Introduction

FreeSWITCH? is available for source compilation on Unix and Linux distributions as well as Windows. Prebuilt binaries are available for installation on some Linux and Unix distributions as well as Windows.

On this page, we'll tell you how to get Freeswitch, and how to install it, in several different ways.

NOTE: If you're new to FreeSWITCH? consider purchasing the FreeSWITCH? book and be sure to read this introductory article: http://www.linuxpromagazine.com/Issues/2009/106/TALK-SOFT.

NOTE: Work is underway to refactor these instructions to reflect the changes in preferred deployment method as of the 1.2 release; please try to avoid changes to this page while this notice is up [since 4-Sep-12] -- jra

Contents

Operating Systems

Linux and Unix

This page is dedicated to Linux and Unix distributions. Everything required to download and install FreeSWITCH? can be found below. Experienced Linux users may want to check the Linux Quick Install Guide.

Mac OS X

The comprehensive Installation and Setup on OS X guide has detailed installation steps and instructions for setting up a 24x7 Mac. OS X is based on BSD Unix so there are similarities to Linux installation.

Windows

Information for source and binary installation is found at Installation for Windows which also contains quick start information. Very little information on this page applies to Windows other than the Source Options.

Preparation

Directory Permissions

FreeSWITCH? can be placed in any directory. However, for Unix and Linux source is typically installed in usr/local/src/freeswitch and binaries in /usr/local/freeswitch. Some systems drop local and use usr/src. These directories are usually root protected after initial OS installation. You should change the owner and group so that root is not needed for installation or execution of FreeSWITCH. You should create and set permissions on /usr/local/freeswitch in this fashion:

```
sudo mkdir /usr/local/freeswitch
sudo chown -R newid:newgroup /usr/local/freeswitch
```

Source Options

The FreeSWITCH? team **strongly** recommends installing from the latest **Git tree** which is extremely stable! Many reported bugs in older versions of FreeSWITCH? are fixed in the current Git version. The developers **require** testing an issue on the current Git version prior to filing a bug report.

FreeSWITCH? can be downloaded from multiple locations, depending on your desired tradeoff between stability and availability of new features. The best method to download the Git source is via Git. The examples in this document use Git.

Which version should I use?

The stable 1.2 series is recommended for use. This is available in Git but also as a tarball. Although the tarball has the latest official stable release, there may be more recent patches to the stable branch in the v1.2.stable Git branch, therefore Git is still the recommended option.

The stable 1.2 series is available in Git branch v1.2.stable. Patches to the 1.2 series will be made here and periodically released as the next 1.2.x version. Since not every patch will mean a newer version you will get the very latest patches to the stable series by using this branch. This branch is a moving target - the checkout date and revision will form part of the version number.

The current stable release in a tarball of FreeSWITCH is 1.2.10, released on 20-May-2013. Older releases and other files are in the archive, along with digital signatures that you can use to verify your download.

If you wish to try an even newer experimental version you can use the Master branch from Git. New features will only be added to the Master branch. This will have the newest, latest, and greatest features, but due to upgrading of the bundled libraries it *might* be less stable than v1.2.stable for a short while. If you use this branch you should test the installation before allowing users to use it to confirm it is stable. You can then use 'git checkout' to install the same tested revision on all your systems.

NOTE that this is a formal change of development practice; Master used to be the recommended deployment target for all users, and now the tarballs and the Git stable branch serve that purpose. Older documentation/tutorials may still exist recommending using Git, but these will now be out-of-date.

Any previous 1.0 release (eg 1.0.4, 1.0.6) is now ancient and will contain a range of bugs that have since been resolved in newer releases, so you should seriously consider upgrading to 1.2 stable.

When upgrading please review the Release Notes page to see whether there are any behaviour changes since previous versions.

Recommended: Git

Git provides access to the latest features and bug fixes added throughout the day. Most often you will only use the **clone** and **pull** commands. There is an excellent Online Git Book available in multiple languages. For help installing Git see Git_Install. The following creates a freeswitch directory and downloads a "clone" of the current source:

```
cd /usr/local/src
git clone git://git.freeswitch.org/freeswitch.git
OR
git clone -b v1.2.stable git://git.freeswitch.org/freeswitch.git
(SEE BELOW)
```

Note: git reset --hard <commit_id> - will revert all local changes and reset the working copy to the specified commit.

Selecting version

There are 2 branches in Git - Master and v1.2.stable

v1.2.stable is the branch for the 1.2 series. It will be the most recent 1.2.x release, plus any patches made since that release.

```
git clone -b v1.2.stable git://git.freeswitch.org/freeswitch.git
```

Master is where the development for the future 1.4 release is being done. New features are added to this branch, but it may be less stable than 1.2.stable at times.

```
git clone git://git.freeswitch.org/freeswitch.git
```

You can also switch branches on an already cloned Git tree:

```
git checkout v1.2.stable
git checkout master
```

If you get an error on GIT PULL on v1.2.stable saying `You asked me to pull without telling me which branch you want to merge with`
Then run this to add the definition for that branch to .git/config:

```
git branch --set-upstream v1.2.stable origin/v1.2.stable
```

Reverting to an Earlier Commit in Git

Sometimes when things go terribly wrong with the latest commits (regression-type bugs), you can always revert to an earlier commit with git. Simply type in:

```
git checkout <revhash>
```

where <revhash> is the hash of an earlier commit. You can find the commit hashes here.

After reverting to a previous version, it is strongly recommended that you (in the following order):

- Report the issue to the FreeSWITCH JIRA
- Run 'git clean -d -f -x' on the root directory of the sources of FreeSWITCH to remove non-git files and spare compilation files.
- Run 'rm -rf /usr/local/freeswitch/{lib,mod,bin}/*' to remove the current binary files in those 3 directories.
- Re-bootstrap the code
- Re-run the configure process

... before you re-build the code from an earlier revision.

To go back to the very latest commit from the development branch, simply type in

```
git checkout master
```

Revert only certain commit

To reverse only the one commit change on newer build

```
git log -1 -p <revhash> | patch -p1 -R
```

freeswitch-contrib

Contributed source may obtained from the freeswitch-contrib tree as follows (this will compile and install updates):

```
 git clone git://git.freeswitch.org/freeswitch-contrib.git
```

Sample configs can be downloaded as well:

```
git clone git://git.freeswitch.org/freeswitch-sample-configs.git
```

The Latest Build site latest.freeswitch.org is currently empty.

Compiling and Installation

Ensure that the required prerequisites are installed. Also check if there are notes for your distribution in Distribution Information and Installation.

Switch to the src/freeswitch Directory

The remaining commands are executed from the directory containing the FreeSWITCH? source (use whatever directory the source is located in):

```
cd /usr/local/src/freeswitch
```

Installed from Git

If Git was used to download, the configuration files must be built before the first compile. ./bootstrap.sh creates many files including modules.conf. Once this is performed it's not normally required to be performed again. If you're installing from a tarball, this script was run before packaging, and you shouldn't need to run it.

```
./bootstrap.sh
```

You can use multiple cores for your bootstrap/config/build, by specifying it at the start, e.g.: (note this may make build errors harder to spot)

```
./bootstrap.sh -j
```

However, its not recommended to use the -j flag for either bootstrap.sh or make, on bootstrap.sh on a slow machine (older machines or say a raspberry pi) -j will fail randomly due to how it is handled. on the make this may also randomly fail. (quoted from Ken Rice in FS-4891)

Configure for Compilation

./configure sets compilation options. Invoke the following command to configure for compilation:

```
./configure
```

You can have the checks cached for each module, by using:

```
./configure -C
```

If you want to install FreeSWITCH to a non-standard location you can simply append the --prefix option to the configure script as shown in this example:

```
./configure --prefix=<example of non standard location>
```

Edit modules.conf

The modules.conf file contains the list of modules to be compiled, some are commented as they are not required. As your FreeSWITCH? configuration becomes more complex additional modules may be required. **If you plan to test the sample IVR after installation, FLITE must be added to the compile as follows:**

1. Use an editor to navigate to and edit /usr/local/src/freeswitch/modules.conf (or whatever directory it was placed into)
2. Search for "flite" and remove the # to uncomment.
3. Save the file.

If this is your first experience with FreeSWITCH? don't add more than FLITE. Adding functions may result in additional prerequisites that must be installed.

Building FLITE requires about 450 megabytes of addressable memory, so if you are building on a device with limited RAM, you'll need to create a swap file (a.k.a. Virtual Memory or paging file) of sufficient size to allow the build to proceed, or simply do not include mod_flite in the build. (Please refer to your operating system's documentation for instructions on how to create a swap file.) All other modules in the default configuration will build successfully on devices with less than 100MB of available RAM.

Compile and Install

Invoke the following commands to compile and create the binaries:

```
make && make install
```

Note: "make install" does not overwrite existing configuration files in freeswitch/conf if a freeswitch.xml file already exists in conf.

Compile and Install Sounds

There are IVR sound prompts and music on hold files that are optional but must be compiled and installed to run the sample IVR. There are four versions available:

- sounds-install moh-install (8 kHz)
- hd-sounds-install hd-moh-install (16 kHz)
- uhd-sounds-install uhd-moh-install (32 kHz)
- cd-sounds-install cd-moh-install (48 kHz)

The cd sounds are recommended since all the sampling rates are provided resulting in fewer problems. Invoke the following command:

```
 make all cd-sounds-install cd-moh-install
```

Ready to Test!

Switch to the freeswitch/bin directory and try some stuff! When you're done read the Getting Started Guide to learn how to configure FreeSWITCH?.

```
cd /usr/local/freeswitch/bin
./freeswitch
```

Upgrading and Re-installation

Updating the Source

If FreeSWITCH? was previously installed using a Git based installation, and only a clean update to the latest tree is required, issue this command:

```
make current
```

The **make current** command will clean your build environment, do a git pull, and then do a make install for you.

Rebuilding

After doing a "git pull" or changing source files, it may be desirable to clean out your build area:

```
make clean modwipe
```

Alternatively, you can execute "make current" which will handle the git pull, clean up, and rebuild in one step:

```
make current
```

Make current *completely'* cleans the build environment and rebuilds FreeSWITCH? so it runs a long time. However, it will **not overwrite** files in a pre-existing "conf" directory. Also, the clean targets leave the "modules.conf" file.

To check if there are any new modules execute:

```
diff build/modules.conf.in modules.conf
```

If a previously disabled module is now enabled, or there is a requirement to rebuild a single module, execute the following (replace mod_name with the name of the module to build):

```
 make mod_name-install
```

Upgrading on-the-fly

It is highly inadvisable to upgrade FreeSWITCH on-the-fly (aka while running) as this can cause unexpected behavior and may potentially hang the entire system.

However it can be done by following these steps below:

After recompilation while FS is still running you can still install:

```
make install
```

Note, this will automatically install, and it even first cleans the build dir and the installation dir, which should even work while FS is running, provided no modules are reloaded or loaded:

```
make current
```

To make the changes of the new build take effect, FS has to be restarted:

```
/usr/local/freeswitch/bin/fs_cli -x shutdown elegant restart
```

This will wait for all traffic to end, yet still allow new traffic. If you want to restart as soon as possible, you can use:

```
/usr/local/freeswitch/bin/fs_cli -x shutdown asap restart
```

This will restart FS as soon as there is no more traffic and it will not allow any new traffic to make this happen as soon as possible.

If you want a faster rebuild, you can make use of make's dependency system. Note that this will not make the version command reflect the actual git revision, as the modules making use of it will not be rebuild if there is no change of them in the git tree.

```
make sync all install
```

Be aware some things might not be rebuilt which might be important to be and there might occur compile errors / runtime errors / segfaults. In my experience, this is rare, but if you encounter any problems, a clean rebuild is advisable to see if the problem persists.

To always reflect the actual git version and revision in every module, there should be a function switch_core_get_version() which would be in an object always rebuilt and giving the actual version in return. So far a preprocessor variable is used.

Troubleshooting

Compile Errors

If you encounter compile errors or other bugs be sure to test it again with the latest Git version, then file a bug report at http://jira.freeswitch.org or email freeswitch-dev@lists.freeswitch.org (we prefer that you use Jira so that we can track all issues). Remember FreeSWITCH? development is always ongoing and requires

contributions from everybody.

Compiling 32-bit Target on 64-bit System

To compile a 32-bit binary on a 64-bit system:

```
CFLAGS=-m32 CXXFLAGS=-m32 LDFLAGS=-m32 ./configure
```

Common Prerequisites

Mandatory

These mandatory prerequisites provide for compiling the standard FreeSWITCH? installation and test the supplied configuration and sample IVR. They are sufficient for many production systems.

- **GIT** or **WGET**
- **AUTOCONF**
- **AUTOMAKE**
- **GAWK**
- **GCC-C++**
- **LIBJPEG-DEVEL** Used by mod_spandsp for basic codecs
- **LIBTOOL**
- **MAKE**
- **NCURSES-DEVEL**

Optional

There are optional modules that can be built and if so may require one or more of the following;

- **curl-devel** for mod_xml_curl
- **expat-devel**
- **libtiff** for fax support
- **libx11-devel** for Mod_skypopen
- **ODBC or UNIX-ODBC and ODBC-devel** see the ODBC page for information
- **OpenSSL** (libssl-dev / openssl-devel) for SIP SSL & TLS and Dingaling
- **python-devel** for the python interface
- **ZLIB and ZLIB-devel**
- **libzrtp** ZRTP encryption support, see the FreeSWITCH? ZRTP page

Distribution Information and Installation

Arch Linux

Arch Linux uses the pacman package manager. it also has a ports-like build system called Arch Build System.

To build FreeSWITCH from the latest git, you use the freeswitch-git package. This package is documented on the Arch Linux Wiki

The packages for FreeSWITCH? stable release (and the zaptel dependency for mod_openzap) are available at the AUR repository:

- Zaptel
- FreeSWITCH

To build these packages follow the instructions of the Makepkg build system.

Alternatively, if you have an AUR helper installed, you may install with

```
packer -S freeswitch-git

yaourt -S freeswitch-git
```

CentOS & RedHat Enterprise Linux

Tested Nov 26, 2010 with CentOS 5.5 x86_64
There were no issues during installation and testing other than EPEL requirements. Installation of Git required adding the EPEL repository to obtain Git.
Tested May 03, 2013 with CentOS 6.4 x86_64
There were no issues during installation and testing.

Note that as of Aug-2012, CentOS6 older than 6.3 may display performance problems on higher-performance installations; if you're planning on needing lots of performance, Ken Rice recommends you stick with 5.x for the nonce. (See the RHEL6 notes at the end of this page for more)

Prerequisites

Use the yum package manager tool or Add/Remove Applications menu function to add the following packages to a basic installation:

- autoconf
- automake
- gcc-c++
- git-core
- libjpeg-devel
- libtool
- make
- ncurses-devel

```
yum install autoconf automake gcc-c++ git-core libjpeg-devel libtool make ncurses-devel pkgconfi
```

To install the optional packages needed to enable all FreeSWITCH modules to be built:

```
yum install unixODBC-devel openssl-devel libogg-devel libvorbis-devel curl-devel libtiff-devel l
```

You will also need to install the following packages if you need the modules listed below:

- mod_event_zmq needs "openssl-devel" and "libuuid-devel" packages.
- mod_erlang_event needs "openssl-devel" and "erlang-devel" packages.
- mod_flite needs "openssl-devel" and "bzip2" packages.
- mod_osp needs "openssl-devel" package and the "OSP Toolkit" see mod_osp.
- mod_snmp needs "openssl-devel" and "net-snmp-devel" packages.
- mod_xml_ldap needs "openssl-devel" and "cyrus-sasl-devel" packages.

YUM Based Installation

FreeSWITCH is now available via yum. You will need to add the FreeSWITCH yum repository

To install the FreeSWITCH Repo

```
rpm -Uvh http://files.freeswitch.org/freeswitch-release-1-0.noarch.rpm
```

Then to install FreeSWITCH with the generic vanilla example configs

```
yum install --nogpgcheck freeswitch-config-vanilla
```

This will give you a base working FreeSWITCH. At the time of this update (2012-FEB-13) these RPMs are Beta but should be working fine. Please report any bugs in them via JIRA

After that you need to install FreeSWITCH sound related files. For that you need to issue following command.

```
yum install freeswitch-sounds*
```

Automatic Startup At Boot

To get FreeSWITCH to start up automatically at system start, just copy the FreeSWITCH init script into the /etc/init.d directory. An example init script is included in the git repository, under the build directory, named as freeswitch.init.redhat. You may need to modify the script to get FreeSWITCH starting up from the directory where the binaries are installed. After the file has been modified to suit your needs, simply run this command:

```
chkconfig --add freeswitch && chkconfig --levels 35 freeswitch on
```

This procedure will also work exactly the same way on Fedora systems.

Installation

Continue with the normal installation steps or the Linux Quick Install Guide Downloads section.

Release(es) 6 and Later

A change by Red Hat in RHEL 6 and later releases was to compile the kernel to run tickless by default. As previously brought up on the mailing lists, it is recommended that the kernel's tickless feature should be disabled for optimum performance when running FreeSwitch under the newer operating system's kernel. You can disable the tickless feature by appending nohz=off to your boot options under GRUB. The GRUB start up configuration file is located under /boot/grub or as /etc/grub.conf. Add it to the line beginning with "kernel". This is also applicable to Fedora releases 8 and later.

Also, when using release 6 and later, make sure to configure with "./configure --without-pgsql", this is to make sure that FreeSWITCH uses it's own curl library, instead of the system provided, and that it doesn't try to use the system provided postgresql libs. If using the system provided versions linking errors will occur. Hopefully this will be auto detected in a near future. Related Jira issues were: FS-3384, FS-3630, FS-3384 and FS-3393.

In addition, FS may not behave 100% correctly in CentOS 6.x, please review Jira issues: FS-4396, FS-4316, FS-4291

UPDATE: We have anecdotal evidence that CentOS 6.3 does not seem to have the problems listed above. If you experience any of the above symptoms with CentOS 6.3 please comment on the Jira cases listed.

Debian

Debian packages

There are precompiled Debian packages available for several Debian releases, including Squeeze, 32 bit and 64 bit. Currently Freeswitch is not in the main Debian repositories because it's very hard to make the Freeswitch code & build system comply with the rules for that.

- Add the following to /etc/apt/sources.list: (for Debian Squeeze)

```
deb http://files.freeswitch.org/repo/deb/debian/ squeeze main
```

- Import the repo signing key

```
 curl http://files.freeswitch.org/repo/deb/debian/freeswitch_archive_g0.pub | apt-key add -
```

or

```
 gpg --keyserver pool.sks-keyservers.net --recv-key D76EDC7725E010CF
 gpg -a --export D76EDC7725E010CF | sudo apt-key add -
```

- Update apt-get.

```
apt-get update
```

- Search for the FreeSWITCH? packages in apt. Choose from the packages listed to complete your FreeSWITCH? installation.

```
apt-cache search freeswitch
```

There's also a meta package to get a reasonably running installation.

```
apt-get install freeswitch-meta-vanilla
```

The packages do not set up the configuration in /etc/freeswitch, and you need to copy files manually:

```
cp -a /usr/share/freeswitch/conf/vanilla /etc/freeswitch
```

Create your own Debian packages

If the precompiled Debian packages don't suite your needs you can build your own custom debian packages.

A new version of the above script which also installs all dependencies and downloads and builds the new seperate sounds & music from github can be found here. Tested on Squeeze with Freeswitch 1.2.10.

Packages can either be installed locally, or upload them to your local Debian repository and install them via APT.

```
 $ dpkg -i $DEB_PACKAGE_NAMES
```

This also allows you to build once and install the packages on multiple servers.

From source

Tested June 28, 2012 with Debian "wheezy" (public beta 18. June 2012) for Raspberry PI ARMv6
There wasa problem using apt-get, gone after sudo apt-get update --fix-missing, install took several hours, compilation without mod_flite, worked fine, find debian .img here: [1]

Tested Nov 27, 2010 with Debian 5.0 AMD64
There were no issues during installation and testing. Benefits from a custom kernel with "CONFIG_HZ_1000=y" and "CONFIG_HZ=1000" defined to avoid performance issues. Follow Debian guidelines [2].

Prerequisites

There are a variety of Debian package manager tools such as dpkg, apt, aptitude, and synaptic. The synaptic GUI tool was used for this installation. The System menu Add/Remove Applications function does not offer the required packages, Add the following to a basic Debian installation:

- autoconf
- automake
- devscripts
- gawk (mawk does not work)
- g++
- git-core
- libjpeg-dev
- libncurses5-dev
- libtool

- make
- python-dev
- pkg-config
- libtiff4-dev
- libperl-dev
- libgdbm-dev
- libdb-dev
- sudo

```
apt-get install autoconf automake devscripts gawk g++ git-core libjpeg-dev libncurses5-dev libto
```

Note some modules might require additional dependencies.

Installation

Continue with the normal installation steps or the Linux Quick Install Guide Downloads section.

Fedora

Tested Nov 12, 2011 with Fedora 16 x86_64
Most modules under Fedora 16 compiles without too much issues with the latest git

On Fedora 14 systems the following may happen:

At this time, the spidermonkey part of compilation fails with many errors such as:

```
"jsapi.c: src/jstypes.h no suitable type for jsint8/JSUint8"
"Fatal error: can't create src/.libs/jsapi.o: permission denied"
```

Two days of time could not resolve this issue. When it is this will be updated. This is apparently a problem that started with other packages and distributions starting October 2010.

FreeBSD

STOP: If you install using the traditional method (git clone/bootstrap/configure/compile) you MUST use gmake. freeswitch will not compile correctly using BSD make.

Tested Nov 30, 2010 with FreeBSD 8.1 (i386)
Tested Mar 22, 2011 with FreeBSD 8.2-RELEASE (amd64)
Tested Jan 12, 2012 with FreeBSD 9.0-RELEASE (i386)
There were no issues during installation and testing, when building from Git source.

Installing via FreeBSD ports collection (TESTING ONLY!): This way all dependencies are downloaded and installed automatically. Currently this port uses freeswitch-1.0.6.tar.gz and **not** the latest git. For GIT see below. This port is maintained by user *rmeese* on #freeswitch and #fusionpbx

```
portsnap fetch update
cd /usr/ports/net/freeswitch-core
make config (enable/disable the modules you need)
make install clean
```

Installing FreeBSD dependencies: AUTOCONF, AUTOMAKE, GCC, GIT, GMAKE, GNUMAKE (installed automatically as a dependency to autoconf), GLIBTOOL, LIBNCURSES, WGET. LIBJPEG is needed for mod_spandsp (installation instructions needed)

```
pkg_add -r autoconf262
pkg_add -r gcc34
pkg_add -r automake19
pkg_add -r git
pkg_add -r gmake
pkg_add -r libtool
pkg_add -r ncurses
pkg_add -r wget
pkg_add -r pkg-config
```

Get source with Git

```
cd /usr/src
/usr/local/bin/git clone  git://git.freeswitch.org/freeswitch.git freeswitch-upstream
cd freeswitch-upstream
```

Build Process After downloading the source with git or with wget

```
./bootstrap.sh
```

Edit modules.conf to enable or disable desired modules.

To avoid failing build because of libodbc:

```
pkg_add -r unixODBC

setenv LDFLAGS -L/usr/local/lib
setenv CPPFLAGS -I/usr/local/include
```

Make sure you are using gmake and not make, it does not build properly with make at the moment on FreeBSD:

```
./configure
gmake install
gmake samples
gmake sounds-install
gmake moh-install
gmake hd-sounds-install
gmake hd-moh-install
```

Quick note on installations requiring spandsp and failing because of libtiff (On FreeBSD):

1. Install tiff from ports.

```
pkg_add -r tiff
```

2. Bash

```
export LDFLAGS=-L/usr/local/lib ; export CPPFLAGS=-I/usr/local/include
```

Or other shells.

```
setenv LDFLAGS -L/usr/local/lib
setenv CPPFLAGS -I/usr/local/include
```

Note: LDFLAGS=-L/usr/local/lib can cause problems if you have a library located there and FreeSWITCH uses a library named the same, in which case the FS build will link against your local lib instead of the desired FS-built lib. An example in FS is $(switch_srcdir)/libs/udns which is used in mod_enum. The FreeBSD port dns/udns installs libudns in /usr/local/lib.

3. Compile spandsp

```
( cd libs/spandsp ; ./configure ; gmake install )
```

4. Return to your normal FreeSWITCH? build

PS: export will work if using bash. For other shells step 2 might differ. (like setenv LDFLAGS -L/usr/local/lib , setenv CPPFLAGS -I/usr/local/include)

Threading Library

There are some known issues with FreeBSD's libpthread implementation and APR that can affect throughput at high volume. A possible fix is to use libmap.conf to remap FreeSWITCH™ and its libraries to use libthr (the better performance threading library)

/etc/libmap.conf

```
[freeswitch]
libc_r.so.5 libthr.so.2
libc_r.so.6 libthr.so.2
libpthread.so.1 libthr.so.2
libpthread.so.2 libthr.so.2
```

I (Vagabond) haven't tested that this actually fixes the throughput issues because I don't have an environment where throughput is that high (50cps+ according to anthm). If anyone does test this, please update this page with your experiences. Some additional information on this problem can be found here and here, in addition please read the libmap.conf man page for your system and be aware of the scheduler change as of 7.1 to ULE by default (this might make things better or worse, I can't find any numbers for the "new" ULE scheduler, only 2006-7 numbers).

Note: If you use fs_cli and experience regular hangs of the client, you may wish to add an identical entry for it.

Note: In FreeBSD 7, libpthread has been removed and libthr is the default implementation.

FreeBSD rc.d script

However the default installation comes with a rc-script, it doesn't meet the requirements of the FreeBSD rc.d framework. I've created (or actually copy/paste it from Postfix) a script that does meets the rc.d requirements. Create the example below in /usr/local/etc/rc.d and give it the name 'freeswitch'

/usr/local/etc/rc.d/freeswitch

```
#!/bin/sh
#
# PROVIDE: freeswitch
# REQUIRE: LOGIN cleanvar
# KEYWORD: shutdown
#
# Add the following lines to /etc/rc.conf to enable freeswitch:
# freeswitch_enable:       Set it to "YES" to enable freeswitch.
#                          Default is "NO".
# freeswitch_flags:        Flags passed to freeswitch-script on startup.
#                          Default is "".
#

. /etc/rc.subr

name="freeswitch"
rcvar=${name}_enable

load_rc_config $name

: ${freeswitch_enable="NO"}
: ${freeswitch_pidfile="/usr/local/freeswitch/run/freeswitch.pid"}

start_cmd=${name}_start
stop_cmd=${name}_stop

pidfile=${freeswitch_pidfile}

freeswitch_start() {
        /usr/local/freeswitch/bin/freeswitch ${freeswitch_flags}
                echo -n "Starting FreeSWITCH: "
}

freeswitch_stop() {
        /usr/local/freeswitch/bin/freeswitch -stop
}

run_rc_command "$1"
```

After creating the file it's time to change the permissions, otherwise it can't be executed. Go to /usr/local/etc/rc.d and issue the following command

```
chmod u-w,ugo+x freeswitch
```

Now it's time for calling the script from /etc/rc.conf. Put the two line below in the rc.conf file. The first one executes the startup script itself, the second one pipes the parameters.

```
freeswitch_enable="YES"
freeswitch_flags="-nc"
```

In this example FreeSWITCH? gets started with the parameters -nc (no console, you can access the console later using fs_cli). If you freewwitch server is on public IP address and not behind a NAT router, add the "-nonat" parameter (This will disable NAT traversal feature of FreeSWITCH). After a reboot you should see something like this in your console:

```
10381 Backgrounding.
```

Gentoo Linux

The FreeSWITCH? ebuilds are maintained by stkn in the freeswitch-overlay at oss.axsentis.de.
Installation instructions: Installation:Gentoo

openSUSE

Tested Nov 23, 2010 with openSUSE 11.3 x86_64
There were no issues during installation and testing.

Prerequisites

Use the YAST tool or zypper to add the following packages to a basic installation, the "base development" group contains most but not all of these items:

- autoconf
- automake
- gcc-c++
- git
- libjpeg-devel
- libtool
- make
- ncurses-devel

Installation

Continue with the normal installation steps or the Linux Quick Install Guide Downloads section.

pfSense package with GUI

pfSense FreeSWITCH? package is available for pfSense 1.2.3. The package includes a GUI for FreeSWITCH's configuration. The naming convention for the GUI has been designed to closely match the XML tag names and file names used in the default XML configuration as closely as possible. Install pfSense 1.2.3 based on FreeBSD7, Then go to System -> Packages and click the '+' add button on the right to install FreeSWITCH? from the packages list. The package is around 50mb because it includes 8khz sounds and music on hold. After the installation FreeSWITCH? will automatically start. By default it will bind to the WAN IP address.

To download pfSense 1.2.3 go to: http://www.pfsense.com/

Screenshots are here: http://portableusbapps.com/images/FreeSWITCH/

The pfSense FreeSWITCH? package is being used on many live systems on dedicated hardware and on some virtual machines.

If you already have a firewall in place and just want to use this package as an easy FreeSWITCH? install you can simply use it as a dedicated device and if desired turn off the firewall in pfSense by going to System -> Advanced -> Disable Firewall then put a check mark in 'Disable all packet filtering.'

Solaris

Solaris use the jds-cbe environment or fix path to working tr (as per FSBUILD-30).

```
export PATH=/usr/xpg4/bin:$PATH
```

- Use Sun Studio 12
- Install SFE (Spec Files Extra)
- Add SFEunixODBC to your Solaris installation

OpenSolaris Nevada

This is the distribution from the opensolaris.org guys, do not mix that up with os200805 which is the new OpenSolaris based distro from SUN.

- Select your development user. I will assume for the time being that the user is called freeswitch.
- Create the target directory for freeswitch

```
mkdir /opt/freeswitch; chown freeswitch /opt/freeswitch
```

- Give your build and development user software installation rights

```
usermod -P "Software Installation" freeswitch
```

- Log out and log back in as the freeswitch user (so that the new permissions become active)

- Download and unpack the new jds-cbe beta package:

```
/usr/sfw/bin/wget http://dlc.sun.com/osol/jds/downloads/cbe/test/desktop-cbe-1.7.0-rc1-x86.tar.b
gtar xvfz desktop-cbe-1.7.0-rc1-x86.tar.bz2"
```

- Install JDS-CBE

```
cd desktop-cbe-1.7.0-rc1
./cbe-install
```

- Leave everything as default. Select the Sun Studio compiler as your default compiler.

- Download the spec-files-extra repository into a subdirectory called SFE

```
svn co https://pkgbuild.svn.sourceforge.net/svnroot/pkgbuild/spec-files-extra/trunk SFE
```

- Load the jds-cbe environment.

```
. /opt/dtbld/bin/env.sh
```

- Compile and install SFEunixodbc.

```
cd SFE
pkgtool --download build SFEunixodbc.spec
```

- Download the FreeSWITCH? tree.

```
git clone git://git.freeswitch.org/freeswitch.git
```

- Prepare the FreeSWITCH? sources.

```
cd freeswitch; ./bootstrap.sh
```

- And edit the modules.conf file to select which modules you would like to have installed.

```
vim modules.conf
```

- Configure FreeSWITCH? sources for 64-bit.

```
CFLAGS=-m64 CXXFLAGS=-m64 LDFLAGS=-m64 ./configure --prefix=/opt/freeswitch --enable-core-odbc-s
--enable-core-libedit-support --enable-64 --with-openssl=/usr/sfw
```

- Or for 32bit.

```
CFLAGS=-m32 CXXFLAGS=-m32 LDFLAGS=-m32 ./configure --prefix=/opt/freeswitch --enable-core-odbc-s
--enable-core-libedit-support --with-openssl=/usr/sfw
```

- Run make.

```
gmake
```

- Install FreeSWITCH? into its target directory.

```
gmake install
```

OpenSolaris os200906

- Edit /etc/system and add the following line, to increase the maximum permissible number of file descriptors per user

```
set rlim_fd_max = 1048576
```

One million FDs ought to be enough for anyone.

- If doing a Git checkout of FreeSWITCH?, it is recommended to create a dedicated ZFS filesystem to hold the git checkout, so that ZFS snapshots can be used on that directory. Create the freeswitch user's

home directory first, then create the ZFS filesystem for it.

```
root@sol:~# mkdir /export/home/freeswitch; zfs create rpool/export/home/freeswitch
```

- Create the FreeSWITCH? user

```
root@osol:~# useradd -b /export/home -P "Software Installation" -s /bin/bash -m freeswitch
```

- Note: OpenSolaris will complain that the username freeswitch is too long, however it does work. Some POSIX utils may truncate the username to freeswit (8 chars)

- Create the target directory for FreeSWITCH?, and set the ownership on that directory, as well as the freeswitch user's home directory

```
root@sol:~# mkdir /opt/freeswitch; chown freeswitch /opt/freeswitch; chown freeswitch:staff /exp
```

- Give the FreeSWITCH? user software installation rights

```
root@sol:~# usermod -P "Software Installation" freeswitch
```

- You can add ulimit -n 262144 to the FreeSWITCH? user's .profile to automatically set a suitable limit

- Log in as the FreeSWITCH? user and download/unpack the new jds-cbe beta package

```
freeswitch@sol:~$ wget http://dlc.sun.com/osol/jds/downloads/cbe/test/desktop-cbe-1.7.0-rc1-x86.
freeswitch@sol:~$ gtar xjf desktop-cbe-1.7.0-rc1-x86.tar.bz2
```

- Install the following packages if they are not already

```
freeswitch@sol:~$ pfexec pkg install SUNWhea SUNWsfwhea SUNWxcu4 SUNWsprot SUNWxwinc SUNWxorg-he
```

- Run the CBE installer

```
freeswitch@sol:~$ cd desktop-cbe-1.7.0-rc1
freeswitch@sol:~/desktop-cbe-1.7.0-rc1$ ./cbe-install
Desktop Common Build Environment (CBE) Installer version 1.7.0-rc1
Using "pkg" packaging system
Checking for required packages...
All required packages are installed.
Starting from pkgbuild 1.1.0 (CBE 1.5) there is no system-wide
build directory (%_topdir), instead, each user has their own.
The default directory is $HOME/packages.
If you wish to use a different build directory, you can define it
in $HOME/.pkgbuildmacros as follows:

%_topdir /path/to/my/build/area

Would you like to do this now? [no]: no
The default topdir (/export/home/freeswitch/packages) will be used
Locating compilers...

The following compilers were found on your system:
```

```
1 - Sun Studio Express 5.10 (Ceres) in /opt/SunStudioExpress/bin
Would you like to configure more compilers for use with the Desktop CBE? [no]: no

The following compilers were found on your system:
1 - Sun Studio Express 5.10 (Ceres) in /opt/SunStudioExpress/bin

The Desktop CBE includes tools for building GNOME, KDE and/or SFE packages
It also includes some optional tools. Please select the tools you wish
to install.

Would you like install the tools for GNOME? [yes]: no
Would you like install the tools for KDE? [yes]: no
Would you like install the tools for SFE? [yes]: yes

The following tools are optional: rsync cvs
Would you like install the optional tools? [yes]: no
```

- As root, create a new ZFS filesystem for the FreeSWITCH? git checkout

```
root@sol:~# zfs create rpool/export/home/freeswitch/fs
root@sol:~# chown freeswitch:staff /export/home/freeswitch/fs
```

- As freeswitch, check out a copy of FreeSWITCH?

```
freeswitch@sol:~$ git clone git://git.freeswitch.org/freeswitch.git fs
```

- Optionally snapshot the fs filesystem (as root). Rolling back to a previous ZFS snapshot is preferable to a make clean

```
root@sol:~# zfs snapshot rpool/export/home/freeswitch/fs@clean
```

- As freeswitch, load the jds-cbe environment and bootstrap the source

```
freeswitch@sol:~$ cd fs/
freeswitch@sol:~/fs$ . /opt/dtbld/bin/env.sh
freeswitch@sol:~/fs$ ./bootstrap.sh
```

- Configure

```
freeswitch@sol:~/fs$ CFLAGS="-g -m64 -I/usr/sfw/include -L/usr/sfw/lib -I/usr/gnu/include -L/usr/
CXXFLAGS="-g -m64 -I/usr/sfw/include -L/usr/sfw/lib -I/usr/gnu/include -L/usr/gnu/lib" \
LDFLAGS="-m64 -L/usr/sfw/lib -R/usr/sfw/lib -L/usr/gnu/lib -R/usr/gnu/lib /usr/lib/64/0@0.so.1" \
./configure --prefix=/opt/freeswitch --enable-64 --with-openssl=/usr/sfw
```

- Build FreeSWITCH

```
freeswitch@sol:~/fs$ gmake
```

- Install FreeSWITCH

```
freeswitch@sol:~/fs$ gmake install
freeswitch@sol:~/fs$ gmake sounds-install moh-install
```

Ubuntu

Visit the Ubuntu Quick Start page.

Cross Compiling for ARM on Linux

These instructions were tested on Linux 2.6.18 (Centos 5) Cross compiling Linux 2.6.21 for ARM using gcc 3.4.6 All commands should be executed from within the main source repository.

Create a modules.conf. (this can be done after configure but it is convenient to do it now)

```
cp build/modules.conf.in modules.conf
```

Decide if you want to include cpp code, if you are running on a small embedded system and don't want to install the 2+MBytes libstdc++.so edit Makefile.am in the root of the source repository and remove all references to switch_cpp.cpp and switch_cpp.h, there is one of each. This will also disable all of FreeSWITCH?'s language modules, edit modules.conf and comment out all the language modules, that is all the lines starting with "language". You can still write C applications with the cpp disabled.

I was not able to get javascript working so you should comment out languages/mod_spidermonkey* from modules.conf and comment out the line "AC_CONFIG_SUBDIRS(libs/js)" in the file configure.in. If you figure out a way to get it working update this page, I wasn't motivated as I did not intend to use it.

If you want to reduce the build time you can comment out other support libraries from configure.in, I only used and tested the following, it saves a lot of build time and I don't know if the other libraries will build:

- libs/srtp
- libs/sqlite
- libs/libedit
- libs/pcre
- libs/apr
- libs/apr-util
- libs/codec/ilbc
- libs/sofia-sip
- libs/libsndfile
- libs/voipcodecs

I already had a version of curl cross compiled so I did not use the version that comes with FreeSWITCH.

Run this command next, it will create all the configure scripts

```
./bootstrap.sh
```

Several of the packages cannot detect capabilities of the Cross Compile environment correctly so configure needs to be called with many options to get it to work correctly. Check that the options I have match your Cross Compile environment. Prior to running this the following environment variables must be set: TARGET_CC - the cross compiler HOSTCC - the host compiler CFLAGS - CFLAGS for the cross compiler CPPFLAGS - CPPFLAGS for the cross compiler LDFLAGS - LDFLAGS for the cross compiler These environment variables also need to be set to point to the appropriate tool in the cross compiler's toolchain CC, CXX, AR, LD, GCC, AS, NM, RANLIB, STRIP, SIZE, OBJCOPY & OBJDUMP. Not sure if they are all

used, but this is what worked for me.

You will need to adjust some setting to fit your environment, set the "path" in ac_cv_path__libcurl_config and --with-curl, or remove them if you use the version that comes with FreeSWITCH?. Set --with-modinstdir to point to where the FreeSWITCH? modules will be on the target filesystem.

```
export config_TARGET_CC="$(TARGET_CC)"; \
   export config_BUILD_CC="$(HOSTCC)"; \
   export config_TARGET_CFLAGS="$(CFLAGS)"; \
   export config_TARGET_LIBS="$(LDFLAGS)"; \
   export CC_FOR_BUILD="$(HOSTCC)"; \
   export CFLAGS_FOR_BUILD=" "; \
   export ac_cv_file__dev_zero=no; \
   export apr_cv_tcp_nodelay_with_cork=yes; \
   export ac_cv_sizeof_ssize_t=4; \
   export ac_cv_file_dbd_apr_dbd_mysql_c=yes; \
   export ac_cv_path__libcurl_config=/path/curl-config; \
   export apr_cv_mutex_recursive=yes; \
   export ac_cv_func_pthread_rwlock_init=yes; \
   export apr_cv_type_rwlock_t=yes; \
       ./configure \
       --target=$(GNU_TARGET_NAME) \
       --host=$(GNU_TARGET_NAME) \
       --build=$(GNU_HOST_NAME) \
       --with-libcurl=/path/install \
       --with-devrandom=/dev/urandom \
       --with-modinstdir=/mod \
```

If you have not done so edit modules.conf so that it will build the modules you desire. If you are unsure, leave this to the defaults. FreeSWITCH? comes with a good set of modules as the default, until you become more familiar with FreeSWITCH? it is advised that you not edit modules.conf. A # at the beginning will omit that module from being built. A list of the modules can be obtained from the Modules page.

Once you are done configuring the build environment you need to run the following command. (Your PATH must point to the cross compiler toolchain)

```
make
```

I found little use for make install as I didn't want the Cross Compiled version installed on the build system. You will need to copy the FreeSWITCH? binary, library and modules directly out of the build tree into where ever you need them.

Cross Compiling for OpenWrt

http://wiki.freeswitch.org/wiki/OpenWrt

Prebuilt Binaries

The FreeSWITCH? team **strongly** recommends installing from the latest **Git tree** which is extremely stable! Many reported bugs in older versions of FreeSWITCH? are fixed in the current Git version. The developers **require** testing an issue on the current Git version prior to filing a bug report.

The FreeSWITCH? project maintains binary packages for a number of Linux distributions on the openSUSE Build Service. All external dependencies required by these FreeSWITCH? packages are either part of the base Linux distros or available in the repository at: http://software.opensuse.org/download/network:/telephony/. If your preferred Linux distro and version is listed there, then we recommend you use these packages.

Special Considerations

ODBC Support

See the ODBC Support document.

ODBC Installed in non-standard locations

As of April 23 2012, a change was made to the configure script to check if the ODBC library is installed. As a consequence, the check fails due to the way how the macro behaves. To resolve this issue, you would issue the following command before you initiate the configure script:

```
export LIBRARY_PATH="<path to ODBC lib here>"
```

No Root Access

If you don?t have root access add --prefix=~/freeswitch to the command. **NOTE: This is a really bad idea on production systems, *not* recommended.**

```
 ./configure --prefix=/usr/local/freeswitch
```

Insert non-formatted text here.

Graphical User Interfaces

See: Freeswitch Gui