# Release Procedures
for SaberNet DCS 2.0

| | |
|---|---|
| **Title**: | SaberNet DCS Release Procedures |
| **Author**: | Seth Remington <sremington@saberlogic.com> |
| **Author**: | Matthew Ranostay <mranostay@saberlogic.com> |
| **Date**: | 2006-08-16 |
| **Revision**: | 1.3 |
| **Description**: | Documentation for procedures to be followed when creating and distributing a release of SaberNet DCS. |

**Contents**

## Version Numbers

SaberNet DCS has a version number made up of three parts: a MAJOR, MINOR, and MACRO number. For example:

```
2.0.1
| | |
| | --> MACRO
| ----> MINOR
------> MAJOR
```

There may be a suffix appended onto the version in the case of an Alpha or Beta release. For example:

```
2.1.0b3  <--- Beta 3
```

The numbers should be incremented according to the following rules:

1. The MACRO number should be incremented by one for every release where the MINOR number does not change. If the MINOR number is incremented then the MACRO number gets reset to 0.
2. The MINOR number is the stable/development indicator and should get incremented when a new stable version is released. Even numbers are stable trees and odd numbers are development trees. So for example, 2.0 would be the stable tree that patches would be applied to and 2.1 would be the development tree that would coexist with the stable tree. When the 2.1 tree became stable it would be incremented to 2.2 and a new 2.3 development tree would be created.

3. The MAJOR number should probably only be incremented for a major re-write ;)

## Updating Version Numbers

The MAJOR, MINOR, and MACRO numbers are set in the sndcs_common/__init__.py file using variables of the same name. Since this is the "common" package containing code used by both the clients and server, both the client and the server will share the same version number. There is a mechanism in the client to make sure the clients are connecting to a server version high enough to support the client. *(NOTE: This is stored in sndcs_client.gtk.REQUIRED_SERVER_VERSION)*

If appending an Alpha or Beta suffix to the MACRO don't add a space between the number and the suffix because the tarball produced by distutils would also contain a space.

## Release Procedure

When creating a new release the following procedure should be followed:

1. Make sure you have the latest sources by running "cvs update".
2. Update the version number in sndcs_common/__init__.py as described in Updating Version Numbers.
3. Add a note to the ChangeLog indicating the release
4. Add a note to NEWS (if it is an important enough release) with a summary of the big changes since the last release.
5. Commit those changes to CVS with "cvs commit".
6. Tag the release in CVS using the command, *'cvs tag release-name'*. If this is an increment of the MINOR number the tag should be a branch (*'cvs tag -b release-name'*) and a new development release (and tag) should be made immediately as well. (See Version Numbers for more info.)
7. Generate the docs: cd docs; ./generate_docs.py (Requires python-docutils and pdflatex.)
8. Run "python setup.py sdist --formats=gztar,bztar,zip" to create the source tarball in gzip, bzip, and zip formats. The resulting tarballs will be in the "dist" directory.
9. Run "python setup.py py2exe" to create Windows distribution, then open "sndcs.iss" with InnoSetup (Project Link: http://www.jrsoftware.org/isinfo.php) and then click "compile". The resulting binary will be in the "Output" directory.
10. Run "python setup.py bdist_rpm" to create a rpm package. The resulting package will be in the "dist" directory.
11. Currently distutils doesn't support Debian packages, so you must create a rpm package and then run "alien" on it.
12. FTP the package(s) onto upload.sourceforge.net in the "incoming" directory.
13. Go to SourceForge's administrative File Releases section and add a release and attach the files.