How to test - read this first

Introduction

On this page:

The IGB project integrates testing and development, and everyone plays a role. As we work on new features, we test them before and after the changes get merged into the development (main-JDK8) branch. This called "agile testing".

The IGB JIRA IGB work board includes testing lanes. When developers modify the code to implement stories described in a JIRA issue, the team performs testing and code review as the issue progresses through the board. This reduces the testing burden at the end of a development cycle when we release a new version to users. Also, it ensures we maintain high quality of intermediate development versions. This lets users use pre-release versions with confidence.

Before we release a new version to users, we create a new Epic in JIRA to track of testing tasks and general progress toward the release.

We then perform general smoke-testing of a release candidate as described in IGB Subsystems pages.

Quickstart - how to test

1. Download the appropriate installer from developer's fork or main IGB repository in Bitbucket. Installers are in the **Downloads** section of the repository.
2. IGB should launch as soon as the installer finishes running.
3. Follow the **Smoke Testing** and **Subsystems** testing protocols in the pages below.
4. If you find a bug, check to see if the bug exists in the currently released version of IGB. This is very important! We need to know when and how the bug was introduced, and knowing when it first appeared helps in solving it. If you find the bug in previous versions, search JIRA for an existing bug report. If the bug is already described, it's a "known bug." Add your finding as a comment to the release testing epic. If doing release testing, add this to the shared release testing document. If testing a new feature, add comments to the JIRA story (if testing a new feature) or report it as a new JIRA issue if it's an entirely new bug.
5. If the bug is new, then report it as a new JIRA issue.

Overview - agile testing

When developers finish a first draft of a new feature or bug fix, they move the corresponding JIRA issue into the "Needs 1st Level Review" column of the Kanban board and un-assign it. Someone else on the team then assigns the issue to themself and moves the issue to "Reviewing" column. Reviewing involves two types of reviewing: code review (discussed elsewhere) and functional testing.

When testing new feature, we always check that the new feature or bug fix works as described in the story. In addition, check subsystems in IGB that might be affected by the change.

If you find a problem, we make a note of it in the corresponding JIRA card, re-assign it to the developer, and move the store back to the **To-Do** lane on the JIRA board. ***Always include a note in the story describing what you did, even if there are no problems***.

For testing, the "deliverable" is a report on what you did to check for errors.

Pre-merge functional testing

Pre-merge functional testing means testing a developer's changes before it is merged into the main-JDK8 branch via a pull request. For functional testing, you should download the developer's branch-specific installer, install it on your system, and test it there. (The developer should have rebased their branch onto the latest main-JDK8 prior to testing.) Assign the corresponding JIRA card to yourself and move it to "First Level Reviewing" while working on it.

Post-merge functional testing

When performing post-merge testing, use the main-JDK8 branch installer to test. As with pre-merge functional testing, if you notice any problems, describe them in the corresponding JIRA card, move it back to the **To-Do** column, and re-assign it to the original developer.

Note: The *main-JDK8 branch installer is a pre-release version of IGB*.

Overview - release testing

Before making an official release of IGB, we test release candidate installers on up-to-date versions of the three major platforms - Mac, Windows, and Linux. This is important because operating systems differ in how they handle security, and this changes with new versions. For example, an IGB installer that worked great on older versions of the Apple OS may become un-installable on a newer version. (Here is an example from release 9.0.2.)

Another reason we test on all three platforms is that IGB is using native file chooser components that are native to each platform. Also, how graphics are handled differ between platforms. For changes to the IGB UI, it is important to check them on each platform.

For release testing, we may set up some computers side-by-side and go through each smoke testing and systems testing protocol as a group, testing each protocol on the three platforms simultaneously. This can save time because it allows comparing behavior across operating systems and uncovers OS-specific problems.

General testing procedures

To check IGB functionality on a system, you should test in two ways:

- **Naive system** - Test on a system that mimics a completely naive users who has never installed IGB on their computer before. This mimics the situation of new IGB users.
- **Current user** - Test on a system that is running a functional copy of the current IGB release. This mimics the situation of existing IGB users.

Note that to test IGB and IGB installations, you need to understand how the IGB installation process works on the system you're testing.

IGB installers are using the Install4J tool to install the compiled IGB code together with a "sandboxed" Java virtual machine that is used only to run IGB. When a user installs IGB, a folder gets created in the part of the file system dedicated to individual users' applications. Administrator (root) privileges are not required to install IGB, so this location is nearly always located in the user's personal or home directory. In addition, be aware that the Install4j installer software *runs every time IGB launches*. When IGB launches, the Install4J runs, too. It sends a request to the IGB server to read a file called "updates.xml." When we release a new version of IGB, we edit this file residing on the server, incrementing the release number as required. Then, when the installer reads this file the next time a user launches IGB, it will invite the user to download and install a new version if available.

Also note that IGB saves information about user preferences to the local system. IGB is using the Java preferences API for this. This means that to mimic a completely naive system, you must clear the user preferences. The easiest way to do this is by launching IGB and running the **Reset Preferences to Defaults** command, available in the **Other Options** tab of the **Preferences** window.

The preferences can also be cleared by deleting the following file, depending on the operating system.

Windows:

Mac: /Users/<user>/Library/Preferences/com.affymetrix.igb.plist

Linux:

Last but not least, IGB stores IGB-specific data in a folder, which is different depending on the operating system.

Windows: C:\Users\<user>\AppData\Roaming\IGB

Mac: /Users/<user>/.igb

Linux: /home/<user>/.igb

To mimic an entirely naive system, you should delete this or move it a new location for safekeeping.

If using a GUI file browser, make sure that the settings allow you to see hidden files.

Naive system quick start

To remove all traces of IGB and mimic a naive system:

- If IGB is already installed, launch it and clear the IGB preferences. (See User's Guide for details.)
- Open the installation directory and run the uninstall script.
- Delete the .igb ("dot-IGB") directory in the ~/home (Mac and Linux) directory or its Windows equipment on your computer.

Current user system quick start

To mimic an existing user of IGB:

- If IGB is already installed, check the version. Is it the latest released version?
- Visit the latest Arabidopsis and human genomes.
  - Use the checkboxes to open data files and then load them by clicking the **Load Data** button.
  - Change the foreground and background colors to mimic a user customizing how data look.
- Create bookmarks for testing.

Installing the test target

The test target is the version of IGB that you are planning to test. This is either a branch-specific installer (in the case of agile testing) or a release candidate (for release testing.)

Download the test target installer and run it.

Note: Instead of installing it in the default location, install the test target version (branch build or release candidate) on your computer in a directory named for the release. In addition, answer "no" when asked if you want to create a shortcut. Following this convention will reduce the chance for conflicts that may appear to be errors but really are only artifacts of the testing process.

What to do if you find an error

Congratulations! Finding errors is awesome!

It is far better for us to find errors - so that we can fix them before releasing the software to our user community.

If you find what appears to be an error, the **first thing** you should do it document how to reproduce it.

Create step-by-step instructions describing how to trigger the error. Make sure that the you can trigger the same error behavior on a completely naive system (see above.) Add these instructions to the existing JIRA card for the particular new feature or bug fix you are testing, or create a new JIRA card if the original card has already been closed or if you think the error you discovered should be documented in a new card. (Feel free to consult a colleague if you are not sure.)

Important: If you open a new JIRA card, give it a title that starts with "Investigate" and then describe the problem.

For an example, see Investigate why IGB hangs on Mac.

Next, find out: When did the error behavior first get introduced? For this, do two things:

- Install the currently released version of IGB on a naive system. Does the current release have the same error?

- Search JIRA for mention of similar behavior. The bug you have found may be a known issue.
- If you find a JIRA report, read it carefully. Was the bug thought to have been fixed? Consult the testing manager or colleagues to decide what to do next. There are many options. Typically, you will re-open the old bug report and move it to the top of the backlog or to the current sprint, depending on severity.
- If you can't find a prior bug report, this means the error could be newly introduced or could be a pre-existing error that has never been reported. Using the target version, develop step-by-step instructions on how to trigger the observed error. If we do not have step-by-step instructions on how to reproduce the error, there is no way we can fix it. So this aspect is essential.
- Next, use your new, step-by-step instructions to find out: Does the error exist in previous versions of IGB? If yes, then you have found an old bug that may not have been reported. If not, it is new. This will provide valuable information about when and how the error was introduced.
- Finally, create an issue for the error in JIRA. If this is a new issue, title it "Investigate and fix ..." (and describe the error). Include the steps required to reproduce the bug. Move the issue to the top of the backlog or to the current spring. If doing release testing (see above) make a note of the error you found in the share testing document. Include a link to the newly created issue.