Sakai QA Home page

Current QA activities in this column

Sakai 19 QA Hub

QA Servers

Meeting Logistics:

- BigBlueButton hosted by BlindsideNetworks
- http://apereo.blindsidenetworks.net/apereo/
- Room "Sakai QA"
- password "apereo" or "apereom" for Meeting Moderators

General Information about QA in this column

Purpose

QA stands for Quality Assurance. In this context the QA mission is primarily to uncover and report software bugs, to verify fixes to software bugs, and to test for regressions. Regressions are features that worked in previous versions, but are broken in the current version.

Pre-requisites for participating

Required - a Jira account is required. Sign up at https://jira.sakaiproject.org

Video: How to Create a Jira (3:26 minutes)

Required - Join Sakai QA email group (sakai-qa@apereo.org) or Sakai Dev group (sakai-dev@apereo.org) to keep up to date with the latest announcements and ways to help with the testing effort.

Optional - **Experienced Testers** - Although, you do not need special permissions to test fixes and to comment on Jira issues, you can not mark the Jira as having its testing completed. To mark the Jira as completed, you need access to a "Tested" button which indicates successful testing of a Jira issue. This button is only available if you are a member of the Jira QA group. Contact sakaicoordinator@apereo.org for more information.

Hints - Our primary tools are Jira, Google docs spreadsheets, and QA servers. Learning how to use Jira and what the fields represent is well worth the time.

Ways to Participate

- Verify bugs that have been fixed.
- Test new features.
- Test existing features to assure changes around them haven't introduced bugs (regressions).
- Create Regression scripts (i.e. step-by-step instructions for performing regression tests.)
- Release testing - testing alpha, beta, and release candidate versions before software is made generally available (GA aka "General Availability" aka production ready).
- Proofread. Make suggestions. Create short videos.
- Ask and answer questions on the email groups.
- Attend the QA meetings to bring up issues that need attention, and to help plan testing for releases.
- Expert QA knowledge needed - special skills are needed for some types of QA testing.

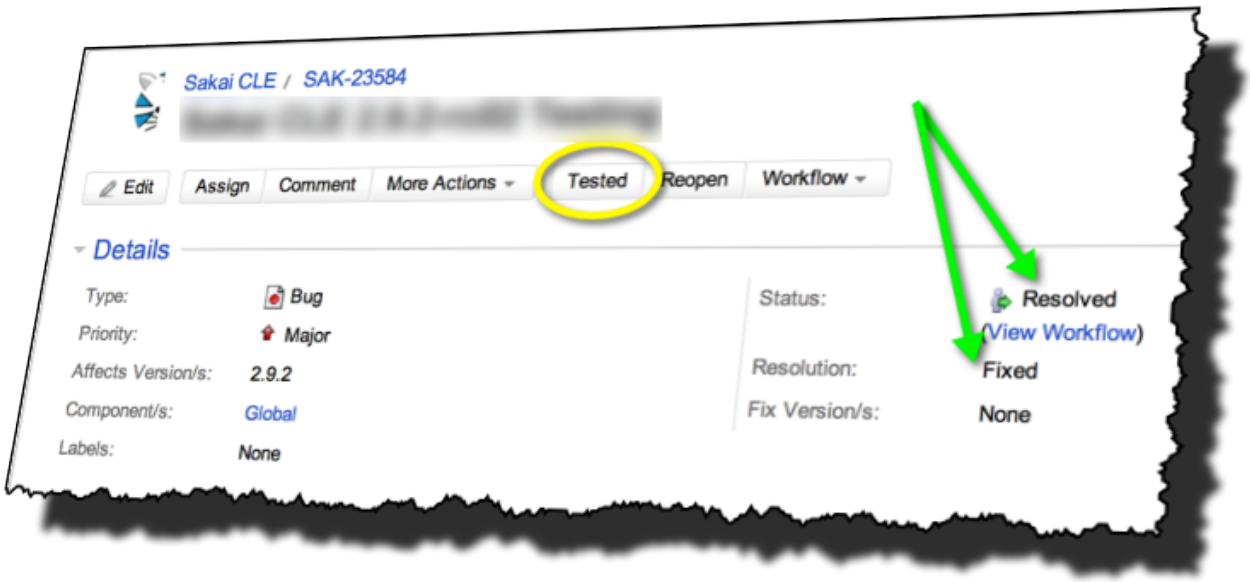
When to Participate

- Anytime. We are almost in constant need of QA testing of one form or another. Contact the sakaicoordinator@apereo.org for help getting started.
- Major Releases - Sakai 11, Sakai 12 and Sakai 19 are all examples of major releases. Major releases typically include significant upgrades to existing features and the addition of new features.
- Maintenance Releases - Sakai 11.4 and Sakai 12.6 are examples of maintenance releases.

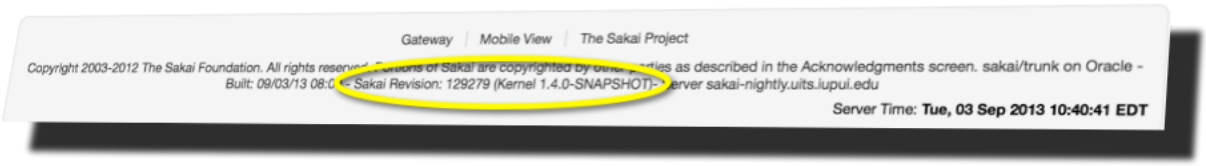
Verifying bug fixes

Testing either takes place on a nightly server, usually trunk, , or on a QA server that is announced when community testing for major or maintenance releases is scheduled. Please be aware that nightly servers are refreshed daily, at midnight Eastern time (-4 or -5 GMT). Please plan your testing appropriately.

Jira bugs which have a resolution of "Fixed" and a status of "Resolved" are ready for testing. If testing is successful click the Tested button. The status will change to Verified. If you don't see a Tested button, even with the correct status and resolution, you may not have permissions in Jira to see the button. Contact sakaicoordinator@apereo.org to request this permission.



In your comments make sure to include which OS/browsers you tested and the Revision number of trunk. The Revision number is important because trunk changes frequently. Look at the bottom of your browser window for the Revision number.



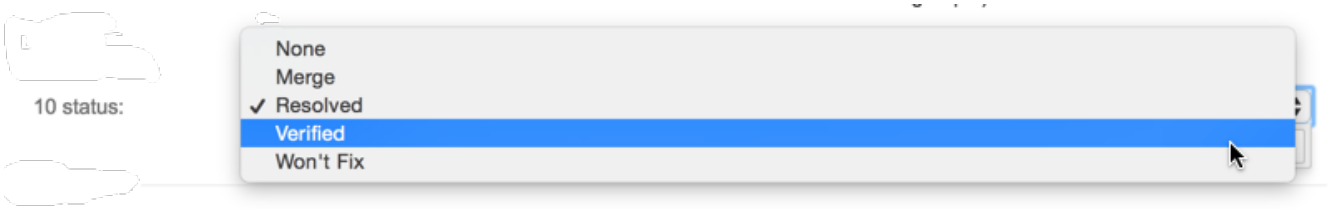
Notes: Some issues may require testing in more than one OS /Browser combination.

Special Case: Master (formerly trunk vs Branch)

The Master branch (previously known as Trunk) has the most recent source code, and is constantly changing as developers contribute fixes and new features. At some point, when we are getting ready to release Sakai as a community-supported version (aka general availability or production release), we create a branch. A branch is a copy of master at a certain point in time, a snapshot. By doing this, we can exercise more control over what goes into a release. We name a branch to correspond with the release. For Sakai 12.0 release we have a 12.0 branch, which we sometimes also refer to as 12.x. For Sakai 19.0 release we have a 19 branch, also called 19.x . Features and fixes almost always start in master (more than 99% of the time. Exceptions are rare.) We test features in master and then when they are verified to work, if they should be included in a branch, we merge them into the branch and re-test the fixes, since the environment between master and the branches can diverge significantly. A fix that works in master, may not work in the branch (although in most cases it does.) Since we do not have enough QA resources to retest every fix, we focus on getting at least all the blocker priority issues retested.

We include in JIRA a status that indicates the state of the ticket with respect to the branch. For example: 12 status. When it is set to Resolved it means that the fix has been merged into the Sakai 12.x branch. Verified means that it has been tested after merging into 12.x . This can be confusing, until you get used to it, because it means that there are at least two fields on a JIRA ticket that have the potential to be set as Resolved or Verified, the main Status of the ticket (see image above), and the branch status. There are only 2 versions of Sakai in development at any one time. Currently that is 12 and 19, therefore we have a 12 status field and 19 status field. When Sakai 19 is released, we will drop support for Sakai 11 and there will be an 19 status field and a 20 status field.

As a QA tester, once you understand what the fields mean, it is okay to set the status yourself to Verified, on both the overall JIRA and for the branch.



A bug that is supposed to be fixed, isn't

If you are verifying a fix and it fails then reopen the issue by clicking the Reopen button. This indicates to the developer that there is more work to be done. If you are not sure if the issue should be reopened then simply leave a comment. Make sure to include the exact steps of how it failed, what server you were on, the revision number if testing was done on trunk, browser/os, etc.

Test New Features

Testing new features is essentially the same as testing for bug fixes. You still want to click the Tested button to move the issue to Verified status if all the testing passes, and Reopen or add comments if you find bugs. Sometimes there are debates about new features and what constitutes a "bug". Those debates happen in the Jira comments and sometimes are brought to other groups like the Sakai Core team for broader discussion, as appropriate/relevant.

Regression testing

Regression testing is a kind of detailed testing. Regressions are features that worked in previous versions of Sakai, but now are broken in the current version. It It is very important to uncover these kinds of problems as soon as possible.

The most up-to-date list of tools for which we have Regression scripts (steps for testing features manually) is kept in Apereo's Google drive. The most recent set is for Sakai 19 RC.1

Sakai 11 Scripts-do we have updated copies?

Release testing - QA Test Hubs

Our goal is to have testing happen as much as possible throughout the Sakai version lifecycle (verifying bug fixes in trunk, testing new features, and regression testing the latest release). However, when we are getting close to having a tagged release (e.g. 12.6, 19.0, 19.1RC, etc) we choose specific periods of time to test, with a Test plan and focus. We call these testing efforts "Test Fests". We have multiple Test Fests to get out a release. Unless otherwise specified we have one "Test Fest" per phase of the release. For example, we might have several Beta versions of the software before the Sakai core team feels confident that it is ready for release. Typically, we then have two or three Release Candidates of the software. A Release Candidate is a version of the software, which if it passes QA testing and Developer scrutiny, will become the next official community release. Each of these Beta releases and Release candidates is a phase of our Software Release process.

What makes a Test Fest unique is that we create a Test plan to guide our testing, a defined period of time to complete the testing, and a review process to assess the state of our testing results. Test Fest's typically take a concentrated effort and we like to have as many testers as possible helping in the effort, sometimes with multiple testers focusing on the same functional areas to provide the highest confidence level we can reasonably achieve. Test Fests are almost always carried out on special QA test servers which have been updated with the correct version of the software which needs testing. The Test Plan will indicate which QA servers are available for testing.

QA team meetings

QA Planning Meetings Tuesdays 2 pm EST BigBlueButton Sakai QA Room.

QA Test Fest Meetings Thursdays 10 AM EST BigBlueButton Sakai QA Room

Expert QA knowledge

- Accessibility Testing
- Localization Testing (different languages)
- Review of patches and fixes
- Testing locally
- Automated testing
- Security testing (requires special permission)

General Testing Tips

Overview

This section serves to provide general testing tips for web applications. It provides basic tests for different elements in web applications to help users think about different ways to test. It is not meant to provide comprehensive documentation on every way to test.

Categorized tips

Data variations

Variations in data that should be used throughout the application in every location that strings can be entered.

- Special characters in text ~!@#$%^&*()
- Non-Latin characters (Cyrillic, Arabic, Japanese, Chinese, Klingon, etc.)
- Languages from right to left (Arabic, Hebrew) [http://www.lipsum.com is a good site to get sample text for both non-latin characters and languages reading from right to left]
- Latin characters with diacritical marks
- Latin characters using ligatures
- Very long strings
- Single character strings
- Scripts/HTML tags
    - <script>alert("xss");</script>
    - Content should be handled gracefully, but not executed when displayed, unless explicitly stated for a feature

Form testing tips

- Submit with no form values
- Submit without required fields filled out
- Submit with improperly formed values (i.e. an improperly formed email in an email address field)
- Submit with Data variations (see Data Variations section)

Login screen testing tips

- Submit with no values
- Valid username, invalid password
- Valid username, valid password
- Invalid username
- Valid username, no password

Drop Down menus

- Drop down is properly sized for menu options
- Default menu option is displayed in drop down on page load
- Selected menu option is displayed in drop down after clicking off menu
- Drop down menu can be navigated through with keyboard and mouse
- Drop down menu appropriately resizes vertically on small screen resolutions
- Entering a character selects the first menu option starting with that character

Combo box

- Combo box is properly sized for menu options
- Default menu option is displayed in drop down on page load
- Can select single item
- Can select multiple items in list
- Can select non-contiguous items
- Drop down menu can be navigated through with keyboard and mouse
- Drop down menu appropriately resizes vertically on small screen resolutions
- Entering a character selects the first menu option starting with that character