PDB2PQR Release SOP

# Preface

This document explains the Standard Operating Procedure (SOP) to release a new version of PDB2PQR. This document is designed to make the process go more smoothly. It should NOT be viewed as a strict Methodology but rather as a list of suggestions and procedures that should be done or accounted for before a new release tar ball is posted on Source Forge and announced on <http://www.poissonboltzmann.org/>.

# Repository

## Branch

When preparing for a new release the trunk (or tag if a bug fix version needs to be released) should be branched into the branches folder. It should be named something obvious such as pdb2pqr-<new version number>. All testing should be done on a clean checkout of the branch. Fixes found during testing should be checked into the branch and merged into the trunk if appropriate. Merging fixes back into a tag is not required and probably not appropriate.

# Build

The branch should be tested to see if it configures and compiles correctly on all the target operating systems. Currently those systems are Linux and Mac OSX 10.6 and newer. Testing for all possible FORTRAN compilers is impractical and unnecessary. Future versions of pdb2pqr will no longer require FORTRAN for compiling the PROPKA component. Testing against the g95 FORTRAN compiler is sufficient. Testing against the GNU compilers for pdb2pka is sufficient.

Remember, these tests should be done on a fresh checkout the branch.

To configure and make a branch:

In the root directory of the branch run ‘./configure’

Verify that configure was able to configure the FORTRAN compiler for PROPKA. (This step will go away when we transition to PROPKA 3.0) Scroll up to the PROPKA section of the configure output and verify that it found a FORTRAN compiler.

Run ‘make’. This step should complete with warnings but no errors.

Run ‘make install’ to create and install the website with the default settings.

# Testing

## Automated Testing

After a successful build automated tests should be run. Run ‘make test’ and ‘make adv-test’. This will test out the most basic case and a more complex case with a pH setting and ligand file. Both tests should run without issue unless changes were made that affect the output of the program. (Version numbers and some atom ordering differences are ignored when comparing expected and actual results.)

## Command Line Testing

From the command line every option should be tested at least once. As there is little overlap in functionality between command line options, testing every combination is not needed.

To test --userff and --usernames make copies of the amber.dat and amber.names files and use them to test those command line options. --ligand can be tested with a mol file in the examples directory.

Version 1.8 will have example user force field and names files in the examples directory. Also, most or all of the command line options will be exercised with a new testing make target.

## Web Site Testing

Like command line testing every available option on the web front end should be tested. All options should also be tested with both opal and non-opal configurations. If we do not have opal set up on the Mac cluster and command line options have not changed the default opal service provided by NCBR may be used for opal testing.

Opal is configured with --with-opal for the pdb2pqr service and --with-apbs-opal for the APBS service.

Opal and non Opal configuration testing should configure the use of APBS. A properly installed and compiled copy of APBS is needed for both Opal and non Opal tests. In both cases the directory of the installed APBS executable must be referenced with the --with-apbs option when running configure.

Incompatible options and invalid files should be tried to ensure that errors are reported correctly by the web interface.

# Documentation

Documentation should be rebuilt and the newly generated files checked in before creating the final tar ball. The documentation must be built only after ‘make clean’ has been run. Any extra files created by the configure script will cause extra documentation pages for standard python modules to be created. (This already happens, but after configure it is worse.) Also it will create documentation pages for configuration py files such as aconf.py. This should be avoided.

Documentation is created by running the genpydoc.sh script in the doc directory.

After generating the new documentation the updated files should be checked in to the branch.

This should be the last thing done before packing up a tar ball so as to reflect any changes to the documentation made during testing.

# Build (Again)

If testing revealed any needed code changes a fresh checkout should be done with another build and automated tests. The main purpose of this is to ensure that you didn’t forget to check anything in and everything needed for deployment is in the branch.

# Deployment

## Tagging

The branch should now be branched into the tag directory of the repository (i.e. tagged) under an appropriately named directory (PDB2PQR-<version>).

## Packaging

After testing is complete and documentation updated another build is done it’s time to package it up.

Checkout a clean copy of the tag. Use the SVN export command to make a copy without the SVN metadata. The files should live in a folder that properly reflects the version of the release. For example “pdb2pqr-1.7.1a “

Tar and gzip with the following command:

tar –zcvf pdb2pqr-1.7.1a.tar.gz pdb2pqr-1.7.1a

## Release

The tar file should be uploaded to Source Forge. A new folder should be created to reflect the version number. The new tar ball should be uploaded to that folder. A copy of ChangeLog (which you should be keeping up to date!) renamed to PDB2PQR-<version>-ReleaseNotes.txt (example PDB2PQR-1.7.1a-ReleasNotes.txt) should also be uploaded to the same folder.

## Announcement

New releases should be announced on [pdb2pqr-users@lists.sourceforge.net](mailto:pdb2pqr-users@lists.sourceforge.net) and <http://www.poissonboltzmann.org/> with the ChangeLog text.