

Contributing to Nightshade

1 Overview

Changes to the Nightshade code-base can be performed on private branches and submitted to Nightshade developers for consideration regarding inclusion in future releases. The following steps are a broad outline of the process for code contribution. The remainder of this document discusses each step in detail. The first 4 steps need to be completed only the first time you setup a new Launchpad account to contribute to Nightshade.

- Register: Create an account on Launchpad and branch nightshade code
- Setup: Installation and configuration of the Bazaar (bzd) source control client
- Initialize the Server Branch: Push a copy of the Nightshade source to your branch.
- Working Branch: Create a local copy of your private branch.
- Code: Code, test and perform a local check-in of changes
- Push: Send changes to the server branch and notify Nightshade developers
- Best Practices: Pain reduction techniques

2 Register

Nightshade is hosted on Launchpad, an open source project host using the Bazaar version control system. You must register an account on <https://launchpad.net> in order to create your own branches of the Nightshade source tree. After you've registered, login to your account and navigate to the Nightshade project at <https://launchpad.net/nightshade>. Click the link titled 'Submit Code'. This will walk you through the steps necessary to create your own copy of the Nightshade code

under your account on Launchpad. Take note of the branch location near the top of the page. Select a 'hosted' branch type and a status of experimental or development.

3 Setup

Download and install the Bazaar client from <https://launchpad.net/bzr>. Prebuilt binaries are available for Windows. For Linux, Bazaar is usually available through the distribution's package manager, such as Yum or Synaptic. Following installation, you must add an SSH key to your account settings on Launchpad. When logged into Launchpad, click your name in the upper right hand corner. Then click the '+' icon near SSH keys and follow the instructions there. You can not modify the branch you created during registration until this is set up properly.

4 Initialize the Server Branch

A server side branch was created in the registration section, but a newly created server branch is empty; it must be initialized. First, create a private local branch of the Nightshade source by entering the command below. If you already have a local copy of the Nightshade source then you may skip this command.

```
bzr branch lp:nightshade [local_path]
```

The `local_path` is any valid path on your local machine where the nightshade source tree will be stored. This step may take a few minutes or longer to complete depending on the available bandwidth. After completion, 'cd' into `local_path` and push the Nightshade source to your private server branch with the following command.

```
bzr push -use-existing lp:branch_path
```

Replace `branch_path` with the path of your server branch noted in step 2.

5 Working Branch

Now that your server branch contains a copy of the Nightshade codebase, you must branch it to create a local copy, or working branch, of your server branch. Use the following command.

```
bzr branch lp:branch_path [local_path]
```

This branch can be used for all your changes to the Nightshade code. The local branch of lp:nightshade created in step 4 may optionally be deleted, as it was only necessary for the initial import of the Nightshade code to your private branch.

6 Code

Changes can now be made to the working branch on your local machine. Changes will not be permanent until committed to the local tree and will not be visible on the server branch until 'pushed' to the server. In either case, changes will not be included in the master Nightshade tree until explicitly merged by a Nightshade administrator. After the changes are complete, enter the command below.

```
bzr commit -m "[descriptive comment]"
```

This causes the changes to be committed to the local branch. If you're not happy with a set of changes, they can be removed by the *bzr revert* command.

7 Push

After all changes are complete and unit tested, use the following command to send the changes to your server branch.

```
bzr push lp:branch_path
```

Now the changes are visible to everyone and can be reviewed by Nightshade developers. Log in to Launchpad and navigate to your branch's overview page. Click the link titled 'Propose for Merging' and enter lp:nightshade as the target branch. Include a detailed description of the changes in the description area. Development will reply by e-mail when the change has been merged or if additional changes are necessary.

8 Best Practices

8.1 Branch Management

If you contribute over the long term, you'll need to occasionally synchronize your personal branch with the Nightshade trunk. If they fall too far out of sync, Nightshade developers will be unable, or at least unhappy, to apply your changes. In the root of your local source tree simply use the merge command.

```
bzr merge lp:nightshade
```

This will pull all the latest changes from the Nightshade trunk into your local tree. If the command completes with no errors, commit the changes to your local branch and optionally push them to your server branch. Occasionally, there will be a non-trivial merge. That is, a section of code that was modified on both the trunk and your own branch. As a result, bzr can not automatically resolve the merge. Consult the Bazaar documentation regarding manual merge resolution. A merge of the trunk should be performed on at least a bi-weekly basis. It is also wise to perform one just before notifying Nightshade developers of your changes.

8.2 Coding Standards

There are a few things that should be avoided in all but a few situations.

- Preprocessor conditionals (i.e. `#ifdefs`). These types of conditionals should occur only in a few isolated places. They should not be used for conditional inclusion of features that are intended for all supported platforms. If a feature needs to be turned on and off then it is usually better to use configuration file settings. If you need to include/exclude code based on platform, use the methods on the `AppSettings` object instead.
- Formatting problems. Although certain portions of the code base are already inconsistent in this regard, please don't make it worse. Use a single tab character for indentation and 1TBS style braces.
- Method definitions in header files. With the exception of template methods, please avoid this. Again, there are lots of these in the code but they can cause problems. Keep declarations in headers and definitions in implementation files.
- Uninitialized members. Initialize all atomic members on class construction (e.g. floats, ints, and especially pointers).

Other issues to consider.

- Default copy constructors. How may your class be used? Can the default copy constructor successfully duplicate all members or will you need to implement one? Be sure you can answer these questions.
- Member variable prefixes. Although Nightshade does not follow a strict Hungarian like notation, we find it valuable to prefix class member variables with ***m_***. All new classes should follow this convention. For existing classes, please follow whatever convention is currently used in them for consistency.