue in Jira of the appropriate #Issue Type
    - The **component** should be set based on where the problem appears to exist. Do not use the Performance **component** however. Only use a 'Performance' label to flag performance related issues so there is a consistent way to filter these issues across projects.
    - The **Assignee** should be left as -Automatic-
    - The **Affects Version** should be set to the **released version** of the instance of Sakai being used.

- The **Fix Version** should be left set to the default of Unknown. The project teams will set the Fix Version once they have had time to review the issue and estimate when they believe they will be able to address it.
- As much of the following information should be included as possible (to avoid the issue being closed as incomplete)
    - Sakai version
    - (bugs) **Steps to reproduce** (detailed and step by step)
    - If disambiguation is necessary, include **Expected Results** and **Actual Results**
    - **Environment details** (DB type and version, tomcat version, OS type and version, Browser and version, etc.)
    - (bugs) URL to the page where the error occurs
    - (bugs) Stacktrace (traceback) if available
    - (bugs) relevant portions of the server logs (do not attach your entire logs, we do not have time to read through 1000s of lines of logs and figure out where the relevant parts are)
    - (bugs) **Screenshots** or **Screencasts** showing the error
    - (feature) Use cases and detailed requirements
    - The desired resolution

For all issues, please provide a simple Summary and Description that can be understood by all in the community, not just yourself or developers. If you need to include some techno-babble, in addition to the plain language, for other developer types to understand the full details that's fine, but consider placing it in a comment on the issue.

If you need to include a large piece of information in an issue, such as a stack trace, please do not put it in the Description field. Instead, add it as an attachment or a comment. (This makes it a lot easier to view and parse issues in most browsers.)

Sometimes developers may uncover bugs in your own code and need to create a Bug issue. If you think the bug does not affect past versions of Sakai and only exists in the code you are actively working on in trunk, then use the tentative release version that trunk will become as the Affects Version (*e.g.*, 2.6 [Tentative]). Or, if you're working in a Jira branch, then use "branch" as the Affects Version.

Definitions

Issue Type

Issue Type	Definition for Sakai
Bug	An error in design or implementation which directly impedes a user from achieving their expected result.
Task	A new capability being added to Sakai.
Feature Request	A desired capability, for inclusion in a future release of Sakai; ideas that come with resources interested in implementing them are more likely to be developed than those offered with the hope that someone else will step forward to do the work.
Contributed Patch	A community-contributed patch to a particular version of Sakai. The origin of such issues may lie in Bugs or Feature Requests which Sakai has not yet evaluated for implementation. Under such circumstances a linked issue is generally created by cloning the original issue in order to track Sakai's work on the issue. [Use at your own risk!]
Branch	An experimental branch of code, which may or may not be merged back into the main code after the experiment completes; identified in SVN by a branch named with the Jira Key.

- Bugs
    - Important to create a detailed description on how to recreate the bug. Please include a step-by-step way to reproduce the bug in the Description field.
- Tasks
    - Tasks are used by project teams to track the addition of or changes to functionality.
    - It is not necessary to set an Affects Version for a Task.
- Contributed Patches
    - Issue is vetted for accuracy and completeness of information.
    - The issue is evaluated by the appropriate project team, just like a Feature Request or Branch, for inclusion in a future release of Sakai.
        - In some cases, a Contributed Patch may run counter to the Sakai design and cannot be incorporated, and maybe marked as Won't Fix and an explanation as to why provided. In this case, the contributor of the Patch may choose to continue developing the patch for future versions of Sakai.
    - Until the patch is integrated in Sakai, the community encourages the contributor to be responsive to comments and feedback provided by other users of the patch and to continue to update the patch to current versions of Sakai
- Feature Requests
    - Issue is vetted for accuracy, completeness of information, relevance to Sakai's overall design, *etc.*
        - A good Feature Request should explain clearly what a user is trying to achieve. Use cases and scenarios can be helpful in communicating that.

- Assigned to appropriate Project Team or Working Group lead, They will evaluate and discuss such issues periodically, when the group reaches an appropriate point in the release cycle to adsorb community input.
    - If the decision is made to not address the issue, then it should be resolved as "Won't Fix" with appropriate comments.
- When a project team is ready to address the issue, it should be Moved to a Task and an initial estimate of the Fix Version set.

Status

Status	Definition for Sakai
Awaiting Review	Issue is waiting for investigation.
Open	Issue is valid and ready to be worked on.
In Progress	Issue is actively being worked on.
Reopened	Issue was thought to be resolved, however, it did not pass QA and needs further work.
Resolved	Issue has been addressed and is ready for QA testing.
Tested	Issue has been tested and has passed. Ready for merging.
Closed	Work on issue is complete, no other activity.

- (Awaiting Review)
    - Contributed patches should receive priority treatment.
    - The issue should be *linked* to other related issues.
    - Check the is *assigned* to a user or group (e.g. Samigo Team or Sakai Core Team) who can resolve it.
    - Adjust the *Priority* depending on overall project and release goals
    - Other attributes should be changed as needed, (*Components* , *Affects Version* , *Security Level* , *Original Estimate* , etc.)
- (Open)
    1. Open tickets should be worked on within 90 days or assigned back to *Unassigned*
    2. Open tickets which have seen no activity in 180 days will be moved back to **Awaiting Review** status
- (Resolved) - General
    1. Resolved tickets which have had no activity in 180 days will be **Closed without testing**
    2. Details in #Resolution
- (Tested) QA is complete on this ticket.
- (Closed) No other activity should or will happen on this ticket

Resolution

Resolution	Definition for Sakai
Unresolved	Issue is under consideration and/or actively being worked on.
Fixed	Issue has been addressed through changes to the design or code. When viewing an issue, the "Subversion Commits" tab provides specific details regarding code changes.
Won't Fix	Issue will not be addressed because it does not match project goals.
Non-Issue	Issue turned out not to be a problem with Sakai.
Duplicate	Issue is a duplicate of a previously submitted issue. A link to the other issue is added so that progress on the issue can be easily accessed.
Incorporated	Issue has been incorporated into another issue. A link to the other issue is added so that progress on the issue can be easily accessed.
Incomplete	Not enough information has been provided to identify the issue.
Cannot Reproduce	Issue cannot be reproduced and more details or steps need to be added before it can be reopened.
No Resources	Issue has no patch provided and there is nobody available to fix the issue at this time.

- (Fixed) Developers should do the following when fixing issues:
    1. All commits to subversion **must** include the corresponding JIRA number as the first word in the commit message. For example: svn commit -m "SAK-1234 fixed the uncaught NPE"

    2. If you are implementing a Feature Request or Contributed Patch, first use **Move** to covert the issue type to a Task
    3. If working in a branch, first merge your branch to trunk before resolving the issue
    4. Update **Fix Version** to the next major release version, do not use alpha or beta releases (e.g. use 2.8.2, 2.9, 2.10 : do NOT use 2.9b07, 2.9a1, trunk)
    5. Add a Test Plan in the "**Test Plan**" field. May be found by clicking the Edit issue button.
    6. Add relevant "**Release Notes**"
    7. Check the boxes, if appropriate, for **Property addition/change required** and/or **Conversion Script Required**
    8. Selects **Merge** for previous supported and affected releases (e.g. 2.8 Status) for bugs only (features are NOT merged back)
- (Reopened) If QA reopens an issue:
    1. QA sets the Affects Version to the version tested
    2. In the process of verifying the issue, if you discover a different problem, create a new issue to capture it, rather than re-opening the current issue and re-using it; reserve re-opening only if the original problem is still present.
    3. Some issues can not be easily verified and may require special testing conditions or input from developers.
- (Duplicate) If it duplicates a previous issue, then the newly opened issue is:
    1. Linked to the original issue as "duplicates"
    2. Commented with a note to look at the linked-to issue to track further progress
    3. Closed with a Resolution of "Duplicate"
    4. Fix Version and should be set to "Unknown"
- (Incorporated) If it is incorporated into a previous issue, then the newly opened issue is:
    1. Linked to the original issue as "incorporated by"
    2. Commented with a note to look at the linked-to issue to track further progress
    3. Closed with a Resolution of "Incorporated"
    4. Fix Version and should be set to "Unknown"
- (Non-Issue) If it turns out that the issue was the result of a mis-understanding of how Sakai operates, then:
    1. An effort is made to clarify the situation.
    2. If the mis-understanding does not suggest a design flaw, then it is Closed with a Resolution of "Non-issue"; otherwise, the design flaw could be captured to Bug or Feature Request.
    3. Fix Version should be set to "Unknown"
- (Incomplete) If insufficient information is provided to describe the issue, then:
    1. A comment is added to ask the user to include the needed information (should list the information needed)
    2. The issue is closed with a Resolution of "Incomplete"; issue can always be Reopened.
    3. Fix Version should be set to "Unknown"
- (Cannot Reproduce) If the issue cannot be reproduced on one of the QA test servers, then:
    1. A comment is entered to indicate what was attempted to reproduce the issue (e.g. "followed steps in issue")
    2. The issue is closed with a Resolution of "Cannot Reproduce"; issue can always be Reopened.
    3. Fix Version should be set to "Unknown"

Branch Merge Status

Each branch has it's own status of the fix for it's branch. The field labels look like 2.9.x Status, 2.8.x Status, etc. These statuses do not tie directly into the overall ticket status and only the Branch manager should update this status.

Resolution	Definition for Sakai
None	No merge needed for this ticket (i.e. only should be applied to trunk or whereever it was committed)
Merge	The code changes related to this ticket should be merged into the x branch.
Resolved	The code changes have been merged
Closed	<needs definition. perhaps a superfluous status?>
Won't Fix	Means the merge to the branch cannot be done and will not be done (usually a reason is given in the comments - this is super rare)

Version

Each Jira issue has an Affects Version and a Fix Version. Generally speaking Sakai uses these values to indicate:

- **Affects Version** - Version(s) in which an **issue is observed**, and should be a released version of Sakai. Do not use unreleased versions.
    - If you find an issue in trunk, and it is a problem that only affects trunk and not previous releases, then please use the next planned release as the Affects Version. If the issue also affects released versions of Sakai, then please include those versions too.
- **Fix Version** - Version(s) for which the issue is **actually fixed** (for Resolved/Closed and Fixed issues). Leave blank for Unresolved issues.

Further notes on Version Numbers

- **Alpha, Beta, and RC tags** (*e.g., alpha03, beta02, RC01*) - After a release is made, the interim alpha, beta, and release candidate versions are merged into the released version. For example, 2.6.0-alpha01, 2.6.0-alpha02, 2.6.0-beta01, 2.6.0-rc01, etc. would be merged into 2.6.0. The merging is necessary for keeping it simple to search, filter, and view issues for released versions of Sakai; however, the original version numbers are still associated with each issue in the database, if there is a need to extract such information later on.
- **Maintenance Branch versions as Affects Versions** (*e.g.*, 2.3.x, 2.4.x) You should generally avoid using maintenance branch version as an Affects Version, unless you are sure an issue does not affect past releases on that branch. This is important because branch versions are moving targets; what affects 2.5.x one day will not necessarily affect it in the future, after the issue is fixed.
- **Maintenance Branch versions as Fix Versions** (*e.g.*, 2.3.x, 2.4.x) - Generally the only branch version a developer needs to be concerned with is the next major release version for changes they are checking in to trunk. Only the Branch Managers should worry about using maintenance branch versions or beta versions as Fix Versions. They will use maintenance branch versions only when the fix is actually checked-in to the branch. It should not be used as an indication that one would like a fix merged (rather one should set the maintenance branch's status to "Merge".)
- **SAK Experimental Branches** (*i.e.*, branch) - Use the generic "branch" version as the Fix Version when working on subtasks under a SAK Experimental Branch. If and when the branch is merged into trunk, please remember to update the Fix Version appropriately (generally this will mean changing it to the next major release at the time the branch is merged to trunk.)
- **Developers**- Can only use actual release version where the fix is made as the Fix Version and indicate it should be merged using the special "Merge" fields (as discussed in the Sakai Core team).
    - For instance if the fix is made in trunk, and trunk is 2.10-SNAPSHOT the fix version should only be set to **2.10 [tentative]**
- **Branch Managers** - Can use any version which is appropriate including beta or alpha versions (which are indicators of when something was merged and what it was merged into).

Maintenance Branch Version Numbers

A ".x" version is used to track issues relating to maintenance branches. For instance, 2.5.x is the version number representing the 2.5 release's maintenance branch.

If someone is basing their deployment on a maintenance branch, then there is the potential for confusion when using it as Affects Version, since the maintenance branch is a moving target. When reporting issues, it is important to indicate the revision number in the issue, so that folks know to which revision of the maintenance branch you are reporting the issue against. Even better is to determine if the issue affects the last maintenance release, and, if it does, use that as the Affects Version instead.

Priority

(Updated for Sakai 12, inspired by Drupal priorities)

The Priority field in Jira is used by Sakai to reflect a combination of issue characteristics, including:

- Number of users affected
- Resources required to resolve

In practice, the Jira Priority field is utilized by Sakai at two times: during prioritization of feature requests/branches/contributed patches for implementation or merging, and when making decisions on what will actually appear in a release. Initial priorities, when an issue is first reported, may be changed to reflect needs as a release moves forward and priorities evolve. Every issue comes in as Major priority and each Jira is triaged to determine the appropriate priority.

As a release date approaches, priorities will also be adjusted - and lowered, if necessary - to reflect the decreasing availability of time and resources.

Priority	Basic Definition for Sakai	Impact on previous releases	Examples
Blocker	Release will not be completed until issue is resolved.	Backported at least 2 major version (if applicable/affects). Potentially more than 2.	• Render a site unusable and have no workaround. • Cause loss/corruption of stored data. (Lost user input, e.g. a failed form submission, is not the same thing as data loss and in most cases is major). • Expose serious security vulnerabilities. • Errors in grading calculations affecting most students/instructors.
Critical	Issue will most likely be resolved for release.	Backported 1 major version back. 2 if applicable/affects and if no code conflict.	• Interfere with normal site visitors' use of the site (for example, content in the wrong language, or validation errors for regular form submissions), even if there is a workaround. • Trigger a Java error through the user interface, but only under rare circumstances or affecting only a small percentage of all users, even if there is a workaround. • Render one feature unusable with no workaround.
Major	Issue should be resolved for release.	Backported 1 major version back if no code conflict	• Admin- or developer-facing bugs with a workaround.

			• Bugs for site visitors that do not interfere with site use, for example, visual layout issues.
- Minor (Previously also Trivial)	Issue may be resolved for release.	Not backported	• Used for cosmetic issues that do not inhibit the functionality or main purpose of the project.

Environment

- A description of the Sakai environment in which the issues was encountered, *e.g.*, web browser, operating system on which Sakai is installed.
- For QA participants this is a good place to note on which test server the issue occurred (*e.g.*, MIT Stable HSQL, MIT Stable Oracle, MIT Stable MysQL, Nightly)
- If you're reporting an issue with a maintenance branch, please include the revision number of your version of the maintenance branch here.

Description

- A detailed description of the issue, including the steps necessary for reproducing the issue.
- You can also add attachments, including screen shots to help illustrate the issue.

Security Level

- Controls the visibility of the issue; currently toggles between viewable by anyone or by just the committers and testers (and the reporter of the issue).

JIRA Notifications

Whenever a Jira issue is modified, the issue's Reporter, Assignee, and any Watchers will receive an email notifying them of the change. To become a Watcher on an issue, go to the issue and view it, and click on the "Watch It" link. Additional notification options include:

- You can subscribe to filters in Jira and receive an email summary of the filter's contents at a time interval you specify. For those interested in a daily summary of updates in the Sakai project in Jira, you should subscribe to the "Updated in Last 24 Hours" filter, and specify a 24-hour or 1-day time interval. (Note that the time of day when you subscribe is the time of day when Jira will send the digest, so you probably want to subscribe to the filter early one morning.)
- Jira can also send on-event notification emails, such as whenever an issue is created, updated, resolved, closed, etc. If you are interested in receiving such emails, then please contact jira-admins@collab.sakaiproject.org.

Jira Spring Cleaning

Every year we should resolve out old issues. I plan to do this by the end of each April as this is the time when we update servers and well it's Spring in some parts of the north. We default to closing out issues that haven't had any updates more than 3 years ago 3 years. So in 2018 we closed out anything older than 2015.

```
This is the JQL to search for these issues. Search for these
updatedDate < "2015/01/01" and project = sak and status in (Open,"Awaiting Review")
```

Then do a "Bulk Change"
Transition Issues

Select "Won't Fix" as a Resolution and add this comment (updating the date) :
Bulk closing issues that have not been updated since 2015 and earlier. Please reopen if this is still an issue and you have new information or if this is a feature you'd like to still have consideration for.