# 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:	2014-11-03
Revision:	1.3.2.15
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. Add any necessary administrative notes to the RELEASE_NOTES
6. Edit the debian/changelog and add an entry with the new version number (use 'date -R' to get the date)
7. Commit those changes to CVS with "cvs commit".
8. 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.)
9. Generate the docs: cd docs; ./generate_docs.py (Requires python-docutils and pdflatex.)
10. 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. (May want to delete MANIFEST if working from an existing sandbox and a new file has been added to the source tree. I've had problems with the MANIFEST file not being rebuild from the MANIFEST.in file and not including new source files.)
11. Run "python setup.py bdist_rpm" to create a rpm package. The resulting package will be in the "dist" directory.
12. Upload the package(s) onto SourceForge (https://frs.sourceforge.net/webupload)

13. Go to SourceForge's administrative File Releases section and add a release and attach the files.

14. Update the "Download Latest Version" section on the main page of the website.

15. (Optional) Submit news item to SourceForge

To create the Debian packages

1. Take the tarball created above, extract it, and cd into the extracted directory.

2. We are going to create very basic config file. Run "python configure_sndcs.py", enter 'b' to configure both the client and the server, and press <enter> to accept all of the defaults. (May want to temporarily move away any existing config files in /etc/sndcs so no live config options get pulled in.)

3. Edit setup.py and change BUILDING_DEB = True

4. Edit the debian/changelog file. Timestamp can be created with the 'date -R' command.

5. Copy the Debian specific start script $(WEBWARE_DIR)/WebKit/StartScripts/Debian overtop of the existing sndcs2httpd file.

6. Run "dpkg-buildpackage -rfakeroot" to create the Debian packages (Note: If get a "debian/rules: Permission denied" error make sure to run "chmod +x debian/rules")

To create the win32 client release

Note: requires all client dependencies to be installed as well as InnoSetup. The latest versions of gtk+, libglade, pygtk can be found at http://ftp.gnome.org/pub/gnome/binaries/win32/

1. Unzip the tarball

2. Open a DOS prompt and 'cd' into the extracted directory

3. We are going to create very basic config file. Run "python configure_sndcs.py", enter 'c' to configure the client, and press <enter> to accept all of the defaults. Make sure the config doesn't pick up any stray values (i.e. namespace) from a previous live install.

4. Recompile the internationalization .po files into .mo files using the win32 version of poedit. The Linux compiled versions don't seem to work on win32.

5. Run the command 'python win32clientsetup.py py2exe' NOTE: the bundle_files option is turned off because it caused the app to crash

6. This will create a win32client directory.

7. From the GTK+ runtime installation copy the lib, etc, and the share/themes directory into the win32client directory

8. Copy the jpeg62.dll file (http://gnuwin32.sourceforge.net/packages/jpeg.htm) into the win32client directory

9. Edit the sndcs2.iss file and correct the version number and file paths

10. (optional) Add win32client.txt file with post install instructions

11. To get gtkspell to work on win32 is a bit of a hack until there is proper win32 support for gtkspell. Grab the libgtkspell.dll file from the Pidgen installer and copy into the win32client directory. "Borrow" The gtkspell.py file from the Gajim project (http://trac.gajim.org/browser/trunk/src/gtkspell.py). This is a wonderful hack that is using ctypes to call the exposed functions in the DLL. I renamed the file to win32gtkspell.py and put it in the sndcs_client/gtk directory. (There may already be a copy there.) Made a couple of small changes to look for the DLL in the right place. Also need to have Aspell (http://aspell.net/win32/ or use what the Pidgin installer installs). Copy the "bin", "data", and "dict" directories into the win32client directory. The "bin" directory will need to be in the path so that the aspell.exe can be found. The InnoSetup installer is set up to add this directory to the system PATH. It uses the SetEnv.exe utility which also must be copied into the win32client directory.

12. Compile and build the installer. It will be in a directory named Output.

13. Rename the installer to sndcs-<version>-client.exe

14. Go to SourceForge's administrative File Releases section and attach the file.

To create the win32 server MySQL release

Note: requires all server dependencies to be installed as well as InnoSetup

1. Unzip the tarball

2. Have Webware installed somewhere on the system

3. Open a DOS prompt and 'cd' into the extracted directory

4. We are going to create very basic config file. Run "python configure_sndcs.py", enter 's' to configure the server, and press <enter> to accept all of the defaults. Enter the current location of the Webware installation

5. Run the command 'python win32serversetup.py py2exe' NOTE: the bundle_files option is turned off because it caused the app to crash

6. This will create a win32server directory.

7. Copy the Webware installation directory into the win32server directory and rename it "Webware" (no version number)

8. Set the webware directory in the config file to "./Webware" (NOTE: probably don't need this anymore since in sndcs/__init__.py when we are looking up the Webware directory we explicitly set it to the Webware dir if we are a frozen exe)

9. Run the command 'python win32serversetup.py py2exe' again

10. Copy the sndcs.mkmodel directory into the win32server directory

11. Change line 11 of WebKit/Cookie.py to "if (pyVer and (pyVer[:2] >= (2, 2)) and not hasattr(sys, 'frozen')):" (See: http://www.voidspace.org.uk/python/weblog/arch_d7_2006_02_25.shtml#e236)

12. *** You have to do this step (at least copy 'sndcs2_web' and maybe 'wap') even if you are using a source tree you've used previously to create an .exe with because no DCS Web updates will take effect otherwise *** Copy everything except CVS, .cvsignore, AppServer, and sndcs2httpd from the sndcs/web directory into the win32server directory (Delete the CVS directories inside the directories that are copied. May need to delete some cruft files inside the ErrorMsgs, Logs, and Sessions directories but need to leave the .cvsignore files there so that InnoSetup doesn't complain about not finding matching files.)

13. Make sure there are no left over passwords in the config file.

14. We need to update the logging.conf file to only use the NTEventLogger. Set [handlers] keys = NT and [logger_root] handlers = NT

15. From the GTK+ runtime installation copy the lib, etc, and the share/themes directory into the win32server directory. Watch because there may already be directories of the same name there so you will need to copy all of the directory contents instead of just copying the whole directory.

16. Compile and build the installer. It will be in a directory named Output.

17. Rename the installer to sndcs-<version>-server-mysql.exe

Internationalization

1. All translatable strings in the app should be wrapped with _('')

2. To extract translatable strings from the glade file perform the following commands. This will create a C header file (I know... just ignore) where "**N_**" is the keyword to the translatable string.

- cd sndcs_client/gtk
- intltool-extract --type=gettext/glade sndcs2.glade

1. Use poedit to create new .po file for language in i18n directory. "e.g. i18n/xx/sndcs.po" (Use '../..' for basepath and '.' as searchpath) (Add **N_** as a source keyword to pick up the translatable string in the Glade .h file)

2. Add the new .mo file to setup.py and win32clientsetup.py data_files for installation

3. On win32 all of the .mo files need to be recompiled using the Windows version of poedit or els they don't seem to work.
4. Test with "LANGUAGE=xx sndcs_gtk"
5. On win32 test with "set LANG=xx" and then "sndcs_gtk.exe"

Hosted by SourceForge

Docutils System Messages

system-message
ERROR/3 in `RELEASE_PROCEDURES.txt`, line 121
Unknown target name: "n".

system-message
ERROR/3 in `RELEASE_PROCEDURES.txt`, line 126
Unknown target name: "n".