Required Packages

First, set up the FreeSWITCH Debian repository

```
# echo "deb http://files.freeswitch.org/repo/deb/debian/ wheezy main" >> /etc/apt/sources.list.d/freeswitch.list
# wget -O - http://files.freeswitch.org/repo/deb/debian/freeswitch_archive_g0.pub | apt-key add -
```

Install the required packages (I'm using freeswitch-meta-default as an example here)

```
# apt-get update
# apt-get install pacemaker freeswitch-meta-default
```

Set Up You Core Database

I've used the same database for all modules, and have found the following minimal required configuration. So, in vars.xml add

```
<X-PRE-PROCESS cmd="set" data="odbc-dsn=[your connection string]" />
```

Then we need to set up sofia and switch. So in switch.conf.xml add

```
<param name="core-db-dsn" value="$$odbc-dsn" />
```

And for each sofia profile

```
<param name="track-calls" value="true" />
<param name="odbc-dsn" value="$$odbc-dsn" />
```

Setting Corosync

The first thing you need to do is set up auth keys. On the first node run

```
# corosync-keygen
```

and copy /etc/corosync/authkey to the second node.

Do everything from here on both nodes

Next set up /etc/corosync/corosync.conf. Mine looks like this, which as you may notice is rather similar to the default. In my setup I had to use the unicast transport due to a switch being funny about multicast.
You'll also need to set START=yes in /etc/default/corosync

Then start corosync

# /etc/init.d/corosync start

Setting up Pacemaker

We'll now set up pacemaker to manage FreeSWITCH and a virtual floating IP address, starting with the IP address.

#crm configure primitive fsip ocf:heartbeat:IPaddr2 params ip="192.168.1.50" cidr_netmask="24"
FreeSWITCH

To monitor the status of FreeSWITCH we use a custom init script. Create /etc/init.d/fssofia on both nodes with the following content, filling in the PROFILES variable as needed.

```bash
### -*- mode:shell-script; indent-tabs-mode:nil; sh-basic-offset:2 -*-
### BEGIN INIT INFO
# Provides: FSSofia
# Required-Start: $network $remote_fs $local_fs
# Required-Stop: $network $remote_fs $local_fs
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: FSSofia
# Description: FSSofia Status
### END INIT INFO
#set -x
FS_CLI_PROG='/usr/bin/fs_cli'
FS_CLI_HOST='127.0.0.1'
FS_CLI_PORT='8021'
FS_CLI_PASS='ClueCon'
PROFILES='external'

usage() {
    echo "Usage: $0 profile1[,profile2[,etc]] {start|stop|status}"
    exit 1
}

fs_cli() {
    $FS_CLI_PROG -H $FS_CLI_HOST -P $FS_CLI_PORT -p $FS_CLI_PASS -x "$1"
}

sofia_profile_started() {
    fs_cli "sofia xmlstatus" | grep "<name>$1</name>" | wc -l
}

if [ $# != 1 ]; then
    usage
fi

CMD=$1

case "$CMD" in
    'start')
        fs_cli "sofia recover"
    ;;
    'stop')
        exit 0
    ;;
    'status')
        for p in $PROFILES; do
            if [ `sofia_profile_started "$p"` -eq 0 ]; then
                echo "$p DOWN"
                exit 3
            fi
        done
        echo "OK"
        exit 0
    ;;
    '*')
        usage
    ;;
esac

Now create the pacemaker resource
Setting Up The Colocation and Order

I didn't have any luck getting this to work with the resource groups that pacemaker provides. The IP failed over but the calls weren't picked up. So instead I set up a colocation rule and an order rule to make sure the failover machine has the IP address before calls are recovered.

```
# crm configure colocation fscolo inf: fsip fs
# crm configure order fscoloorder-1 0: fsip:start fs:start
```

And that's all I had to do. You can test this by deliberately crashing you main freeswitch servers using

```
# fs_cli -x 'fsctl crash'
```

While monitoring the cluster resources with

```
# crm_mon -Ar
```

References

Of course, I didn't come up with all this all on my own :)

- Enterprise_deployment_IP_Failover
- Enterprise Deployment with Corosync