Aeon GUI beta testing 

# Pull Request & Release Candidate testing Guidelines

The purpose of Pull Request (PR) testing is to find, if any, pre-merge issues with a build from upstream patches or new patches to Aeon core code. Sometimes, new issues are created by merging upstream or new code. We do this testing because we want to find if there are specific Operating System (OS) issues or if there are general issues with the build process or function of the soon to be integrated code. Beta testing helps streamline the PR merge process so that updates and changes can be done in a timely manner. Testing of candidates must be done after creating your build testing environment using the guidelines found on the Aeonix GUI Readme.md located [here](https://github.com/aeonix/aeon-gui#compiling-the-aeon-gui-from-source).. Note that dependencies do change over time and please verify your build environment reflects the latest changes. Also be sure to follow any OS specific guidelines for the build process such as the Windows addition of “ cd build & make deploy “. Below are the steps that are requested with results for individual building and testing:

# Pull Request Building:

## Create a new folder and pull the latest PR from the Aeonix GitHub repo:

### “ git clone <https://github.com/aeonix/aeon>-gui.git ”

## Move to the Aeon folder and fetch the PR you are testing:

### “ cd aeon-gui && git fetch origin pull/37/head:PR-37 ”

## Checkout the PR you are testing:

### “ git checkout PR-37 ”

## Update the modules for that PR:

### “ git submodule init && git submodule update ”

## Build the PR candidate

### “ ./build.sh ”

# Release Candidate building:

## Create a new folder and pull the latest PR from the Aeonix GitHub repo:

### “ git clone <https://github.com/aeonix/aeon>-gui.git ”

## Move to the Aeon folder:

### “ cd aeon-gui “

## Build the release candidate:

### “ ./build.sh ”

If the gui builds properly, you should be left with the following binaries in your build folder as noted on the command prompt after successful completion:

* Aeon-GUI
* Aeond
* Depending on OS, additional .dll files or others will be included with the build

If your build was not successful, please search on the Aeon GitHub bug list to see if there are any similar build errors. If you do not find them, search the Monero GitHub bug list to see if there are any issues related to yours. Chances are that someone else has had the same issue in the past, especially with PR builds. If you do not find this bug anywhere, try building the latest Monero release and see if you have the same issue. If you do have the same build errors, create a bug in the Monero GitHub. If you do NOT have the same error building on Monero, please consult the Aeon discord or Aeon IRC for assistance before creating an Aeon bug. Chances are someone else may have seen this issue on the same build process and OS as you. If needed, create a bug on the Aeon GitHub for your issue.

If your build of the GUI was successful, we need to test it now. Each candidate needs testing for all possible functions. Of course, not everyone can test every function and every little detail, so we suggest at least testing the following on your desired OS and report back the findings. After successful completion, please test the following:

1 – Do the GUI open properly?

2 – Do the GUI close properly with no errors.?

3 – Are the executables named properly with no spelling errors?

4 – If applicable, does the supplied daemon run properly?

5 – Does the GUI and Daemon built show the correct build version?

* You can check the hash version at the header of the CLI daemon and on the GUI at the bottom of the “Settings” tab and compare them the latest PR commit or Release commit.



6 – Daemon testing (when applicable):

* Run the daemon and sync blockchain from your latest blockchain file.
* Run the daemon and sync from block 0.
* Run the daemon and sync from a desired block height you enter.
* Once synced check as many commands as you can with the daemon CLI specifically “version” and “status”. See additional commands [here](https://github.com/AEONCommunity/CLI-Usage/blob/master/Daemon%20commands.md).
* After the daemon syncs, run the daemon for at least 24 hours and check status and block height for errors. Run the daemon for a long as possible to verify no longer term issues arise from the build.
* Close daemon with “exit” and verify the daemon closes with no errors.
* Report any issues on the PR or if a release version contact devs via IRC or Discord channel.

7 – Wallet GUI testing:

* Open the GUI and create a new wallet in the language of your choice. Make sure to save the seed and keys for regenerating later. Make sure the wallet syncs with the daemon properly, it should start automatically.
* Open the GUI and open an already generated wallet file. Make sure it syncs with the daemon properly.
* Restore a wallet from seed or keys, verify it restores with no errors and at the proper height you enter (if not starting from block 0).
* Open the GUI and connect to a remote node and verify it syncs properly with no errors.
* Send and Receive a transaction on both local daemon and remote node and verify no issues.
* Check as many additional functions of the wallet as possible. Open all tabs and check for spelling errors or any non-functioning tabs or functions.
* Report any issues found on the PR or to devs on the IRC channel or Discord.

9 – When you have tested as much as you can, please report back findings for PR testing on the actual pull request on GitHub. If testing a release, feel free to comment on the announcement thread. If you have any issues at all with the testing of build or functional process, please contact the dev team on IRC first then any other available resource. Dev team members are most active on IRC. If unable to contact on IRC, use other media platforms to contact the community and dev team. Report listings with the following information so that we can determine if a Release or PR is good to go:

* VM build or Standard OS build type
* Operating system you used to build and test
* Any specific build variables you used
* Processor type, core count used, and RAM used
* Language used for testing wallet
* Any issues you encountered on the build process
* Any issues you encountered on the testing process
* Any comments for stability or enhancements for the build
* Any additional comments from build/testing