Freeswitch

An open source phone system

Author: A.J.Gillette
Date: Oct 08, 2009
Revision: 1.3

Table of Contents

Introduction....3
Installation....3
Preparing for installation....3
Step 1: Basic pre-requisites....3
Step 2: For python development....3
Step 3: For TLS development....3
Step 4: For mod_fax development....4
Step 5: For skype development....4
Getting the source....4
Step 1: Getting the source....4
Step 2: Adding support for H323....4
Step 1: Install ptlib....4
Step 3: Building the source....5
Step 4: Configuring the system....5
Registrations:....5
Access Control:....7
SIP profiles:....8
Call Routing and Dialplans:....9
Auto Attendants and IVR's:....10
Dial by name:....11
Starting the system....12
Connecting to the Event/API interface....12
Using the sound card – mod_portaudio....13
Running as a daemon....15
Connecting to the console when running as a daemon....17

Introduction

Freeswitch is an open source communications system that supports both H323 and SIP. This configuration note is meant to be a primer for those not familiar with freeswitch. It will cover the basic installation and configuration of some of the key system components.

Installation

When freeswitch is included as part of the HD workplace products, the installation will be part of the overall system installation and much of what is included here will not be necessary. That said there is much to be learned about the capabilities of the freeswitch platform by performing a raw system installation at least once. Freeswitch is a very flexible platform that may be useful for other applications within your enterprise. The following summary was extracted from the freeswitch WIKI which can be found at

http://wiki.freeswitch.org/wiki/Installation_Guide

The installations steps that follow assume that the system is being installed on a fresh copy of Ubuntu Linux.

Preparing for installation

Ubuntu is shipped on a CD and as such installs with the bare minimum number of packages. In order to build and install freeswitch some additional packages need to be installed. You should be able to copy the following lines and paste them directly into a terminal window. You may not need all of the following packages but for consistency its a good idea to install them all.

Step 1: Basic pre-requisites

apt-get install build-essential subversion subversion-tools automake1.9 gcc-4.1 autoconf make wget libtool g++ libncurses5 libncurses5-dev libgdbm-dev libdb4.4-dev libperl-dev

Step 2: For python development

apt-get install python2.5 python2.5-dev

Step 3: For TLS development

apt-get install libgnutls-dev

Step 4: For mod_fax development

apt-get install libtiff4-dev

Step 5: For skype development

apt-get install libx11-dev

Getting the source

The next part of the installation is to get the freeswitch source. We recommend that the source be placed in the "/usr/src" directory on your linux target machine. The installation process will install freeswitch in "/usr/local/freeswitch". It is important to note that this will give you two places where configurations can be modified and saved. A common mistake is to make configuration changes in the wrong place. The copy of freeswitch that will be executed at startup is the one in "/usr/local/freeswitch" so configuration changes should be made in "/usr/local/freeswitch/conf" for that version. To execute freeswitch manually from the command line you execute the binary file which can be found at "/usr/local/freeswitch/bin". Change to that directory and execute the file by typing "./freeswitch".

Step 1: Getting the source

There are two ways to obtain the source. The SVN checkout will provide the source from the repository but it will be the head of the code stream and not guaranteed to work. The best way to obtain the source is to download the latest release tarball and uncompress it into the "/usr/src" directory. The tarball can be found at the following URL.

http://files.freeswitch.org/freeswitch-1.0.4.tar.gz

Copy the file to your "/usr/src" directory and type the following command to uncompress it into it's own directory.

```
tar -xzvf freeswitch-1.0.4.tar.gz
```

Step 2: Adding support for H323

The Freeswitch system does not support H.323 by default, it has to be enabled prior to the system being built. The following commands will install the appropriate support packages for the h.323 installation. PT lib will not build without installing flex and bison both of which are available on the default ubuntu repository. Install them with apt-get.

Step 1: Install ptlib

```
Cd /usr/src/freeswitch-1.0.4/build
svn co https://opalvoip.svn.sourceforge.net/svnroot/opalvoip/ptlib/trunk ptlib
apt-get install flex
apt-get install bison
cd ptlib
./configure --prefix=/usr
make
make install
```

The easiest way to proceed with installing the h.323 support is to run the provided script in the "/usr/src/freeswitch-1.0.4/build" directory. You will need to change the properties of the script and then execute it using the following commands.

```
Cd /usr/src/freeswitch-1.0.4/build
chmod 777 buildopal.sh
./buildopal.sh
```

The opal library contains the core of the H.323 software but an additional module must be compiled in order for freeswitch to make use of the opal library. To make sure this module is built when you build freeswitch you must edit the "modules.conf" file in the "/usr/src/freeswitch-1.0.4" directory.

Step 3: Building the source

Change directory to the new freeswitch directory and start the build process by typing the command ".configure". This will check your system to make sure that you have all the packages necessary to build the source code. The steps above are meant to make sure that you do have all the necessary packages but in the event that you don't, you cann install the additional packages by typing "apt-get install <package_name>" at the command line. Note that the <package_name> portion of the command above is replaced with the actual package name from the error message that you get when the ".configure" fails.

When the make has completed successfully you will install the freeswitch software to the default "/usr/local/freeswitch" directory by typing "make install". Prompts and music on hold can be installed by typing "make sounds-install" and "make moh-install" respectively.

```
cd /usr/local/freeswitch-1.0.4
./configure
make
make install
make sounds-install
make moh-install
```

Step 4: Configuring the system

Registrations:

The complete configuration for a freeswitch system is contained in the "/usr/local/freeswitch/conf" directory. Backing up this directory backs up the entire system configuration. Registration credentials for devices that will be registering with freeswitch can be found in the "/usr/local/freeswitch/conf/directory/default" directory. Notice that there is a separate XML file for each device. The file below is for extension 1000 and is called "1000.xml".

```
<include>
  <user id="1000">
    <params>
      <param name="password" value="$${default_password}"/>
      <param name="vm-password" value="1000"/>
    </params>
    <variables>
      <variable name="toll_allow" value="domestic,international,local"/>
      <variable name="accountcode" value="1000"/>
      <variable name="user_context" value="default"/>
      <variable name="effective_caller_id_name" value="Extension 1000"/>
      <variable name="effective_caller_id_number" value="1000"/>
      <variable name="outbound_caller_id_name" value="$${outbound_caller_name}"/>
      <variable name="outbound_caller_id_number" value="$${outbound_caller_id}"/>
      <variable name="callgroup" value="techsupport"/>
    </variables>
  </user>
</include>
```

Things to note about this file are the “user id” and the “password”. Each phone/device will have to be configured to register with this freeswitch using an auth-userame and auth-password. The auth-username for this device will be 1000 and the default auth-password will be “$${default_password} “. The two dollar signs indicates a global variable which can be found in the “/usr/local/freeswitch/vars.xml” file. The default password is “ClueCon”. The “callgroup” entry is used to place this device in a ring-group or hunt-group. All other devices in this group must have an identical line.

Once a phone is registered and the system is running, you can test it by dialing “9998” which will play the music from the popular game tetris. Dialing “3050” will connect the phone to the built in conference unit. The following is an actual registration file for our in house phone system.

```
<include>
  <user id="5508">
    <params>
      <param name="password" value="$${default_password}"/>
      <param name="vm-password" value="5508"/>
    </params>
    <variables>
      <variable name="toll_allow" value="domestic,international,local"/>
      <variable name="accountcode" value="5508"/>
      <variable name="user_context" value="default"/>
      <variable name="effective_caller_id_name" value="Fred Gillette"/>
      <variable name="effective_caller_id_number" value="5508"/>
      <variable name="outbound_caller_id_name" value="$${outbound_caller_name}"/>
      <variable name="outbound_caller_id_number" value="$${outbound_caller_id}"/>
      <variable name="callgroup" value="appeng"/>
    </variables>
  </user>
</include>
```

The other important piece of this configuration file is the “effective_caller_id_name” and the “effective_caller_id_number”. These two fields are used by the dial-by-name feature and should contain the users name and phone number.

Access Control:

Access control is how the freeswitch protects itself from unwanted calls and registrations. The access control list exists in the form of a configuration file which can be found in the "/usr/local/freeswitch/conf/autoload-configs/" directory. You will need to edit this file to allow inbound calls. The file to edit is "/usr/local/freeswitch/conf/autoload-configs/acl.conf.xml". You will need to add a line to accept local calls. In the example below the public subnet is configured to be "192.168.216.x" so the line added is " <node type="allow" cidr="192.168.216.1/32"/>" to allow inbound calls from the local SIP phone system whose address is "192.168.216.1".

"acl.conf.xml"

```
<configuration name="acl.conf" description="Network Lists">
  <network-lists>
    <!--
        These ACL's are automatically created on startup.

        rfc1918.auto  - RFC1918 Space
        nat.auto      - RFC1918 Excluding your local lan.
        localnet.auto - ACL for your local lan.
        loopback.auto - ACL for your local lan.
    -->

    <list name="lan" default="allow">
      <node type="deny" cidr="192.168.42.0/24"/>
      <node type="allow" cidr="192.168.42.42/32"/>
    </list>

    <!--
       This will traverse the directory adding all users
       with the cidr= tag to this ACL, when this ACL matches
       the users variables and params apply as if they
       digest authenticated.
    -->
    <list name="domains" default="deny">
      <node type="allow" domain="$${domain}"/>
      <node type="allow" cidr="192.168.216.1/32"/>
    </list>

  </network-lists>
</configuration>
```

SIP profiles:

The SIP profiles can be found in "/usr/local/freeswitch/conf/sip_profiles". There are separate files for internal and external interfaces. A default freeswitch installation will have only one NIC with the internal and external sip profiles configured on different ports. Freeswitch installations can be configured for two NIC's and will use the standard port 5060 on each interface. Adding the following lines to the "/usr/local/freeswitch/vars.xml" file will define constants that can be used to reference each interface.

```
<X-PRE-PROCESS cmd="set" data="system_internal_ip=192.168.215.1"/>

<X-PRE-PROCESS cmd="set" data="system_external_ip=192.168.217.53"/>
<X-PRE-PROCESS cmd="set" data="system_sip_port=5060"/>
```

The lines above, assume that you have pre-configured your Ubuntu OS to have two NIC's where your external NIC is 192.168.217.53 and your internal NIC is 192.168.215.1. Now that we have global variables that we can use in the SIP profiles we can take a look at the two profiles provided. There will be one profile for each physical interface, called "internal.xml" and "external.xml". If we look at "external.xml" the following are the lines of interest.

```
<param name="sip-port" value="$${system_sip_port}"/>
<param name="rtp-ip" value="$${system_external_ip}"/>
<param name="sip-ip" value="$${system_external_ip}"/>
<param name="ext-rtp-ip" value="$${system_external_ip}"/>
<param name="ext-sip-ip" value="$${system_external_ip}"/>
```

Now in the "internal.xml" file, the lines of interest are:

```
<param name="ext-rtp-ip" value="$${system_internal_ip}"/>
<param name="ext-sip-ip" value="$${system_internal_ip}"/>
```

You will probably need to restart freeswitch in order for the changes to take effect. The changes will cause freeswitch to start a user agent (UA) on each of the specified interfaces.

The default configuration for freeswitch uses one NIC so in that case the internal IP and external IP are the same. Two UA's listen on the same port so the internal port will be 5060 but the external port will be 5080.

Call Routing and Dialplans:

Calls find their way into the freeswitch system either from locally connected phones or from external SIP devices. The “SIP Profiles” section explained how “sofia profiles” are used to route calls to dial plans. Dial Plans are a combination of XML and Regular Expressions that define rules that will be used to route calls. Calls exist as a call half and are connected to another call half via a collection of applications. Each rule in a dial plan will contain a line that identifies which application will handle the call. The most common of these applications is “bridge” which connects the call half to another SIP destination.

In Freeswitch there is no one dialplan, but rather a series of dial plans that can be threaded together to produce the end result. The primary dialplan is called default.xml and can be found in the “/usr/local/freeswitch/conf/dialplan” directory. The other dial plan of interest is external.xml which contains the dial plan for external callers. If the inbound call is allowed the caller is directed to default.xml where the bulk of the dial plan resides.

Auto Attendants and IVR's:

An auto attendant is included with the basic freeswitch configuration. You can access it by dialing 4000 from any registered phone. The file that contains the configuration is in /usr/local/freeswitch/conf/autoload_configs" and is called "ivr.conf.xml". It references a file that contains phrase macros that although interesting, are not necessary for most configurations. The listing below is for a simpler version of an auto-attendant that uses pre-recorded prompt files.

```
<configuration name="ivr.conf" description="IVR menus">
  <menus>
    <!-- demo IVR setup -->
    <!-- demo IVR, Main Menu -->
    <menu name="demo_ivr"
                greet-long="ivr/ivr-welcomeDay.wav"
                greet-short="ivr/ivr-welcomeDay.wav"
                invalid-sound="ivr/ivr-that_was_an_invalid_entry.wav"
                exit-sound="voicemail/vm-goodbye.wav"
                timeout="10000"
                inter-digit-timeout="2000"
                max-failures="3"
                max-timeouts="3"
                digit-len="4">
      <entry action="menu-exec-app" digits="1" param="transfer 5777 XML default"/>    <!-- Mike Pascoe -->
      <entry action="menu-exec-app" digits="2" param="transfer 5515 XML default"/>    <!-- Dan Rusheleau -->
      <entry action="menu-exec-app" digits="3" param="transfer 5577 XML default"/>    <!-- Chris Charlebois -->
      <entry action="menu-exec-app" digits="4" param="transfer 5519 XML default"/>    <!-- Guy Dubuc -->
      <entry action="menu-exec-app" digits="5" param="transfer 5780 XML default"/>    <!-- Judy Wood -->
      <entry action="menu-exec-app" digits="8" param="javascript dial_by_name_directory.js"/>
      <entry action="menu-exec-app" digits="/^(55[0-9]{2}|57[0-9]{2})$/" param="transfer $1 XML features"/>
      <entry action="menu-top" digits="9"/>          <!-- Repeat this menu -->
    </menu>
  </menus>
</configuration>
```

The file "ivr-welcomeDay.wav" was copied from an EAS to the directory "/usr/local/freeswitch/sounds/en/us/callie/ivr/8000". All prompts for the IVR application should be in this directory. The action "menu-top" causes the IVR to replay the main message. Pressing "8" in this example will transfer the caller to the "dial_by_name_directory" java script that will ask the caller to spell the last name of the person being called.

Dial by name:

A dial by name application is included with the freeswitch tarball in the contribs folder. The “contrib” folder can be found in the top level of your source code tree. In my case I extracted the freeswitch source to “/apps/src/freeswitch-1.0.4” so the dial-by-name application is in “/apps/src/freeswitch-1.0.4/contrib/nmartin/dial_by_name_directory”. Copy the file “dial_by_name_directory.js” to the working scripts directory “/usr/local/freeswitch/scripts”. Copy the phase file “dial_by_name_directory.xml” to “/usr/local/freeswitch/conf/lang/en/demo” so that it can be found by the java script when it executes. The previous section had a sample IVR file that would route callers to the dial_by_name application by pressing 8. If you want to be able to access the application directly, add the following to your default dialplan.

```
<extension name=”dial_by_name”>
   <condition field=”destination_number” expression=”^6000$>
      <action application=”javascript” data=”dail_by_name_directory.js” />
   </condition>
</extension>
```

This will route calls to extension 6000 to the dial_by_name service. Note that the java script actually parses the user registration files in “/usr/local/freeswitch/conf/directory/default” and that is is important that the “effective_caller_id_name” be set to “firstname lastname”. It is also important that the “effective_caller_id_number” field is set to the users extension. The name is used to find the user and the number is where the caller will be transferred to. When configuring your voicemail you have the option to record your name. If you record your name it will be played back to the user before the user is transferred.

Starting the system

There are several ways to start the freeswitch system. The executable can be found in "/usr/local/freeswitch/bin" and is called "freeswitch". The commands below can be used to start the system from a terminal window.

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

This works fine for a development environment but for a production system you will want freeswitch to start at boot.

Connecting to the Event/API interface

Freeswitch provides an event channel that also makes it possible to inject api commands. The default socket is 8021. So the command to connect to the channel would be:

```
netcat 192.168.219.1 8021
```

Users must then authenticate using the command:

```
auth 1cp3050
```

The response from the system will then be:

```
Content-Type: command/reply
Reply-Text: =OK accepted
```

You are now connected and can issue commands or monitor events. To monitor a new event type:

```
event <event_name>
```

A good event to monitor for is the CHANNEL_CREATE event which will fire for each call half that is created. To monitor for this event type:

```
event CHANNEL_CREATE
```

If you then make a call you will see the CHANNEL_CREATE event displayed on the event channel. For a list of events that can be monitored from the event socket, check on the WIKI at:

http://wiki.freeswitch.org/wiki//Event_list

Using the sound card – mod_portaudio

The "mod_portaudio" code in freeswitch provides a way to treat the sound card as a SIP end point. Through trial and error and conversations with the designers on the IRC chat, it has been determined that most if not all sound cards only partially work on linux systems. The people on the IRC chat suggested using external USB sound cards and our testing has verified that these cards are much more reliable.

The portaudio module is not part of the default installation, so freeswitch must be re-built with the module enabled in order to make use of the modules features. To do this you must first edit the modules file in your build directory. My file was located in "/usr/src/freeswitch-1.0.4".

```
#endpoints/mod_dingaling
endpoints/mod_iax
#endpoints/mod_portaudio
endpoints/mod_sofia
endpoints/mod_loopback
```

Remove the "#" from the portaudio line, save the file and rebuild freeswitch by typing "make". When the make has completed type "make install". Restart freeswitch and the module should be installed. Type "pa" into the console and you should get a list of portaudio commands. "pa listdev" will display the sound cards available.

```
freeswitch@192.168.217.53@internal> pa devlist
0;/dev/dsp;16;16;r,i,o
1;/dev/dsp1;16;0;
```

You should be able to select one device and use it for both input and output.

```
freeswitch@192.168.217.53@internal> pa indev #0
indev set to 0

freeswitch@192.168.217.53@internal> pa outdev #0
outdev set to 0
```

Now you can make a test call using the "pa call" command.

```
freeswitch@192.168.217.53@internal> pa call 1000
```

Extension 1000 should ring and when you answer you should get two way audio from the phone to the sound card. I connected a headset for the test. In order to call the sound card we need to give it and extension and program some dial plan entries. Create the following dial plan entry in "/usr/local/freeswitch/conf/default.xml":

```
<extension name="test">
   <condition field="destination_number" expression="^3$">
     <action application="bridge" data="portaudio"/>
   </condition>
 </extension>
```

When you call extension 3 you should see “Ring Ready” on the console. Type “pa answer” to hear ring back and answer the call.

There is an excellent example page on the freeswitch wiki which also shows how to configure freeswitch to be a dedicated soft-phone. The freeswitch wiki can be found at http://wiki.freeswitch.org enter the key word “softphone” into the search box or go directly to the example page http://wiki.freeswitch.org/wiki/Freeswitch_softphone

Running as a daemon

To get freeswitch to run as a daemon you have to create a script in the "/etc/init.d" directory. The script below runs as the root user and starts freeswitch at boot time.

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          skeleton
# Required-Start:    $local_fs $remote_fs
# Required-Stop:     $local_fs $remote_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Description:       Freeswitch debian init script.
# Author: Matthew Williams
#

# Do NOT "set -e"

# PATH should only include /usr/* if it runs after the mountnfs.sh script
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/local/bin
DESC="Freeswitch"
NAME=freeswitch
DAEMON=/usr/local/freeswitch/bin/$NAME
DAEMON_ARGS="-nc"
PIDFILE=/usr/local/freeswitch/log/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME

FS_USER=root
FS_GROUP=root

# Exit if the package is not installed
[ -x "$DAEMON" ] || exit 0

# Read configuration variable file if it is present
[ -r /etc/default/$NAME ] && . /etc/default/$NAME

# Load the VERBOSE setting and other rcS variables
. /lib/init/vars.sh

# Define LSB log_* functions.
# Depend on lsb-base (>= 3.0-6) to ensure that this file is present.
. /lib/lsb/init-functions

#
# Function that starts the daemon/service
#
do_start()
{
        # Set user to run as
    if [ $FS_USER ] ; then
          DAEMON_ARGS="`echo $DAEMON_ARGS` -u $FS_USER"
    fi
        # Set group to run as
    if [ $FS_GROUP ] ; then
      DAEMON_ARGS="`echo $DAEMON_ARGS` -g $FS_GROUP"
    fi

    # Return
    #   0 if daemon has been started
    #   1 if daemon was already running
    #   2 if daemon could not be started
    start-stop-daemon --start --quiet --pidfile $PIDFILE --exec $DAEMON --test > /dev/null -- \
        || return 1
    start-stop-daemon --start --quiet --pidfile $PIDFILE --exec $DAEMON -- \
        $DAEMON_ARGS \
        || return 2
    # Add code here, if necessary, that waits for the process to be ready
```

```
	# to handle requests from services started subsequently which depend
	# on this one.  As a last resort, sleep for some time.
}

#
# Function that stops the daemon/service
#
do_stop()
{
	# Return
	#   0 if daemon has been stopped
	#   1 if daemon was already stopped
	#   2 if daemon could not be stopped
	#   other if a failure occurred
	start-stop-daemon --stop --quiet --retry=TERM/30/KILL/5 --pidfile $PIDFILE --name $NAME
	RETVAL="$?"
	[ "$RETVAL" = 2 ] && return 2
	# Wait for children to finish too if this is a daemon that forks
	# and if the daemon is only ever run from this initscript.
	# If the above conditions are not satisfied then add some other code
	# that waits for the process to drop all resources that could be
	# needed by services started subsequently.  A last resort is to
	# sleep for some time.
	start-stop-daemon --stop --quiet --oknodo --retry=0/30/KILL/5 --exec $DAEMON
	[ "$?" = 2 ] && return 2
	# Many daemons don't delete their pidfiles when they exit.
	rm -f $PIDFILE
	return "$RETVAL"
}

#
# Function that sends a SIGHUP to the daemon/service
#
do_reload() {
	#
	# If the daemon can reload its configuration without
	# restarting (for example, when it is sent a SIGHUP),
	# then implement that here.
	#
	start-stop-daemon --stop --signal 1 --quiet --pidfile $PIDFILE --name $NAME
	return 0
}

case "$1" in
  start)
	[ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
	do_start
	case "$?" in
		0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
		2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
	esac
	;;
  stop)
	[ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"
	do_stop
	case "$?" in
		0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
		2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
	esac
	;;
  #reload|force-reload)
	#
	# If do_reload() is not implemented then leave this commented out
	# and leave 'force-reload' as an alias for 'restart'.
	#
	#log_daemon_msg "Reloading $DESC" "$NAME"
	#do_reload
	#log_end_msg $?
	#;;
  restart|force-reload)
```

```
        #
        # If the "reload" option is implemented then remove the
        # 'force-reload' alias
        #
        log_daemon_msg "Restarting $DESC" "$NAME"
        do_stop
        case "$?" in
          0|1)
                do_start
                case "$?" in
                        0) log_end_msg 0 ;;
                        1) log_end_msg 1 ;; # Old process is still running
                        *) log_end_msg 1 ;; # Failed to start
                esac
                ;;
          *)
                # Failed to stop
                log_end_msg 1
                ;;
        esac
        ;;
  *)
        #echo "Usage: $SCRIPTNAME {start|stop|restart|reload|force-reload}" >&2
        echo "Usage: $SCRIPTNAME {start|stop|restart|force-reload}" >&2
        exit 3
        ;;
esac
```

Note Once you have the script in place you need to tell the system to run it for each of the run levels. To do that type “update-rc.d freeswitch defaults”

Connecting to the console when running as a daemon

To connect to the console when running freeswitch as a daemon you need to use the “fs_cli” command which can be found in the “/usr/local/freeswitch/bin” directory. The default password the actual command will be:

./fs_cli -H <ip address> -p <password>

./fs_cli -H 192.168.219.1 -p ClueCon