FreeSWITCH 简要使用教程

日积月累才有成长，拨苗助长毁人难悔

18621575908

上海宁卫信息技术有限公司

1

FreeSWITCH 简要使用教程
(草本)

李浩
18621575908

2014 年 10 月 12 日

http://www.freeswitch.net.cn
http://www.opentips.net.cn
QQ: 1354608370
Mail: lihao@nway.com.cn
Mobile:18621575908

目录

一、 通信发展历程

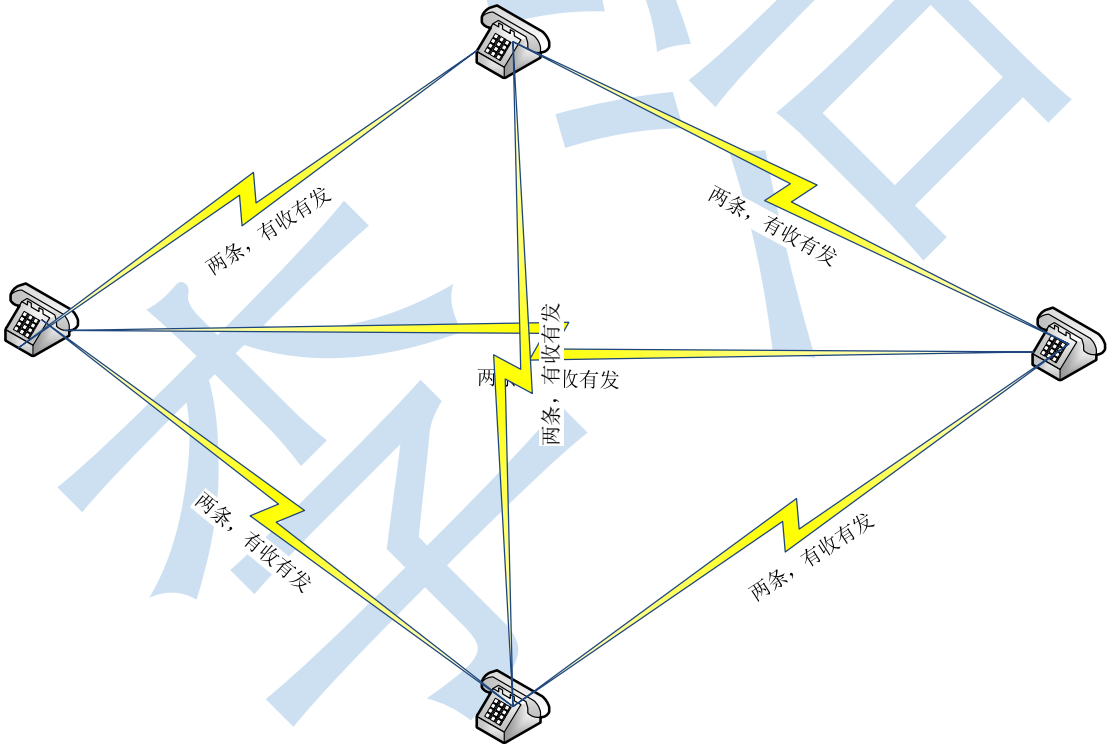
1. 电话的发明者

– 贝尔 –1876 贝尔实验室

1888 年，德国青年物理学家海因里斯.赫兹（H.R.Hertz）用电波环进行了一系列实验，发现了电磁波的存在，他用实验证明了麦克斯韦的电磁理论。这个实验轰动了整个科学界，成为近代科学技术史上的一个重要里程碑，导致了无线电的诞生和电子技术的发展。

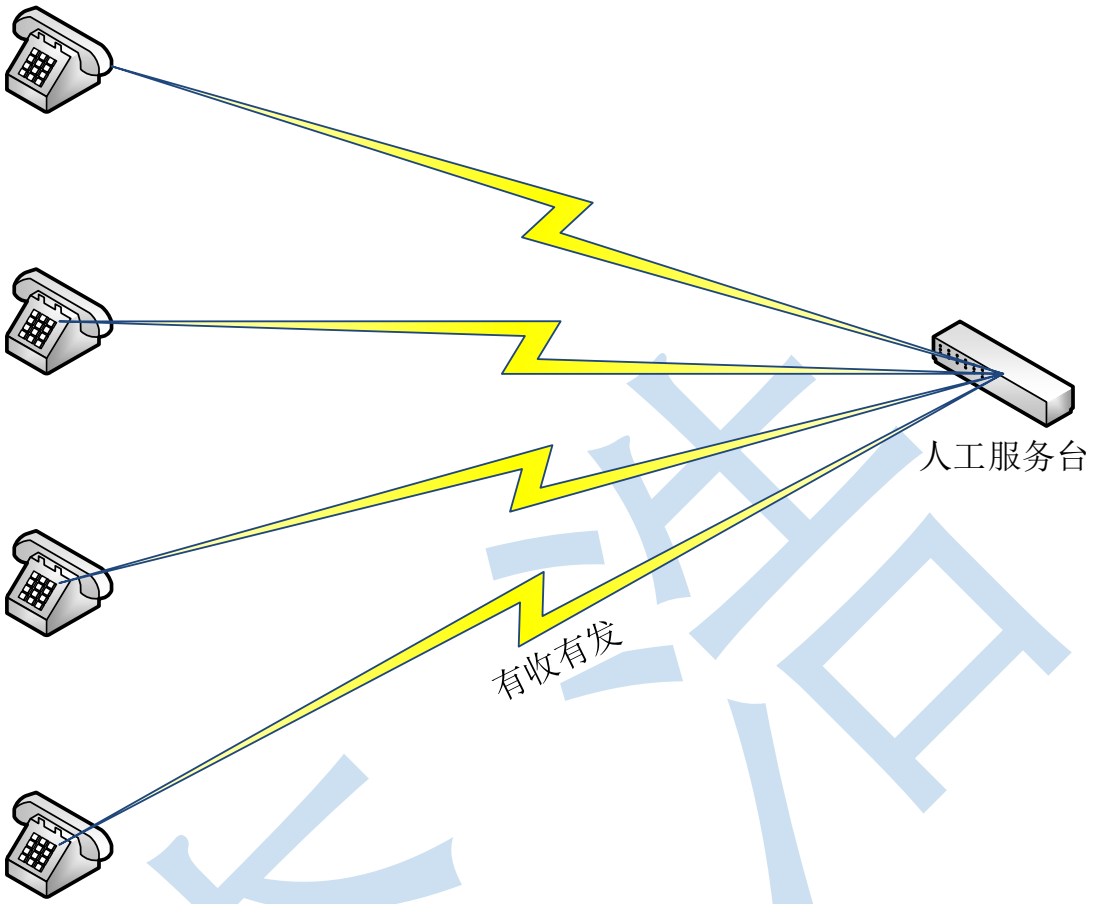
2. 第一代结构

4 人通话为例



在这种结构下，每条线都是专线，一户需要其它三户进行通话，则都是要拉三对专线，成本非常大。假如从北京到上海要打电话，北京和上海各有两户，那么就需要在一千多公里的里程中，拉三对线，而且为了传输性能和效率，必须是铜线，那么这个成本更是具大。

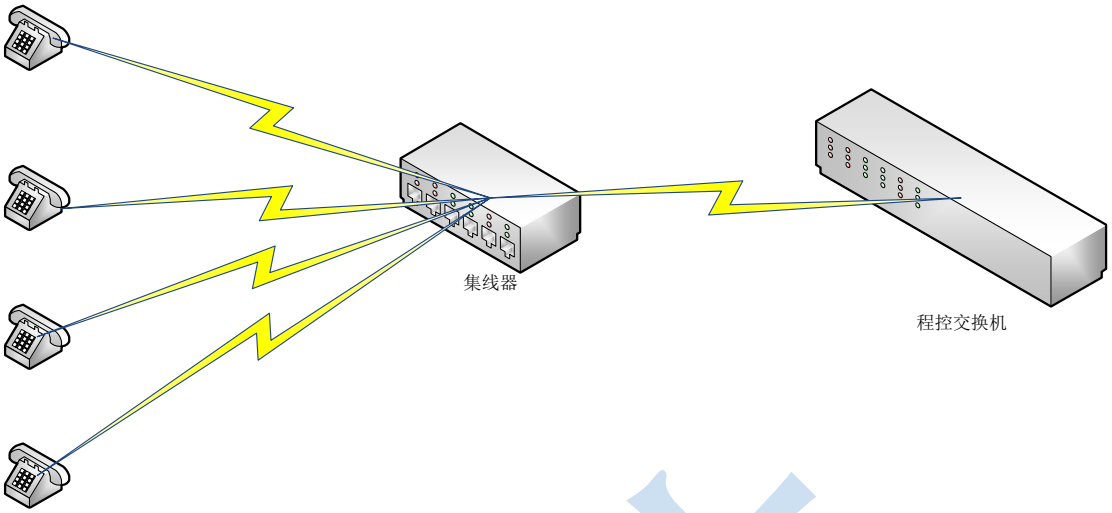
3. 第二代



这种结构下，由统一的人工服务台先接听，然后通过跳线帽把某两路的线跳线直接直连，故而相比前一代，拉的线数量有所减少，在成本上有很大的降低。

4. 第三代

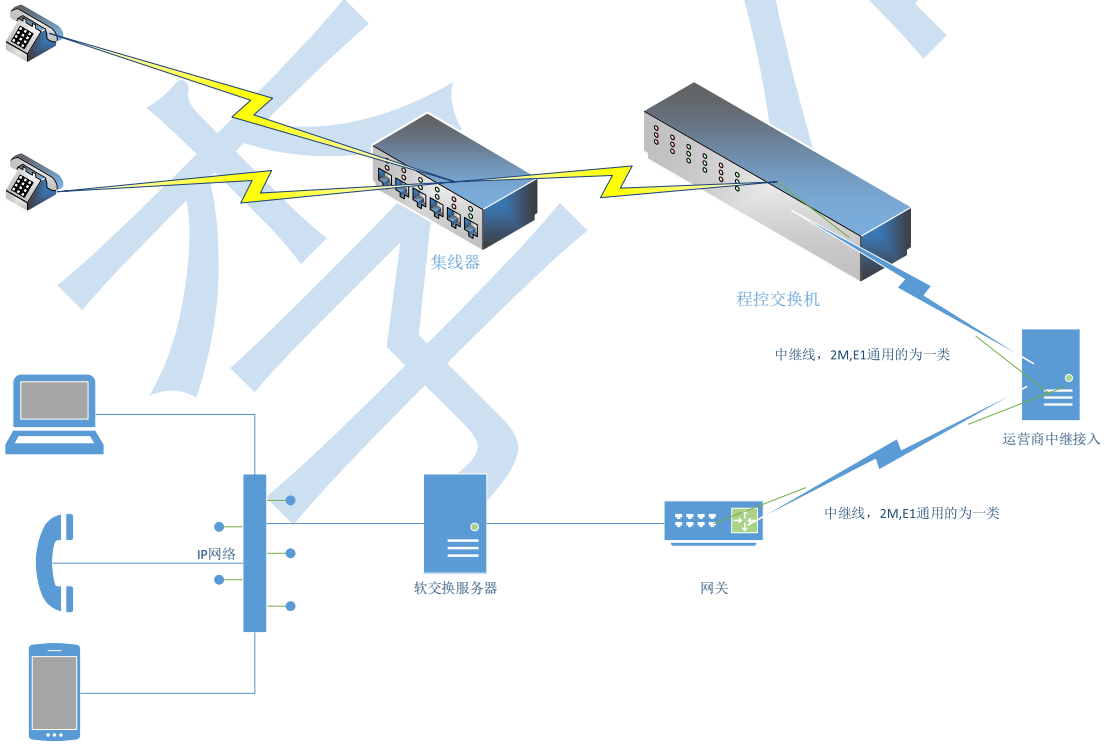
程控交换机



这种结构通过可编程控制器，对接入的各方电话进行控制，取代人工服务台，大大地方便了使用者。

5. 第四代

IP+多路复用+ 无线



这种结构即当前流行的方案，在个别老旧的系统中还是会有相关的程控，但更多的使用 ip 化方式进行相对应的 ip 化传输。国内的运营商也有个口号“光进铜退”即是在这种情况下出现的。现在有一部分民用核心网也使用 ip 传输。

二、 呼叫中心和 IPPBX 的区别

IPPBX 的说明

IPPBX 则是以呼叫为主体的系统，可在各种呼叫通信中占主体，包含各种呼叫功能，大概的以以下为例，呼入呼出、IVR、录音、留言、监听、多方通话等等。但它的作用仅用于呼叫，不和业务相挂钩。而在 IPPBX 的开源产品以 FreeSwitch、Asterisk 为最主要的两个系统。IPPBX 基本都是 B2BUA，即信令媒体都一个系统中处理。
另外有个东西叫 SIP Proxy，这个作为代理，和我们的主体无关，略过。

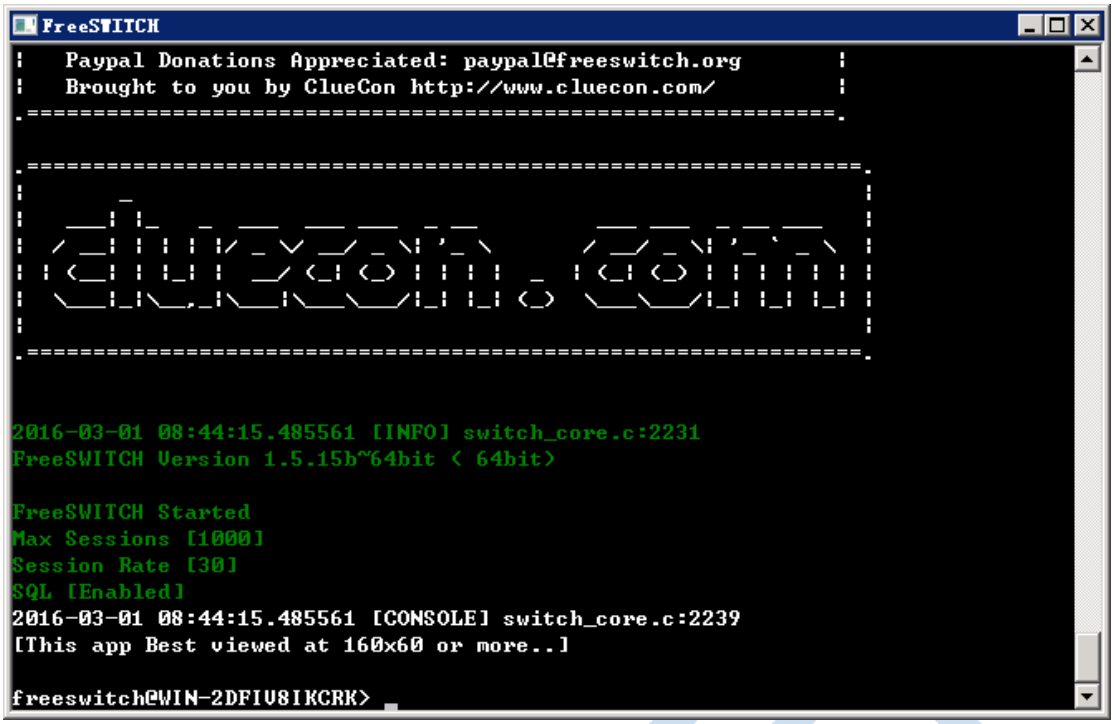
呼叫中心的说明

简单说来，呼叫中心以业务为中心，呼叫为辅助手段的系统。比如外呼营销业务中的去电弹屏、工单、客户相关管理系统的结合，或者呼入的来电弹屏，CRM、工单、订单等等，我们可以采用中间件的模式来完成这项工作。当然他需要有一定的对接开发工作量。

三、 FreeSWITCH 的安装

A. Windows 下安装

Windows 下的 FreeSWITCH 安装非常简单，从以下地址：
http://files.freeswitch.org/windows_installer/installer/
或
~~http://pan.baidu.com/s/1jI6oQG2 下载 FSGui 使用~~
中下载相应的版本的后，双击并一步一步 next 点击下去后，高版本的 win 系统需要用管理员运行，那么如果没有意外，就会出现 FreeSWITCH 启动后的控制台，如图：



在这里需要注意一点，要在防火墙中允许相关的端口，以便让应用能跑起来。

B. CentOS 下安装

以 CentOS6 为例

#epel repo

yum -y install wget

wget http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm

#32 bit

 # wget http://dl.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm

rpm -ivh epel-release-6-8.noarch.rpm

#

在测试环境使用 yum 安装，遇到一个问题：

#Error: Cannot retrieve metalink for repository: epel. Please verify its path and try again

#处理很简单，修改文件“/etc/yum.repos.d/epel.repo”， 将 baseurl 的注释取消，

mirrorlist 注释掉。即可。

yum clean all

yum makecache

yum install -y libtool git subversion autoconf automake libtool gcc-c++ ncurses-devel make

yum -y install expat-devel openssl-devel libtiff-devel libX11-devel unixODBC-devel libssl-devel python-devel
yum -y install zlib-devel libzrtpcpp-devel alsa-lib-devel libogg-devel libvorbis-devel perl-libs gdbm-devel
yum -y install libdb-devel uuid-devel @development-tools patch

yum -y install ldns-devel libidn-devel unbound-devel
yum -y install libjpeg-devel
yum -y install pcre-devel
yum -y install speex-devel
yum -y install sqlite-devel curl-devel libedit-devel

cd /usr/local/src
 # To build from Master, the latest source code:
#git clone https://freeswitch.org/stash/scm/fs/freeswitch.git

OR

To build from the current release source code:

git clone -b v1.2.stable https://freeswitch.org/stash/scm/fs/freeswitch.git

cd /usr/local/src/freeswitch

 # The -j argument spawns multiple threads to speed the build process
./bootstrap.sh -j

 # if you want to add or remove modules from the build, edit modules.conf
#vi modules.conf
 # add a module by removing '#' comment character at the beginning of the line
 # remove a module by inserting the '#' comment character at the beginning of the line containing the name of the module to be skipped

#使用 postgresql 时，应指定以下一些

#./configure --enable-core-pgsql-support --host=i686 #host 后边的值为 uname -m

./configure -C

```
make && make install
make all install cd-sounds-install cd-moh-install
make sounds-install moh-install
make moh-install
ln -sf /usr/local/freeswitch/bin/freeswitch /usr/bin/
ln -sf /usr/local/freeswitch/bin/fs_cli /usr/bin/
```

#安装服务

```
cp build/freeswitch.init.redhat /etc/init.d/freeswitch
chmod 755 /etc/init.d/freeswitch
vim /etc/init.d/freeswitch
```

```
#
PROG_NAME=freeswitch
PID_FILE=${PID_FILE-/var/run/freeswitch/freeswitch.pid}
#FS_USER=${FS_USER-freeswitch}
FS_USER=${FS_USER-root}
FS_FILE=${FS_FILE-/usr/local/freeswitch/bin/freeswitch}
FS_HOME=${FS_HOME-/usr/local/freeswitch}
LOCK_FILE=/var/lock/subsys/freeswitch
FREESWITCH_ARGS="-nc"
RETVAL=0
```

#保存

```
mkdir /var/run/freeswitch
chkconfig --add freeswitch
```

C. Debian 安装

以 Debian8 为主,顺便可以体验下视频会议融屏

1. 安装 Debian 8

2. 通过图形管理界面更改 ip 地址为静态的或通过命令更改,前提,加载安装盘,先装个 vim,

在 debian 下，vi 我认为不好用

修改 /etc/network/interfaces ，如：

auto eth0 #设置设备名称
iface eth0 inet static #设置接口类型，static 为静态 ip，或者为 dhcp
address 192.168.1.1 #接口地址
netmask 255.255.255.0 #掩码
gateway 192.168.1.254 #网关

为了让配置生效，运行

/etc/init.d/networking restart

3. 配置 debian 源

vim /etc/apt/sources.list

变成以下内容：

 # deb cdrom:[Debian GNU/Linux 8.1.0 _Jessie_ - Official amd64 DVD Binary-1 20150606-14:19]/ jessie contrib main

#deb cdrom:[Debian GNU/Linux 8.1.0 _Jessie_ - Official amd64 DVD Binary-1 20150606-14:19]/ jessie contrib main

#deb http://security.debian.org/ jessie/updates main contrib
#deb-src http://security.debian.org/ jessie/updates main contrib

deb http://mirrors.163.com/debian jessie main non-free contrib
deb http://mirrors.163.com/debian jessie-proposed-updates main contrib non-free
deb http://mirrors.163.com/debian-security jessie/updates main contrib non-free

deb http://security.debian.org jessie/updates main contrib non-free
#以上禁掉 cdrom 和 deb 原生源，改为 163 的，加快速度

 # jessie-updates, previously known as 'volatile'
 # A network mirror was not selected during install. The following entries
 # are provided as examples, but you should amend them as appropriate
 # for your mirror of choice.
#
 # deb http://ftp.debian.org/debian/ jessie-updates main contrib
 # deb-src http://ftp.debian.org/debian/ jessie-updates main contrib

4. 接下来那么就按官方的基本文档走了

a. 添加 freeswitch 的源

15

```
echo "deb http://files.freeswitch.org/repo/deb/debian/ jessie
main" > /etc/apt/sources.list.d/99FreeSWITCH.test.list
wget -O - http://files.freeswitch.org/repo/deb/debian/key.gpg |apt-key add -
apt-get update
```

b. 安装依赖

```
DEBIAN_FRONTEND=none APT_LISTCHANGES_FRONTEND=none apt-get install -y --force-
yes freeswitch-video-deps-most
```

c. git FreeSWITCH 的源码

```
git config --global pull.rebase true
```

```
git clone https://freeswitch.org/stash/scm/fs/freeswitch.git freeswitch.git
cd freeswitch.git
./bootstrap.sh -j
./configure -C
```

d. 开启 mod_av 模块

```
perl -i -pe 's/#applications\/mod_av/applications\/mod_av/g' modules.conf
```

e. 编译并 install

```
make
make install
make cd-sounds-install
make cd-moh-install
make samples
```

f. 修改内核参数

```
/etc/sysctl.d/vid.conf
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
kernel.core_pattern = core.%p
```

系统调用使之生效

sysctl -w net.core.rmem_max=16777216
sysctl -w net.core.wmem_max=16777216
sysctl -w kernel.core_pattern=core.%p

g. 将 mod_av 加载

fs_cli 中 load mod_av 或 autoload_configs/modules.conf.xml 中加一行,

<load module="mod_av"/>

四、 FreeSWITCH 的音频通话

A. FreeSWITCH 启动及查看

Freeswitch 一般安装路径在 linux 下的/usr/local/freeswitch，可执行程序位于 /usr/local/freeswitch/bin 下，配置文件位于 /usr/local/freeswitch/conf 中
先看下 FreeSWITCH 的启动参数

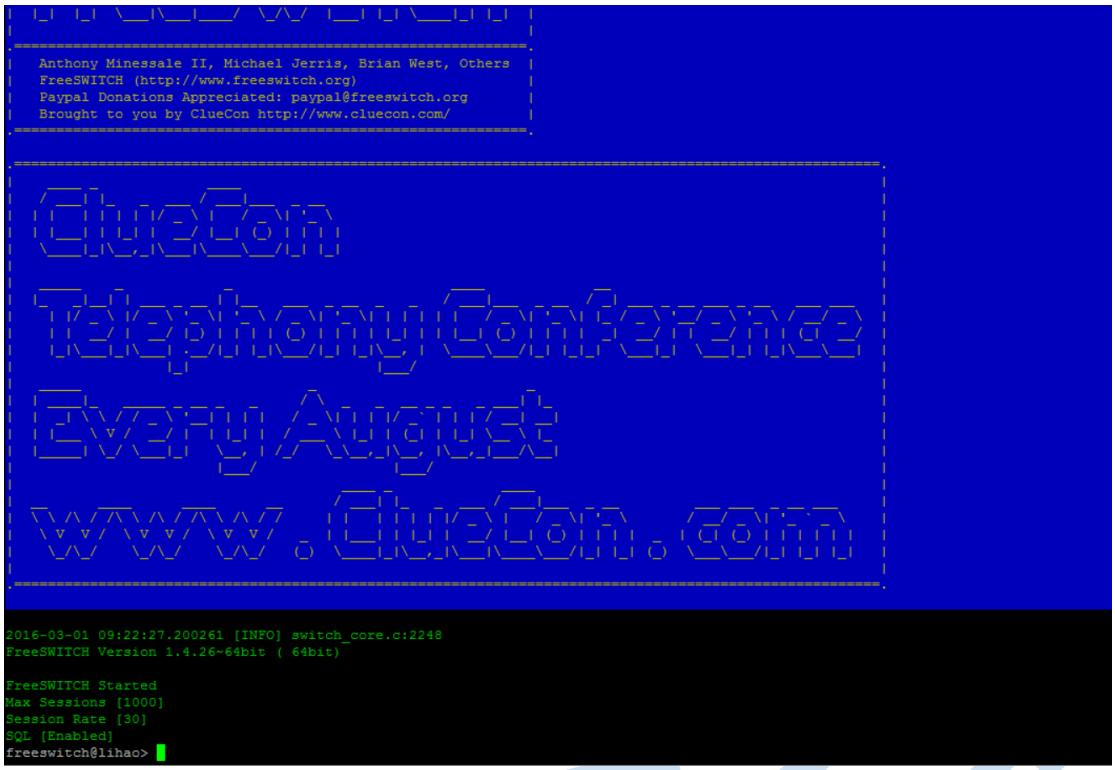
```
root@lihao:~# freeswitch -help
Usage: freeswitch [OPTIONS]

These are the optional arguments you can pass to freeswitch:
        -nf                     -- no forking
        -reincarnate            -- restart the switch on an uncontrolled exit
        -reincarnate-reexec     -- run execv on a restart (helpful for upgrades)
        -u [user]               -- specify user to switch to
        -g [group]              -- specify group to switch to
        -core                   -- dump cores
        -help                   -- this message
        -version                -- print the version and exit
        -rp                     -- enable high(realtime) priority settings
        -lp                     -- enable low priority settings
        -np                     -- enable normal priority settings
        -vg                     -- run under valgrind
        -nosql                  -- disable internal sql scoreboard
        -heavy-timer            -- Heavy Timer, possibly more accurate but at a cost
        -nonat                  -- disable auto nat detection
```

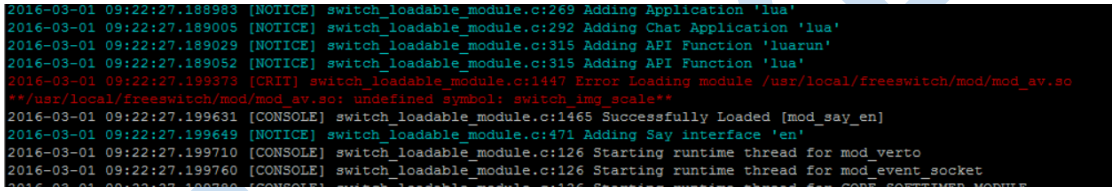
-nonatmap -- disable auto nat port mapping
-nocal -- disable clock calibration
-nort -- disable clock clock_realtime
-stop -- stop freeswitch
-nc -- do not output to a console and background
-ncwait -- do not output to a console and background but wait until the system is ready before exiting (implies -nc)
-c -- output to a console and stay in the foreground

Options to control locations of files:
-base [basedir] -- alternate prefix directory
-cfgname [filename] -- alternate filename for FreeSWITCH main configuration file
-conf [confdir] -- alternate directory for FreeSWITCH configuration files
-log [logdir] -- alternate directory for logfiles
-run [rundir] -- alternate directory for runtime files
-db [dbdir] -- alternate directory for the internal database
-mod [moddir] -- alternate directory for modules
-htdocs [htdocsdir] -- alternate directory for htdocs
-scripts [scriptsdir] -- alternate directory for scripts
-temp [directory] -- alternate directory for temporary files
-grammar [directory] -- alternate directory for grammar files
-certs [directory] -- alternate directory for certificates
-recordings [directory] -- alternate directory for recordings
-storage [directory] -- alternate directory for voicemail storage
-cache [directory] -- alternate directory for cache files
-sounds [directory] -- alternate directory for sound files

如果不带参数，我们直接启动应用的话即 linux 系统中命令行中直接输入 freeswitch，一切顺利的话，我们就启动了应用。在系统使用前，不带参数的启动是必要的，特别是自己有相关修改的情况下，这样可以让我们发现并找出问题。



通过鼠标翻页可以看看我们的系统动行有没有错，这样不用去另外分析日志等，如：



很红的错误，不管是因为什么错，这样列出来后，就可以有针对性的去处理和解决。

当然，我们一般使用的话，是需要把 FreeSWITCH 放到后台运行，则一般是：

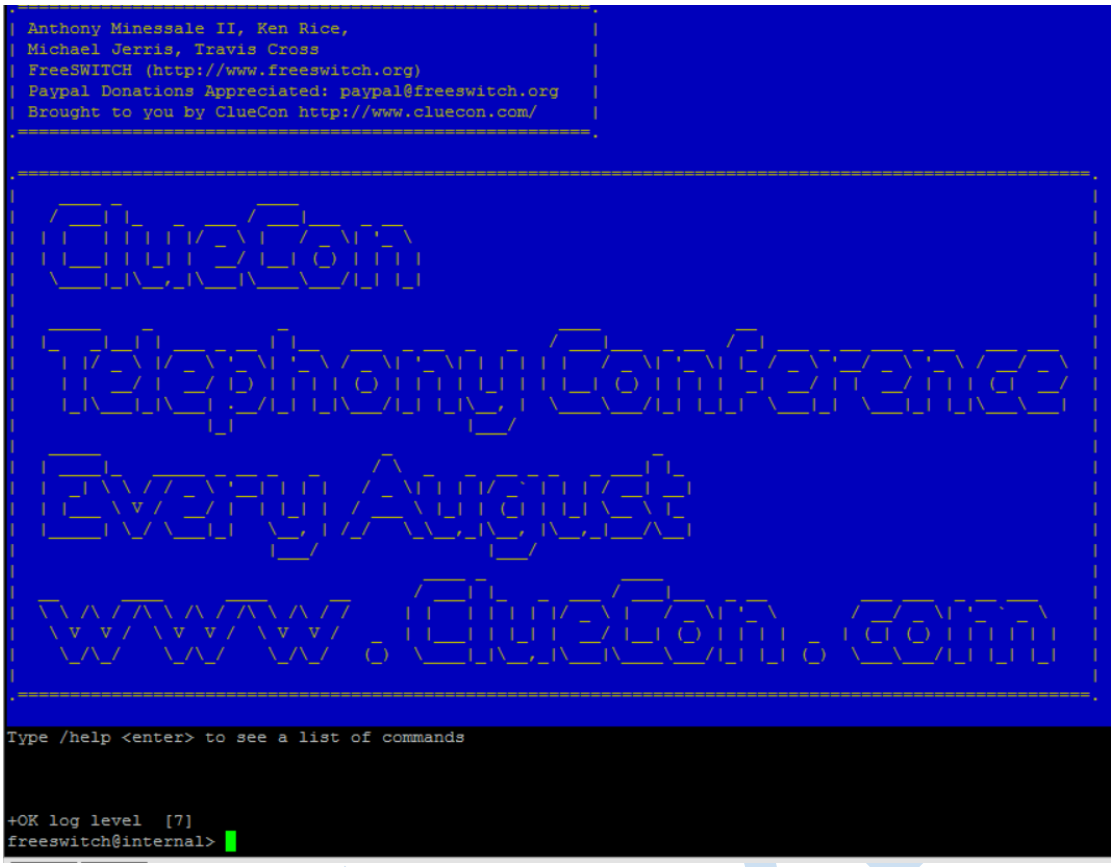
freeswitch –nc

查看相关日志，则是 fs_cli 命令

root@lihao:~# freeswitch -nc

12858 Backgrounding.

root@lihao:~# fs_cli



查看启动与否

如遇异常或要查看 freeswitch 启动得到底对不对，则可以用 netstat –anp | grep freeswitch

root@lihao:~# netstat -anp | grep freeswitch

tcp 0 0 192.168.1.209:5060 0.0.0.0: LISTEN 12898/freeswitch*

tcp 0 0 192.168.1.209:5061 0.0.0.0: LISTEN 12898/freeswitch*

tcp 0 0 192.168.1.209:5061 0.0.0.0: LISTEN 12898/freeswitch*

tcp 0 0 192.168.1.209:5066 0.0.0.0: LISTEN 12898/freeswitch*

tcp 0 0 192.168.1.209:5066 0.0.0.0: LISTEN 12898/freeswitch*

tcp 0 0 192.168.1.209:5067 0.0.0.0: LISTEN 12898/freeswitch*

tcp 0 0 192.168.1.209:8081 0.0.0.0: LISTEN 12898/freeswitch*

tcp 0 0 192.168.1.209:8082 0.0.0.0: LISTEN 12898/freeswitch*

```
tcp        0      0 192.168.1.209:7443      0.0.0.0:*               LISTEN
12898/freeswitch
tcp        0      0 192.168.1.209:7443      0.0.0.0:*               LISTEN
12898/freeswitch
tcp        0      0 192.168.1.209:5080      0.0.0.0:*               LISTEN
12898/freeswitch
tcp        0      0 192.168.1.209:5081      0.0.0.0:*               LISTEN
12898/freeswitch
tcp6       0      0 ::1:5060                :::*                    LISTEN
12898/freeswitch
tcp6       0      0 ::1:5061                :::*                    LISTEN
12898/freeswitch
tcp6       0      0 :::8021                 :::*                    LISTEN
12898/freeswitch
tcp6       0      0 ::1:5080                :::*                    LISTEN
12898/freeswitch
tcp6       0      0 ::1:5081                :::*                    LISTEN
12898/freeswitch
udp        0      0 192.168.1.209:5060      0.0.0.0:*
12898/freeswitch
udp        0      0 192.168.1.209:5067      0.0.0.0:*
12898/freeswitch
udp        0      0 192.168.1.209:5080      0.0.0.0:*
12898/freeswitch
udp        0      0 0.0.0.0:1337            0.0.0.0:*
12898/freeswitch
udp        0      0 239.255.255.250:1900    0.0.0.0:*
12898/freeswitch
udp        0      0 192.168.1.209:59575     192.168.1.1:5351        ESTABLISHED
12898/freeswitch
udp6       0      0 ::1:5060                :::*
12898/freeswitch
udp6       0      0 ::1:5080                :::*
12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     166284   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     166291   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     170300   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     170309   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     164438   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     170310   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     164434   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     170301   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     166290   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     170293   12898/freeswitch
```

```
unix  3      [ ]         STREAM     CONNECTED     170291   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     170292   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     164437   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     170294   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     166283   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     164435   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     166286   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     166942   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     166943   12898/freeswitch
unix  3      [ ]         STREAM     CONNECTED     166285   12898/freeswitch
```


fs_cli 连接不了本机的 freeswitch

检查配置文件下的 autoload_config/event_socket.conf.xml 中的 ip,密码等信息，以及查看默认的 8021 端口是否启动

查看本机 freeswitch 的运行状态

在 fs_cli 中输入 status

```
freeswitch@internal> status
UP 0 years, 0 days, 0 hours, 14 minutes, 55 seconds, 802 milliseconds, 151 microseconds
FreeSWITCH (Version 1.4.26  64bit) is ready
0 session(s) since startup
0 session(s) - peak 0, last 5min 0
0 session(s) per Sec out of max 30, peak 0, last 5min 0
1000 session(s) max
min idle cpu 0.00/99.37
Current Stack Size/Max 240K/8192K

freeswitch@internal>
```

查看 sip 相关的状态

在 fs_cli 中输入 sofia status

```
freeswitch@internal> sofia status
                    Name          Type                                       Data      State
=================================================================================================
           external-ipv6       profile                  sip:mod_sofia@[::1]:5080      RUNNING (0)
           external-ipv6       profile                  sip:mod_sofia@[::1]:5081      RUNNING (0) (TLS)
external-ipv6::example.com      gateway                   sip:joeuser@example.com      NOREG
           192.168.1.209         alias                                 internal      ALIASED
               internal2       profile          sip:mod_sofia@192.168.1.209:5067      RUNNING (0)
               internal2       profile          sip:mod_sofia@192.168.1.209:5061      RUNNING (0) (TLS)
                external       profile          sip:mod_sofia@192.168.1.209:5080      RUNNING (0)
                external       profile          sip:mod_sofia@192.168.1.209:5081      RUNNING (0) (TLS)
    external::example.com      gateway                sip:joeuser@example.com      NOREG
          external::lihao      gateway             sip:FreeSWITCH@38.29.23.22      NOREG
           internal-ipv6       profile                  sip:mod_sofia@[::1]:5060      RUNNING (0)
           internal-ipv6       profile                  sip:mod_sofia@[::1]:5061      RUNNING (0) (TLS)
                internal       profile          sip:mod_sofia@192.168.1.209:5060      RUNNING (0)
                internal       profile          sip:mod_sofia@192.168.1.209:5061      RUNNING (0) (TLS)
=================================================================================================
5 profiles 1 alias

freeswitch@internal>
```

添加一个新的用户

在 /usr/local/freeswitch/conf/directory/default/ 下有默认的 1000-1019 共 20 个帐号，可以通过 copy 并修改其中的 user_id 来实现增加新的号码

FreeSWITCH 中的路由配置

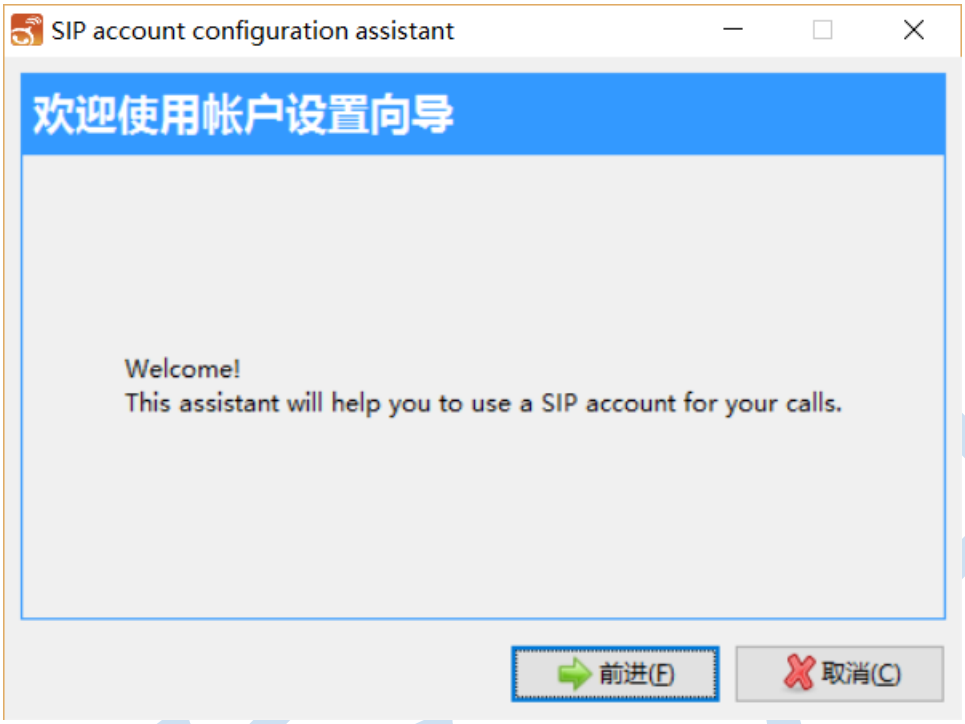
可以修改 /usr/local/freeswitch/conf/dialplan/default.xml 来修改默认路由，也可以往 /usr/local/freeswitch/conf/dialplan/default 下添加新的路由文件来实现
外线来电路由则是由/usr/local/freeswitch/conf/dialplan/public.xml 中配置
路由配置规则

```
<extension name="Local_Extension">
      <condition field="destination_number" expression="^(.*)$">
        <action application="export" data="dialed_extension=$1"/>
        <action application="export"
data="exten_record_file=$${recordings_dir}/${strftime(%Y-%m-%d)}/${caller_id_number}.$1.${st
rftime(%H-%M-%S)}.${uuid}.wav"/>
        <action application="export" data="nolocal:execute_on_answer=dsr
${exten_record_file}"/>
        <action application="set" data="record_file=${exten_record_file}"/>
        <action application="log" data="CRIT execute dsr ${exten_record_file}"/>
        <action application="sleep" data="1000"/>
        <action application="bridge" data="user/${dialed_extension}"/>
      </condition>
    </extension>
```

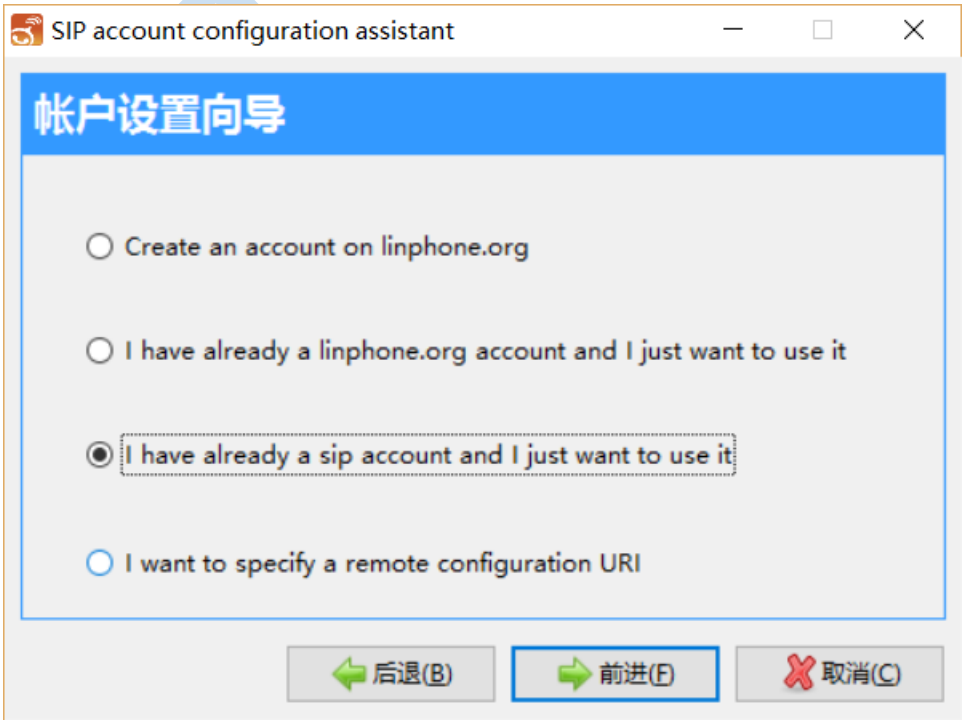
以上为一例，装所有前边如果有路由都匹配不上的路由，都进入到这个路由中，示例中是使用笔者的语音实时质检识别模块，同时进行录音，如果不需要使用它，则将<action application="export" data="nolocal:execute_on_answer=dsr ${exten_record_file}"/>中的 dsr 换为 record_session 即可。

B. linphone 配置

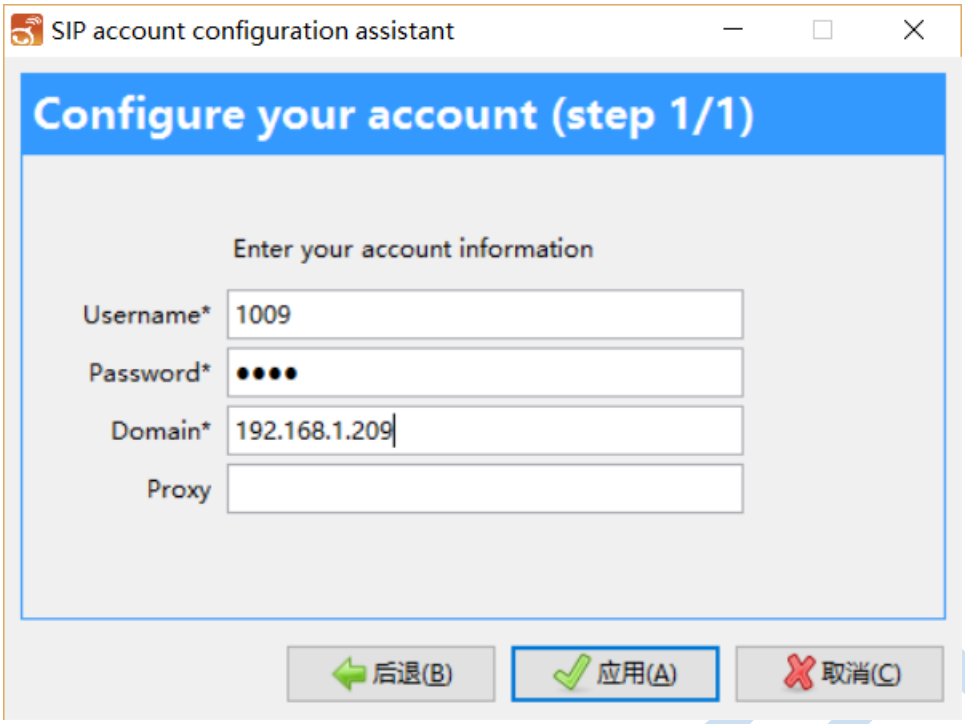
打开 linphone -> options ->首选项 ， 在第一个 tab 中 sip 帐户管理中，点击 wizard



前进



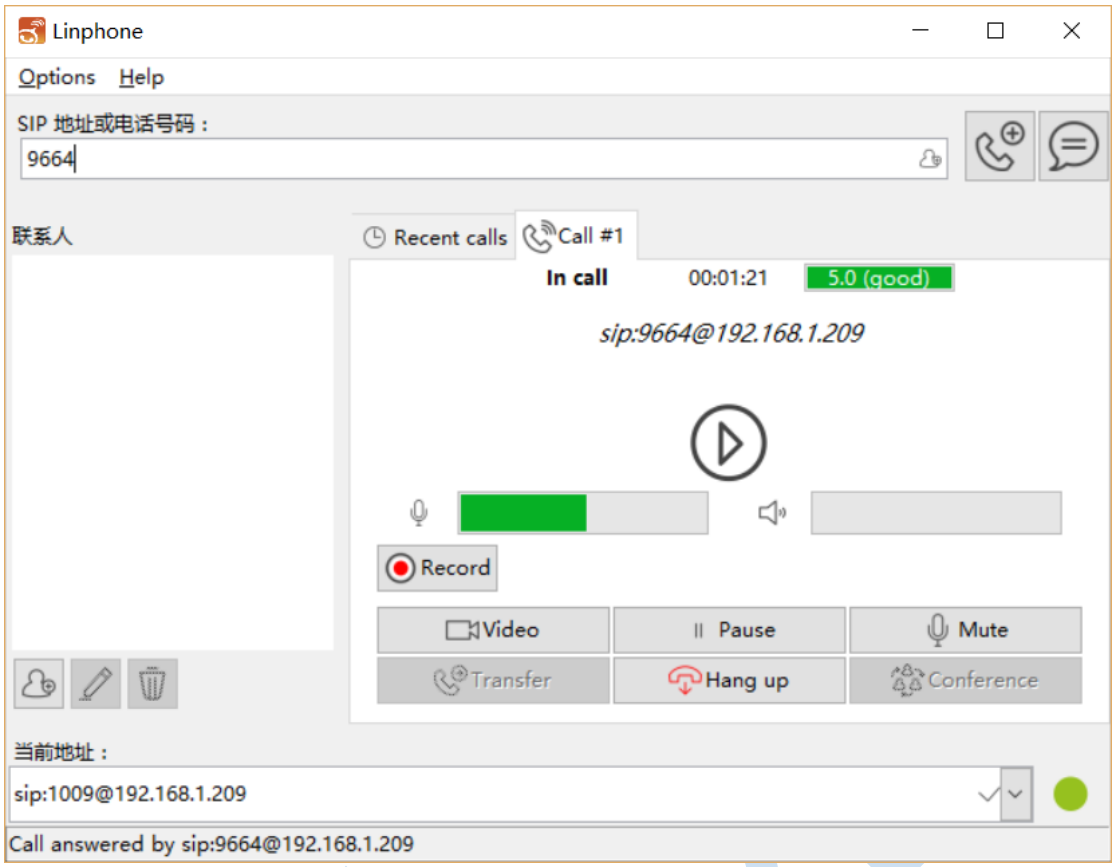
前进



应用

C. linphone 通话及 FreeSWITCH 日志查看

按以上配置就可以进行语音测试了
我们呼叫系统自带放音短号 9664



而同时 Freeswitch 的日志上就会出现相关的 debug，notice 等信息。

如遇到红色的报警，并提示要 sleep 10 秒的，则是由于我们使用了默认密码，可以修改 Conf/vars.xml 中的

<X-PRE-PROCESS cmd="set" data="default_password=1234"/>

或修改 conf/dialplan/default.xml 中

```
<condition field="${default_password}" expression="^1234$" break="never">
    <action application="log" data="CRIT WARNING WARNING WARNING WARNING WARNING WARNING WARNING WARNING WARNING "/>
    <action application="log" data="CRIT Open $${conf_dir}/vars.xml and change the default_password."/>
    <action application="log" data="CRIT Once changed type 'reloadxml' at the console."/>
    <action application="log" data="CRIT WARNING WARNING WARNING WARNING WARNING WARNING WARNING WARNING WARNING "/>
    <action application="sleep" data="10000"/>
</condition>
```

五、 使用 FreeSWITCH 作为视频通话服务器

A. 配置视频相关

FreeSWITCH 默认支持 Google VP8 编码，在编译时，如果遇到 yuv,vpx 等的问题，可以按以下的方式解决

编译 libyuv

git clone https://freeswitch.org/stash/scm/sd/libyuv.git
cd libyuv
make -f linux.mk CXXFLAGS="-fPIC -O2 -fomit-frame-pointer -Iinclude/"
make install
cp /usr/lib/pkgconfig/libyuv.pc /usr/lib64/pkgconfig/

编译 VPX

cd ..
git clone https://freeswitch.org/stash/scm/sd/libvpx.git
cd libvpx
./configure --enable-pic --disable-static --enable-shared
make
make install
cp /usr/local/lib/pkgconfig/vpx.pc /usr/lib64/pkgconfig/

编译 OPUS

cd ..
git clone https://freeswitch.org/stash/scm/sd/opus.git
cd opus
./autogen.sh
./configure
make
make install
cp /usr/local/lib/pkgconfig/opus.pc /usr/lib64/pkgconfig

编译 libpng

git clone https://freeswitch.org/stash/scm/sd/libpng.git
cd libpng

```
./configure
make
make install
cp /usr/local/lib/pkgconfig/libpng*/usr/lib64/pkgconfig/
```

编译 openh264


```
git clone https://freeswitch.org/stash/scm/sd/openh264.git
cd openh264/
make ENABLE64BIT=Yes
make install
cp openh264.pc /usr/lib64/pkgconfig/

pkg-config --list-all | grep openh264
```

编译 FreeSWITCH

```
cd freeswitch
echo "codecs/mod_openh264" >> modules.conf
./configure --prefix=/usr/local/freeswitch
make
make install
```

测试

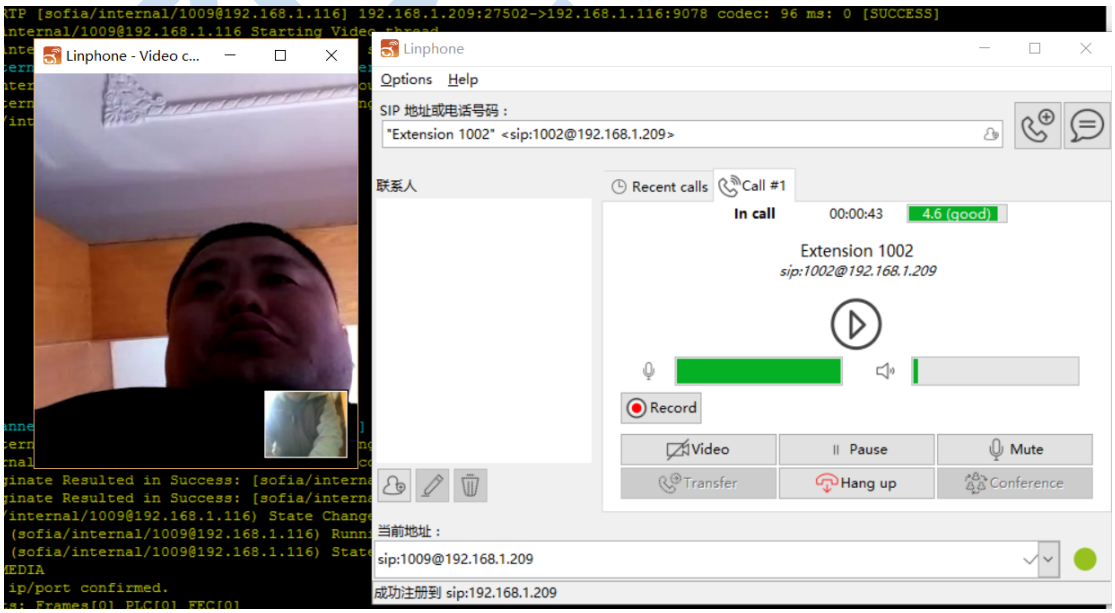
一。修改 autoload_configs/modules.conf.xml

注释 mod_h26x 添加 mod_openh264

二。修改 vars.xml codec 中添加 H264 编码

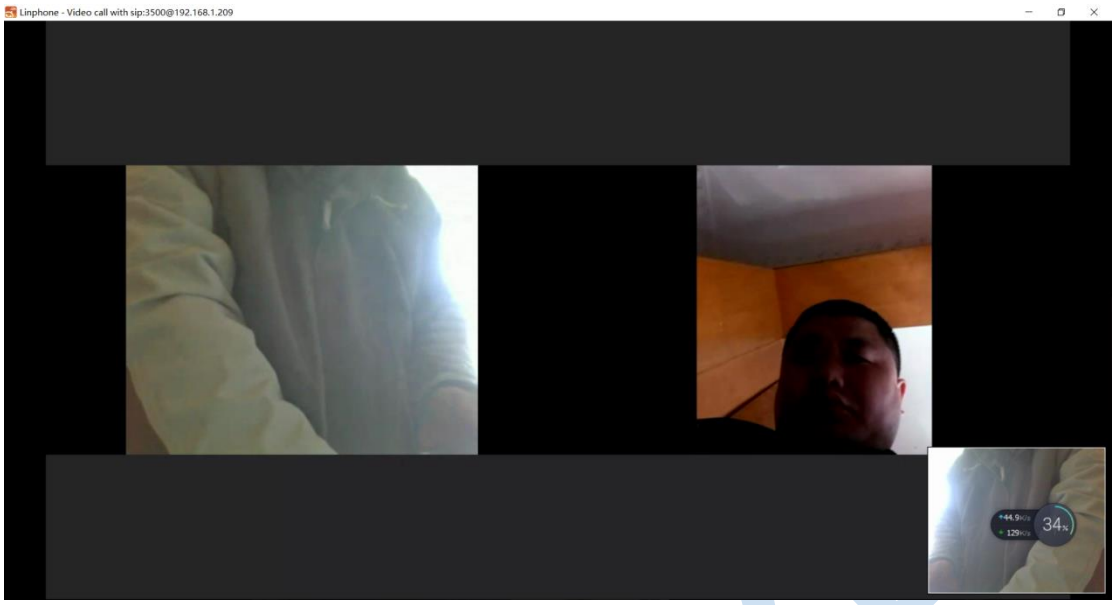
B. Linphone 配置视频通话

在以上的服务端配置了 vp8 和 h264 后，那么我们的客户支持这两种编码其一时，就可以视频通话了



C. FreeSWITCH 视频会议相关

由于 FreeSWITCH 开发组开发视频相关时，选择 livav 这个从 ffmpeg 库中分裂出来的库，而 libav 只在 Debian 8 下表现良好，所以这个方面只支持 Debian 8 系统下，加载成功 mod_av 即可，然后呼 3500 即可加入到默认的视频会议。



六、 FreeSWITCH 与外线连接

A. 与 sangoma 板卡相连

详见：
http://wiki.sangoma.com/wanpipe-freeswitch-ftdm-installation

http://wiki.sangoma.com/wanpipe-freeswitch-config

那么就可以通过

```
Originate freetdm/wp1/a/18621575908 &echo
```

去呼叫电话号码了

B. 与网关或 Voip 外线连接

在 /usr/local/freeswitch/conf/sip_profiles/external 下添加新的网关配置文件，如：

```
< include >
< gateway name="lihao">
 < param name="realm" value="210.83.80.xx"/>
 < param name="proxy" value="210.83.80.xx"/>
 < param name="register" value="false"/>
< / gateway>
< / include>
```


然后在 fs_cli 中，

sofia profile external rescan

通过 sofia status 就可以看到刚才的外线，那么接下来就可以：

```
originate sofia/gateway/lihao/18621575908 &echo
```

去呼叫电话号码了

七、 FreeSWITCH 与 WEBRTC

A. 什么是 WEBRTC

WebRTC，名称源自网页实时通信（Web Real-Time Communication）的缩写，是一个支持网页浏览器进行实时语音对话或视频对话的技术，是谷歌 2010 年以 6820 万美元收购 Global IP Solutions 公司而获得的一项技术。

B. 让 FreeSWITCH 支持 WEBRTC

开启 /usr/local/freeswitch/conf/sip_profiles/internal.xml 中的

```
<param name="ws-binding"    value=":5066"/>
<param name="apply-candidate-acl" value="121.40.40.40"/>
<param name="apply-candidate-acl" value="rfc1918.auto"/>
 <param name="apply-candidate-acl" value="localnet.auto"/>
```

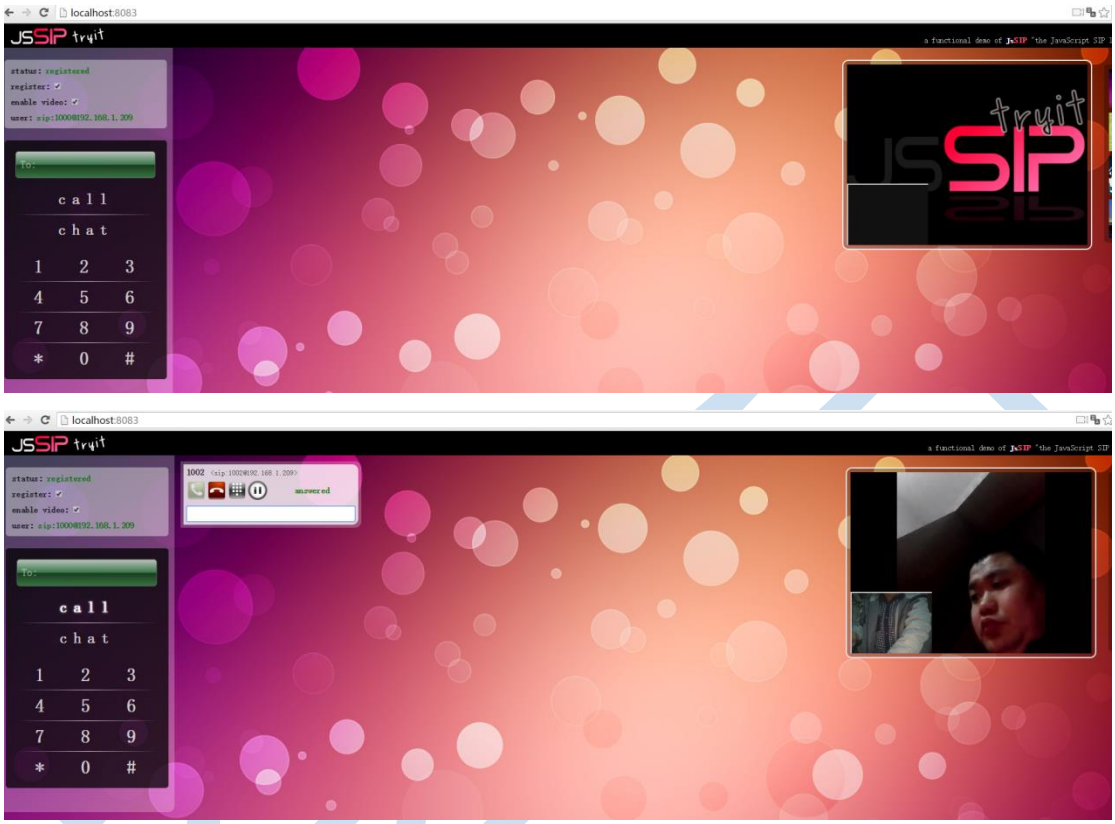
即由 5066 端口来响应来自于 WEBRTC 的请求。

C. 使用 Jssip 来实现 webrtc 通话

将 Jssipi 的压缩包到如 D:\WEBRTC\Jssip 下，运行

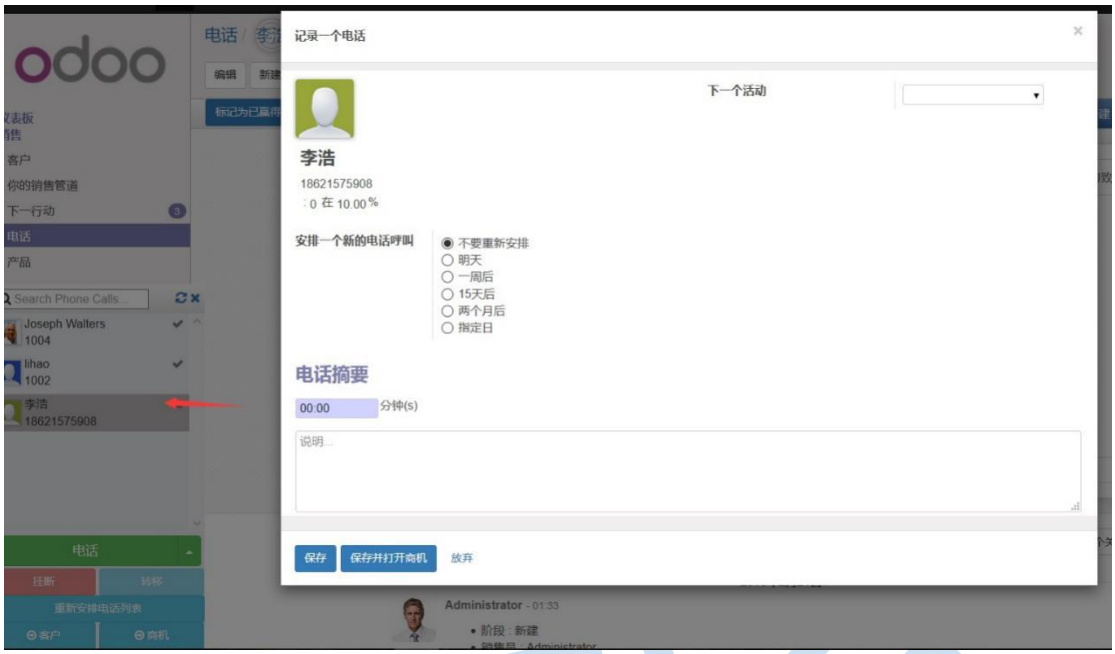
python -m SimpleHTTPServer 8083

那么我们就可以在 chrome 中该问 localhost:8083
从而注册到 FreeSWITCH 中



D. Sip.js 与 Odoo 与 FreeSWITCH 结合

基于以上的配置，可以与非常庞大的开源的 ERP 系统 Odoo 结合起来



八、 FreeSWITCH 的彩铃和 IVR

A. 来去电回应

Answer
Bridge
Playback
Loopback
。。。。。。

B. Ring 的格式及转码

PCM
G711u
G729
Opus
ILBC
MP3

C. IVR 配置

conf/ivr_menus/demo_ivr.xml

以及路由中指定
<action application="ivr" data="demo_ivr"/>

九、 FreeSWITCH 的 API 与 APP

可以通过 show api 和 show application 来查看 FreeSWITCH 内部的 api 与 app.app 从理论上来说是作用于通道，不论是在 dialplan 还是通过 esl outbound 连接过来的通话线程，诸如：
echo
playback
bridge
answer
hangup
transfer
log
set
export
record
record_session
。。。。。。。
而 api 则主要可以认为是对整个系统的操作，不局限于某一个单个的通道，如果要在单个通道进行操作则需要提供 uuid,如:
originate
uuid_transfer
uuid_bridge
uuid_break
uuid_kill
uuid_broadcast
uuid_answer
.....
如果要在 esl inbound 应用中调用，则一般需要采用 bgapi/api originater user/1000 1001 等

十、 FreeSWITCH Inbound 连接

```
#include <esl.h>


int main(void)
{
	esl_handle_t handle = {{0}};
```

```
    esl_connect(&handle, "localhost", 8021, NULL, "ClueCon");

    esl_send_recv(&handle, "api status\n\n");

    if (handle.last_sr_event && handle.last_sr_event->body) {
        printf("%s\n", handle.last_sr_event->body);
    } else {
        printf("%s\n", handle.last_sr_reply);
    }

    esl_disconnect(&handle);

    return 0;
}
```

十一、 FreeSWITCH Outbound 连接

```
#include <esl.h>

static void mycallback(esl_socket_t server_sock, esl_socket_t client_sock, struct sockaddr_in *addr,
void *user_data)
{
    esl_handle_t handle = {{0}};
    int done = 0;
    esl_status_t status;
    time_t exp = 0;

    esl_attach_handle(&handle, client_sock, addr);

    esl_log(ESL_LOG_INFO, "Connected! %d\n", handle.sock);

   esl_filter(&handle, "unique-id", esl_event_get_header(handle.info_event, "caller-unique-id"));
    esl_events(&handle, ESL_EVENT_TYPE_PLAIN, "SESSION_HEARTBEAT CHANNEL_ANSWER
CHANNEL_ORIGINATE CHANNEL_PROGRESS CHANNEL_HANGUP "
                    "CHANNEL_BRIDGE CHANNEL_UNBRIDGE CHANNEL_OUTGOING
CHANNEL_EXECUTE CHANNEL_EXECUTE_COMPLETE DTMF CUSTOM conference::maintenance");

    esl_send_recv(&handle, "linger");

    esl_execute(&handle, "answer", NULL, NULL);
```

```
    esl_execute(&handle, "conference", "3000@default", NULL);

    while((status = esl_recv_timed(&handle, 1000)) != ESL_FAIL) {
        if (done) {
            if (time(NULL) >= exp) {
                break;
            }
        } else if (status == ESL_SUCCESS) {
            const char *type = esl_event_get_header(handle.last_event, "content-type");
            if (type && !strcasecmp(type, "text/disconnect-notice")) {
                const char *dispo = esl_event_get_header(handle.last_event, "content-
disposition");
                esl_log(ESL_LOG_INFO, "Got a disconnection notice dispostion: [%s]\n", dispo ?
dispo : "");
                if (dispo && !strcmp(dispo, "linger")) {
                    done = 1;
                    esl_log(ESL_LOG_INFO, "Waiting 5 seconds for any remaining events.\n");
                    exp = time(NULL) + 5;
                }
            }
        }
    }

    esl_log(ESL_LOG_INFO, "Disconnected! %d\n", handle.sock);
    esl_disconnect(&handle);
}

int main(void)
{
    esl_global_set_default_logger(7);
    esl_listen_threaded("localhost", 8040, mycallback, NULL, 100000);

    return 0;
}
```

十二、 FreeSWITCH 与 LUA

A.什么是 Lua

Lua 是一个小巧的脚本语言。是属于由 c 实现的嵌入式语言，需要由其它语言发起并调用.

B. 在 FreeSWITCH 中如何调用 Lua

Mod_lua，那么就可以通过 lua 去调用相关脚本
默认执行的脚本属于/usr/local/freeswitch/scripts
可以用如 lua a.lua 来执行某个 lua 脚本

C. 使用 lua 与数据库协助 FreeSWITCH 管理用户

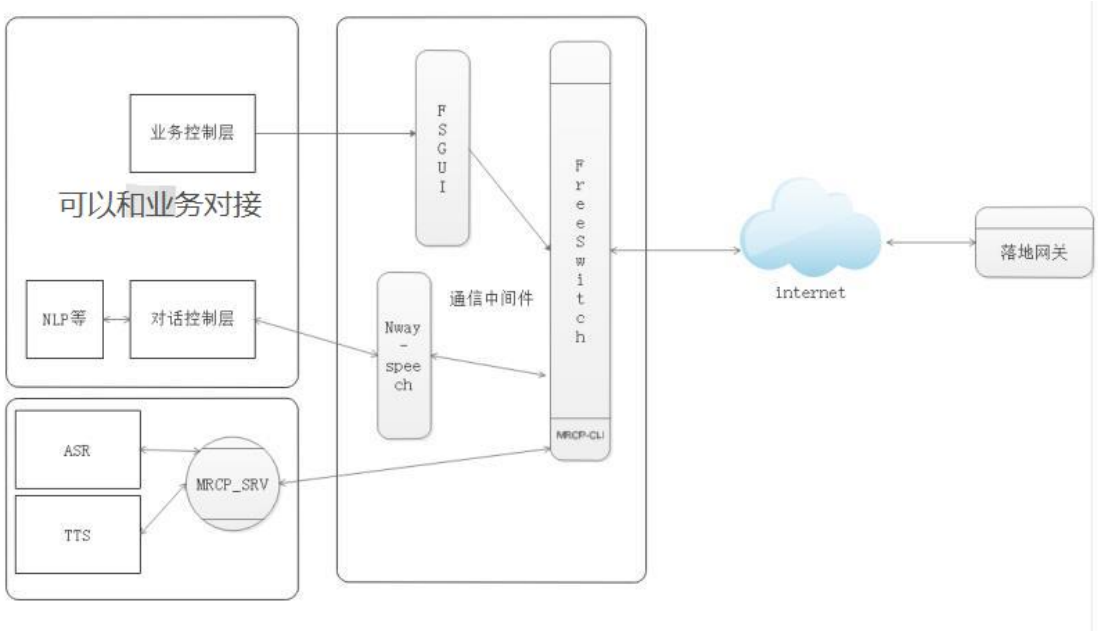
配置 conf/autoload_config/lua.con.xml
使用 gen_dir_user_xml.lua

十三、 其它与 FreeSWITCH 相关的开发语言

不建议使用 java 开发这类的，协议中采用的类似于文本模式的，默认的 esl-java 库相对封装的简单，在性能和扩展上存在一定的问题。
Python
C++
C
Php
Go lang
Ruby
等语音都可以按照 event socket 的传输协议进行封装使用

十四、智能客服、外呼

a. 系统架构图



b. 智能客服

客服由于其业务扩散的特点，要做好智能客服，远比外呼系统要复杂。因为呼入时，对方的逻辑是围绕着他打进电话的需求来，可能会有方言、口音、平时交流的一些习惯等，智能客服需要围绕着呼入方的思路把电话流程引正且最终成为为呼入用户提供面向客服业务的服务。

B1. 技术特点

沟通点比较发散，所以问题点多，比较难应对

方言、口音、年龄、性别等带来的问题点更多，识别和处理均有难度

面向多业务时，要分清业务，同时不同业务间要跳转，达到一个类似真人的应对，有难度

呼入时会有呼入方进行大量的数据识别，对语音识别有挑战。

B2. 业务特点

要和业务强耦合

对 nlp 的要求较高

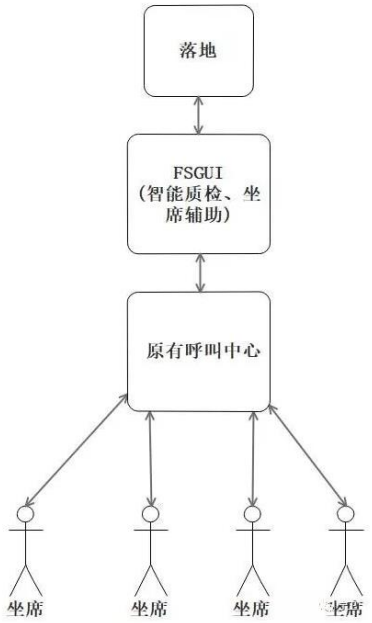
B3. 技术类型

上图采用的是 MRCP 协议，在 FreeSwitch 中使用 unimrcp 协议进行通信，但也可以自己抓取媒体流，然后采用一些公有云中的 sdk,http 等接口来完成。

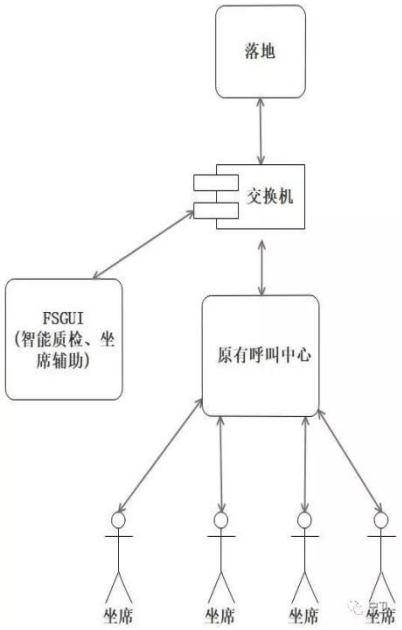
十五、语音实时识别

作为语音通信，我们在这里含两种不同的架构的语音实时识别，用于电话实时质检或座席辅助。

串联的用于把这语音实时识别系统作为落地使用，架构图如下：



并联的用于不改变原有的系统架构和使用，只是在交换机处做一个端口镜像，将原系统中的数据推送过来，由我们的系统进行抓包处理



十六、FSGui 介绍

FSGui 是由上海宁卫信息技术有限公司自主研发的新一代的呼叫平台，它将云呼叫、VOIP、PSTN、IMS、DID、IVR 等集成在一起，实现将 IP 网络和传统通信通过语音完美结合在一起。并为第三方呼叫及事件查询提供 RESTful 接口。

整个系统分为如下的结构

应用说明：

WEBServer nway_pbx_web 用于处理所有的 web 访问，但不包括 restful 接口

AUTHServer nway_pbx_auth 用于处理 FreeSwitch 的 Register 消息

PBXServer nway_pbx 业务主应用，用于处理路由，网关，IVR 等呼叫业务层

FreeSwitch

Postgresql

Redis

具体参考 http://www.freeswitch.net.cn/100.html

附录：

安装问题

源码快速 git 地址

我们镜像了一个 FreeSwitch 的源码镜像

```
https://git.oschina.net/nwaycn/freeswitch.git
```

到底如何选择一个版本

在 freeSwitch 使用中，大的版本号，如 1.5 这种奇数版本是开发版，1.4 这种版本是稳定版。如果没有太多的视频类的业务需求，只是语音层面的，建议使用 1.4.26 即可。更高版本在我们做语音通信中，并不是特别好的一个选择。

如何去编译某个模块

Freeswitch 中包含了大量的模块，合计 15 大类，在 150-200 个功能 module 间，如果要编译，则在源代码中 modules.conf 中修改配置，以便生成对应的 Makefile，重新 configure 后，再编译.
当然如果已完成了编译，则可以使用 make mod_xxx-install 来随时编译某个模块及复制到 $FSPATH/mod 下

如何选择一个操作系统

不建议使用 linux 外的其它系统，即使是 linux 也建议是 centos/redhat,debian，其它的不建议使用，因为开发者的能力，开源各框架的跨平台能力等，都需要重新斟酌。

如何在 centos 上安装 libyuv,vpx,opus,libpng,libav

```
1. 编译 libyuv
2. git clone https://freeswitch.org/stash/scm/sd/libyuv.git
3. cd libyuv
4. make -f linux.mk CXXFLAGS="-fPIC -O2 -fomit-frame-pointer -Iinclude/"
5. make install
6. cp /usr/lib/pkgconfig/libyuv.pc /usr/lib64/pkgconfig/
7. 编译 VPX
8. cd ..git clone https://freeswitch.org/stash/scm/sd/libvpx.git
9. cd libvpx./configure --enable-pic --disable-static --enable-shared
10. make
11. make install
12. cp /usr/local/lib/pkgconfig/vpx.pc /usr/lib64/pkgconfig/
13. 编译 OPUS
14. cd ..git clone https://freeswitch.org/stash/scm/sd/opus.git
15. cd opus./autogen.sh./configure
16. make
17. make install
18. cp /usr/local/lib/pkgconfig/opus.pc /usr/lib64/pkgconfig
19. 编译 libpng
20. git clone https://freeswitch.org/stash/scm/sd/libpng.git
21. cd libpng ./configure
22. make
23. make install
24. cp /usr/local/lib/pkgconfig/libpng*/usr/lib64/pkgconfig/
25. 编译 libav
26. git clone https://freeswitch.org/stash/scm/sd/libav.git
27. cd libav
28. ./configure                          #CFLAGS="-fPIC" ./configure --enable-pic -
    -enable-shared
29.
30. make                     # make CXXFLAGS="-fPIC"
31.
32. make install
```

如何在 centos 上快速源码编译一套 freeswitch

```
1. yum  -y  install wget
2. wget http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-
   8.noarch.rpm
3. yum install -y git subversion autoconf automake libtool gcc-c++ ncurses-
   devel make
4. yum -y install expat-devel openssl-devel libtiff-devel libX11-
   devel unixODBC-devel libssl-devel python-devel
5. yum -y install zlib-devel libzrtpcpp-devel alsa-lib-devel libogg-
   devel libvorbis-devel perl-libs gdbm-devel
6. yum -y install libdb-devel uuid-devel @development-tools patch
7. yum -y install ldns-devel libidn-devel unbound-devel
8. yum -y install libjpeg-devel sqlite-devel libcurl-devel
9. yum -y install pcre-devel libuuid-devel bison-devel bison
10. yum -y install speex-devel libtheora-devel yasm nasm libedit-devel
11. cd /usr/local/src
12. git clone -b v1.4 https://git.oschina.net/nwaycn/freeswitch.git fs1.4
13. cd fs1.4
14. ./bootstrap.sh -j
15. ./configure -C
16. make && make install
17. make all install cd-sounds-install cd-moh-install
18. make sounds-install moh-install
19. make moh-install
20. ln -sf /usr/local/freeswitch/bin/freeswitch /usr/bin/
21. ln -sf /usr/local/freeswitch/bin/fs_cli /usr/bin/
```

如何让 freeswitch 支持 h264

编译 openh264

```
git clone https://freeswitch.org/stash/scm/sd/openh264.git
cd openh264/
make ENABLE64BIT=Yes
make install
cp openh264.pc /usr/lib64/pkgconfig/
pkg-config --list-all | grep openh264
```

编译 openh264 模块

```
cd freeswitch
echo "codecs/mod_openh264" >> modules.conf
./configure --prefix=/usr/local/freeswitch
make
make install
```

测试

1. 修改 autoload_configs/modules.conf.xml

注释 mod_h26x 添加 mod_openh264

2. 修改 vars.xml codec 中添加 H264 编码

如何让 freeswitch 支持 postgresql

1. 安装 postgresql-devel 相关包

```
./configure --enable-core-pgsql-support
```

使用问题

如何增加一个分机帐号

如果是源码安装的，在/usr/local/freeswitch/conf/directory/default 下有默认的 20 个帐号，可以类似的添加一个新的 extension.xml 增加

如何动态增加一个分机帐号

a. 改变 mod_sofia 内部有关帐户认证机制，从数据库中直接取
b. 采用 lua 脚本，通过 lua 脚本从数据库中获取，需运行 fs 前配置 lua.conf.xml

```
<param name="xml-handler-script" value="gen_dir_user_xml.lua"/>

<param name="xml-handler-bindings" value="directory"/>
```

c. 采用 xml_curl 使用其它语言开发相对应的 service 来完成动态帐号

<param name="gateway-url" value="http://127.0.0.1:3000" bindings="directory"/>

FreeSWITCH 使用域名注册

/usr/local/freeswitch/conf/sip_profiles/internal.xml 中

43

```
<param name="force-register-domain"value="$${domain}"/>

<param name="force-subscription-domain"value="$${domain}"/>

<param name="force-register-db-domain"value="$${domain}"/>
```

这些注释掉即可

有关透传号码及由平台发起呼叫或回拨

透传号码，这只是个技术名词，在运营商的交互中，用户侧看到的其实都是虚的号码，物理地址才是绝对的，所以不要去多想，在一些业务场景，比如我们在做 400 电话或 95 外呼等时，运营商许可我们是可以透传相关的号码出局的。以下即由 gw 这个落地呼叫 18621575908 并透传 13671947488 这个号码，接通后转给 1000 这个内部分机

```
originate {origination_caller_id_name='13671947488',origination_caller_id_nu
mber=13671947488,effective_caller_id_name=13671947488,effective_caller_id_nu
mber=13671947488}sofia/gateway/gw/018621575908 &bridge(user/1000)
```

在 dialplan 中配置时采用 set 即可

如何采用 esl inbound 处理路由

所有的通话直接送给 park，由 park 这个 freeswitch event 产生后送到 inbound 应用中处理

<action application="park"/>

如何采用 esl outbound 处理路由

在对应的路由中使用 socket

即类似于

<action application="socket" data="127.0.0.1:8083 async full"/>

如何向一个正在通话的通道中送 dtmf

如果是 inbound 则用 `uuid_send_dtmf`
如果是 outbound 则用 `send_dtmf`

44

如何配置 mrcp

```
<include>
  <profile name="xfmrcpv2asr" version="2">
    <param name="client-ip" value="172.100.0.217"/>
    <param name="client-port" value="8101"/>
    <param name="server-ip" value="172.100.0.200"/>
    <param name="server-port" value="5070"/>
    <param name="sip-transport" value="udp"/>
    <param name="rtp-ip" value="172.100.0.200"/>
    <param name="rtp-port-min" value="30000"/>
    <param name="rtp-port-max" value="58000"/>
    <param name="rtcp" value="1"/>
    <param name="rtcp-bye" value="2"/>
    <param name="rtcp-tx-interval" value="5000"/>
    <param name="rtcp-rx-resolution" value="1000"/>
    <param name="codecs" value="PCMU PCMA L16/96/8000"/>
    <synthparams>
    </synthparams>
    <recogparams>
    </recogparams>
  </profile>
</include>
```

Freeswitch 配置外呼并录音

```
<include>
  <extension name="call out">
    <condition field="destination_number" expression="^(\d+)$">
      <action application="set" data="RECORD_TITLE=Recording ${destination_number} ${caller_id_name} ${strftime(%Y-%m-%d %H:%M)}"/>
        <action application="set" data="RECORD_COPYRIGHT=(c) 2015"/>
        <action application="set" data="RECORD_SOFTWARE=FreeSwitch"/>
        <action application="set" data="RECORD_ARTIST=FreeSwitch"/>
        <action application="set" data="RECORD_COMMENT=FreeSwitch"/>
        <action application="set" data="RECORD_DATE=${strftime(%Y-%m-%d %H:%M)}"/>
```

```
        <action application="set" data="RECORD_STEREO=true"/>
        <action application="set" data="record_sample_rate=8000"/>
        <action application="record_session" data="/home/voices/records/${strftime(%Y-%m-%d)}/${strftime(%Y-%m-%d-%H-%M-%S)}_$1_${caller_id_name}.wav"/>
        <action application="export" data="dialed_extension=$1"/>
        <action application="set" data="call_timeout=30"/>
        <action application="set" data="payment=0"/>
        <action application="set" data="service_nibble=0"/>
        <action application="set" data="hangup_after_bridge=false"/>
        <action application="set" data="continue_on_fail=NORMAL_TEMPORARY_FAILURE,USER_BUSY,NO_ANSWER,TIMEOUT,NO_ROUTE_DESTINATION"/>
        <action application="set" data="continue_on_fail=false"/>
        <action application="export" data="RECORD_STEREO=false"/>
        <action application="bridge" data="sofia/gateway/gw1/$1"/>
    </condition>
  </extension>
</include>
```

如果要在接通时才录音，则将 record_session 改为

```
<action application="set" data="execute_on_answer=record_session /home/voices/records/${strftime(%Y-%m-%d)}/${strftime(%Y-%m-%d-%H-%M-%S)}_$1_${caller_id_name}.wav"/>
```

ESL 中获取是呼入 fs 还是由 fs 呼出的

Call-Direction,这个消息头是标明 inbound 或 ounbound

ESL 中如何收 DTMF

在 lua 或 outbound 应用中，使用类似：

```
esl_execute(handle, "play_and_get_digits",
                    "4 5 3 5000 # 'say:请输入您的密码，以井号结束' "
                    ERROR_PROMPT " charge_password ^\\d{4}$", uuid);
```

注：最少为 4 位最多为 5 位重复 3 次 5000 毫秒超时，如果不到最高位的结束符为#，say 为播放的语音，可以换成 wav 文件，ERROR_PROMPT 为输入无效时的提示音，可以为 wav，charge_password 为存在 fs 系统中的变量，此处用^\\d{4}$代表 4 位整型数字

如果是在 inbound 应用中，则需要按 uuid 进行缓存 dtmf

代码重启 fs

```
fs_cli -x "fsctl shutdown restart"
```

允许或限制多终端注册


```
<param name="multiple-registrations" value="true"/>
```

如何设置一个 FS 服务器支持的并发数？

Switch.conf.xml 中的 max-sessions

如何设置一个 FS 服务器每秒呼叫数

Switch.conf.xml 中的 sessions-per-second

如何设置一个 FS 服务器的 rtp 端口范围

Switch.conf.xml 中的

```
<param name="rtp-start-port" value="16384"/>
<param name="rtp-end-port" value="32768"/>
```

如何修改一个编码的 ptime

Switch.conf.xml 中的

```
<default-ptimes>
    <!-- <codec name="G729" ptime="40"/> -->
</default-ptimes>
```

如何一直保持某个呼入不被挂断

park_after_bridge=true
hangup_after_bridge=false
set 以上两个值

将接通的电话转至 conference

```
uuid_transfer uuid -both 'conference:3000' inline
```

从 fs_cli 查看相关具体的事件


```
/event plain CHANNEL_HANGUP
```

中止当前某个通道上的操作

```
  uuid_break xxxx all
```

查看 fs 中相关 sip profile 信息

```
sofia status profile  external
```

开启 sip 包跟踪

```
1.  sofia profile internal siptrace on
2.  sofia global siptrace on
```

变更日志级别

```
console loglevel 7
```

发送（180 RINGING）的效果

在 dialplan 中配置
ring_ready
或使用 esl inbound
uuid_ring_ready

重新注册网关

```
sofia profile external register gw
```

Fs 呼叫时传真实的彩铃

```
originate {return_ring_ready=true}sofia/gateway/gw/18621575908 &bridge(user/8001)
```

fs 监听某个通话

48

eavesdrop
在 default.xml 中就有示例

```
<extensionnameextensionname="global"continue="true">
  <condition>
    <actionapplicationactionapplication="info"/>
    <actionapplicationactionapplication="db"data="insert/spymap/${caller_id_number}/${uuid}"/>
    <actionapplicationactionapplication="db"data="insert/last_dial/${caller_id_number}/${destination_number}"/>
    <actionapplicationactionapplication="db"data="insert/last_dial/global/${uuid}"/>
  </condition>
</extension>

<extensionnameextensionname="eavesdrop">
  <conditionfieldconditionfield="destination_number"expression="^88(.*)$|^\*0(.*)$">
    <actionapplicationactionapplication="answer"/>
    <actionapplicationactionapplication="eavesdrop"data="${db(select/spymap/$1$2)}"/>
  </condition>
</extension>
```

使用 esl 监听

```
api originate sofia/default/1001@nway.com &eavesdrop(uuid)
```

需要在 esl inbound 应用中记录之前通话的 uuid

Fs 同步系统时间

```
fsctl sync_clock

fsctl sync_clock_when_idle
```

优化一、采用内存数据库

Linux 下

备份原来的 db

mv /usr/local/freeswitch/db /usr/local/freeswitch/db_old

创建新的目录

mkdir /usr/local/freeswitch/db

挂载内存库

mount -t tmpfs tmpfs /usr/local/freeswitch/db

WINDOWS 下

使用相关工具，做一个内存盘

在磁盘中做一个基本的安装目录后，进行配置且保存

做一个服务，把磁盘中的安装目录拷到内存盘中，并运行 freeswithconsole.exe 即可。

优化二、使用 jemalloc

a. 下载

https://github.com/jemalloc/jemalloc/archive/4.0.4.tar.gz

b. 解压

tar zxvf jemalloc-4.0.4.tar.gz

c. 编译

cd jemalloc-4.0.4

./configure --prefix=/usr/local

make

make install

d. 添加到 etc/profile 中

export LD_PRELOAD=/usr/local/lib/libjemalloc.so

保存后，source /etc/profile

这样在重启相关应用后，就会用 Jemalloc 等代替标准库的 malloc 等进行内存分配和释放

FreeSWITCH 与线路网关对接(IP 认证)

50

```
<include>

<gateway name="lihao">

    <param name="realm" value="210.83.80.xx"/>

    <param name="proxy" value="210.83.80.xx"/>

    <param name="register" value="false"/>

</gateway>

</include>
```

在$FS_PATH/conf/sip_profiles/external/下建一个文件如 nway.xml,内容如以上 xml 内容，然后 freeswitch -stop && freeswitch -nc 或者运行 fs_cli 后：

```
sofia profile external rescan reloadxml
```

FreeSWITCH 与线路采用密码验证

```
<include>

    <gateway name="nway">

        <param name="username" value="51531234"/>

        <param name="password" value="23332ddefrr"/>

        <param name="realm" value="220.196.xx.xxx"/>

        <param name="proxy" value="220.196.xx.xxx"/>

        <param name="register" value="true"/>

    </gateway>

</include>
```

其它同上

如何设置最长通话时间

```
originate {execute_on_answer=\'sched_hangup +200\'}user/1000 &endless_playback(\'welcome.wav\')
```


FreeSwitch 中用户不经过认证即可注册成功

一般来说，FreeSwitch 中的 SIP 用户都需要通过用户名和密码进行认证后才能注册成功，并进行通话。若有特殊需要，也可以设置为无认证即可使用，具体设置如下 ：
打开 \conf\sip_profiles\internal.xml ，将如下两条设置去掉注释即可，即：

```
1.  <param name="suppress-cng" value="true"/>
2.  <param name="accept-blind-reg" value="true"/>
3.  <param name="accept-blind-auth" value="true"/>
```

同时在 acl.conf.xml 中添加相对应的 ip

如何设置不听远程的彩铃，按自己的设置放彩铃

ignore_early_media=true
ringback=a.wav

设置呼转的号码是多个且同时振铃，当有一个接听后，其它就不再振铃

<action application="bridge"
data="sofia/internal/1000@192.168.0.183,sofia/sip/1001@192.68.0.183"/>

设置呼转的号码是多个且顺序振铃，当有一个接听后，其它就不再振铃

<action application="bridge"
data="sofia/internal/1000@192.168.0.183|sofia/sip/1001@192.68.0.183"/>

某个路由必须走某种编码

absolute_codec_string=PCMA:PCMU

52

如何在外呼时，让其送出的号码不是’0000000’

在全局的 vars.xml 中有类似

```
1.  <X-PRE-PROCESS cmd="set" data="outbound_caller_name=fsgui"/>
2.  <X-PRE-PROCESS cmd="set" data="outbound_caller_id=18621575908"/>
```

或在每个用户的配置 xml 中

```
1.  <variable name="outbound_caller_id_name" value="$${outbound_caller_name}"/>

2.  <variable name="outbound_caller_id_number" value="$${outbound_caller_id}"/>
```

控制通话的音量

```
1.  <action application="set_audio_level" data="write 4"/>
2.  <action application="set_audio_level" data="read 4"/>
```

write 是用来设置对应 session 的所说的话的音量，那么 read 就是设置对应 session 听到的音量 level.

fs 转发客户端的自定义头

客户端添加自定义 SIP 头必须是 X-开头 freeswitch 才能帮转发 ，而且建立通话的时候 需要在 leg-b 上加 sip_copy_custom_headers=true 如：

```
<action application="bridge" data="{sip_copy_custom_headers=true}user/${dial
ed_extension}"/>
```

如何使用 postgresql 记录 freeswitch 话单

在 postgresql 某数据库中先建一个表

```
1.  CREATE TABLE cdr(
2.    ID SERIAL PRIMARY KEY,
```

```
  local_ip_v4 CHAR(16),
  caller_id_name CHAR(20),
  caller_id_number CHAR(20),
  outbound_caller_id_number CHAR(20),
  destination_number CHAR(20),
  context CHAR(20),
  start_stamp timestamp default NULL,
  answer_stamp timestamp default NULL,
  end_stamp timestamp default NULL,
  duration INT,
  billsec INT,
  hangup_cause CHAR(30),
  uuid CHAR(50),
  bleg_uuid CHAR(50),
  accountcode CHAR(20),
  read_codec CHAR(10),
  write_codec CHAR(10)
);
```

然后在 conf/autoload_configs/cdr_pg_csv.conf.xml 中配置相关的数据库信息和 sip 头

修改 sdp 中的 fs 名称

profile 文件中

```
<param name="username" value="nway"/>
```

如何做一个 fs 的级联

reeswitch 只需要在 A 上配置一个到 B 的网关（将下列内容存成 XML 文件放到 conf/sip_profiles/external/b.xml）：

```
<include>
        <gateway name="b">
                <param name="realm" value="192.168.1.B"/>
                <param name="username" value="1000"/>
                <param name="password" value="1234"/>
        </gateway>
</include>
```

然后在 B 中配置路由里处理对应的呼入，而在 A 中配置路由

```
<extension name="B">
  <condition field="destination_number" expression="^0(.*)$">
```

```
    <action application="bridge" data="sofia/gateway/b/$1"/>
  </condition>
</extension>
```

Fs 中如果放公网需要开放的端口（默认）

5060,5080 sip 端口
5066,7443 如果开通 websocket(webrtc)
Switch.conf.xml 中定义的 rtp 端口范围

由平台先呼 a 再呼 b 时，先放彩铃再听回铃再接通

```
originate {i_early_media=true}user/1020 set:instant_ringback=true,set:transf
er_ringback=/wav/a.wav,bridge:sofia/gateway/gw/18621575908 inline
```

平台外呼后放音再转座席

```
originate user/1001 playback:/li.wav,bridge:user/1003 inline
```

如何调整 jitterbuffer

Sofia profile 中

```
<param name="auto-jitterbuffer-msec" value="60"/>
```

Dialplan 中

```
<action application="set" data="jitterbuffer_msec=60:200:20"/>

<action application="answer"/>
```

FreeSwitch 网关轮询模块 mod_distributor

mod_distributor 采用加权轮询分配方式把呼叫分配给网关。可以通过 xml 文件来配置多个网关列表。

编辑 modules.conf 且添加以下行:

```
applications/mod_distributor
```


然后

```
make mod_distributor && make mod_distributor-install
```

配置 FreeSwitch 自动加载该模块

$FS_CONF/autoload_configs/modules.conf.xml 中添加一行

```
<load module="mod_distributor"/>
```

使用方法

有好几种方法可以使用这个模块，不过它通常被用于 bridge 这个 application,如,你有两个网关定义去分发,那么 dailstring 可以如此:

```
<action application="bridge"
data="sofia/gateway/${distributor(distributor_list)}/${destination_number}"/>

-- or--

<action application="bridge"
data="sofia/external/${destination_number}@${distributor(distributor_list)}"/>
```

在 fs_cli 中重新加载 distributor.conf.xml:

```
Reloadxml distributor_ctl reload
```

mod_distributor 通过配置文件中返回的节点值来工作，如:

```
<configuration name="distributor.conf" description="Distributor Configuration">
```

```
<lists>
  <list name="test" total-weight="10">
    <node name="foo1" weight="1"/>
    <node name="foo2" weight="9"/>
  </list>
</lists>
</configuration>
```


在名称为"test"的列表中,总权重是 10,而且所有节点的权重值合为 10,当一个分发呼叫到 test 时，它会第一时间返回 foo1,在后九次会返回 foo2.再然后下一次返回 foo1,9 次 foo2,一直循环下去。

如果是一个交替列表的可以如下方式:

```
<list name="alternating" total-weight="2">
    <node name="alt_one" weight="1"/>
    <node name="alt_two" weight="1"/>
</list>
```

在上述的例子中，每个呼叫到分发(alternating)将在 alt_one and alt_two 中交替。

你可以使用这个方式发送到多个网关实现简单的负载均衡。我们可以看看三个不同的网关定义成 gateway1,gateway2,gateway3.把每三个呼叫送给每一个网关，你需要在 distributor.conf.xml 中配置:

```
<list name="3gw" total-weight="3">
    <node name="gateway1" weight="1"/>
```

```
    <node name="gateway2" weight="1"/>
    <node name="gateway3" weight="1"/>
  </list>
```

57 定义路由如下：

```
<extension name="3-way gateway distro">
  <condition field="destination_number" expression="^(.*)$">
    <action application="bridge" data="sofia/gateway/${distributor(3gw)}/$1"/>
  </condition>
</extension>
```

这个分发模块支持按网关名称去使用，每个呼叫到达路由时将触发分发模块变换使用不同的 gw 去响应呼叫。

当你的拨号路由在上一个 bridge 时失败，你想继续呼叫且在在 xml 路由中循环：

```
<extension name="3-way gateway distributor">
  <condition field="destination_number" expression="^(.*)$">
    <action application="set" data="continue_on_fail=true"/>
    <action application="set" data="hangup_after_bridge=true"/>
    <action application="bridge" data="sofia/gateway/${distributor(3gw)}/$1" loop="3"/>
  </condition>
</extension>
```

如果有不可达的网关，应在路由中排除不可达网关

```
<action application="bridge" data="sofia/gateway/${distributor(<listname> ${sofia(profile
<profilename> gwlist down)})}/$1"/>

--or--
```

注意：括号是可选变量：${foo(bar)} 等同于 ${foo bar}

```
<action application="bridge" data="sofia/gateway/${distributor <listname> ${sofia profile
<profilename> gwlist down}}/$1"/>

--or--

<action application="bridge" data="sofia/gateway/${expand(distributor <listname> \${sofia(profile
<profilename> gwlist down)})}/$1"/>
```

API Command

如何执行以上命令在命令行中呢:

```
expand distributor <listname> ${sofia profile <profilename> gwlist down}
```

遇到本机 8021fs_cli 连 fs 不上

一般是预计是因为解析 localhost 解析不了，在 event_socket.conf.xml 中将 ip 部分改为 127.0.0.1 即可

使用 webrtc 时没声音或提示 Remote Address Error!

Sip profile 中添加

```
1.  <param name="apply-candidate-acl" value="121.40.40.40"/>
2.  <param name="apply-candidate-acl" value="rfc1918.auto"/>
3.  <param name="apply-candidate-acl" value="localnet.auto"/>
```

遇到总是提示 domain 被 acl 拒绝

Sip profile

```
<param name="apply-candidate-acl" value="ipv4"/>
```

Acl.conf.xml

```
4.  <list name="ipv4" default="deny">
5.      <node type="allow" cidr="0.0.0.0/0"/>
6.  </list>
```

59

刚安装好，使用时总是延时十秒才呼叫

因为使用了默认的密码 1234,一是改密码，二是处理 dialplan/default.xml 中的以下

```
<condition field="${default_password}" expression="^1234$" break="never">

     <action application="log" data="CRIT WARNING WARNING WARNING WARNING
WARNING WARNING WARNING WARNING WARNING "/>

     <action application="log" data="CRIT Open $${conf_dir}/vars.xml and
change the default_password."/>

     <action application="log" data="CRIT Once changed type 'reloadxml' at
the console."/>

      <action application="log" data="CRIT WARNING WARNING WARNING WARNING
WARNING WARNING WARNING WARNING WARNING "/>

      <action application="sleep" data="10000"/>

</condition>
```

修改默认密码

Conf/vars.xml 中

```
      <X-PRE-PROCESS cmd="set" data="default_password=1234"/>
```

Webrtc 中 candidate 多个 ip 地址

```
      originate {media_webrtc=true}user/1000 &echo
```

fs 在内网，但要处理公网上的请求

sip protfile 中

```
ext-rtp-ip = “47.104.124.127”

ext-sip-ip = “47.104.124.127”
```


关闭 rtp 自动调整

```
{disable_rtp_auto_adjust=true}
```

修改默认的 sip 端口

在 vars.xml 中

```
<X-PRE-PROCESS cmd="set" data="internal_sip_port=15062"/>
<X-PRE-PROCESS cmd="set" data="internal_tls_port=15061"/>
<X-PRE-PROCESS cmd="set" data="external_sip_port=15080"/>
<X-PRE-PROCESS cmd="set" data="external_tls_port=15081"/>
```

切记，公网一定要改默认端口

ULIMIT 配置

```
ulimit -c unlimited # The maximum size of core files created.
ulimit -d unlimited # The maximum size of a process's data segment.
ulimit -
f unlimited # The maximum size of files created by the shell (default option
)
ulimit -i unlimited # The maximum number of pending signals
ulimit -n 999999    # The maximum number of open file descriptors.
ulimit -q unlimited # The maximum POSIX message queue size
ulimit -
u unlimited # The maximum number of processes available to a single user.
ulimit -
v unlimited # The maximum amount of virtual memory available to the process.

ulimit -x unlimited # ???
ulimit -s 240         # The maximum stack size
```

```
11. ulimit -l unlimited # The maximum size that may be locked into memory.
12. ulimit -a           # All current limits are reported.
```

在哪里去检查语音通话的质量

事件 channel_hangup_complete 中有相关信息

如何查看已注册的相关分机

```
show registrations

sofia status profile internal reg
```

在 dialplan xml 中检查文件是否存在

```
<extension name="play-news-announcements">

  <condition expression="${file_exists(${sounds_dir}/news.wav)}" expression="tru
e"/>

    <action application="playback" data="${sounds_dir}/news.wav"/>

    <anti-action application="playback" data="${soufnds_dir}/no-news-is-good-
news.wav"/>

  </condition>

</extension>
```

如何调整 fs_cli 中日志显示的级别

```
console loglevel 4

0 - CONSOLE

1 - ALERT

2 - CRIT

3 - ERR
```

```
4 - WARNING

5 - NOTICE

6 - INFO

7 - DEBUG
```


呼叫保持和恢复

```
uuid_hold off <uuid>



uuid_hold toggle uuid
```

expand 的使用

```
expand originate sofia/internal/1001%${domain} 9999
```

会将${domain}这个变量在具体使用时替换成真实的参数

limit_execute 的使用

```
<extension name="outbound">

  <condition field="destination_number" expression="^1?[2-9]\d{2}[2-9]\d{6}$">

    <action application="limit_execute" data="hash outbound carrier1 5 bridge
sofia/gateway/carrier1/${destination_number}" />

    <action application="limit_execute" data="hash outbound carrier2 5 bridge
sofia/gateway/carrier2/${destination_number}" />

  </condition>

</extension>
```

两个 gateway 各最大 5 线，则通过这种方式，来分配线路

控制呼叫频率

10 分钟这个 ip 呼叫 5 次

```
<action  application="limit"  data="hash ${sip_received_ip} ${destination_number} 5/600"  />
```

控制呼出总数


```
<action application="limit" data="db outgoing gw1 10" />

<action application="bridge" data="sofia/gateway/gw1/${destnum}" />
```

重新加载 external 配的网关

```
sofia profile external rescan reloadxml

sofia profile external restart reloadxml
```

呼叫保持和恢复

```
uuid_hold off <uuid>

uuid_hold toggle uuid
```

让通话接通后放音

```
session:setVariable("execute_on_answer","sched_broadcast +1 /usr/local/freeswitch/sounds/lihao.wav bleg");
```

如何让 fs 回复一个值，如 486

```
session:execute("respond","486")
```

放在内网的 goip 注册到公网中的 fs 如何呼叫

```
Originate {sip_invite_req_uri=sip:18621575908@default}user/1000 &echo
```

如何判断是由先挂机

```
sip_hangup_disposition

recv_bye ， send_bye
```

如何快速查看 fs 使用中的通道变量

在源码中有 python esl 示例，如

```
cd /opt/freeswitch/libs/esl/

make pymod

cd python

python events.py > lihao.log
```

这样在 lihao.log 中就可以查看各时间点的通话，或者自己再加些时间和私有标记

Freeswitch 通道变量

Info variable name	channel variable name	Description
Channel-State	state	Current state of the call
Channel-State-Number	state_number	Integer
Channel-Name	channel_name	Channel name
Unique-ID	uuid	uuid of this channel's call leg
Call-Direction	direction	Inbound or Outbound
Answer-State	state	.
Channel-Read-Codec-Name	read_codec	the read codec variable mean the source codec
Channel-Read-Codec-Rate	read_rate	the source rate
Channel-Write-Codec-Name	write_codec	the destination codec same to write_codec if not transcoded

Channel-Write-Codec-Rate	write_rate	destination rate same to read rate if not transcoded
Caller-Username	username	.
Caller-Dialplan	dialplan	user dialplan like xml, lua, enum, lcr
Caller-Caller-ID-Name	caller_id_name	.
Caller-Caller-ID-Number	caller_id_number	.
Caller-ANI	ani	ANI of caller, frequently the same as caller ID number
Caller-ANI-II	aniii	
Caller-Network-Addr	network_addr	IP address of calling party
Caller-Destination-Number	destination_number	Destination (dialed) number
Caller-Unique-ID	uuid	This channel's uuid
Caller-Source	source	Source module, i.e. mod_sofia, mod_openzap, etc.
Caller-Context	context	Dialplan context
Caller-RDNIS	rdnis	Redirected DNIS info. See transfer application
Caller-Channel-Name	channel_name	.
Caller-Profile-Index	profile_index	.

Caller-Channel-Created-Time	created_time	.
Caller-Channel-Answered-Time	answered_time	.
Caller-Channel-Hangup-Time	hangup_time	.
Caller-Channel-Transfer-Time	transfer_time	.
Caller-Screen-Bit	screen_bit	.
Caller-Privacy-Hide-Name	privacy_hide_name	.
Caller-Privacy-Hide-Number	privacy_hide_number	
	initial_callee_id_name	
variable_sip_received_ip	sip_received_ip	.
variable_sip_received_port	sip_received_port	.
variable_sip_authorized	sip_authorized	.
variable_sip_mailbox	sip_mailbox	.
variable_sip_auth_username	sip_auth_username	.

variable_sip_auth_realm	sip_auth_realm	.
variable_mailbox	mailbox	.
variable_user_name	user_name	.
variable_domain_name	domain_name	.
variable_record_stereo	record_stereo	.
variable_accountcode	accountcode	.
variable_user_context	user_context	.
variable_effective_caller_id_name	effective_caller_id_name	.
variable_effective_caller_id_number	effective_caller_id_number	.
variable_caller_domain	caller_domain	.
variable_sip_from_user	sip_from_user	.
variable_sip_from_uri	sip_from_uri	.
variable_sip_from_host	sip_from_host	.
variable_sip_from_user_stripped	sip_from_user_stripped	.
variable_sip_from_tag	sip_from_tag	.
variable_sofia_profile_name	sofia_profile_name	.
variable_sofia_profile_do	sofia_profile_domain_n	.

main_name	ame	
variable_sip_full_route	sip_full_route	The complete contents of the Route: header.
variable_sip_full_via	sip_full_via	The complete contents of the Via: header.
variable_sip_full_from	sip_full_from	The complete contents of the From: header.
variable_sip_full_to	sip_full_to	The complete contents of the To: header.
variable_sip_req_params	sip_req_params	.
variable_sip_req_user	sip_req_user	.
variable_sip_req_uri	sip_req_uri	.
variable_sip_req_host	sip_req_host	.
variable_sip_to_params	sip_to_params	.
variable_sip_to_tag	sip_to_tag	.
variable_sip_to_user	sip_to_user	.
variable_sip_to_uri	sip_to_uri	.
variable_sip_to_host	sip_to_host	.
variable_sip_contact_params	sip_contact_params	.
variable_sip_contact_use	sip_contact_user	.

r		
variable_sip_contact_port	sip_contact_port	.
variable_sip_contact_uri	sip_contact_uri	.
variable_sip_contact_host	sip_contact_host	.
variable_sip_invite_domain	sip_invite_domain	.
variable_channel_name	channel_name	.
variable_sip_call_id	sip_call_id	.
variable_sip_user_agent	sip_user_agent	.
variable_sip_via_host	sip_via_host	.
variable_sip_via_port	sip_via_port	.
variable_sip_via_rport	sip_via_rport	.
variable_presence_id	presence_id	.
variable_sip_h_P-Key-Flags	sip_h_p-key-flags	This will contain the optional P-Key-Flags header(s) that may be received from calling endpoint.
variable_switch_r_sdp	switch_r_sdp	The whole SDP received from calling endpoint.
variable_remote_media_i	remote_media_ip	.

p		
variable_remote_media_port	remote_media_port	.
variable_write_codec	write_codec	.
variable_write_rate	write_rate	.
variable_endpoint_disposition	endpoint_disposition	.
variable_dialed_ext	dialed_ext	.
variable_transfer_ringback	transfer_ringback	.
variable_call_timeout	call_timeout	.
variable_hangup_after_bridge	hangup_after_bridge	.
variable_continue_on_fail	continue_on_fail	.
variable_dialed_user	dialed_user	.
variable_dialed_domain	dialed_domain	.
variable_sip_redirect_contact_user_0	sip_redirect_contact_user_0	.
variable_sip_redirect_contact_host_0	sip_redirect_contact_host_0	.
variable_sip_h_Referred-	sip_h_referred-by	.

By		
variable_sip_refer_to	sip_refer_to	The full SIP URI received from a SIP Refer-To: response
variable_max_forwards	max_forwards	.
variable_originate_disposition	originate_disposition	.
variable_read_codec	read_codec	.
variable_read_rate	read_rate	.
variable_open	open	.
variable_use_profile	use_profile	.
variable_current_application	current_application	.
variable_ep_codec_string	ep_codec_string	This variable is only available if late negotiation is enabled on the profile. It's a readable string containing all the codecs proposed by the calling endpoint. This can be easily parsed in the dialplan.
variable_rtp_disable_hold	rtp_disable_hold	This variable when set will disable the hold feature of the phone.

variable_sip_acl_authed_by	sip_acl_authed_by	This variable holds what ACL rule allowed the call.
variable_curl_response_data	curl_response_data	This variable stores the output from the last curl made.
variable_drop_dtmf	drop_dtmf	Set on a channel to drop DTMF events on the way out.
variable_drop_dtmf_masking_file	drop_dtmf_masking_file	If drop_dtmf is true play specified file for every tone received.
variable_drop_dtmf_masking_digits	drop_dtmf_masking_digits	If drop_dtmf is true play specified tone for every tone received.
sip_codec_negotiation	sip_codec_negotiation	sip_codec_negotiation is basically a channel variable equivalent of inbound-codec-negotiation. sip_codec_negotiation accepts "scrooge" & "greedy" as values. This means you can change codec negotiation on a per call basis.

Caller-Callee-ID-Name	-	-
Caller-Callee-ID-Number	-	-
Caller-Channel-Progress-Media-Time	-	-
Caller-Channel-Progress-Time	-	-
Caller-Direction	-	-
Caller-Profile-Created-Time	profile_created	-
Caller-Transfer-Source	-	-
Channel-Call-State	-	-
Channel-Call-UUID	-	-
Channel-HIT-Dialplan	-	-
Channel-Read-Codec-Bit-Rate	-	-
Channel-Write-Codec-Bit-Rate	-	-
Core-UUID	-	-
Event-Calling-File	-	-
Event-Calling-Function	-	-
Event-Calling-Line-	-	-

Number		
Event-Date-GMT	-	-
Event-Date-Local	-	-
Event-Date-Timestamp	-	-
Event-Name	-	-
Event-Sequence	-	-
FreeSWITCH-Hostname	-	-
FreeSWITCH-IPv4	-	-
FreeSWITCH-IPv6	-	-
FreeSWITCH-Switchname	-	-
Hunt-ANI	-	-
Hunt-Callee-ID-Name	-	-
Hunt-Callee-ID-Number	-	-
Hunt-Caller-ID-Name	-	-
Hunt-Caller-ID-Number	-	-
Hunt-Channel-Answered-Time	-	-
Hunt-Channel-Created-Time	-	-
Hunt-Channel-Hangup-	-	-

Time		
Hunt-Channel-Name	-	-
Hunt-Channel-Progress-Media-Time	-	-
Hunt-Channel-Progress-Time	-	-
Hunt-Channel-Transfer-Time	-	-
Hunt-Context	-	-
Hunt-Destination-Number	-	-
Hunt-Dialplan	-	-
Hunt-Direction	-	-
Hunt-Network-Addr	-	-
Hunt-Privacy-Hide-Name	-	-
Hunt-Privacy-Hide-Number	-	-
Hunt-Profile-Created-Time	profile_created	-
Hunt-Profile-Index	-	-

Hunt-RDNIS	-	-
Hunt-Screen-Bit	-	-
Hunt-Source	-	-
Hunt-Transfer-Source	-	-
Hunt-Unique-ID	-	-
Hunt-Username	-	-
Presence-Call-Direction	-	-
variable_DIALSTATUS	-	-
variable_absolute_codec_string	-	-
variable_advertised_media_ip	-	-
variable_bridge_channel	-	-
variable_bridge_hangup_cause	-	-
variable_bridge_uuid	-	-
variable_call_uuid	-	-
variable_current_application_response	-	-
variable_direction	-	-
variable_inherit_codec	-	-

variable_is_outbound	-	-
variable_last_bridge_to	-	-
variable_last_sent_callee_id_name	-	-
variable_last_sent_callee_id_number	-	-
variable_local_media_ip	-	-
variable_local_media_port	-	-
variable_number_alias	-	-
variable_originate_early_media	-	-
variable_originating_leg_uuid	-	-
variable_originator	-	-
variable_originator_codec	-	-
variable_outbound_caller_id_number	-	-
variable_recovery_profile_name	-	-

variable_rtp_use_ssrc	-	-
variable_session_id	-	-
variable_sip_2833_recv_payload	-	-
variable_sip_2833_send_payload	-	-
variable_sip_P-Asserted-Identity	-	-
variable_sip_Privacy	-	-
variable_sip_audio_recv_pt	-	-
variable_sip_cid_type	-	-
variable_sip_cseq	-	-
variable_sip_destination_url	-	-
variable_sip_from_display	sip_from_display	'User' element of SIP From: line
variable_sip_from_port	-	-
variable_sip_gateway	-	-
variable_sip_gateway_name	-	-

variable_sip_h_P-Charging-Vector	-	-
variable_sip_local_network_addr	-	-
variable_sip_local_sdp_str	-	-
variable_sip_network_ip	-	-
variable_sip_network_port	-	-
variable_sip_number_alias	-	-
variable_sip_outgoing_contact_uri	-	-
variable_sip_ph_P-Charging-Vector	-	-
variable_sip_profile_name	-	-
variable_sip_recover_contact	-	-
variable_sip_recover_via	-	-
variable_sip_reply_host	-	-

variable_sip_reply_port	-	-
variable_sip_req_port	-	-
variable_sip_to_port	-	-
variable_sip_use_codec_name	-	-
variable_sip_use_codec_ptime	-	-
variable_sip_use_codec_rate	-	-
variable_sip_use_pt	-	-
variable_sip_via_protocol	-	-
variable_switch_m_sdp	-	-
variable_transfer_history	-	-
variable_transfer_source	-	-
variable_uuid	-	-

选择 G711 还是 G729？

G711 的特点是在 FS 中即使有转换也性能高，但带宽占用大，同时也是以前运营商数字中继的主要编码，G729 是占用带宽小，但运算量相对大，而且会存在一定的音频损失，一般是没太多关系的，但最终用于智能客服、智能外呼等场景建议尽量用 G711.

添加 sip 头，用于非标的一些 sip server

> 添加

```
<action application="set" data="sip_h_X-myphone=18621575908"/>

去除

<action application="unset" data="sip_h_X- myphone"/>
```

81

强行注销一个 sip 分机或重启

```
sofia profile <profile_name> flush_inbound_reg [<call_id>|<user@host>] [reboot]

sofia profile internal flush_inbound_reg 18621575908@10.0.0.21
```

让 fs 内核使用 postgresql 数据库

以下文件
db.conf.xml
fifo.conf.xml
voicemail.conf.xml
switch.conf.xml
internal.xml
external.xml

```
<param name="odbc-dsn" value="pgsql://hostaddr=127.0.0.1 dbname=freeswitch user=postgres
password='123456' options='-c client_min_messages=NOTICE' application_name='freeswitch'"/>
```

录音最短时间

```
session:execute("set", "RECORD_MIN_SEC=10");
```

当 b 路挂机后继续走路由

```
<action application="set" data="hangup_after_bridge=true"/>

<action application="set"
data="continue_on_fail=USER_NOT_REGISTERED,NORMAL_TEMPORARY_FAILURE,USER_BUSY,NO_ANSWE
R,TIMEOUT,NO_ROUTE_DESTINATION"/>
```

freeswitch 将 sip 日志写入文件

```
<param name="sip-trace" value="no"/> 改为：
```

```
<param name="sip-trace" value="true"/>
```

sofia.conf.xml 里面：
<global_settings> 下面增加：

```
<param name="tracelevel" value="alert"/>
```


如何设置 P-Asserted-Identity

```
<action application=”export” data=”sip_h_ P-Asserted-Identity=+86186215759O8@1.1.1.1” />
```

让 freeswitch 通话进行变声

```
application/mod_soundtouch
```

限制 5080 送入需要认证才能呼叫

```
<X-PRE-PROCESS cmd="set" data="external_auth_calls=true"/>
```

让客户端定时发送注册包

```
<param name="sip-force-expires" value="1800"/>
```

让 fs 转发 info

```
<param name="liberal-dtmf" value="true"/>

<param name="dtmf-type" value="info"/>
```

fs1.6.7 以后默认不转码处理

```
<X-PRE-PROCESS cmd="set" data="media_mix_inbound_outbound_codecs=true"/>
```

或者拨号计划

```
<action application="export" data="media_mix_inbound_outbound_codecs=true"/>

<action application="bridge" data="user/$1@${domain_name}"/>
```

调试 xml_curl

```
xml_curl debug_on
```

用于控制 originate 的一些参数

originate_retries 重拨次数
originate_retry_sleep_ms 重拨间隔
originate_continue_on_timeout 超时继续执行 originate_retries
fail_on_single_reject=失败原因 或者 !失败原因 出现失败原因 继续执行 originate_retries
hangup_on_single_reject=true 挂断主叫
group_confirm_key=#,group_confirm_file=b.wav

示倒，用 pocketsphinx 实现的说省会城市就放音

```
1.  -- Used in parse_xml
2.  function parseargs_xml(s)
3.     local arg = {}
4.     string.gsub(s, "(%w+)=([\"']?)(.-)%2", function (w, _, a)
5.                           arg[w] = a
6.                       end)
7.     return arg
8.  end
9.
10. -- Log
11. function log(text)
12.     freeswitch.consoleLog("INFO","\n log..................... " .. text)
13. end
14.
15. function parseargs_xml(s)
16.    local arg = {}
17.    string.gsub(s, "(%w+)=([\"']?)(.-)%2", function (w, _, a)
18.                           arg[w] = a
19.                       end)
20.    return arg
21. end
22. tab = "    "
23. function print_table(t, i)
24.     local indent ="" -- i 缩进，当前调用缩进
25.     for j = 0, i do
```

```
26.         indent = indent .. tab
27.     end
28.     for k, v in pairs(t) do
29.         if (type(v) == "table") then -- type(v) 当前类型时否 table 如果是，则需要递归，
30.             log(indent .. "< " .. k .. " is a table />")
31.             print_table(v, i + 1) -- 递归调用
32.             log(indent .. "/> end table ".. k .. "/>")
33.         else -- 否则直接输出当前值
34.
35.             log(indent .. "<" .. k .. "=" .. v.."/>")
36.         end
37.     end
38. end
39. function parse_xml(s)
40.    local stack = {};
41.    local top = {};
42.    table.insert(stack, top);
43.    local ni,c,label,xarg, empty;
44.    local i, j = 1, 1;
45.    while true do
46.       ni,j,c,label,xarg, empty = string.find(s, "<(%/?)(%w+)(.-)(%/?)>", i);
47.       if not ni then
48.             break
49.       end
50.       local text = string.sub(s, i, ni-1);
51.       if not string.find(text, "^%s*$") then
52.             -- log("\n text .. " .. text )
53.             --return text
54.             table.insert(top, text);
55.       end
56.       if empty == "/" then
57.             --log("\n top .. " .. parseargs_xml(xarg))
58.             table.insert(top, {label=label, xarg=parseargs_xml(xarg), empty=1});
59.       elseif c == "" then
60.             top = {label=label, xarg=parseargs_xml(xarg)};
61.             --log("\n xarg .. " .. xarg)
62.             table.insert(stack, top);
63.       else
64.         local toclose = table.remove(stack);
65.         top = stack[#stack];
66.         if #stack < 1 then
```

```
67.             error("nothing to close with "..label);
68.         end
69.         if toclose.label ~= label then
70.             error("trying to close "..toclose.label.." with "..label);
71.         end
72.         table.insert(top, toclose);
73.       end
74.       --log("\n i = " .. tostring(i))
75.       i = j+1;
76.    end
77.    local text = string.sub(s, i);
78.    if not string.find(text, "^%s*$") then
79.       --log( "FOUND STRING .... " .. text)
80.       table.insert(stack[stack.n], text);
81.    end
82.    if #stack > 1 then
83.       error("unclosed "..stack[stack.n].label);
84.    end
85.    --print_table(stack[1][2],0)
86.    return stack[1];
87. end
88.
89. function getResults(s)
90.    local xml = parse_xml(s);
91.    --log(tostring(xml[1]))
92.     log(tostring(xml))
93.    local stack = {}
94.    local top = {}
95.    table.insert(stack, top)
96.    --log("\n xml 2..... " .. xml[2]:serialize())
97.    top = {grammar=xml[2].xarg.grammar, score=xml[2].xarg.score, text=xml[2][
   1][1]}
98.    table.insert(stack, top)
99.    return top;
100.    --return xml
101. end
102.
103. -
   - This is the input callback used by dtmf or any other events on this sessio
   n such as ASR.
104. function onInput(s, type, obj)
105.    --
   freeswitch.consoleLog("info", "Callback with type " .. type .. "\n");
106.    if (type == "dtmf") then
```

```
107.       freeswitch.consoleLog("INFO", "DTMF Digit: " .. obj.digit .. "\n");
108.    else if (type == "event") then
109.      local event = obj:getHeader("Speech-Type");
110.      if (event == "begin-speaking") then
111.         freeswitch.consoleLog("INFO", "\n begin-
    speaking \n" .. obj:serialize() .. "\n");
112.         -- Return break on begin-
    speaking events to stop playback of the fire or tts.
113.         return "break";
114.      end
115.      --DETECTED_SPEECH     detected-speech
116.      --
    freeswitch.consoleLog("INFO", "\nNo Body: " .. obj:serialize() .. "\n");
117.
118.      if (event == "detected-speech") then
119.         --
    freeswitch.consoleLog("INFO", "\nNo Body: " .. obj:serialize() .. "\n");
120.         --log("\n" .. obj:serialize())
121.         if (obj:getBody()) then
122.            --log("Text body:" .. obj:getBody())
123.            freeswitch.consoleLog("INFO","\n Text body:" .. obj:getBody())
124.            -
    - Pause speech detection (this is on auto but pausing it just in case)
125.            session:execute("detect_speech", "pause");
126.            --local speech_output = obj:getBody()
127.            results = getResults(obj:getBody())
128.            --if (results ~= nil) then
129.            if (results.grammar ~= nil) then
130.                --log("Heard: confidence = " .. results.score .. "\n")
131.                --log("Heard :" .. results.text)
132.                local city = results.text[1]
133.                freeswitch.consoleLog("INFO","\n Heard:" .. city)
134.
135.             else
136.                 --log("Can not heard")
137.                 freeswitch.consoleLog("INFO","\n Can not heard!"  )
138.             end
139.            -
    - Parse the results from the event into the results table for later use.
140.            --results = getResults(obj:getBody());
141.         end
142.         return "break";
143.      end
144.       end
```

```
145.    end
146. end
147.
148.
149. --Used to map returned names to extension numbers
150. extensions = {
151.    ["MATTHEW_WILLIAMS"] = 1000,
152.    ["MATT_WILLIAMS"] = 1000,
153.    ["JOSEPH_SMITH"] = 1001,
154.    ["JOE_SMITH"] = 1000
155.    }
156.
157. -- Create the empty results table.
158. results = {};
159.
160. -- Answer the call.
161. session:answer();
162. freeswitch.consoleLog("info", "Into ASR... \n");
163. prefix = "/usr/local/freeswitch/sounds/wav/"
164. sound = prefix .. "welcome.wav"
165. session:execute("playback",sound)
166. -- Register the input callback
167. session:setInputCallback("onInput");
168. -- Sleep a little bit to give media time to be fully up.
169. session:sleep(200);
170. -- Start the detect_speech app.  This attaches the bug to fire events
171. session:execute("detect_speech", "pocketsphinx zh nway");
172.
173. -- Magic happens here.
174. -
    - It would be ok to loop like 3 times and error to the operator if this does
    n't work or revert to reading names off with TTS.
175. while (session:ready() == true) do
176.    session:sleep(100);
177.    -- Who are they looking for?
178.    -- This sleep is what blocks till the detected-
    speech event.  This has to give you enough time to speak plus get the result
    s.
179.    session:sleep(3000);
180.    session:sleep(3000);
181.    -- If the results aren't null and we have an extension in the table.
182.    if (results.text ~= nil) then
183.       -- Letting the caller know we are trying.
```

```
184.        -
    - It's critical to stop the detect_detect otherwise it will continue to fire
     speech events and waste resources.
185.        -- session:execute("detect_speech", "stop");
186.        -- Transfer the call to the extension out of the lua table.
187.        -
    - session:execute("transfer", extensions[results.text] .. " XML home");
188.        local city = results.text[1]
189.        if city == "北京" then
190.           sound = prefix .. "beijing.wav"
191.           session:execute("playback",sound)
192.        end
193.        if city == "上海" then
194.           sound = prefix .. "shanghai.wav"
195.           session:execute("playback",sound)
196.        end
197.        if city == "哈尔滨" then
198.           sound = prefix .. "haabin.wav"
199.           session:execute("playback",sound)
200.        end
201.        if city == "沈阳" then
202.           sound = prefix .. "shenyang.wav"
203.           session:execute("playback",sound)
204.        end
205.        if city == "长春" then
206.           sound = prefix .. "changchun.wav"
207.           session:execute("playback",sound)
208.        end
209.        if city == "石家庄" then
210.           sound = prefix .. "shijiazhuang.wav"
211.           session:execute("playback",sound)
212.        end
213.         if city == "济南" then
214.           sound = prefix .. "jinan.wav"
215.           session:execute("playback",sound)
216.        end
217.        if city == "郑州" then
218.           sound = prefix .. "zhengzhou.wav"
219.           session:execute("playback",sound)
220.        end
221.        if city == "南京" then
222.           sound = prefix .. "nanjing.wav"
223.           session:execute("playback",sound)
224.        end
```

```
225.        if city == "杭州" then
226.           sound = prefix .. "hangzhou.wav"
227.           session:execute("playback",sound)
228.        end
229.        if city == "福州" then
230.           sound = prefix .. "fuzhou.wav"
231.           session:execute("playback",sound)
232.        end
233.        if city == "武汉" then
234.           sound = prefix .. "wuhan.wav"
235.           session:execute("playback",sound)
236.        end
237.        if city == "长沙" then
238.           sound = prefix .. "changsha.wav"
239.           session:execute("playback",sound)
240.        end
241.        if city == "南昌" then
242.           sound = prefix .. "nanchang.wav"
243.           session:execute("playback",sound)
244.        end
245.        if city == "广州" then
246.           sound = prefix .. "guangzhou.wav"
247.           session:execute("playback",sound)
248.        end
249.        if city == "南宁" then
250.           sound = prefix .. "nanning.wav"
251.           session:execute("playback",sound)
252.        end
253.        if city == "贵阳" then
254.           sound = prefix .. "guiyang.wav"
255.           session:execute("playback",sound)
256.        end
257.        if city == "昆明" then
258.           sound = prefix .. "kuiming.wav"
259.           session:execute("playback",sound)
260.        end
261.        if city == "成都" then
262.           sound = prefix .. "chengdu.wav"
263.           session:execute("playback",sound)
264.        end
265.        if city == "西安" then
266.           sound = prefix .. "xian.wav"
267.           session:execute("playback",sound)
268.        end
```

```
269.        if city == "太原" then
270.           sound = prefix .. "taiyuan.wav"
271.           session:execute("playback",sound)
272.        end
273.        if city == "合肥" then
274.           sound = prefix .. "hefei.wav"
275.           session:execute("playback",sound)
276.        end
277.        if city == "兰州" then
278.           sound = prefix .. "lanzhou.wav"
279.           session:execute("playback",sound)
280.        end
281.        if city == "西宁" then
282.           sound = prefix .. "xining.wav"
283.           session:execute("playback",sound)
284.        end
285.        if city == "呼和浩特" then
286.           sound = prefix .. "huhehaote.wav"
287.           session:execute("playback",sound)
288.        end
289.        if city == "乌鲁木齐" then
290.           sound = prefix .. "wulumuqi.wav"
291.           session:execute("playback",sound)
292.        end
293.        if city == "拉萨" then
294.           sound = prefix .. "lasa.wav"
295.           session:execute("playback",sound)
296.        end
297.        if city == "银川" then
298.           sound = prefix .. "yinchuan.wav"
299.           session:execute("playback",sound)
300.        end
301.        if city == "重庆" then
302.           sound = prefix .. "chongqing.wav"
303.           session:execute("playback",sound)
304.        end
305.        if city == "天津" then
306.           sound = prefix .. "tianjin.wav"
307.           session:execute("playback",sound)
308.        end
309.        if city == "海口" then
310.           sound = prefix .. "haikou.wav"
311.           session:execute("playback",sound)
312.        end
```

```
313.       if city == "春节快乐" or city == "春节愉快" then
314.          sound = prefix .. "chunjie.wav"
315.          session:execute("playback",sound)
316.       end
317.
318.    end
319.    -- We didn't have them in our directory table.
320.    -- Clear any results we have just in case.
321.    results = {};
322.    -- Resume detect_speech.
323.    sound = prefix .. "repeat.wav"
324.    session:execute("playback",sound)
325.    session:execute("detect_speech", "resume");
326. end
```

Auto Changing audio port 是什么设置导致的？

```
disable-rtp-auto-adjus
```

有的移动 ims 没有彩铃

Sip profile 中

```
<param name="enable-100rel" value="true"/>
```

这种方法只是试一试的思路，不代表百分百肯定解决

无法二次拨号，DTMF 不能用

```
<action application="export" data="nolocal:execute_on_answer=start_dtmf"/>
```

这种方法只是试一试的思路，不代表百分百肯定解决

接通后报工号

```
uuid_broadcast 1332f73c-389f-11e9-b67d-ebae59d3667d
file_string:///home/voices/default/job_number_prefix.wav!/home/voices/defaul
t/1.wav!/home/voices/default/0.wav!/home/voices/default/1.wav!/home/voices/d
efault/job_number_suffix.wav both
```

fs 的 invite 中的几个头参数

sip_invite_req_uri　　用来修改 request line 中的部分

sip_invite_to_uri　　用来修改 to

sip_invite_from_uri　　用来修改 from

FreeSwitch 模块 mod_unimrcp 配置数据库化初探

在我们使用 FreeSwitch 对接 mrcp 时，最常用的方式就是 mod_unimrcp 模块，但 mod_unimrcp 模块需要对 conf/mrcp_profiles/这里进行对应的 mrcp 配置，在这里一定要记住，这里的 unimrcp 是作为 client 去和其它第三方的 mrcp server 对接，由 mrcp server 和 asr/tts 打交道。

不管是 asr 还是 tts，对于 unimrcp 的配置都差不多，只是在一些参数调整上会存在一定的差别。以下示例为一个使用科大讯飞的 mrcp v2 的 unimrcp 在 FreeSwitch 中的配置示例：

```
1. <include>
2.   <profile name="xfmrcpv2asr" version="2">
3.     <param name="client-ip" value="172.100.0.217"/>
4.     <param name="client-port" value="8101"/>
5.     <param name="server-ip" value="172.100.0.200"/>
6.     <param name="server-port" value="5070"/>
7.     <param name="sip-transport" value="udp"/>
8.     <param name="rtp-ip" value="172.100.0.200"/>
9.     <param name="rtp-port-min" value="30000"/>
```

```
10.     <param name="rtp-port-max" value="58000"/>
11.     <param name="rtcp" value="1"/>
12.     <param name="rtcp-bye" value="2"/>
13.     <param name="rtcp-tx-interval" value="5000"/>
14.     <param name="rtcp-rx-resolution" value="1000"/>
15.     <param name="codecs" value="PCMU PCMA L16/96/8000"/>
16.     <synthparams>
17.     </synthparams>
18.     <recogparams>
19.     </recogparams>
20.   </profile>
21.</include>
```

但这样存在一个问题，每增加一个服务器对接，都需要去在 conf/mrcp_profiles 下做类似的配置，那么可不可以直接配置到数据库中呢？肯定可以啦。

问题在 mod_unimrcp.c 这个实现中去对以下几个函数进行数据库化的处理：

process_profile_config

process_rtp_config

process_mrcpv1_config ，只有 version 为 1 才调用

process_mrcpv2_config ，只有 version 为 2 才调用

mod_unimrcp_do_config

把这些函数按我们数据库结构进行查询，然后一一对应匹配，那么就把这些内容给数据库化了。