百问 FreeSwitch

余洪涌 2013-03-08 修订

qq:109559405

tel:13606060253

目录

致谢：

感谢 CCAV，感谢 GOOGLE，感谢 FreeSwitch 群，感谢 freeswitch.org，感谢 freeswitch.org.cn
感谢 ~！ @#￥%……&*（）—+

声明：

首先需要声明，本文档不是严谨的学术书籍，
是针对实际研发和部署使用过程中的一些问题的总结，很多的问题答案只是一种解决办法，不一定是最佳答案。由于环境的不同，问题的答案甚至在各位读者的测试环境中都无法重现。甚至是错误的答案。:）
因此，本书尽量描述清楚问题的出现环境。
本书的测试环境是 基于 FreeSwitch 的 windows 版的 1.2.1 的源码上编译的。

VOIP 基础部分

0. 本文档读者对象是哪些？

毫无疑问 FreeSwitch 百问的读者肯定是技术人员，他们可能是：

A. 对 voip 有兴趣没有基础的伙计
B. 对 FreeSwitch 有兴趣站在门口的伙计
C. 计划把 FreeSwitch 从实验所或者研发中心部署应用到实网系统中的伙计
D. 准备使用 FreeSwitch 做 IPPBX 的伙计
E. 准备开发 IP 呼叫中心的伙计
F. 对 FreeSwitch 进行运营维护的伙计

1. 研究 VOIP/FreeSwitch 之前需要哪些基础知识？

假如你不开发应用，只是使用 FreeSwitch，通常熟悉 SIP 就够。
假如要你开发应用，熟悉 C/C++是必须的。
虽然 FreeSwitch 支持很多语言的开发接口，但是实际上基本都是在 C/C++的接口上通过 SWIG 进行扩展的。通过 swig 包装支持多种脚本语本语言控制呼叫流程,如
perl php lua python ruby java tcl 等。

事件套接字 使用 Event Socket 可以使用任何其它语言通过 Socket 方式控制呼叫流程、扩 展 FreeSWITCH 功能

通俗讲 FreeSwitch 是也是一个 VOIP，目前主流的 VOIP，甚至包括运营商的 IMS 毫无例外都是基于 SIP 协议的。因此要搞定 FreeSwitch，SIP 协议是必须的。

2. VOIP 基础设施有哪些？

以太网络，这个是废话，没有这咋实现 IP 通信呢。

IP 话机,相当于普通电话机，不过不是普通电话机的电话线 RJ11 的接口 而是网线 RJ45 接口.

软电话，安装在电脑上的（通常是 windows 下）模拟 ip 话机的软件。当然安卓和 linux 下也有软电话。

SIP Server/IPPBX，完成 VOIP 交换的设备或者安装在 pc 上的软件。

IP 网关，通常是完成 VOIP 网络 到 PSTN E1 数字接口或者模拟电话线接口转换的设备

3. SIP 基本问题有哪些？

SIP 协议广义上是包括 SIP 协议 ，RTP 协议和 SDP 协议。

他们的关系可以这样描述：

SIP 是信令控制，相当于电信里面的 7 号信令，或者相当于发号施令的领导角色，

RTP 是媒体传输，相当于干体力活的民工角色

SDP 是媒体描述，相当于描述使用哪些或者哪种语音和视频编码进行协商。

举例说明：a 用户使用电话打给 b 手机

SIP 是信令控制 相当于 a 举起电话机 然后 b 的号码 和 通话之后挂机的动作。

RTP 是媒体传输 相当于 电话通了 a 和 b 语音通话的动作。

SDP 是媒体描述 相当于 a 和 b 都使用普通话进行对话。假如 a 和 b 没有一个共同 的语言 会导致协商不成功。就无法通话。比如 a 是中国人，b 是英国人，a 问：会说汉语吗。b 听不懂，b 问：can you speak English？ a 也不懂。最终只能放弃通话。

SIP 是控制命令，是具有文本的，通过网络抓包是可以直接看出内容，这样调试排查问题就比较方便

SIP 协议 默认使用 UDP 协议。但是也支持 TCP 协议，有些变态的系统或者软电话只能支持 TCP，比如微软的 LYNC

SIP 默认是使用 5060 端口。但是通常也可以指定其他端口。比如 5061 之类。

有些软件，比如 tom 版本的 skype 启动之后也会使用 5060 端口，这样导致了其他软电话或者 FreeSwitch 启动失败。

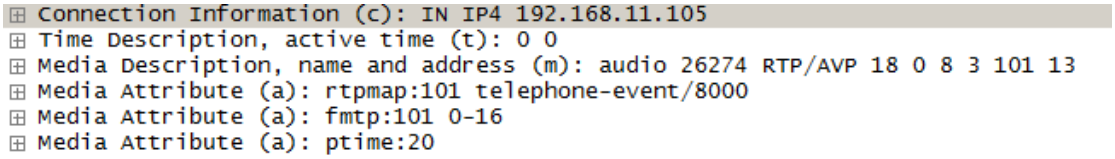
通常系统有很多线路可以并发使用，但是他们都是使用一个端口（默认是 5060）进行信令控制的。

SIP 使用模式 有两种：一种是注册模式，大家每个人分配一个分机号码和密码，然后都注册到一个 SIP SERVER /IPPBX 上。呼叫的时候只要呼叫分机号码就可以
另外一种是点对点模式： 通过呼叫 软电话的或者 IP 话机的 IP+端口（默认 5060）的模式。
标准的 sip 地址格式是： sip:1234@IP:PORT
其中 1234 是号码

4. RTP 基本问题有哪些？

RTP 的默认端口是没有指定的，通常系统有很多线路可以并发使用，每一路通常都会 占用两个端口 一个是 RTP 一个是 RTCP，
RTP 媒体传输的机器的 IP 和 SIP 的信令控制的 IP 可以不是同一个。虽然很多情况下他们是一个，表示信令控制和媒体传输都是一台机器上发起的。但是大的系统里面往往他们的 ip 不是同一个。小系统：一个部门就是一个人，这个人既是领导（SIP）也是干苦力活的（RTP）。
因为一个机器的 cpu 毕竟计算能力有限。而一个大的系统相当于一个部门很多人，领导（SIP）通常一个就够，干苦力活的（RTP）需要很多人。
SIP 的流量通常很少，但是 RTP 的流量通常是巨大的，因为要传输语音流个视频流。所谓：领导一句话，手下干活累死人。

RTP 的语音流是一份一份（从时间上看是离散的）进行传输的。
一份就是一个 UDP 包，通常情况下一个包包括了 20ms 或 30 毫秒的语音。每个包的负载大小根据语音编码器不同，在几十 Byte 到几百 Byte 之间
这样的话，假如是 20ms 一个包，1 秒的语音要发 50 次，可见流量是很大的。
这个 RTP 的 ip 通过抓包可以看出来。
参考下图：

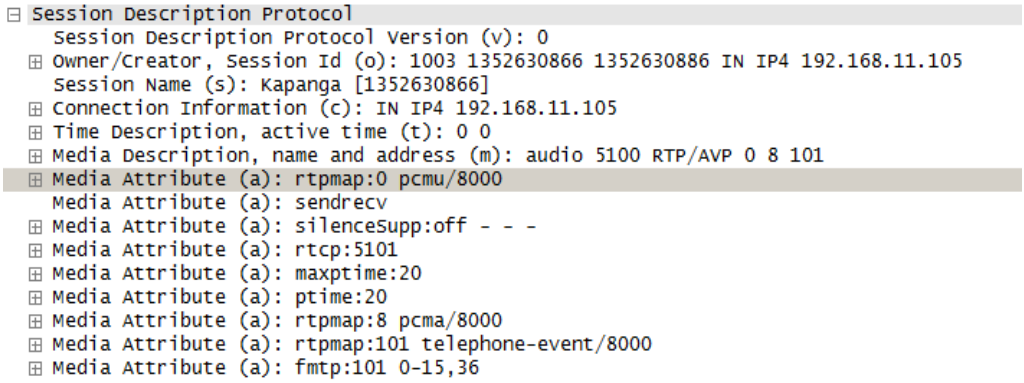


其中的 Connection Information 后面的 192.168.11.105 就是 RTP 的 IP 地址
audio 后面的 26274 就是 RTP 的端口。

5. SDP 基本问题有哪些？

SDP 是对媒体的描述，描述了 RTP 的 IP 和端口 PORT
说明自己这边有哪些编码器（相当于一个人会说哪些语言）
也描述了是否支持视频（说白了就是有视频编码器）。
也描述了是否支持按键。

很多情况下呼叫建立不成功都是因为 SDP 协商不成功导致。这样需要通过抓包才能看出。
参考下图：



其中的 Connection Information 后面的 192.168.11.105 就是 RTP 的 IP 地址
audio 后面的 5100 就是 RTP 的端口。
RTP/AVP 后面的 0 8 101 表示支持的 编码 0 表示 G.711 Ulaw, 8 表示 G.711 的 Alaw，101 是表示支持电话按键事件
后面的很多 Media Attribute 是对媒体属性的具体描述。
总结一下：
SDP 的核心有几个：

A. 说明媒体的来源 IP 地址
B. 说明媒体的来源的端口 port
C. 说明自己有哪些媒体能力（能说几个语言）
D. 包括对 dtmf 按键的是否支持是说明。

一旦两个软电话要对话，就要对比交换上面这些信息。

6. 常用的支持语音的软电话有哪些？

Windows 下常见的软电话有：
Xlite/ eyebeam,kapanga,linphone,jitsi，PJSIP 等等
Linux 下有 Linphone，PJSIP 等
安卓系统下常用的软电话有：sipdroid,CSipSimple,jPhoneLite

7. 常用的支持视频的软电话有哪些？

Windows 和 Linux 下的软电话大多也支持视频。
Windows 下常见的软电话有：
Xlite/ eyebeam,kapanga,linphone,jitsi 等等
Linux 下有 Linphone 等
本人使用 kapanga 和 Linphone 测试过 FreeSwitch 的视频通话。

8. 常见语音编码器有哪些？

常用的语音编码器有
G.711 Ulaw 和 Alaw 这个是绝大多数的 VOIP 都支持的。G.711 要是都不支持，就不能称为 VOIP，所以刚刚开始测试使用某个系统或者软电话，使用 G.711 准没有错。
G.729 这个是公网上使用最多的，但是由于有授权，有些软电话不支持，或者需要采购才能使用。

iLBC 大多数软电话都支持，而且没有授权限制
SPEEX 大多数软电话都支持，网络上也有源码，siri 和科大的语音识别的的语音传输就是使用这个编码，flash player 也支持这个编码
GSM 大多数软电话都支持，在公网上假如 G.711 带宽太大，其他编码又没有的情况，可以考虑使用这个。

9. 常见视频编码器有哪些？

H.263 最基本的视频，大多数的支持视频的软电话都支持
其中 H.263 又细分为 H.263 , H.263+/ H.263-1998 等
MP4
H.264 (MPEG-4 part 10)效果最好

10. PSTN 和 VOIP 区别有哪些？

PSTN 是电路交换，是时分，话音质量是可靠的可控的，无论闲死还是忙死，话音质量都是一样的。
VOIP 是数据包交换，话音质量有用以太网的天生属性，质量是不可靠的。通常网络情况下话音效果不错，但是一旦网络繁忙，话音质量就立刻下降。因此 VOIP 的运营最大的难点是在于系统的稳定性。

11. PSTN 常用信令有哪些

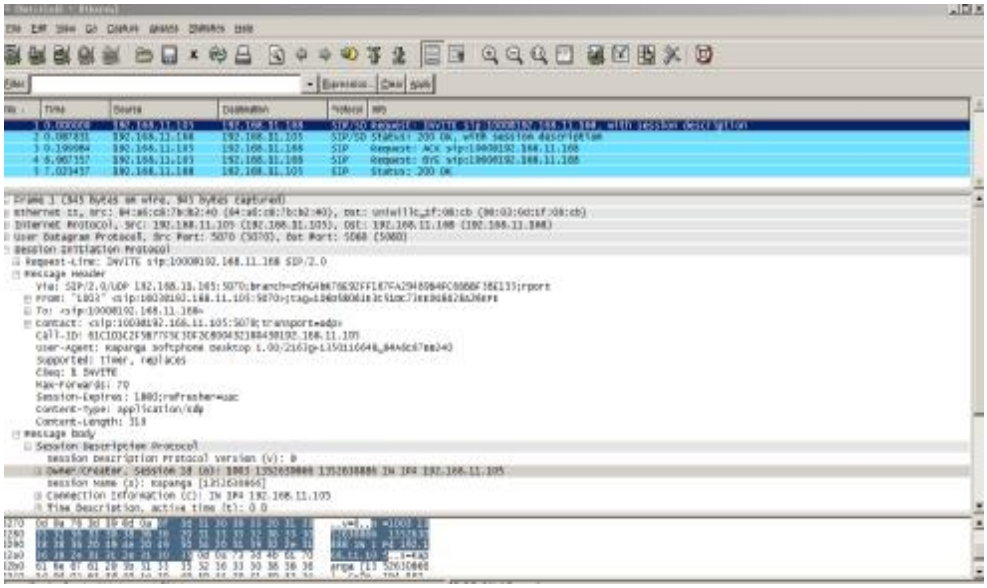
常用的 PSTN 信令有 ISDN PRI 和 SS7
其中区别在于 ISDN PRI 是一个 E1（30 路话路）使用其中的 16 时隙作为信令控制
SS7 是 7 号信令，是很多 E1，最大可以到 2^12 路（4096）话路共用某个 E1 的某个时隙（通常是 16 时隙，作为信令 LINK）作为信令控制，为了提高可靠性，通常使用两个 E1 线路上的时隙（即两个 LINK）作为信令控制。

12. VOIP 的系统开发和测试有哪些常用工具？

最常用的通常是网络抓包程序

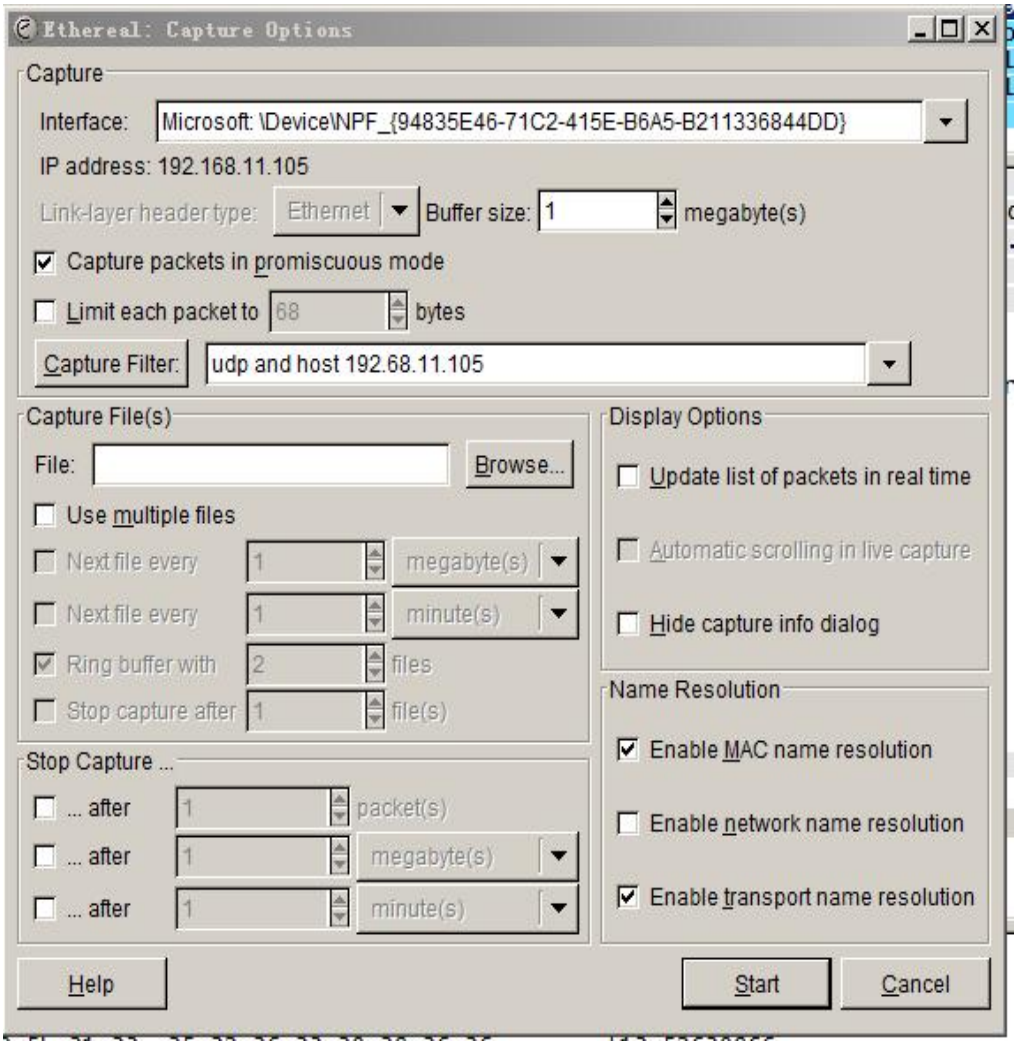
Ethereal + WinPcap 或者 wireshark + WinPcap

有用 sip 协议是文本字节流模式，抓包之后，直接使用工具里面的查看窗口直接查看，如下图：



13. 如何使用 Ethereal 对指定机器进行抓包分析？

在 Capture 菜单中选择 Option 子菜单。然后在出现的对话框中选择网卡（Interface）如下图：



然后在 Capture Filter 里面输入 udp and host 192.168.11.105
Udp 表示 只是抓 udp 的包，因为 sip 默认是基于 udp，rtp 也是基于 udp
Host 192.168.11.105 表示只抓 ip 是 192.168.11.105 的数据包（从本机器发给 192.168.11.105，或者从 192.168.11.105 发给本机）。
其中 192.168.11.105 也可以是本机地址。
然后点 start 开始抓包。

14. 如何使用 Ethereal 对指定端口进行抓包分析？

类似上面问题，
在 Capture Filter 里面输入 udp and port 5060
Udp 表示 只是抓 udp 的包
Port 5060 表示只是抓 5060 端口上的包。
因为 sip 默认是基于 udp，默认是 5060 端口。以上的命令就是只是抓 sip 信令控制的包。

15. Ethereal 能对本机内的通信进行抓包吗?

本机的内地通信实际上没有经过网卡,因此 Ethereal 无法实现对本机内的通信进行抓包,比如本机的软电话呼叫本机上的 FreeSwitch,就无法抓包。

16. Ethereal 能对其他机器之间的通信进行抓包吗?

一般情况下,假如 A 机器和 B 机器进行通信,而你要在 C 机器上对 A 机和 B 机的通信进行抓包,是抓不到包的
比如 a 机的软电话呼叫 B 本机上的软电话,你要在 C 机器上对他们的通信进行抓包,是抓不到包。

17. Windows 下使用啥命令工具可看哪个 port 被谁占用?

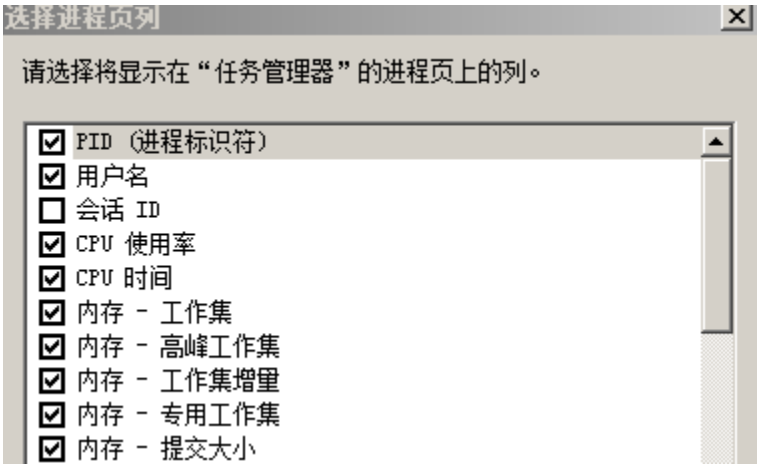
首先:在 windows 命令行下:输入
netstat -oan >c:\tmp.txt

然后打开 c:\tmp.txt,如下图:

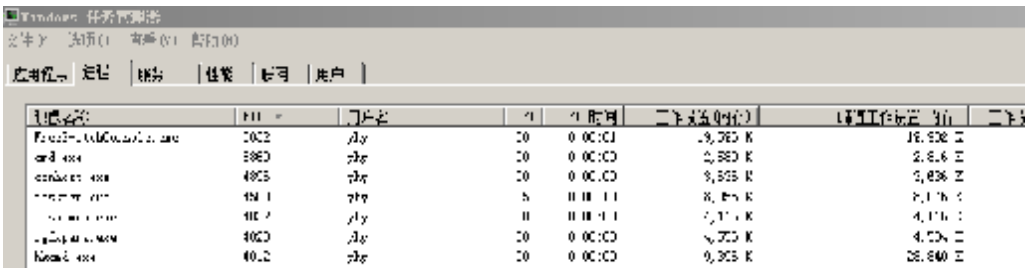
协议	本地地址	外部地址	状态	PID
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING	844
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING	4
TCP	0.0.0.0:1025	0.0.0.0:0	LISTENING	484
TCP	0.0.0.0:1026	0.0.0.0:0	LISTENING	928
TCP	0.0.0.0:1027	0.0.0.0:0	LISTENING	996
TCP	0.0.0.0:1028	0.0.0.0:0	LISTENING	576
TCP	0.0.0.0:1029	0.0.0.0:0	LISTENING	600
TCP	127.0.0.1:2735	127.0.0.1:2736	ESTABLISHED	3444
TCP	127.0.0.1:2736	127.0.0.1:2735	ESTABLISHED	3444
TCP	127.0.0.1:8021	0.0.0.0:0	LISTENING	6032
TCP	192.168.11.105:139	0.0.0.0:0	LISTENING	4
TCP	192.168.11.105:5060	0.0.0.0:0	LISTENING	6032
TCP	192.168.11.105:5080	0.0.0.0:0	LISTENING	6032
UDP	127.0.0.1:60908	*:*		3244
UDP	127.0.0.1:62781	*:*		6032
UDP	127.0.0.1:62782	*:*		6032
UDP	127.0.0.1:62783	*:*		6032
UDP	127.0.0.1:62784	*:*		6032
UDP	127.0.0.1:62785	*:*		6032
UDP	127.0.0.1:62786	*:*		6032
UDP	192.168.11.105:137	*:*		4
UDP	192.168.11.105:138	*:*		4
UDP	192.168.11.105:1900	*:*		3244
UDP	192.168.11.105:5060	*:*		6032
UDP	192.168.11.105:5080	*:*		6032

可以看到 UDP 5060 端口的占用的 PID 是 6032。
从图中我们看到 6032 还占用了 5080 等端口。

打开任务管理器，点查看菜单，然后点选择列。。。在出现的对话框中勾选 PID，如下图：



然后在进程标签页总找出 PID 是 6032 的进程名称。如下图:



从图中可以看出是 FreeSwitchConsole.exe。我们知道这个是 FreeSwitch 的启动进程。

18. 如何根据使用的编码器计算 VOIP 需要的带宽？

VOIP 的数据包 包括 SIP 数据包和 RTP 数据包，SIP 只是传输控制命令，跟 RTP 的语音媒体数据包比较起来简直就是九牛一毛，完全可以忽略不计。

下面仅仅计算 RTP 的数据包

RTP 的数据包=以太网地址+IP 类型+IP 包头+UDP 包头+RTP 包头+语音编码之后的负载

以 G.711 alaw 编码器为例

以太网地址 12 字节

IP 类型 2 字节

IP 数据包头:20 字节

UDP 协议的报头: 8 个字节

RTP 包头是 12 字节

语音负载：160 字节：一个包 20ms，8K 的采样频率，每个采样点使用 alaw 编码之后是 1 个字节。因此 20ms 的语音在编码之后的负载是 160 字节。

总共是 12+2+20+8+12+160=214 字节

1 秒的语音就是 214*（1000/20）=214*50=10700 字节,按照流量计算就是 10700*8/1024=84Kpbs 左右。

以 G.729 编码器为例:

假如是 G.729. 一个包 20ms 其他不变，只是语音编码之后变成了 8 字节，因此语音负载是 8 字节，

可以算出来：总共是 12+2+20+8+12+8=62 字节

语音就是 62*（1000/20）=214*50=10700 字节,按照流量计算就是 10700*8/1024=24Kpbs 左右。

一个 10Mbps 带宽，有效的带宽能达到 4-5M 左右，

以 G.711 alaw 为例子:一线需要的带宽是 84Kbps.因此可以支持的线路数是：5*1000/84=60 线左右。但是考虑到业务通常还有带宽需求，因此，30 线是比较保险的。

以 G.729 为例子:一线需要的带宽是 24Kbps.因此可以支持的线路数是：5*1000/24=200 线左右。

但是考虑到业务通常还有带宽需求，比如 CRM 的弹屏通常包括了用户 360 度的信息和历史记录，数据量也是不小。因此，10M 的带宽，100 线是比较保险的。

另外要注意的是 ADSL 的带宽上行和下行是不对称的，而 VOIP 的语音通话的开销是求上下行一样的带宽。因此假如并发使用的线路多了，而且碰到单边效果不好的情况，很可能的带宽不对称导致的。

19. 如何测试你的系统的 WAN 的进出口带宽？

有很多网站都可以提供带宽测试

http://www.speedtest.net/

是比较好用的一个

上面有很多 host 可以选择进行测试。

比如你是联通的宽带，可以选择上面的联通的 host 进行测试。

根据本人测试的结果还是比较符合实际情况。

20. 如何理解 VOIP 的 NAT 穿透？

这个问题解释起来比较复杂。

具体来说可以分为下面几个问题：

a. 啥是 NAT
b. 啥是 NAT 穿透
c. 为啥需要 NAT 穿透
d. 如何实现 NAT 穿透
e. FS 如何解决 NAT 穿透。

先看问题 a. 啥是 NAT。

NAT(Network Address Translators)，网络地址转换：网络地址转换是在 IP 地址日益缺乏的情况下产生的，它的主要目的就是为了能够地址重用。NAT 分为两大类，基本 的 NAT 和

NAPT(Network Address/Port Translator)。

最开始 NAT 是运行在路由器上的一个功能模块。

NAT 的产生基于如下事实：一个私有网络的节点中只有很少的节点需要与外网连接。那么这个子网中其实只有少数的节点需要外网的 IP 地址，其他的节点的 IP 地址应该是可以重用的。

因此，基本的 NAT 实现的功能很简单，在子网内使用一个保留的 IP 子网段，这些 IP 对外是不可见的。子网内只有少数一些 IP 地址可以对应到真正全球唯一的 IP 地址。如果这些节点需要访问外部网络，那么基本 NAT 就负责将这个节点的子网内 IP 转化为一个全球唯一的 IP 然后发送出去。(基本的 NAT 会改变 IP 包中的原 IP 地址，但是不会改变 IP 包中的端口)

关于基本的 NAT 可以参看 RFC 1631

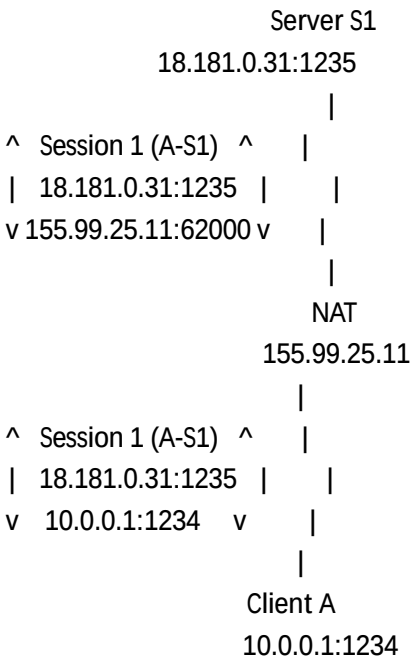
另外一种 NAT 叫做 NAPT，从名称上我们也可以看得出，NAPT 不但会改变经过这个 NAT 设备的 IP 数据报的 IP 地址，还会改变 IP 数据报的 TCP/UDP 端口。

现在大大多数路由器都是 NAPT

上面描述很拗口，归纳一下可以这样理解：一个公司的局域网内的很多机器通过路由器的一个公网 IP，来使用外网上的其他服务。路由器上要做 ip 转换啥的。这样外网上的机器看到的都是这个公司我公网 ip 了。

然后看问题 b. 啥是 NAT 穿透,描述起来比较麻烦，我试图通过下面的例子来说明：

看下图：



有一个私有网络 10.*.*.*，Client A 是其中的一台计算机，这个网络的网关（一个 NAT 设备）的外网 IP 是 155.99.25.11(应该还有一个内网的 IP 地址，比如 10.0.0.10)。如果 Client A 中的某个进程（这个进程创建了一个 UDP Socket,这个 Socket 绑定 1234 端口）想访问外网主机 18.181.0.31 的 1235 端口，那么当数据包通过 NAT 时会发生什么事情呢？

首先 NAT 会改变这个数据包的原 IP 地址，改为 155.99.25.11。接着 NAT 会为这个传输创建一个 Session（Session 是一个抽象的概 念，如果是 TCP，也许 Session 是由一个 SYN 包开始，以一个 FIN 包结束。而 UDP 呢，以这个 IP 的这个端口的第一个 UDP 开始，结束呢，呵呵， 也许是几分钟，也许是几小时，这要看具体的实现了）并且给这个 Session 分配一个端口，比如 62000，然后改变这个数据包的源端口为 62000。所 以本来是（ 10.0.0.1:1234->18.181.0.31:1235 ） 的数据包到了互联网上变为了（155.99.25.11:62000->18.181.0.31:1235）。

一旦 NAT 创建了一个 Session 后，NAT 会记住 62000 端口对应的是 10.0.0.1 的 1234 端口，以后从 18.181.0.31 发送到 62000 端口的数据会被 NAT 自动的转发到 10.0.0.1 上。(注意：这里是说 18.181.0.31 发送到 62000 端口的数据会被转发，其他的 IP 发送到这个端口的数据将被 NAT 抛弃）这样 Client A 就与 Server S1 建立以了一个连接。

然后看问题 c.为啥需要 NAT 穿透，这个问题是因为下面 3 个原因导致：

1. VOIP 是基于 UDP 通信的，没有固定的连接，对比来看 http web，ftp 啥的是基于 tcp，有固定的连接。客户端发起通信之后通路就通了，大家可以收发数据。因此 http，ftp 啥的通常不需要 NAT 穿透。
2. VOIP 比如 sip，信令控制和媒体流是分开的。
3. VOIP 的媒体流的 ip 和端口是在 sdp 里面描述的。Sdp 是客户端在发起呼叫的时候造好的。此时根本不知道外网的 ip 和外网将要使用端口（语音 rtp 包都还没有发呢）。

因此服务端收到的 sdp 里面的 ip 和 port 都是客户端内网的 ip 和 port。将来服务器发包的时候会发到这个 ip 和 port 上，这样显然是错误的，因此需要一些办法来纠正这个问题，这个办法就是 NAT 穿透。

下面看问题 d.如何实现 NAT 穿透。

实现 NAT 穿透在 VOIP 上可以这样描述：假设 VOIP 服务端，比如 FS，在公网上，能跟在其他公司内部局域网上的 VOIP 客户端进行 VOIP 通话。

实现这个功能 大多是基于 ICE 和 STUN 之类的办法，都需要第 3 方服务器的支持。

我们的希望是通过服务器，在服务器上自己就实现。

而不是借助第 3 方服务器之类的办法。

下面就就是介绍这个原理：

举例如下:

1. SIP Server IP 60.10.58.193 sip 端口是 5060
2. SIP Client :Sip 软电话 内网地址:10.10.10.70, sip 端口是 5200
3. 路由器 公司内网地址:10.10.10.2,路由器外网地址: 218.107.192.68
4. sip 通信过程:

   SIP Client 发起 sip 信令.

   SIP 里面包括 sdp 的数据协议.

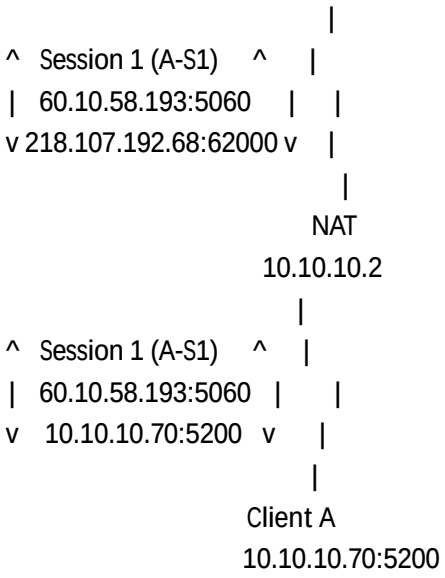
   SDP 协议里面的 RTP 地址填: 10.10.10.70

   RTP PORT= 10140

   SIP Server 的信令过程

Server SIP

60.10.58.193:5060

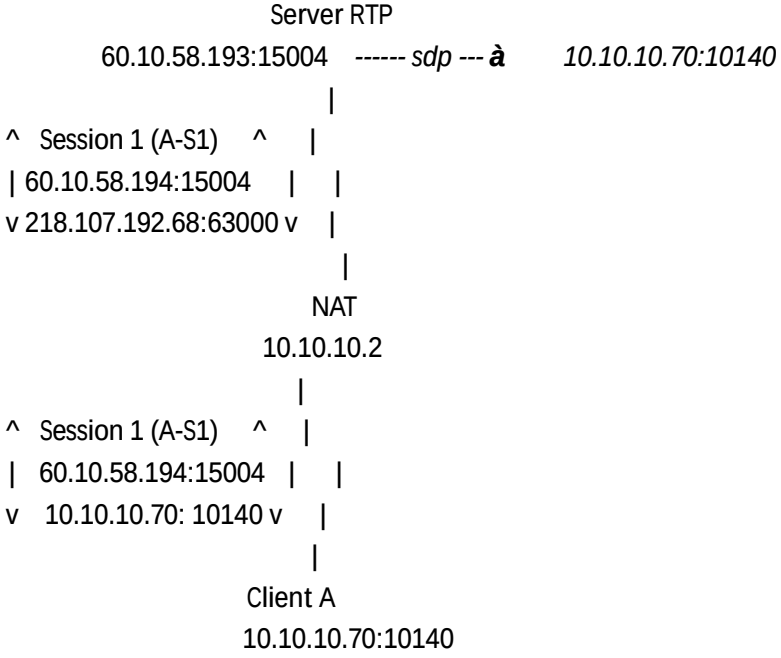


说明：经过路由器之后：sip client 的端口 5200 变成了，62000（这个是路由器分配的）

5. RTP 的话音通信过程

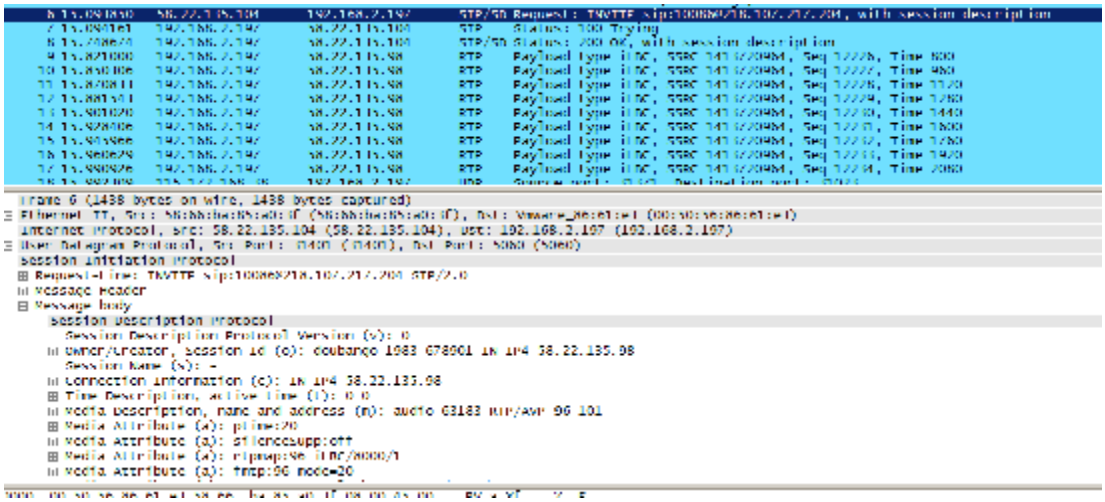
注意: Server RTP 原来是根据 SDP 协议里面的地址 10.10.10.70: 10140 发送 RTP,需要改为根据收到的 RTP 的来源发送 RTP.才能实现 NAT 穿透.

这是通过 server 上当判断收到的 rtp 的包和 SDP 里面的参数不一样时,会上报事件給应用程序,然后允许应用程序重新设置 RTP 参数才能实现的.



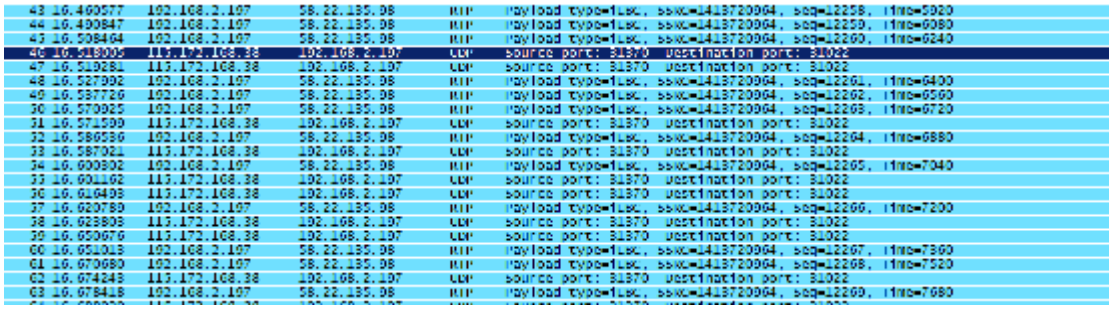
最后我们看问题 e. Fs 如何解决 NAT 穿透。

参见下面的 eyebeam 和公网的 FS 通信的 fs 服务端 VOIP 抓包：
FS 的 ip 是 192.168.2.197
VOIP 客户端的 SIP 地址是 58.22.135.104，



VOIP SIP 发给 FS 的 INVITE 的 SDP 的里面的媒体的 IP 是 58.22.135.98，端口是 63183
因此 FS 发 RTP 到 58.22.135.98 了。这个时候，VOIP 客户端是听不见声音的。

后续，fs 开始收到客户端的 rtp 的语音数据包
如下：



这个会话的 rtp 的语音数据包 ip 变成了 115.172168.38，端口是 31370
因此 FS 发 RTP 到 115.172168.38 了。然后 115.172168.38 这个 ip 上的路由器把包转发给内网的 voip 客户端，这个时候，VOIP 客户端就可以听见声音了。

一句话归纳是：服务端的语音包要发到哪里去是要能跟随哪里来语音包进行变动，才能支持 NAT 穿透。或者简单一点:哪里去 by 哪里来

FreeSwitch 基础和配置部分

21. FreeSwitch 是什么？

FreeSWITCH 是一个开源的电话交换平它具有很强的可伸缩性，从一个简单的软电话客户端到运营商级的软交换设备。他可以是一个 SIP SERVER，也可以通过他实现很多协议转换。也可以实现 VOIP 的 IVR 或者呼叫中心。

22. FreeSwitch 是谁发起开发的？

Anthony Minessale 在 2005 年的时候认为 Asterisk 存在许多问题
而修复这些问题需要很多时间。于是他想从头创建一个 Asterisk 2.0
后面就变成了 FreeSWITCH，因此从某种意义上说 FreeSWITCH 是 Asterisk 2.0。一开始没有人认真考虑他的问题，因此他就自己开发，牛人通常就是这样。呵呵

23. FreeSwitch 历史是什么？

2005 开始怀孕
2007 年发表 1.0
2010 年发布 1.0.6
2012 年发布 1.2.X 和 1.3.X

24. FreeSwitch 能做啥？

FreeSWITCH 几乎无所不能可以用作，一个简单的交换引擎、一个 PBX，
一个媒体网关或媒体支持 IVR 的服务器等。
它支持 SIP、H323、Skype、Google Talk，RTMP 等协议，支持板卡 E1 接口，这样就可以实现通过运营商打电话到手机或者固定电话之上。

25. FreeSwitch 如何与其他系统集成？

FreeSwitch 跟其他系统集成基本上是通过他的支持的协议接口进行集成的
最主要的集成方式就是通过 SIP 协议。
可以作为一个分机注册到其他系统上，也可以其他系统作为分机注册到 FreeSwitch 上以

实现互通，甚至可以不通过注册直接使用点对点方式进行通讯。
实际使用过程中：集成过以下第 3 方的系统或者设备
FreeSwitch==SIP==毅航公司 ISX1000/4000 系列多媒体可编程交换机
ISX1000/4000 系列的新驱动支持 iLBC 编码器，可以和 FreeSwitch 公网集成，从而实现 FreeSwitch 的落地出口。

FreeSwitch==SIP==东进公司 Keygoe 系列多媒体可编程交换机

FreeSwitch==SIP==Dialogic 公司 HMP 多媒体可编程软交换系列

26. FreeSwitch 最新版稳定本号（2013-01-21）是什么？

在 http://files.freeswitch.org/上最新的稳定版本是 1.2.5.3：

通过 git 上去取,最新的版本据说是 1.3.3

27. FreeSwitch 支持哪些操作系统？

主流的 LINUX 和 WINDOWS 操作系统都支持。都有现成的安装包。
这个对初学者比较有利。
很多初学者不熟悉 LINUX 系统，使用 WINDOWS 可以迅速入门，安装配置到运行起来呼叫通第一个软化电话前后不超过 10 分钟。会比较有成就感;-)！

28. 去哪里下载 FreeSwitch 安装包和源码？

在 http://files.freeswitch.org/ 下载比较方便，初学者可以现在已经编译好的二进制安装包。 使用 GIT 去下载源码很麻烦。因此不建议

29. FreeSwitch 在 windows 下如何安装？

以 1.2.3 版本为例，下载 freeswitch1.2.3.msi
然后双击运行，然后点 next
然后在 END-User License 界面 勾选 I Accept。。。
然后点 next，然后选择 Complete 完全安装（反正没有多大），然后点 Install

就开始安装拷贝。 拷贝完成 点 ok 完成安装。
假如机器上有 360 之类的还要点允许访问和记住选择。
FreeSwitch 会安装到 C:\Program Files\FreeSWITCH 目录下
假如机器上之前安装过去其他版本，请先卸载，然后完全删
除 C:\Program Files\FreeSWITCH 目录 以及目录下的所有东西，然后再安装，
假如 C:\Program Files\FreeSWITCH 目录 以及目录下的有文件东西将可能导致
安装不成功

30. FreeSwitch 在 LINUX 下如何编译和安装？

以下是本人在 centos 6.3 上的安装步骤：
install freeswitch on centos 6.3:

1）先到 http://files.freeswitch.org/ 下载 源码包 freeswitch-1.2.5.3.tar.bz2
2）然后 tar jxvf freeswitch-1.2.5.3.tar.bz2
假如是.gz 结尾的请使用-z 命令： tar zxvf freeswitch-1.2.4.tar.gz

3）核查必须安装工具：
sudo yum install git autoconf automake libtool ncurses-devel libjpeg-devel
4）核查可选的安装工具：
sudo yum install expat-devel openssl-devel libtiff-devel libX11-devel unixODBC-devel libssl-devel python-devel zlib-devel libzrtpcpp-devel alsa-lib-devel libogg-devel libvorbis-devel perl-libs gdbm-devel libdb-devel uuid-devel @development-tools
5) ./rebootstrap.sh
6) ./configure
7) vi modules.conf 添加需要的额外模块比如 mod_rtmp 模块 默认 是没有 not found rtmp endpoint 和 ilbc codec 两个模块的
8) make
9) make all install cd-sounds-install cd-moh-install
cd /usr/local/freeswitch/conf/autoload_configs
vi modules.conf.xml 根据需要增加相应的模块，比如 mod_rtmp 和 ilbc 模块
10) cd /usr/local/freeswitch/bin/freeswitch
./freeswitch

说明：
默认 FreeSwitch 安装在/usr/local/freeswitch/，然而如果不想使用默认安装路径，可以在 ./configure 的时候 添加“--prefix=”参数改变 FreeSwitch 的安装路径。
./configure --prefix=/usr/freeswitch

31. FreeSwitch 在 windows 下如何编译？

1.2.1 或者之后的版本需要使用 VS2010 进行编译。

本人使用 VS2010 旗舰版（安装 VS2010 的时候 C#也要安装，否则有些 FreeSwitch 的工程无法打开）。

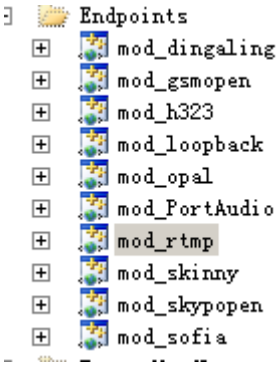
VS2010 运行之后，打开源码包里面的 Freeswitch.2010.sln。打开之后可能会出现

点确定不管他。

然后开始编译。

注意：有许多模块，比如 rtmp_mod 模块默认是没有编译产生的，需要手工自己在模块对应的工程右键点编译。

如下图：



或者在配置管理器里面勾选对应的模块

如下图：

编译之后会在工程文件所在的目录下产生
\Win32\Debug 目录 （Debug 版本）
或者 Win32\Release 目录（Release 版本）
他们的目录下包括了运行的 exe 文件 dll 和对应的配置文件目录（跟安装包安装产生的文件基本一样，额外多了一下 pdb 之类的文件）。
直接运行 Debug 或者 Release 目录下的 FreeSwitchConsole.exe
就可以启动

32. FreeSwitch 在 windows 下如何安装到 C 盘之外？

目录下，而且变态的安装包不能修改安装目录。只能安装到 C:\Program Files\FreeSWITCH 这个目录。
要搞到其他目录运行可以在安装之后，拷贝整个 FreeSWITCH 目录到其他盘比如 d:
然后运行 d:\ FreeSWITCH 目录下的 FreeSwitchConsole.exe。

33. FreeSwitch 在 windows 下如何启动？

双击 FreeSwitchConsole.exe 就可以
可以把这个文件的快捷方式放到 启动 里面（在程序里面右键点启动，然后点打开）。
然后把 windows 设置成自动登录（需要修改注册表）
这样万一机器重启，机器就自动登录了，然后 FreeSwitchConsole.exe
会自动启动。

34. FreeSwitch 在实际使用部署的时候如何启动比较安全？

带上-nonat 参数启动比较安全
FreeSwitchConsole.exe -nonat

35. FS_CLI 跟 FreeSwitch 是啥关系？

FS_Cli 是 FreeSwitch 一个客户端控制界面，
可以在 FS_Cli 上对 FreeSwitch 进行管理，比如日志级别设置，查看日志，执行呼叫等操

作。FS_Cli 是通过 ESL 接口对 FreeSwitch 进行管理。FS_Cli 也可以执行 APP 模拟进行发起呼叫，播放语音等功能。

FS_Cli 有快捷按键，F1-F12,功能对应如下:

```
<key name="1" value="help"/>
<key name="2" value="status"/>
<key name="3" value="show channels"/>
<key name="4" value="show calls"/>
<key name="5" value="sofia status"/>
<key name="6" value="reloadxml"/>
<key name="7" value="console loglevel 0"/>
<key name="8" value="console loglevel 7"/>
<key name="9" value="sofia status profile internal"/>
<key name="10" value="sofia profile internal siptrace on"/>
<key name="11" value="sofia profile internal siptrace off"/>
<key name="12" value="version"/>
```

FS_Cli 默认是连接到本机的机器上，也可以连接到其他机器上的 FreeSwitch 进行管理。

36. 在 FS_CLI 上如何拨打测试分机？

FS> originate sofia/profile/internal/1000 &echo (拨打 1000 并执行 echo 程序）
FS> originate user/1000 &echo (同上）
FS> originate sofia/profile/internal/1000 9999 (相当于在软电话 1000 上拨打 9999)
FS> originate sofia/profile/internal/1000 9999 XML default (同上)

37. FreeSwitch 能跟哪些外部协议对接？

它支持 SIP、H323、Skype、Google Talk，RTMP 等协议
实际本人测试过 sip 和 RTMP。

38. FreeSwitch 如何跟 PSTN 对接，实现落地？

有两个办法：

办法 1. FreeSwitch 通过 SIP 接到第三方的 VOIP 网关上，VOIP 网关通过 E1 接口接到 PSTN 上，通常 VOIP 网关可以支持 ISDN PRI 和 SS7 信令。比如上面提到的通过 ISX1000/4000 进行落地。

办法 2.使用支持 FreeSwitch 的 E1 接口卡，在机器上插这种卡，安装卡驱动。然后安装

FreeSwitch ，再进行协议配置才能使用。

39. 已经有哪些硬件板卡支持FreeSwitch 跟运营商的 E1 电路 对接？

目前已知的 sangoma 板卡可以 ISDN PRI 和 SS7 信令。

如何在 LINUX 的 freeswitch 平台上安装 sangoma 请参考：
http://www.voip88.com/article-1202-1.html
如何安装在 Windows 的 freeswitch 平台上安装 sangoma 请参考：
http://wiki.sangoma.com/fs-windows-freeswitch-compile-isdn
SS7 信令 的支持软件是商业版本，是要额外收费的，根据了解，1 个 E1 端口差不多是 1K 的 授 权 费 用 ， 根 据 http://wiki.sangoma.com/wanpipe-freeswitch#sangoma-freetdm-ss7-library-libsngss7 上面的介绍，ss7 不支持 tup，仅仅支持 isup：

Sangoma FreeTDM SS7 Library (libsng_ss7)
Features
Sangoma's SS7 Library uses Continuous Computing's (Trillium) MTP2/3 and ISUP stacks to provide a commercial grade SS7 interface to FreeSWITCH, via the FreeTDM channel driver.

List of supported variants
- ISUP (ITU/ANSI)
- MTP3 (ITU/ANSI)
- MTP2 (ITU/ANSI)
- SCCP API

40. FreeSwitch 默认配置启动之后占用哪些端口？

FreeSwitch 启动之后，占用以下这些端口：
SIP 5060 5080
RTP:16384-32768
TCP：1935 假如启动 mod_rtmp 模块
8021 等
具体如下：

```
TCP    0.0.0.0:1935
TCP    127.0.0.1:8021
TCP    192.168.11.105:5060
```

TCP 192.168.11.105:5080
UDP 127.0.0.1:50621
UDP 127.0.0.1:50622
UDP 127.0.0.1:50623
UDP 127.0.0.1:50624
UDP 192.168.11.105:5060
UDP 192.168.11.105:5080

假如这些端口已经被占用，将可能导致启动错误。

41. FreeSwitch 在多个 IP 机器上如何指定运行在某个 IP 上？

修改\conf\sip_profiles\ internal.xml 和 external.xml

```
<param name="sip-ip" value="$${local_ip_v4}"/>
<param name="sip-port" value="$${internal_sip_port}"/>
<param name="rtp-ip" value="$${local_ip_v4}"/>
```

对应于 internal.xml 和 external.xml 的 sip 的 ip 和 rtp 的 ip

42. FreeSwitch 常用目录有哪些？

主要目录：

mod 可加载模块
sounds 声音文件,使用 playback() 时默认的寻找路径
log 日志，CDR 等
recordings 录音，使用 record() 时默认的存放路径
conf 配置文件目录

43. FreeSwitch 基本配置文件有哪些？

在/Conf 目录下：
vars.xml 文件：一些常用变量默认分机密码=1234，codec，sip ，ip，port 等
/sip_profiles
/autoload_configs
/dialplan
/directory
/dialplan/default.xml 缺省的拨号计划
/directory/default/*.xml SIP 用户，每用户一个文件

/sip_profiles/internal.xml 一个 SIP profile，或称作一个 SIP-UA，监听在本地 IP 及端口 5060，一般供内网用户使用

/sip_profiles/externa.xml 另一个 SIP-UA，用作外部连接，端口 5080

/autoload_configs/modules.conf.xml 配置当 FreeSWITCH 启动时自动装载哪些模块

44. FreeSwitch 如何设置日志级别？

FS 的日志分为两种：

a.一种是显示在界面上的日志修改：

在 FS_CLI 管理界面上：

FS>console loglevel 级别

级别 从 0-7， 比如 6 设置成 INFO 级别

基本越高 日志越大比如设置成 7，DEBUG 级别。几乎每个操作都很多日志。

输入之后，会返回当前的级别提示如下：

FS> console loglevel 0
+OK log level 0 [0]
+OK console log level set to CONSOLE

FS> console loglevel 7
+OK log level 7 [7]
+OK console log level set to DEBUG

FS> console loglevel 6
+OK log level 6 [6]
+OK console log level set to INFO

假如要看 sip 的详细日志，使用以下命令：

sofia profile internal siptrace on 打开 sip 日志

sofia profile internal siptrace off 关闭 sip 日志

假如要一开始就设置日志级别，比如默认启动级别改为 4（warnning），需要 修改 vars.xml 文件里面：

<X-PRE-PROCESS cmd="set" data="console_loglevel=4"/>

b.另外一种是在\log 下输出的文件日志：比如要修改为 wanning，需要修改需要一下文件 conf\autoload_configs\logfile.conf.xml，将里面的

<map name="all" value="debug,info,notice,warning,err,crit,alert"/>

改为：

<map name="all" value="warning,err,crit,alert"/>

45. FreeSwitch 如何看有多少用户注册上来？

在 FS_CLI 管理界面上：

FS>sofia status profile internal

(显示多少用户已注册）

假如 刚刚启动，没有人注册上来：

提示如下：

。。。。。

ZRTP-PASSTHRU	false
AGGRESSIVENAT	false
STUN-ENABLED	true
STUN-AUTO-DISABLE	false
CALLS-IN	0
FAILED-CALLS-IN	0
CALLS-OUT	0
FAILED-CALLS-OUT	0
REGISTRATIONS	0

我使用 eyebeam 测试，实际上开了两个软电话，注册两个上来。控制台看到是 2 注册上来。 如下图：

```
freeswitch@internal> sofia status profile internal
================================================================================
Name                    internal
Domain Name             N/A
Auto-NAT                false
DBName                  sofia_reg_internal
Pres Hosts              192.168.11.105,192.168.11.105
Dialplan                XML
Context                 public
Challenge Realm         auto_from
RTP-IP                  192.168.11.105
SIP-IP                  192.168.11.105
URL                     sip:mod_sofia@192.168.11.105:5060
BIND-URL                sip:mod_sofia@192.168.11.105:5060
HOLD-MUSIC              local_stream://moh
OUTBOUND-PROXY          N/A
CODECS IN               iLBC,PCMA
CODECS OUT              iLBC,PCMA
TEL-EVENT               101
DTMF-MODE               rfc2833
CNG                     13
SESSION-TO              0
MAX-DIALOG              0
NOMEDIA                 false
LATE-NEG                false
PROXY-MEDIA             false
ZRTP-PASSTHRU           false
AGGRESSIVENAT           false
STUN-ENABLED            true
STUN-AUTO-DISABLE       false
CALLS-IN                0
FAILED-CALLS-IN         0
CALLS-OUT               0
FAILED-CALLS-OUT        0
REGISTRATIONS           2
```

最后的一行可以看到 数目是 2，表示 2 注册上来。
这里面还有 sip 的 ip，拨入和拨出系统的编码器等等很多有用信息。

46. FreeSwitch 如何看有哪些用户注册上来？

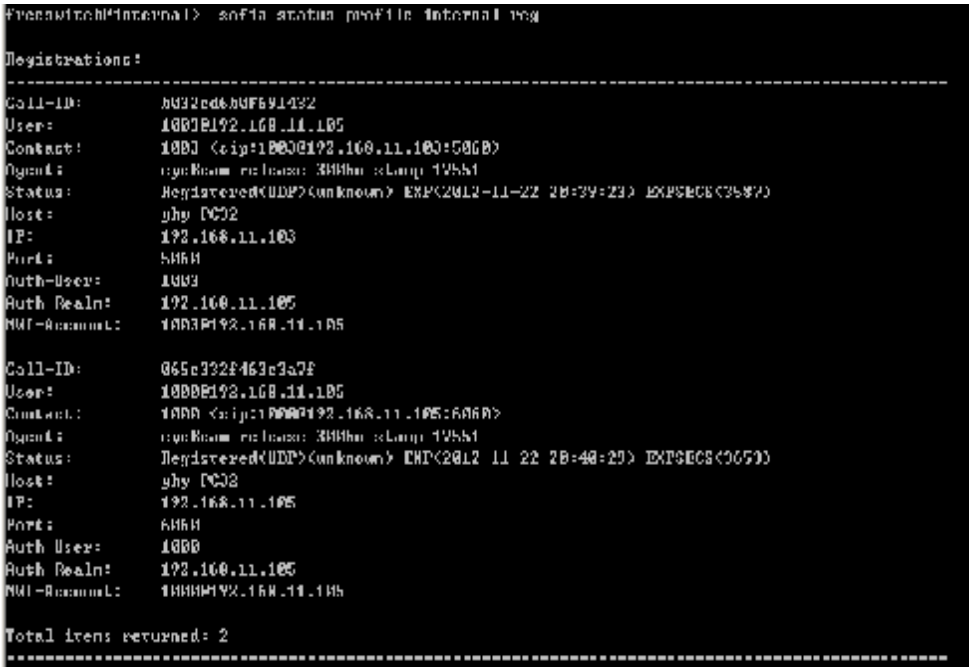
在 FS_CLI 管理界面上：
FS> sofia status profile internal reg 显示哪些用户已注册

刚刚启动 FreeSwitch
FS> sofia status profile internal reg 提示如下：

+OK log level [7]
freeswitch@internal> sofia status profile internal reg

Registrations:
==========
==========
Total items returned: 0

启动两个软电话之后，
FS> sofia status profile internal reg 提示如下：



这个可以看到注册上来 的机器的 ip 地址分机号码等详细信息。

我使用 eyebeam 测试，实际上注册两个上来。控制台看到是 2 个注注册上来。

\db\ sofia_reg_internal.db 里面保存的是注册的信息。
假如碰到意外情况，可能存在有 软电话注册上来之后，一直在里面的情况。
看这个行：
Status: Registered(UDP)(unknown) EXP(2012-11-22 20:45:46) EXPSECS(3655)
最后的秒数目 就是注册有效期。

47. FreeSwitch 默认配置启动之后有哪些默认注册用户和密码是多少？

FreeSwitch 默认配置启动
默认有 1000-1019 20 个帐号
他们的默认密码是 1234.
使用软电话可以注册上去进行呼叫。

48. FreeSwitch 如何设置不需要密码认证？

conf\sip_profiles\ internal.xml 里面的：

```
<param name="accept-blind-reg" value="true"/>
<param name="accept-blind-auth" value="true"/>
```

修改之后使用软电话可以以任何号码和密码注册上去，但是假如没有帐号，呼叫是不行的。
假如要允许 系统没有的用户可以随便注册和呼叫有一种办法是使用下面 lua 脚本的办法：
先修改 conf\autoload_configs\lua.conf.xml 文件，结果如下：

```
<configuration name="lua.conf" description="LUA Configuration">
  <settings>
  <param name="xml-handler-script" value="gen_dir_user_xml.lua"/>
  <param name="xml-handler-bindings" value="directory"/>
  </settings>
</configuration>
```

然后创建 gen_dir_user_xml.lua 文件并且存到 \scripts 目录下, gen_dir_user_xml.lua 内容如下：

```
freeswitch.consoleLog("notice", "Debug from gen_dir_user_xml.lua, provided params:\n" .. params:serialize() .. "\n")

local req_domain = params:getHeader("domain")
local req_key    = params:getHeader("key")
```

```
local req_user      = params:getHeader("user")


   XML_STRING =
[[<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<document type="freeswitch/xml">
   <section name="directory">
      <domain name="]] .. req_domain .. [[">
         <user id="]] .. req_user    .. [[">
            <params>
               <param name="password" value="]] .. req_user .. [["/>
               <param name="vm-password" value="]] .. req_user .. [["/>
               <param                                                    name="dial-string"
value="{sip_invite_domain=${dialed_domain},presence_id=${dialed_user}@${dialed_d
omain}}${sofia_contact(${dialed_user}@${dialed_domain})}"/>
            </params>
             <variables>
               <variable name="toll_allow" value="domestic,international,local"/>
               <param name="accountcode" value="]] .. req_user .. [["/>
               <variable name="user_context" value="default"/>
                <param  name="effective_caller_id_number"  value="]]  ..  req_user  ..
[["/>
               <param   name="effective_caller_id_name"   value="Extension   ]]   ..
req_user .. [["/>>
               <variable      name="outbound_caller_id_name"      value="Extension
5678"/>
               <variable name="outbound_caller_id_number" value="5678"/>
               <variable name="callgroup" value="techsupport"/>
         </variables>
            </user>
         </domain>
      </section>
   </document>]]

      -- comment the following line for production:
  freeswitch.consoleLog("notice",  "Debug  from  gen_dir_user_xml.lua,  generated
XML:\n" .. XML_STRING .. "\n")
```

之后使用随便的分机号码 和跟分机号码一样的密码注册 就可以使用系统。

49. FreeSwitch 默认配置启动之后有哪些分机比较有用？

FreeSwitch 默认配置启动之后

下面的这些分机会经常用到

9196 echo，回音测试

9195 echo，回音测试，延迟 5 秒

5000 示例 IVR

30xx 电话会议比如 3000，3001.拨入之后假如是第一个人会听音乐

50. 如何手工添加 FreeSwitch 分机？

比如要添加一个 13606060253 的 11 位分机。

首先在 conf/directory/default 目录下 增加一个用户配置文件，配置文件可以参考已经有的配置文件。比如 建立一个 13606060253.xml 内容如下：

```
<include>
  <user id="13606060253">
    <params>
      <param name="password" value="13606060253"/>
      <param name="vm-password" value="13606060253"/>
    </params>
    <variables>
      <variable name="toll_allow" value="domestic,international,local"/>
      <variable name="accountcode" value="13606060253"/>
      <variable name="user_context" value="default"/>
      <variable name="effective_caller_id_name" value="Extension 13606060253"/>
      <variable name="effective_caller_id_number" value="13606060253"/>
      <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>
```

然后修改拨号计划(Dialplan)使其它用户可以呼叫到它。

先要修改 conf\dialplan\public.xml 修改为任意号码拨入，

public.xml 里面 public_extensions 修改如下：

```
<extension name="public_extensions">
  <condition field="destination_number" expression="^([0-9]\d+)$">
<action application="transfer" data="$1 XML default"/>
  </condition>
</extension>
```

然后 修改 conf\dialplan\default.xml

在开头部分增加：

```
<extension name="Local_Extension2">
```

```
        <condition field="destination_number" expression="^([0-9]\d+)$">
      <action application="export" data="dialed_extension=$1"/>
      <action application="set" data="call_timeout=10"/>
       <action application="set" data="record_sample_rate=8000"/>
       <action application="export" data="RECORD_STEREO=false"/>
      <action application="set" data="hangup_after_bridge=true"/>
      <action application="set" data="continue_on_fail=true"/>
      <action application="bridge"
data="user/${dialed_extension}@${domain_name}"/>
      <action application="bridge"
data="${rtmp_contact(default/${dialed_extension}@$${domain})}"/>
      <action application="bridge"
data="${rtmp_contact(default/${dialed_extension}@$${domain})}"/>
        </condition>
      </extension>
```

最后重启 FreeSwitch。

51. FreeSwitch 拨号计划的正则表达式有哪些是最常用的模式？

拨号计划使用 perl 的正则表达式
常用的 匹配 模式如下：
^ 表示开始匹配，^123 表示匹配 123 开头
$表示结束匹配 456$表示匹配 456 结束
| 表示或者，匹配任何一个
[] 表示匹配其中的任意一个字符
[0-9] 等于匹配 [0123456789]
\d 等于 匹配[0-9]
\d+ 等于匹配 1 个或多个数字
\d* 等于匹配 0 个或多个前面的字符

52. FreeSwitch 默认配置如何修改拨号计划设置没有注册上来不走留言信箱？

修改/conf/dialplan 目录下的 default.xml 文件
在 <context name="default"> 之后加入：

```
<extension name="Local_Extension2">
      <condition field="destination_number" expression="^(1[01][0-9][0-9])$">
   <action application="export" data="dialed_extension=$1"/>
   <action application="set" data="call_timeout=10"/>
   <action application="set" data="hangup_after_bridge=true"/>
   <action application="set" data="continue_on_fail=false"/>
   <action application="bridge" data="user/${dialed_extension}@${domain_name}"/>
     </condition>
  </extension>
```

其中的
destination_number 是分机号码
这个例子表示分机号码是 1000-1199 共 200 个

53. FreeSwitch 默认配置加载哪些编码器？

使用 show codec 命令可以看到系统加载的编码器

```
FS> show codec
type,name,ikey
codec,ADPCM (IMA),mod_spandsp
codec,AMR,mod_amr
codec,G.711 alaw,CORE_PCM_MODULE
codec,G.711 ulaw,CORE_PCM_MODULE
codec,G.722,mod_spandsp
codec,G.723.1 6.3k,mod_g723_1
codec,G.726 16k,mod_spandsp
codec,G.726 16k (AAL2),mod_spandsp
codec,G.726 24k,mod_spandsp
codec,G.726 24k (AAL2),mod_spandsp
codec,G.726 32k,mod_spandsp
codec,G.726 32k (AAL2),mod_spandsp
codec,G.726 40k,mod_spandsp
codec,G.726 40k (AAL2),mod_spandsp
codec,G.729,mod_g729
codec,GSM,mod_spandsp
codec,H.261 Video (passthru),mod_h26x
codec,H.263 Video (passthru),mod_h26x
codec,H.263+ Video (passthru),mod_h26x
codec,H.263++ Video (passthru),mod_h26x
codec,H.264 Video (passthru),mod_h26x
codec,LPC-10,mod_spandsp
codec,PROXY PASS-THROUGH,CORE_PCM_MODULE
codec,PROXY VIDEO PASS-THROUGH,CORE_PCM_MODULE
```

codec,RAW Signed Linear (16 bit),CORE_PCM_MODULE
codec,Speex,mod_speex

26 total.
但是要注意，有加载不一定就能使用。
具体能否使用还要看 vars.xml 里面的配置

54. FreeSwitch 默认配置哪些编码器能使用？

上面说到系统加载了很多编码器，但是不一定能使用，具体能使用哪些编码器，要看 conf/ vars.xml 配置文件里面的下面的参数：

<X-PRE-PROCESS cmd="set" data="global_codec_prefs=G722,PCMU,PCMA,GSM"/>
<X-PRE-PROCESS cmd="set" data="outbound_codec_prefs=PCMU,PCMA,GSM"/>

global_codec_prefs 是全局能使用的编码器。
outbound_codec_prefs 是 FreeSwitch 拨出的时候使用的编码器

55. FreeSwitch 如何设置修改默认配置添加支持 G.729 ，iLBC 等编码器？

首先 查看：conf\autoload_configs 目录下的 modules.conf.xml 配置文件
看看是否有打开注释加载到系统，注释了就是没有加载的。
比如 mod_ilbc 默认是没有加载的。
而 mod_g729 默认是加载的。
然后在
conf/ vars.xml 配置文件里面的看下面的参数：

<X-PRE-PROCESS cmd="set" data="global_codec_prefs=G722,PCMU,PCMA,GSM"/>
<X-PRE-PROCESS cmd="set" data="outbound_codec_prefs=PCMU,PCMA,GSM"/>

假如没有就加上：

<X-PRE-PROCESS cmd="set" data="global_codec_prefs=G722,PCMU,PCMA,GSM,G729,iLBC"/>
<X-PRE-PROCESS cmd="set" data="outbound_codec_prefs=PCMU,PCMA,GSM,G729,iLBC"/>

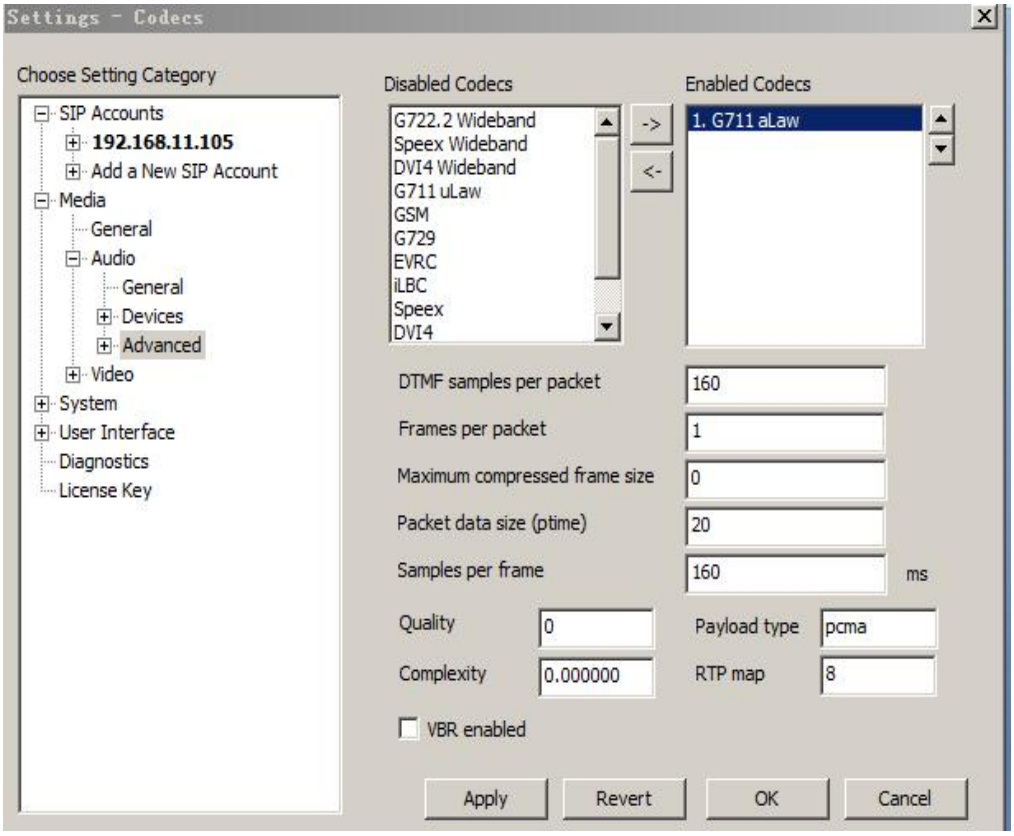
然后保存。重启 FreeSwitch 就可以。
然后有人说我加了 G729，但是还是不行，那是因为 FreeSwitch 的 G729 只能支持透传的方式
不能转码，导致呼叫到 IVR，或者两个软电话一个是 G729 一个不是 G729 也不能通话。
要想使用 G729 通话，只能两个软电话都是 G729 的情况才行。

56. 如何看 FreeSwitch 当前支持哪些语音和视频编码器？

结合使用 show codec 命令可以看到系统加载的编码器
和 sofia status profile internal
或者假如是 5080 端口 sofia status profile external

57. 软电话上如何指定编码器

大多数软电话上都能指定使用编码器
我们强烈建议在内网测试，只要指定 G.711 alaw 语音编码器就可以。简单明了，有问题也好查，因为 SDP 的协商过程太过复杂，初学者很晕。
以 eyebeam 为例子，在 eyebeam 上点右键，点 setting 然后 点 media，audio，advance 如下图：



（通常情况下视频是不需要的，可以都去掉，只有当需要视频功能的时候才加上 h.263）

58. 如何实现然 FreeSwitch 进行转码？

FreeSwitch 的默认配置是不进行转码的，比如软电话 a 呼叫 软电话 b，a 只有 G.711

alaw，b 只有 G.711 ulaw。他们都注册到 FreeSwitch 上，注册是成功的，但是无论 a 呼叫 b 还是 b 呼叫 a 都是不行的。但是 a 呼叫 a 自己是可以的，b 呼叫 b 自己也是可以的。

要让 FreeSwitch 支持分机间能转码通话

需要修改：

conf\sip_profiles\ 目录下的 internal.xml

或 external.xml （5080 端口上的）下面两个参数都设置为 false：

<param name="inbound-late-negotiation" value="false"/>

<param name="inbound-zrtp-passthru" value="false"/>

59. FreeSwitch 哪些编码器不支持转码？

在 FreeSwitch 里面的 G.729 和 G.723.1 AMR 等都是只能透传的。

无法实现转码也无法实现到 ESL 的 IVR。

60. 如何解决 FreeSwitch 的 G.729 的转码？

要实现 G.729 的转码需要自己开发编码器，即在 FreeSwitch 的源码上进行修改。

有人已经做成功这个事情。

有编译好的二进制包可以使用

在 FS 的讨论 190435825 群里面有做好的 二进制安装包,

可以下载 FreeSWITCH1.2.0-G729.rar

或者自己增加这个编码器。

或者网络上购买别人做好的这个编码器。成都的一家公司有出售这个源码。

61. FreeSwitch 如何编程修改源码，手工添加编码器？

根据 http://www.linuxidc.com/Linux/2012-08/68045.htm

上面的介绍：

以模块方式加载的编码在目录 src\mod\codecs 下面，所以如果我们想添加自己的编码，在此目录下参考其他实现即可，freeswitch 对新编码的添加接口比较简单，

自己增加编解码器步骤如下：

注册几个四个回调：

init

encode

decode

destroy
然后通过 switch_core_codec_add_implementation
把这几个回调的实现注册进去。

62. FreeSwitch 如何设置修改默认配置才能支持视频通话？

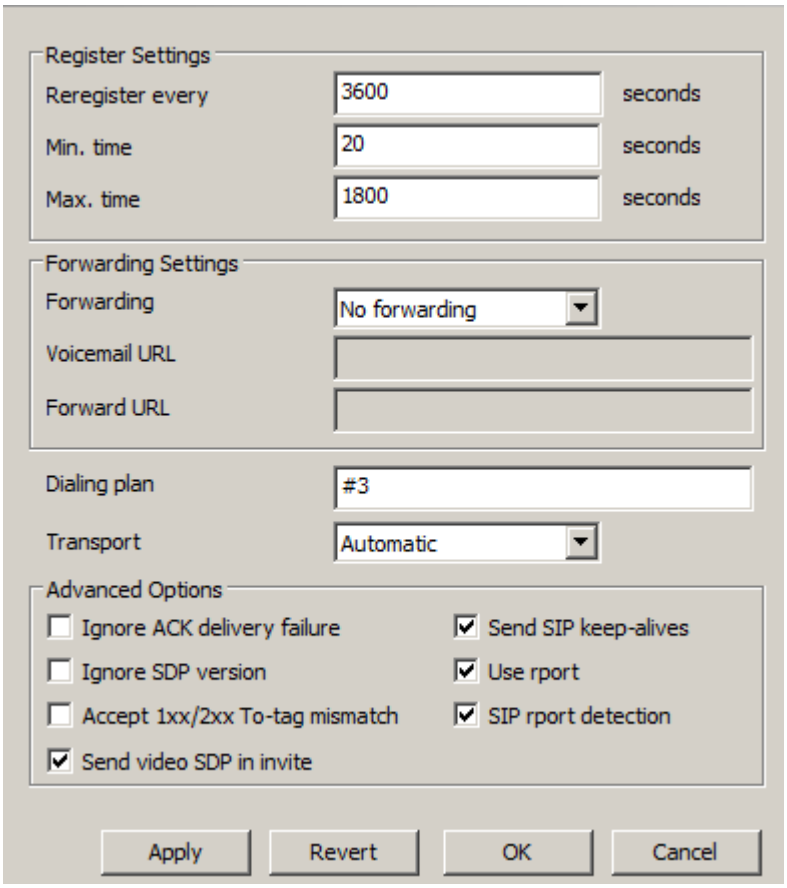
在 conf/ vars.xml 配置文件里面修改下面的参数，增加 H263 或者 H264 编码：

<X-PRE-PROCESS cmd="set" data="global_codec_prefs=G722,PCMU,PCMA,GSM"/>
<X-PRE-PROCESS cmd="set" data="outbound_codec_prefs=PCMU,PCMA,GSM"/>

改为：

<X-PRE-PROCESS cmd="set" data="global_codec_prefs=PCMU,PCMA,GSM,H263"/>
<X-PRE-PROCESS cmd="set" data="outbound_codec_prefs=PCMU,PCMA,GSM,H263"/>

然后保存。重启 FreeSwitch 就可以。
要测试支持视频通话，软电话要支持。先要有视频的编码器，然后
在 eyebeam 里面还要设置如下：



另外：FreeSwitch 的视频通话是不支持转码的，因此使用的时候需要所有的软电话

都设置为一样的视频编码器，比如都设置为 H263 编码器

63. FreeSwitch 如何实现在两个网卡上不同的网段上的分机互通？

在两个网卡上 不同的 网段上,与运行两个网段上的 软电话互通.
这是一个实际应用环境中会经常碰到的问题
比如：
192.168.1.X 和 192.168.11.X 网段
1000 分机 register on internal 192.168.1.X 5060
1002 分机 register on external 192.168.11.X 5060
拨打测试：
external 1002 分机呼叫 ==>internal 1000 ok
internal 1000 分机 呼叫 ==>external 1002 fail

需要修改文件 internal.xml 和 external.xml

位置：/usr/local/freeswitch/conf/sip_profiles
internal.xml 修改内容：
<param name="rtp-ip" value="192.168.1.101"/>
<param name="sip-ip" value="192.168.1.101"/>
<param name="sip-port" value="5060"/>
<param name="inbound-late-negotiation" value="false"/>
<param name="inbound-zrtp-passthru" value="false"/>

external.xml
<param name="rtp-ip" value="192.168.11.101"/>
<param name="sip-ip" value="192.168.11.101"/>
<param name="sip-port" value="5060"/>
<param name="inbound-late-negotiation" value="false"/>
<param name="inbound-zrtp-passthru" value="false"/>
打开 sip_profile/internal.xml 文件，反注释相同的行：
<param name="force-register-domain" value="$${domain}"/>
<param name="force-register-db-domain" value="$${domain}"/>
<param name="dbname" value="share_presence"/>
<param name="presence-hosts" value="$${domain}"/>
打开 sip_profile/external.xml 文件，反注释下面的行：
<param name="force-register-domain" value="$${domain}"/>
<param name="force-register-db-domain" value="$${domain}"/>
<param name="dbname" value="share_presence"/>
<param name="presence-hosts" value="$${domain}"/>

结论：默认的拨号计划，只能呼叫注册在 192.168.1.X 网段上 的 internal 上的

分机。要呼叫 192.168.11.X 上的号码，需要通过 gateway 的方式设置，或者安装下面的方法修改拨号计划：

注意下面的黑体部分其中的 192.168.1.102 就是内网的网卡 ip 地址（不是 192.168.11.X）：

```
<extension name="Local_Extension2">
<condition field="destination_number" expression="^([0-7]\d+)$">
<action application="export" data="dialed_extension=$1"/>
<action application="set" data="call_timeout=30"/>
<action application="set" data="record_sample_rate=8000"/>
<action application="export" data="RECORD_STEREO=false"/>
<action application="set" data="hangup_after_bridge=false"/>
<action application="set" data="continue_on_fail=true"/>
<action application="bridge" data="user/${dialed_extension}@${domain_name}"/>
<action application="bridge" data="sofia/external/${dialed_extension}%192.168.1.101"/>
<action application="bridge"
data="${rtmp_contact(default/${dialed_extension}@$${domain})}"/>
<action application="bridge"
data="${rtmp_contact(default/${dialed_extension}@$${domain})}"/>
</condition>
</extension>
```

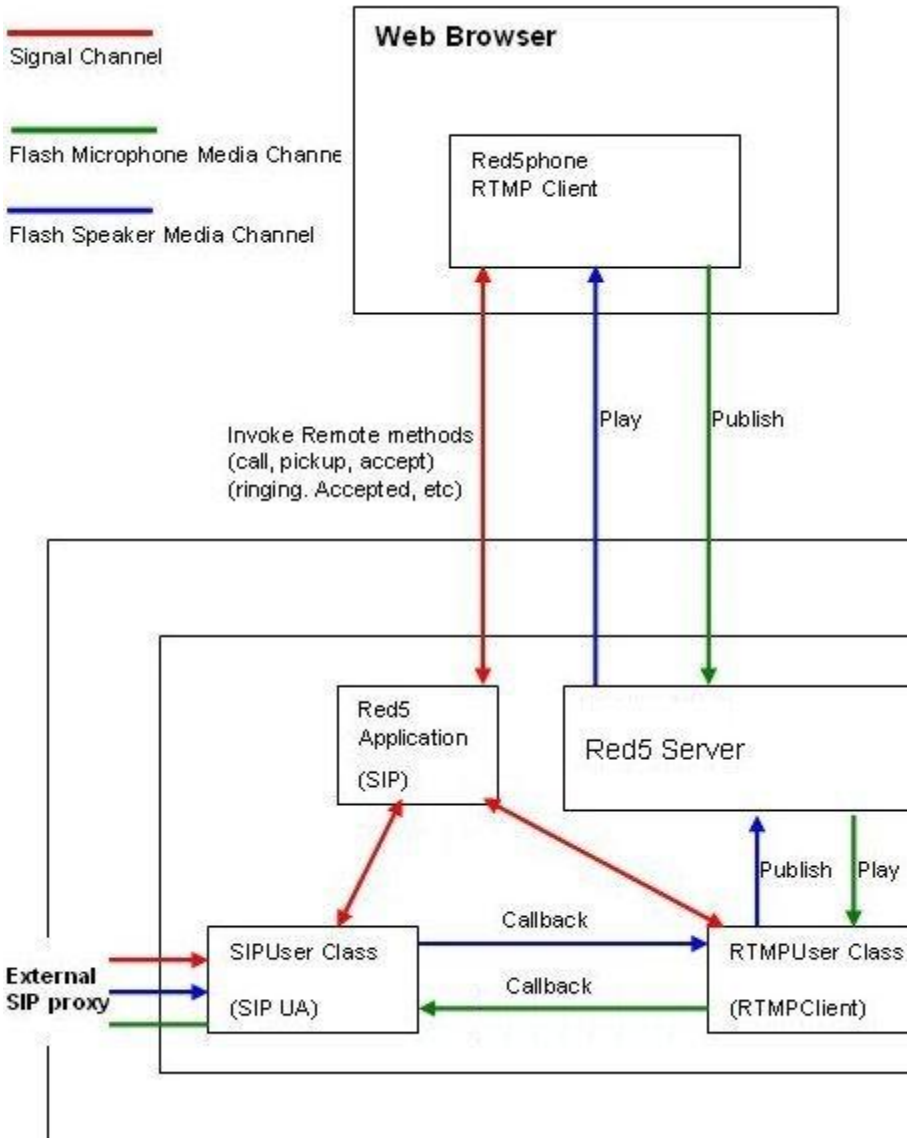
FreeSwitch FlashPhone 部分

64. 什么是 flash phone/SIP?

Flash phone 就是通过网页上的 flash 插件来实现语音媒体通话的功能。作为搞 voip sip 的研发人员，了解这个有一定的必要性。

我们认为 支持 rtmp 协议能进行语音通话的系统都可以叫做 flash phone。

其代表是开源的 red5phone ，它使用 java 基于 red5 媒体服务器上开发的 实现 flash 到 sip 转换网关。网址是：
http://code.google.com/p/red5phone/
red5phone 基本系统结构图如下：



从图中可以大概了解 FLASH PHONE 的服务端的工作原理

65. 为啥需要 flash phone/SIP?

Flash sip 或者 flash phone 的优势是在浏览器里面使用 flash player 插件不需要新安装插件。这对用户是非常方便的。

66. 啥情况下下需要 flash phone?

出现的用途有几种场景：

A. 访问系统的用户是非特定的用户或者游客。他们通过网页上对系统进行访问。使用 activeX 插件的形式，用户会不信任，也不会安装。

因此可以通过 Flash sip 或者 flash phone 来对系统进行语音访问。

因为绝大多数的机器浏览器都安装了 Flash player 插件。

通俗一点描述：用户浏览某个网站，对里面的某个产品有兴趣，可以点产品边上的电话按键马上和客服或者销售人员进行语音沟通。

以前类似的业务是通过 web 800 或者 web call 让用户输入手机号码然后系统呼叫用户的手机跟坐席进行通话来实现的。许多用户不想泄漏手机，不会输入手机号码。

B. B/S 结构的呼叫中心，系统的客服或者坐席是经常变换机器的，他们今天可能在公司上班，明天可能在家里上班。

C. 有些地方的运营商不允许使用 SIP 进行 VOIP 通讯。比如山东聊城 联通的宽带就是禁止 使用 SIP 进行 VOIP 通讯。这个时候通过 Flash phone 作为媒体通讯是合适的。

67. 使用 flash phone 使用什么协议？

flash phone 使用 adobe 公司的 flash player
插件，跟系统使用 RTMP 进行通信
RTMP 是基于 TCP 协议开发的一种媒体协议。
客户端就是 flash player 或者基于 AIR 的应用程序，服务器可以说 FMS，RED5, 和 FreeSwitch 的 MOD_RTMP

68. 使用 flash phone 需要注意哪些问题？

由于 RTMP 是基于 TCP 协议开发的一种媒体协议，
TCP 的协议 网络影响特别大。
因此使用 flash phone 的时候需要注意：

A. 系统的服务端和客户端最好能在一个运营商内使用。跨运营商带宽是不能保证的，延时也会明显加大。在使用 tcp 进行媒体传输通讯的时候影响会显著放大。
B. 假如要公众服务的系统，最好是架设在 BGP 的四线接入的机房内部，才能有效保证 接入的带宽和延时。
C. RTMP 默认是使用 1935 端口。机器上要保证这个端口是没有进程其他占用的。
D. flash phone 测试呼叫 9195 延迟 5 秒的回音测试，是不行的，有 BUG。

69. FreeSwitch 如何增加 RTMP 接口协议模块以实现对 flash phone 的支持？

FreeSwitch 默认是不提供 mod_rtmp 模块
需要自己编译。

编译好 mod_rtmp 模块
之后通过修改 \conf\autoload_configs
目录下的 modules.conf.xml 配置文件：
<!-- <load module="mod_rtmp"/> -->
改为：
<load module="mod_rtmp"/>
然后保存。重启 FreeSwitch 就可以使用。

70. FreeSwitch 的 flash 配置文件是哪个，如何配置 RTMP 的端口？

\conf\autoload_configs 目录下的
rtmp.conf.xml 就是这模块的配置文件。
配置文件里面的：
<param name="bind-address" value="0.0.0.0:1935" />
其中的 1935 就是 rtmp 模块发默认端口。假如 1935 端口被占用，
就要修改这个参数。

71. FreeSwitch 的flash 配置文件如何配置不需要 login 就可以呼叫其他分机？

首先：
修改 \conf\autoload_configs 目录下的 rtmp.conf.xml 配置文件。
配置文件里面的：
<param name="auth-calls" value="true" />
参数 改为：
<param name="auth-calls" value="false" />
然后不用 flash phone 上 login 就可以呼叫。
但是这样是有风险的，我们建议除非是内网使用，否则不要设置为 false。
然后在 flex 代码里面 在 makeCall 呼叫的时候 account 设置为”,flex 代码 makeCall2 函数如下：

```
public function makeCall2(number:String, account:String):int {
        if (netConnection != null)
        {
            hangup(now_uuid);
            if (incomingNetStream == null)
            {
                setupStreams();
```

```
                    }
                    if(loginok)
                        netConnection.call("makeCall", null, number, account);
                    else
                        netConnection.call("makeCall", null, number, '');


                    return 0;
                }
                return -1;
            }
```

注意上面的黑体部分：

```
    if(loginok)
                        netConnection.call("makeCall", null, number, account);
                    else
                        netConnection.call("makeCall", null, number, '');
```

假如登录过，那么使用 account 帐号作为主叫进行呼叫，假如没有登录过，那么使用”帐号作为主叫进行呼叫，这个时候被叫看到的主叫号码是 默认的主叫号码:’0000000000’.没有登录假如使用 netConnection.call("makeCall", null, number, account);进行呼叫是呼叫不通的。

72. FreeSwitch 的 flash phone 使用啥工具进行修改开发？

提供的 支持 rtmp 的 代码在 \clients\flex 目录下
flash phone 是基于 FLEX 开发的目前支持
Adobe Flash Builder 4.6 开发。

73. FreeSwitch 的 flash phone 代码哪些是最有用的？

FreeSwitch 默认提供的代码很庞大，功能很多，尤其是 JS 部分，给初学者带来很大的困惑。
实际上基本的通话功能只要很少的 code，简化之后的 code 只有 300 多行。

整个 demo 可以到 ftp 42.121.124.34 上匿名下载 fs_flex.rar 上去下载。

74. FreeSwitch 的 flash phone 如何呼叫分机？

flash phone 连接 connect 到系统之后，可以呼叫已经注册到 FreeSwitch 上的分机。
呼叫的 方法是 直接呼叫：
sip:分机@FreeSwitch 的 IP

75. FreeSwitch 的 flash phone 如何查哪些机器连接上来？

在 fs cli 界面上输入：
FS>rtmp status profile default sessions

假如都没有用户连接 connect 上来，提示如下：
Profile: default
I/O Backend: tcp
Bind address: 0.0.0.0:1935
Active calls: 0

Sessions:
uuid,address,user,domain,flashVer

假如用户已经 connect 上来,但是没有 login，提示类似如下：
Profile: default
I/O Backend: tcp
Bind address: 0.0.0.0:1935
Active calls: 0
Sessions:
uuid,address,user,domain,flashVer
3e0d3f92-0980-427f-8c49-786f5c6c6f62,192.168.1.102:16671,,,WIN 11,4,402,287

假如用户已经 connect 上来，并且已经 login 提示类似如下：
Profile: default
I/O Backend: tcp
Bind address: 0.0.0.0:1935
Active calls: 0

Sessions:
uuid,address,user,domain,flashVer
3e0d3f92-0980-427f-8c49-786f5c6c6f62,192.168.1.102:16671,1001,192.168.1.102,WIN 11,4,402,287

76. FreeSwitch 的 flash phone 如何查注册上来的分机？

在 fs cli 界面上输入：
FS> rtmp status profile default reg
假如没有用户注册上来（login）提示如下：
Profile: default
I/O Backend: tcp
Bind address: 0.0.0.0:1935
Active calls: 0

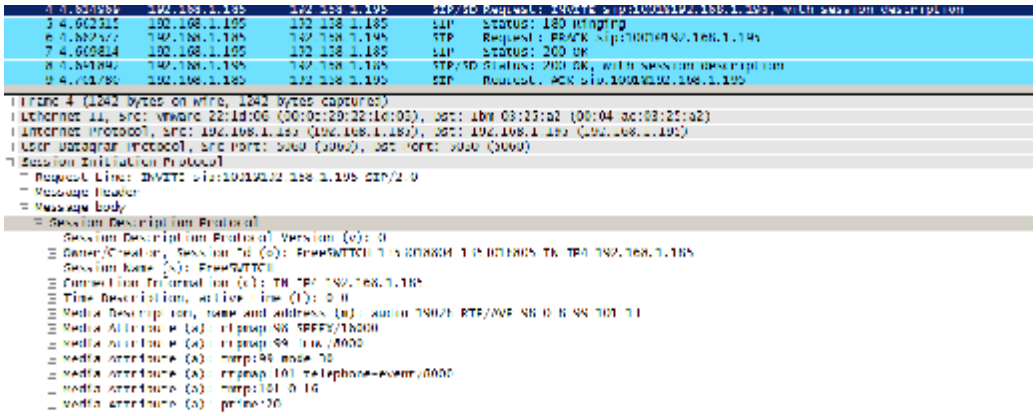
假如有用户注册上来提示如下：
Profile: default
I/O Backend: tcp
Bind address: 0.0.0.0:1935
Active calls: 0

Registrations:
user,nickname,uuid
1002@192.168.1.102,1002@192.168.1.102,6873e7ae-572b-4166-871c-54bc3189acb7
1001@192.168.1.102,1001@192.168.1.102,3e0d3f92-0980-427f-8c49-786f5c6c6f62

其中 1002 和 1001 就是注册上来的分机

77. FreeSwitch 的啥情况下 iLBC 编码不能使用？

本人测试的 1.2.1 1.2.3 1.2.5.3 的版本
由于 FreeSwitch 的某个 bug，导致某些情况下（比如 flash phone 客户端） 通过 iLBC 编码呼叫其他系统，或者其他 软电话分机 的时候，INVITE 里面的 SDP 是错误的。如下图：



实际上 iLBC 编码 的 RTP MAP 应该是 98.

在这里 flash phone 默认的 SPEEX 的 RTP MAP 变成了 98.而

iLBC 编码 的 RTP MAP 变成了 99.从而导致无法跟其他系统（比如毅航的 ISX 系列）集成使用，比如会造成单通。

这个时候需要通过修改源码，重新编译 mod_sofia 模块 替换之后 才能使用。

具体修改的地方是 sofia_glue.c 文件里面的sofia_glue_set_local_sdp函数里面的下面部分：

```
if (!tech_pvt->payload_space) {
        int i;

        tech_pvt->payload_space = 98; //yhy2012-11-16 98是iLBC的RTM MAP，应该改为其他比如97
```

Ver 1.2.3:

mod_rtmp.c

```
//yhy2012-12-23 带宽或者采样rate 由16000改为8000.
    #define SPEEX_BAND_YHY   8000

switch_core_timer_init(&tech_pvt->timer, "soft", 20, (SPEEX_BAND_YHY / (1000 / 20)),
switch_core_session_get_pool(session));

    /* Initialize read & write codecs */
    if (switch_core_codec_init(&tech_pvt->read_codec, /* name */ "SPEEX",
          /* fmtp */ NULL,    /* rate */ SPEEX_BAND_YHY, /* ms */ 20, /* channels */ 1,
        /* flags */ SWITCH_CODEC_FLAG_ENCODE | SWITCH_CODEC_FLAG_DECODE,
        /* codec settings */ NULL, switch_core_session_get_pool(session)) !=
SWITCH_STATUS_SUCCESS) {
        switch_log_printf(SWITCH_CHANNEL_LOG, SWITCH_LOG_ERROR, "Can't initialize read
codec\n");

        return SWITCH_STATUS_FALSE;
    }

    if (switch_core_codec_init(&tech_pvt->write_codec, /* name */ "SPEEX",
```

```
            /* fmtp */ NULL,    /* rate */ SPEEX_BAND_YHY, /* ms */ 20, /* channels */ 1,
            /* flags */ SWITCH_CODEC_FLAG_ENCODE | SWITCH_CODEC_FLAG_DECODE,
            /* codec settings */ NULL, switch_core_session_get_pool(session)) !=
SWITCH_STATUS_SUCCESS) {
            switch_log_printf(SWITCH_CHANNEL_LOG, SWITCH_LOG_ERROR, "Can't initialize write
codec\n");

            return SWITCH_STATUS_FALSE;
        }
```

在修改代码之后 测试

iLBC 和 speex@8000h@20i 不能同时设置到 编码器里面，否则:

eyebeam(软电话的)呼叫会导致 ilbc 的 rtpmat=97 ， speex 的 rtpmap=98

flashphone 的呼叫会导致 speex 的 rtpmap=97，ilbc 的 rtpmat=98 导致不一致的错误

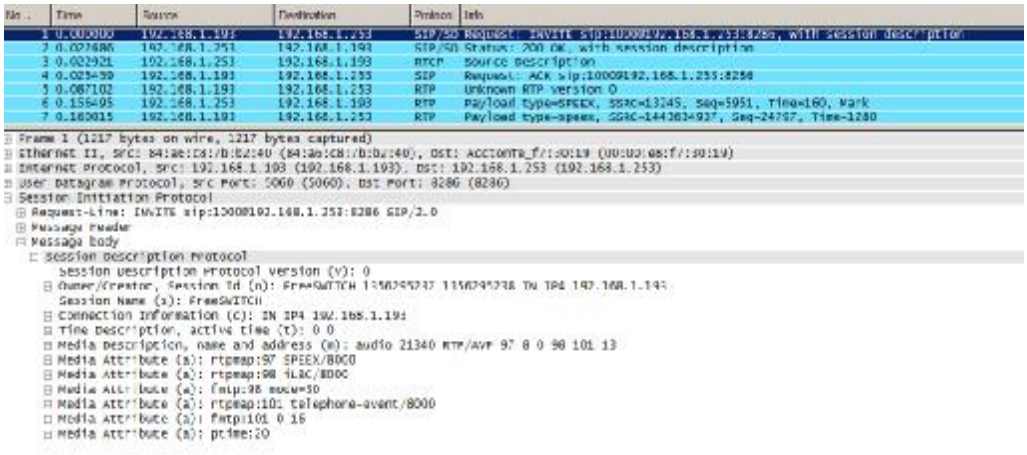
只能 使用 iLBC 或者 SPEEX 里面的一个

在修改代码之后 测试

假如没有设置 speex@8000h@20i ， flash 呼叫内部分机的时候 SDP 不会出现:

speex 的 rtpmap=97

但是假如是 flash 呼叫 外部的没有注册上来的 sip，那么会出现 speex 的 rtpmap=97 的 sdp 如下图： 呼叫 sip:1000@192.168.1.253:8286



78. 软电话分机如何实现对 FreeSwitch 的 flash phone 的呼叫？

flash phone 连接到 FreeSwitch 的 MOD_RTMP 之后。

需要 login 才能被呼叫。

软电话分机要呼叫，需要修改拨号计划:

在 conf\dialplan 目录下的 default.xml 文件里面修改:

在

```
<!-- http://wiki.freeswitch.org/wiki/Dialplan_XML -->
<include>
  <context name="default">
```

这之后

增加：

```
    <extension name="Local_Extension2">
        <condition field="destination_number" expression="^(10[01][0-9])$">
      <action application="export" data="dialed_extension=$1"/>
      <action application="set" data="call_timeout=10"/>
      <action application="set" data="hangup_after_bridge=true"/>
      <action application="set" data="continue_on_fail=true"/>
      <action application="bridge"
data="${rtmp_contact(default/${dialed_extension}@$${domain})}"/>
      <action application="bridge"
data="${rtmp_contact(default/${dialed_extension}@$${domain})}"/>
        </condition>
      </extension>
```

注意两条 bridge 命令，因为测试发现 nod_rtmp 有 bug。导致呼叫到 flash 客户端有时候不成功。continue_on_fail=true 的情况下

万一 bridge 不行，我们就再来一次 bridge！

不行甚至可以考虑再来一次。

79. 如何实现同时支持呼叫分机和 flash client 分机?

类似上面：

```
<include>
  <context name="default">

  <extension name="Local_Extension2">
      <condition field="destination_number" expression="^(10[01][0-9])$">
   <action application="export" data="dialed_extension=$1"/>
   <action application="set" data="call_timeout=10"/>
   <action application="set" data="hangup_after_bridge=true"/>
   <action application="set" data="continue_on_fail=true"/>
   <action application="bridge" data="user/${dialed_extension}@${domain_name}"/>
   <action application="bridge"
```

```
data="${rtmp_contact(default/${dialed_extension}@$${domain})}"/>
 <action                                                    application="bridge"
data="${rtmp_contact(default/${dialed_extension}@$${domain})}"/>
        </condition>
      </extension>
```

注意有 3 条 bridge 命令，第一次是呼叫软电话的分机
要是没有软电话分机注册上，就呼叫 flash client 的分机。

FreeSwitch 高级配置部分

80. 没有注册的 实现对 FreeSwitch 的分机的呼叫？

修改：acl.conf.xml
比如允许 192.168.11.X 网段拨入到软电话或者 flash client，
acl.conf.xml 修改如下，然后没有注册的呼叫注册的
可以直接呼到默认的配置 5080 端口就行

```
<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">
          <!-- domain= is special it scans the domain from the directory to build the ACL -->
          <node type="allow" domain="$${domain}"/>
          <node type="allow" host="192.168.11.0" mask="255.255.255.0"/>
          <!-- use cidr= if you wish to allow ip ranges to this domains acl. -->
          <!-- <node type="allow" cidr="192.168.0.0/24"/> -->
        </list>

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

注意上面的黑体部分。

81. FreeSwitch 为啥会没有发挂机信号给 A leg?

有两种情况会导致这个问题：

A. <action application="set" data="hangup_after_bridge=true"/> 设置成了 false

B. Invite 里面的 from 1005@fs ip 地址发起的会导致被叫挂机,fs 不会转发挂机信号给主叫。因此跟第三方集成的时候，from 的地址要保证是对的。

82. FreeSwitch 如何修改默认配置才能拨打外部的 SIP 电话或者 SIP 网关?

在 conf\sip_profiles\external 目录下

建立：

gw1.xml 内容如下：

```
    <gateway name="gw1">
        <param name="realm" value="192.168.11.103:5060"/>
        <param name="username" value="5678"/>
        <param name="password" value="1234"/>
        <param name="register" value="false" />
</gateway>
```

其中 ip 地址和端口是外拨网关的 ip 和 sip 的端口

然后 在 conf\dialplan\default 建立一个 call_out.xml

内容如下：

```
<include>
  <extension name="call out">
    <condition field="destination_number" expression="^9(\d+)$">
      <action application="bridge" data="sofia/gateway/gw1/$1"/>
    </condition>
  </extension>
</include>
```

表示拨打 9 开头的号码都走外拨网关 gw1 的路由

83. 外部的 SIP 网关如何拨打到某个分机？

在 conf/dialplan/public 目录下建立 my_did.xml 文件内容如下：

```
<include>
  <extension name="public_did">
    <condition field="destination_number" expression="^(你的接入号码)$">
      <action application="transfer" data="1000 XML default"/>
    </condition>
  </extension>
</include>
```

之后拨打你的接入号码 就会到分机上。

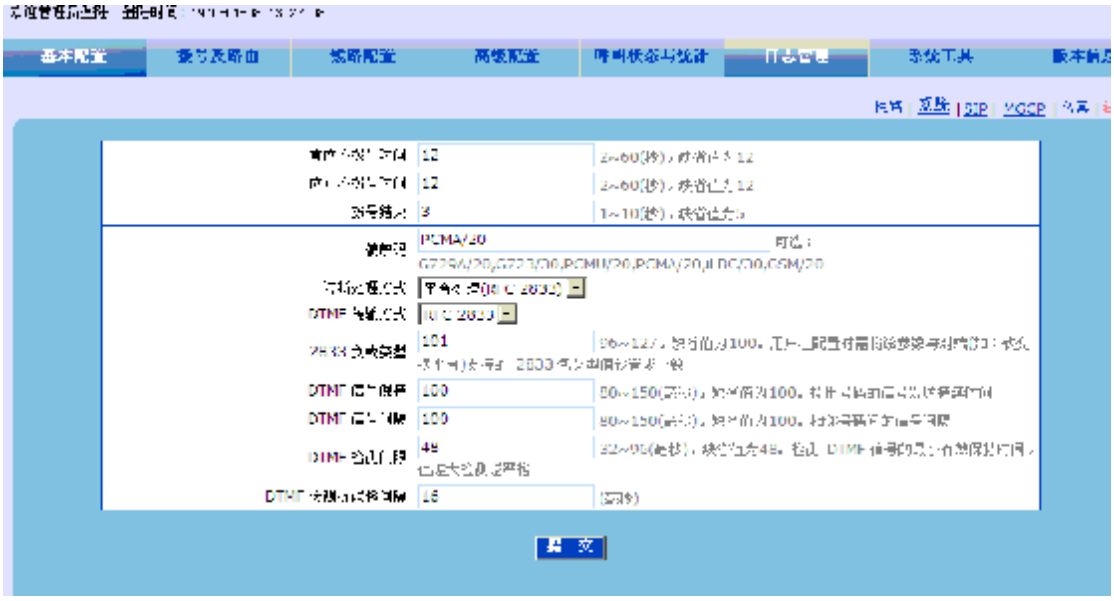
84. FreeSwitch 如何和讯时网关 MX8 进行集成？

假设：
FreeSwitch 的 ip 是 192.168.1.236
mx8 的 ip 是 192.168.1.251

首先：解决拨入的问题：配置 MX8，使他以分机的形式注册到 FreeSwitch 上。有用户拨入的时候指向 外联模式设置的接入号码。从而进入 ivr 系统。
主要配置界面如下：

Mx8 ip 配置界面：

MX8 系统配置界面：



MX8 SIP 配置界面：

192.168.1.236 是 FS 的 ip

注册用户名是 1005 默认密码是 1234

MX8 拨出路由配置界面：

默认所以的 ip 拨入都从 PSTN 第一线 拨出：

MX8 线路号码配置界面：

MX8 线路绑定号码配置界面：
其中的 12396 是 ivr 系统的接入号码，
假如线路有开反极性检测 可以勾选反极性检测

其次解决拨出的问题，通常 mx8 是作为网关出口。因此创建
conf\sip_profiles\external\gw1.xml
内容如下：

```
<gateway name="gw1">
        <param name="realm" value="192.168.1.251:5060"/>
        <param name="username" value="5678"/>
        <param name="password" value="1234"/>
        <param name="register" value="false" />
</gateway>
```

然后在需要拨打外线的时候，
分为两种情况
1） 分机上外拨：
从某一分机上呼出：打外部电话就需要加先拨 9
修改拨号计划，创建新 XML 文件：
conf/dialplan/default/call_out.xml :

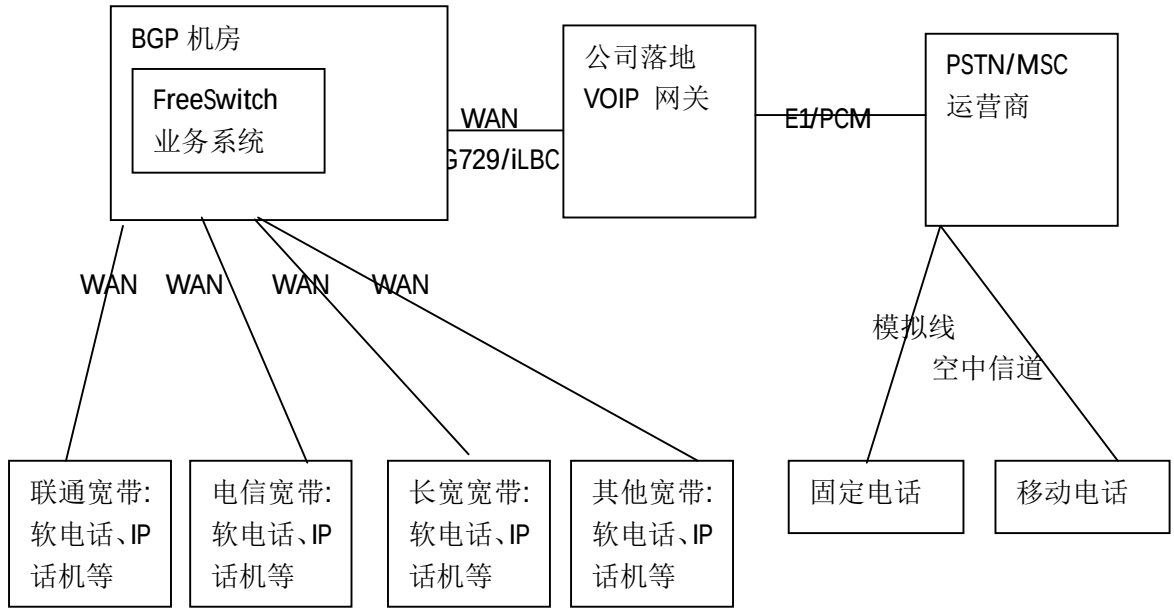
```
<include>
    <extension name="call out">
    <condition field="destination_number" expression="^9(\d+)$">
        <action application="bridge" data="sofia/gateway/gw1/$1"/>
        </condition>
    </extension>
</include>
```

2） 假如 是 esl 编程,ivr 外拨 可以判断被叫号码，比如 被叫号码超过 4 位的时候
sprintf(cmd_tmp,"bgapi originate {origination_uuid=%s,originate_timeout=60}sofia/gateway/gw1/%s %s\n\n",uuid,called,system_caller);
res=esl_send_recv(&handle,cmd_tmp);

85. FreeSwitch 公网运营如何设计？

FreeSwitch 运行在公网，意味着

A. 假如是对公众提供服务，软电话分机分布在全国各地，各个运营商都可能存在，因此各个运营商的接入带宽延时是要考虑的。我们建议部署在 BGP 4 线机房
B. 意味这软电话在公司内网，需要 NAT 穿透才能使用，因此需要去掉 –nonat 启动
C. FreeSwitch 公网运行的一种建议方案如下图：



86. FreeSwitch 公网服务在防火墙之后如何部署？

部署在防火墙之后意味着 FS 运行的机器上是没有公网 IP 的，只有内网 IP。公网 IP 是通过 IP 映射或者端口映射到 fs 的机器上。

理论上启动 fs 的时候不带任何参数自动支持 NAT，但是本人测试这个方法不靠谱，尤其麻烦的是不稳定，有点象段誉公子的六脉神剑，经常不灵光。

从而导致一方单通，甚至双方都没有声音的情况。

比较保险的做法是修改以下的配置文件：

internal.xml 里面的：

```
<param name="ext-rtp-ip" value="auto-nat"/>
<param name="ext-sip-ip" value="auto-nat"/>
```

直接修改为公网地址，比如 218.107.217.204 是外网地址，修改如下：

```
<param name="ext-rtp-ip" value="218.107.217.204"/>
<param name="ext-sip-ip" value="218.107.217.204"/>
```

对应的 external.xml 配置里面也有这个参数可以根据需求进行修改。

假如是端口映射，那么需要在路由器上映射下面这些端口到 fs 的机器上：

5060	UDP & TCP	SIP UAS	Used for SIP signaling (Standard SIP Port, for default Internal Profile)
5080	UDP & TCP	SIP UAS	Used for SIP signaling (For default "External" Profile)

16384-32768	UDP	RTP/ RTCP multimedia streaming	语音、视频数据传输用端口

87. FreeSwitch 公网运营有哪些需要特别考虑的？

除了考虑实网部署的问题之外
公网运营考虑的问题 还有带宽核算和
安全性，
带宽的问题主要是编码器 的使用
G729 默认是没有转码的
因此考虑 iLBC 编码是合适的

88. FreeSwitch 公网运营环境下哪些情况下测试过？

联通 WCDMA 3G 数据通道。
电信 CDMA 2000 3G 数据通道。
电信宽带用户。
联通宽带用户。
长宽宽带用户。
移动（铁通）宽带用户。
客户软电话在公司内部通过 WIFI 上网的环境测试

89. FreeSwitch 如何禁止 IP 地址发生改变后，自动重启 sofia 模块？

修改文件 sofia.conf.xml
位置：/usr/local/freeswitch/conf/autoload_configs/sofia.conf.xml
修改内容：
<param name="auto-restart" value="false"/>
该属性设置的目的是防止 FS 在检测到 IP 地址发生改变后，自动重启 sofia 模块。

90. FreeSwitch 公网运营部署如何配置用户帐号密码？

默认用户的认证密码和语音邮箱登陆密码都异常简单（默认为 1234）。修改密码也可以修改配置文件，但是终归不方便 尤其是存在多个 fs 的情况下 .

具体方法可以参考以下建议：

1、 删除默认的静态 XML 配置，通过 mod_xml_curl 模块使用后台数据库中动态的数据。

2、 手动修改配置文件中的用户名和密码。

3、 通过运行 scripts/perl/randomize-passwords.pl 修改。

4、 通过 lua 脚本实现。

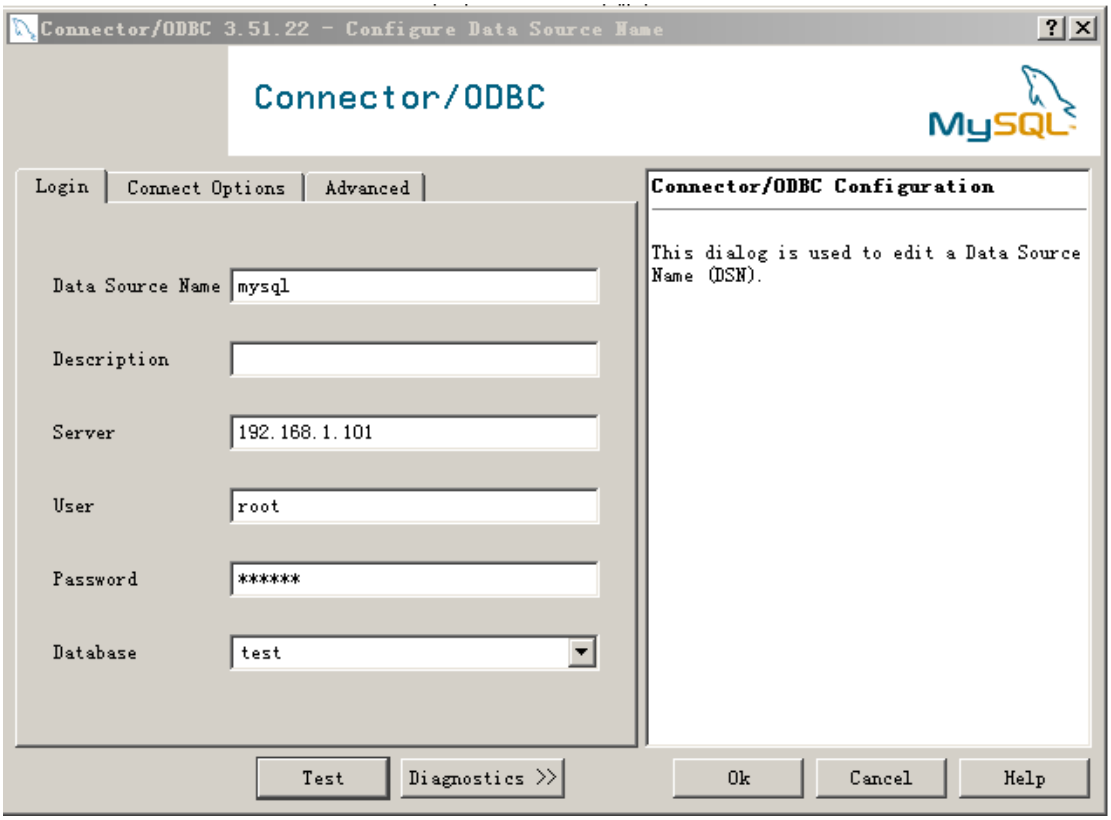
其中 4、 通过 lua 脚本实现 比较简单，以帐号数据存放在 mysql 数据库为例，过程说明如下：

1）在 mysql 数据库中建立如下表：

```
CREATE TABLE      userinfo(
   `id` int(11) NOT NULL auto_increment,
     username  varchar(64) NOT NULL default '',
     password        varchar(64) NOT NULL default '',
     updatedate       timestamp NOT NULL default CURRENT_TIMESTAMP ,
   PRIMARY KEY    (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=gbk;
```

其中 username 就是分机号码，password 就是分机对应的密码

2）创建 mysql ODBC 数据源，以 windows 版本为例内容如下：



其中的 mysql 是数据源名称，下面要用到。

3）修改 conf\autoload_configs\lua.conf.xml 文件，结果如下：

```
<configuration name="lua.conf" description="LUA Configuration">
  <settings>
  <param name="xml-handler-script" value="gen_dir_user_xml.lua"/>
  <param name="xml-handler-bindings" value="directory"/>
  </settings>
</configuration>
```

4）创建 gen_dir_user_xml.lua 文件并且存到 \scripts 目录下，gen_dir_user_xml.lua 内容如下：

```
local req_domain = params:getHeader("domain")
local req_key    = params:getHeader("key")
local req_user   = params:getHeader("user")

local dbh = freeswitch.Dbh("mysql","root","123456")
if dbh:connected() == false then
  freeswitch.consoleLog("notice", "gen_dir_user_xml.lua cannot connect to database" ..
dsn .. "\n")
  return
end
local my_query = string.format("select password from userinfo where username='%s' limit
1", req_user)
assert (dbh:query(my_query, function(u) -- there will be only 0 or 1 iteration (limit 1)
  XML_STRING =
[[<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

```
<document type="freeswitch/xml">
  <section name="directory">
    <domain name="]] .. req_domain .. [[">
      <user id="]] .. req_user    .. [[">
        <params>
          <param name="password" value="]] .. u.password .. [["/>
           <param                                                    name="dial-string"
value="{sip_invite_domain=${dialed_domain},presence_id=${dialed_user}@${dialed_do
main}}${sofia_contact(${dialed_user}@${dialed_domain})}"/>
        </params>
        <variables>
          <variable name="user_context" value="default"/>
        </variables>
      </user>
    </domain>
  </section>
</document>]]
end))
```

其中"mysql" 是数据源名称，"root" 是帐号，"123456" 是 mysql 的 root 帐号的密码。

91. FreeSwitch 如何配置帐号密码在 oracle 数据库里面？

上面的问题解决了用户的帐号和密码保存在 mysql 数据库里面的问题。Lua 脚本通过 odbc 去取进行认证。

但是假如帐号和密码保存在 oracle 数据库里面的，那么 fs 由于代码的 bug 就不能使用。

修改修改两个地方：

1. 修改 switch_odbc.c 把里面原来下面的代码：

   strcpy((char *) sql, "select 1 ");

   改为：

   strcpy((char *) sql, "select 1 from dual"); //yhy2013-03-07 to support oracle database

   然后重新编译 FS。这样的修改可以同时支持 mysql 和 oacle，不过这样就不能支持 sqlserver 了。

   “select 1” 的语法是 支持 mysql 和 sqlserver。

   “select 1 from dual” 的语法是 支持 mysql 和 oracle。

   这个 sql 语法是数据库心跳的握手语法。

2. 上一个问题的支持 mysql 面的 lua 的脚本也是不行的，要修改为下面的脚本：

```
-- gen_dir_user_xml.lua
-- example script for generating user directory XML
-- comment the following line for production:
freeswitch.consoleLog("notice", "Debug from gen_dir_user_xml.lua, provided params:\n" .. params:serialize() .. "\n")

local req_domain = params:getHeader("domain")
local req_key    = params:getHeader("key")
local req_user   = params:getHeader("user")

local dbh = freeswitch.Dbh("test","system","test")
if dbh:connected() == false then
   freeswitch.consoleLog("notice", "gen_dir_user_xml.lua cannot connect to database" .. dsn .. "\n")
   return
end

-- it's probably wise to sanitize input to avoid SQL injections !
local my_query = string.format("select fsname,fspassword from fsuserinfo where fsname='%s' ", req_user)

freeswitch.consoleLog("notice", "sql=" .. my_query .. "\n")
local my_pass =" "
local my_query = "select password from userinfo where username = " .. req_user

-- set variable - or print to console if no session is available
local function sv(key, val)
            print(key .. " : " .. val)
  my_pass=val;
end

assert(dbh:query(my_query, function(row)
      for key, val in pairs(row) do       -- in this example only one row with one column will be returned
          sv(key, val)                    -- so here key = 'user'
      end
end))

 XML_STRING =
[[<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<document type="freeswitch/xml">
   <section name="directory">
      <domain name="]] .. req_domain .. [[">
         <user id="]] .. req_user    .. [[">
```

```
            <params>
              <param name="password" value="]] .. my_pass .. [["/>
               <param                                                    name="dial-string"
value="{sip_invite_domain=${dialed_domain},presence_id=${dialed_user}@${dialed_do
main}}${sofia_contact(${dialed_user}@${dialed_domain})}"/>
            </params>
            <variables>

             <variable name="user_context" value="default"/>
           <variable name="transfer_fallback_extension" value="operator"/>
             <variable name="toll_allow" value="domestic,international,local"/>
           <param name="accountcode" value="]] .. req_user .. [["/>
            <param name="effective_caller_id_number" value="]] .. req_user .. [["/>
             <param name="effective_caller_id_name" value="]] .. req_user .. [["/>>
            </variables>
          </user>
        </domain>
    </section>
</document>]]

    -- comment the following line for production:
    freeswitch.consoleLog("notice", "Debug from gen_dir_user_xml.lua, generated
XML:\n" .. XML_STRING .. "\n")
```

92. FreeSwitch 的 SDP 有啥缺陷？

Alaw 和 ulaw 没有 m=rtpmap 0/8000 之类 的详细描述
导致呼叫到 kapanga 上，kapanga 接通就挂机。
没有 rtpmat 描述 导致呼叫到 red5 phone 等不行

93. FreeSwitch 的 SDP 有啥特殊的？

跟其他的软电话之类的对比 多了 rtpmap=13 静音的说明。FreeSwitch 如何做集群？
多个 FS 服务器如何做集群有很多方法，下面这个是基于 ACL 的方法：
假设 你有 FSA (10.0.0.1) 分机 1000-1099
和 FSB (10.10.0.1) 分机 1100-1199.
首先我们要修改 autoload_configs/acl.conf.xml
FSA 上在 <list name="domains" ...>上增加：
<node type="allow" cidr="10.10.0.1/32"/>

FSB 上在 <list name="domains" ...>上增加：

```
<node type="allow" cidr="10.0.0.1/32"/>
```

然后

Now the boxes are allowed to talk to each other, they still don't know about the extensions each FreeSWITCH is serving. Therefore you need to tell FreeSWITCH where to send which calls. This is called "routing" and is performed by the bridge application using the internal sofia profile (sip_profiles/internal.xml). Despite the name internal, it will handle calls not for the local switch just fine.

在 FSA 上的 conf/dialplan/default.xml 里面增加：

```
<extension name="Dial to FSB">
  <condition field="destination_number" expression="^(11\d\d)$">
    <action application="bridge" data="sofia/internal/$1@10.10.0.1"/>
  </condition>
</extension>
```

表示 11XX 的分机会到 FSB 的 IP 上

在 FSB 上的 conf/dialplan/default.xml 里面增加：

```
<extension name="Dial to FSA">
  <condition field="destination_number" expression="^(10\d\d)$">
    <action application="bridge" data="sofia/internal/$1@10.0.0.1"/>
  </condition>
</extension>
```

表示 11XX 的分机会到 FSB 的 IP 上

最后

在 FSA 的 conf/dialplan/public.xml 里面增加

```
<extension name="Calls fromFSB">
  <condition field="destination_number" expression="^(10\d\d)$">
    <action application="transfer" data="$1 XML default"/>
  </condition>
</extension>
```

这个表示把 FSB 打过来的来电分机号码转到 XML default 从而进入正常的拨号计划

在 FSB 的 conf/dialplan/public.xml 里面增加

```
<extension name="Calls from FSA">
  <condition field="destination_number" expression="^(11\d\d)$">
    <action application="transfer" data="$1 XML default"/>
  </condition>
</extension>
```

这个表示把 FSA 打过来的来电分机号码转到 XML default 从而进入正常的拨号计划

94. FreeSwitch 对内存 cpu 需求如何？

基本的机器就可以，目前主流的双 四核的 cpu，8G 内存配置的 pc Server 机器，可以支持 300 先线并发带转码。
在 P950 CPU 2G 内存配置的笔记本上测试可以支持 30 路的转码（G.711--SPEEX）。
假如内存不够，会导致 FreeSwitch 崩溃。
在 3612 QM CPU4G 内存配置的笔记本上测试可以支持 200 路的 IVR。

FreeSwitch ESL 编程部分

95. FreeSwitch ESL 编程能做啥用？

本人的理解假如没有 ESL 编程控制（或者其他的脚本编程控制比如 Erlang），FreeSwitch 仅仅只会是一个 SIP Server。有了 ESL 编程的控制，FreeSwitch 的功能可以扩展到整个呼叫中心的复杂应用。ESL 编程的控制可以完成最基本的 IVR 系统的全部功能，包括但不限于如下 10 个 IVR 基本功能内容：

1） 接受用户拨入，应答呼叫
2） 系统拨出到用户，
3） 挂断用户呼叫
4） 连接用户 a 和 b 进行媒体通话
5） 断开用户 a 和 b 的媒体通话，但是用户 a 和 b 的信令都没有断开
6） 播音（同时可以获取用户按键）
7） 录音（同时可以获取用户按键）
8） 参加会议
9） 会场控制，设置某个人静音或者允许某个人说话
10） 会场录音的开启和停止。

下面的 ESL 编程控制内容就是针对上面这些基本的 IVR 应用进行描述的。
由于 ESL 原生使用 C/C++语言，本人使用 C/C++进行 ESL 的 IVR 编写。
使用 C/C++的好处是当 ESL 有 bug 崩溃的时候可以直接 debug，从而定位问题。

96. FreeSwitch 默认播音和录音目录在哪里?

播放文件的默认目录 ：
\sounds\en\us\callie
录音文件的默认目录:
\sounds\en\us\callie

97. FreeSwitch 如何指定录音文件和目录？

可以在 record 的时候写上绝对路径
比如 d:/record/1.wav

98. FreeSwitch 如何指定 alaw 播音文件的格式?

FreeSwitch 的播音除 wav 文件之外，其他是根据文件扩展名来区分格式的。
国内通常的 ivr 系统都是使用单声道的 8000Hz alaw 编码的语音格式
要做到然 FreeSwitch 能支持这种格式的语音有两种办法：
A. 文件名称的扩展名叫做 .alaw
B. 使用工具给这些文件加上 wav 头，然后 扩展名叫做 .wav
假如 是其他格式的语音类似参考修改。
具体可以在 fs_cli 下使用 show files 命令查看。然后使用合适的语音文件扩展名

99. FreeSwitch 如何指定 alaw 录音文件的格式?

国内通常的 ivr 系统都是使用单声道的 8000Hz alaw 编码的语音格式
要做到能录制到这种格式的语音：
需要以下几个步骤：

先在拨号计划里面加上：

```
<action application="set" data="record_sample_rate=8000"/>
        <action application="export" data="RECORD_STEREO=false"/>
```

来指定是 8000HZ 和单声道。默认是立体声的。

比如：

```
<extension name="socket">
        <condition field="destination_number" expression="^12396$">
        <action application="set" data="record_sample_rate=8000"/>
        <action application="export" data="RECORD_STEREO=false"/>
        <action application="socket" data="127.0.0.1:8084 async full"/>
        </condition>
    </extension>
```

然后：
在 uuid_record 或者 record 的时候
录音的时候指定是 .alaw 的扩展名，这样就指定是 alaw 编码的语音。
假如是.wav 扩展名。则是线性 16bit 没有编码的语音。

100. FreeSwitch 录音文件，回放时发现声音很小，如何解决？

录音生成的 WAV 文件，和实际录音对比声音只是小一点点。
使用 cooledit 播放声音是几乎正常的。
回放录音文件时发现声音很小，可以通过设置增益来实现:

```
uuid_audio <uuid> [start [read|write] [mute|level <level>]|stop]
```

level 参数从 -4 to 4, 0 是默认声音大小. 4 是最大

101. FreeSwitch 如何实现支持视频的录制和播放？

FreeSwtich 通过 模块 fsv 支持视频的录制和播放，此模块提供两个 app,record_fsv 或者 record 和 play_fsv, 一个录像，一个播放，其实现在目录 src\mod\applications\mod_fsv/mod_fsv.c

使用方法：
录像：
dialplan 中 调用 app record_fsv 或者 record
<action application="record_fsv" data="file.fsv"/>
参数为录音文件名。
播放视频：
dialplan 中调用 app play_fsv
<action application="play_fsv" data="file.fsv"/>
参数同样为 文件名

在 api 里面 使用：

```
uuid_record <uuid> start file.fsv
```

uuid_record also can record video, but at this time it only record one leg of audio and video：只能录制一个 leg 的视频。

会场里面的录制：recording in conference is also possible with ：

```
conference 3000 record file.fsv
```

说明： fsv 表示是"FreeSWITCH Video"的视频，里面的格式是 线性 pcm 的语音和视频编码的原始 rtp 包的组合，文件头格式如下：

```
struct file_header {
        int32_t version;
        char video_codec_name[32];
        char video_fmtp[128];
```

```
        uint32_t audio_rate;
        uint32_t audio_ptime;
        switch_time_t created;
};
```

The frames are stored in the order received. A frame header, presumably 4 bytes in length (but stored as an "int", in native endian), is used; the top bit is one if frame is video, zero if audio; the 31 lower bits are used to store the frame length.帧按顺序保存，开始 4byte 是帧大小。假如第一个 bit 是 0 表示此帧是语音，假如是 1 表示此帧是视频。 其他 31bit 才是真正的帧长度

Try call 9193 and 9194 shipped with the default dialplan with a video phone.

102. FreeSwitch ESL LIB 例子有哪些？

API 开发 demo

source ： libs/esl 目录下

testserver.c 外联模式例子

testclient.c 内联模式例子

fs_cli.c fs client 的 code

103. FreeSwitch ESL LIB 例子 windows 下如何方便建立 VS 工程？

以 testserver.c 为例子建立 VS 工程

Windows 的 ESL LIB 例子 demo ,有一个已经存在的工程：

\libs\esl 目录下 fs_cli.2010.vcxproj

我们可以在这个工程的基础上很快建立 例子的工程

使用 Vs2010 打开这个过程

然后把 testserver.c

文件里面的内容拷贝覆盖 fs_cli.c 里面的内容

然后编译，保存就可以了。

104. FreeSwitch ESL LIB 例子 windows 下使用有哪些要注意？

Windows 下使用 Socket 的程序在使用 Socket 之前必须调用 WSAStartup 函数

以 testserver.c 例子为例：在 main()最开始的地方的

初始化前要增加下面这些代码：

```
WSADATA Data;
 if( WSAStartup(MAKEWORD(2, 1), &Data) != 0)
 {
        return false;
}
```

否则 testserver.c 启动会马上退出。
如下面：

```
int main(void)
{
 WSADATA Data;
 if( WSAStartup(MAKEWORD(2, 1), &Data) != 0)
 {
        return 0;
 }

 esl_global_set_default_logger(7);
 esl_listen_threaded("localhost", 8084, mycallback, 100000);

 return 0;
}
```

mycallback 的函数，每个呼叫会多一个线程。
因此 mycallback 里面是多线程的。
除非特别说明，以后的函数代码除了内联模式的外拨线程之外
全部都是在 mycallback 函数里面调用执行的。

105. FreeSwitch 什么时候需要用到内联模式编程？

内联模式在有 ivr 系统外拨（originate）到分机或者其他电话的时候用到。
内联模式初始化如下：

```
int main(void)
{
 WSADATA Data;
 if( WSAStartup(MAKEWORD(2, 1), &Data) != 0)
 {
        return 0;
 }
 esl_global_set_default_logger(6);

 int res=esl_connect(&ghandle, "127.0.0.1", 8021, NULL, "ClueCon");
```

```
if(res!=0)
{
    disp_msg("System Start esl_connect res=%d fail",res);
    return 0;
}
。。。。。
```

ghandle 是全局的 ，将来通过 ghandle 可以 产生 uuid，从而进行外拨呼叫

106. FreeSwitch 内联模式如何实现 IVR 全业务外拨用户功能？

FreeSwitch 的 bridge 是连接有一个用户 桥接另外一个用户。不能用来实现 ivr 的外拨功能。比如外拨一个用户让播放提示语音，让他参加会议。

要实现一个 ivr 全业务的外拨功能意味着：外拨接通之后的被叫用户可以进入 ivr，可以和其他用户通话，或者参加会议。还要求外拨以后可以随时挂掉正在外拨的呼叫。

要实现以上功能需要如下步骤：

1）首先，使用下面的函数获取一个新的 UUID

```
int get_uuid(char*uuid)
{
    uuid[0]=0;
    esl_send_recv(&ghandle, "api create_uuid\n\n");
        if (ghandle.last_sr_event && ghandle.last_sr_event->body)
    {
        disp_msg("get_uuid:[%s]\n", ghandle.last_sr_event->body);
        strcpy(uuid,ghandle.last_sr_event->body);
        return 0;
    }
    return 1;
}
```

ghandle 是全局的

这个 uuid 将来用来外拨，或者用来挂掉正在外拨的呼叫

2）创建一个专门负责呼叫的线程：CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)CallOutWorkThreadRealTime,(LPVOID)(index),0, NULL);

该线程新建立一个内联模式的连接到 fs，然后 根据被叫号码的长度判断是内部分机呼叫 还是外拨到网关。

主要代码如下：

```
int CallOutWorkThreadRealTime(void* arg)
{
    。。。。
    res=esl_connect(&handle, "127.0.0.1", 8021, NULL, "ClueCon");
```

```
        if(res!=0)
        {
            disp_msg("esl_connect esl_connect res=%d fail",res);
            return 0;
        }

        if(strlen(called)>4)
            sprintf(cmd_tmp,"bgapi originate {origination_uuid=%s,originate_timeout=60}sofia/gateway/gw1/%s %s\n\n",uuid,called,system_caller);
        else
            sprintf(cmd_tmp,"bgapi originate {origination_uuid=%s,originate_timeout=30}sofia/internal/%s%c%s %s\n\n",uuid,called,'%',fsserverip,system_caller);
        res=esl_send_recv(&handle,cmd_tmp);
        disp_msg("CallOutThread esl_send_recv:[%s],res=%d",cmd_tmp,res);

        if (handle.last_sr_event && handle.last_sr_event->body)
        {
            disp_msg("[%s],index=%d", handle.last_sr_event->body,index);
        }
        else
        {
            disp_msg("[%s],index=%d,last_sr_reply", handle.last_sr_reply,index);
        }
        disp_msg("CallOutThread uuid=[%s]",uuid);
        esl_filter(&handle, "unique-id", uuid);
        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");
 。。。。。
```

说明：originate_timeout=60 表示外拨超时时间，被叫超过这个时间没有应答，ivr 系统就收到 CHANNEL_HANGUP。

3）然后使用 check_makecall_event()和 check_makecall_body()函数监听 下面 3 个事件：

"CHANNEL_HANGUP" 用户没有接，挂机了

"CHANNEL_PROGRESS" 用户开始振铃，这个事件不是一定有的，假如设置自动应答，可能就没有

"CHANNEL_ANSWER"用户开始应答

获取这几个事件是创建一个专门负责呼叫的线程的最核心的功能，假如不需要获取上面

的 3 个事件，就不需要创建一个专门负责呼叫的线程，可以直接使用全局的 ghandle 来执行“bgapi originate...”实现外拨，被叫用户接通之后也会到外联的 callback 函数里面。但是完备的 ivr 系统，在外拨的过程中，需要获取上面的这些事件进行控制。

check_makecall_event()和 check_makecall_body()函数如下：

```
int check_makecall_body(const char*eventbody,char*dtmfbuf,int maxdtmf,char*
endchar,int *channel_execute)
{
 char tmp[128];
 unsigned int i,len;

 if (eventbody==NULL) return 0;
 len=strlen(eventbody);
 if(len>64) len=64;
 strncpy(tmp,eventbody+strlen("Event-Name: "),len);
 tmp[len]=0;
 for(i=0;i<strlen(tmp);i++) if(tmp[i]=='\n' || tmp[i]=='\r') {tmp[i]=0;break;}
 disp_msg("Event-Name:[%s]", tmp);
 if(strcmp(tmp,"CHANNEL_EXECUTE")==0) *channel_execute=1;
 if(strcmp(tmp,"CHANNEL_HANGUP")==0) return -98;
 if(strcmp(tmp,"CHANNEL_PROGRESS")==0) return EV_ALERTING;
 if(strcmp(tmp,"CHANNEL_ANSWER")==0)  return EV_CONNECTED;
 return 0;
}
int check_makecall_event(esl_handle_t *handle,int timer)
{
 int done = 0,c=0;
 esl_status_t status;
 time_t exp = 0;
 char dtmf[128];
 int res=0;
 int  channel_execute=0;
 dtmf[0]=0;
 time_t time1=time(NULL);

 while((status = esl_recv_timed(handle, 1000)) != ESL_FAIL)
 {
  c++;
  if(timer>0 && (time(NULL)-time1>=timer)){res=100;break;}
  if (status == ESL_SUCCESS)
  {
   const char *type = esl_event_get_header(handle->last_event,
"content-type");
```

```
                if (type)
                {
                    if(strcasecmp(type, "text/disconnect-notice")==0)
                    {
                        const char *dispo = esl_event_get_header(handle->last_event,
"content-disposition");
                        disp_msg("Got a disconnection notice dispostion: [%s]", dispo ?
dispo : "");
                        if (dispo && strcmp(dispo, "linger")==0)
                        {
                            res=-99;
                            break;
                        }
                    }
                    if(strcasecmp(type, "text/event-plain")==0)
                    {
                        const char *eventbody=esl_event_get_body(handle->last_event);

 if((res=check_makecall_body(eventbody,dtmf,1,"#",&channel_execute)))
                        {
                            disp_msg("check_makecall_body res=%d.",res);
                            if(res)     break;
                        }
                    }

                }
            }
        }
        disp_msg("check_makecall_event res=%d",res);
        return res;
}
```

4）最后，特别说明：

假如用户挂机本线程结束。

假如用户接通本线程也结束。

假如用户接通，FreeSwitch 将自动马上呼叫 system_caller，通过外联模式的 callback 进入到处理的 ivr 函数里面。

这样就统一了拨入和拨出的 ivr 处理。将拨出当作拨入一起处理。处理 ivr 的全业务。（为啥这样做，是因为一来这样统一处理比较方便，二来本人测试，直接在外拨的线程里面处理 ivr 的业务，会有莫名其妙的情况发生。）

在外联模式 callback 函数中，可以通过下面的办法来知道是 用户拨入还是系统拨出：

```
 int dir=1;//call in
 if(esl_event_get_header(handle.info_event, "call-direction"))
 {
 disp_msg("dir:%s",esl_event_get_header(handle.info_event, "call-direction"));
```

```
if(strcmp(esl_event_get_header(handle.info_event,"call-direction"),"outbound")==0) dir=0;
}
```

107. FreeSwitch 如何设置支持其他机器内联模式到 FS？

修改 conf\autoload_configs\event_socket.conf.xml 配置文件：
将里面的 <param name="listen-ip" value="127.0.0.1"/> 改为
<param name="listen-ip" value="0.0.0.0"/>
这样其他机器上的 esl 客户端可以内联到 FS 的 ip 上。而不仅仅是本机的 127.0.0.1 的 ip。

另外将来的代码 esl_connect(&ghandle, fsip, 8021, NULL, "ClueCon");
fsip 不能是127.0.0.1 因为是fs的网卡上的ip。

108. FreeSwitch 如何设置支持 SOCKET EVENT API 外联模式编程？

在 \conf\dialplan\default.xml 配置文件里面加上：

```
<extension name="socket">
        <condition field="destination_number" expression="^12396$">
        <action application="socket" data="127.0.0.1:8084 async full"/>
        </condition>
    </extension>
```

其中的 12396 表示接入号码。也就是上面内联模式里面的 system_caller。
之后，拨打 12396 的号码就会到本地的 8084 端口的服务流程（用户自己控制的 ivr）上
说明：
假如只有用户拨入到 ivr 系统，没有 ivr 系统拨出到用户的情况，使用外联模式一种就足够了，不需要使用 内联模式。

109. FreeSwitch 如何支持外联到其他机器的 ESL 客户端？

修改上面拨号计划里面的

```
<action application="socket" data="127.0.0.1:8084 async full"/>
```

类似 内线模式的 ip，上面的 127.0.0.1 ip 可以设置为其他机器的 ip，那么 ESL 的客户端就可以运行在其他机器上。当有电路来的时候，fs 会外联到这个指定的 ip 上。
另外在代码里面esl 的客户端侦听的ip要设置为如下：
esl_listen_threaded(“0.0.0.0”, 8084, mycallback, 100000);

110. FreeSwitch ESL 外联模式 同步和异步模式有啥区别？

A. 设置区别：

在 \conf\dialplan\default.xml 配置文件里的<extension name="socket"> 章节里面：

异步模式参数：

<action application="socket" data="127.0.0.1:8084 async full"/>

同步模式参数：

<action application="socket" data="127.0.0.1:8084 sync full"/>

B. 使用区别：主要区别在于 执行 app 或者 api 的时候异步模式是马上返回的。而同步模式是阻塞的。一直到条件满足函数才会返回的。

C. 例子： 以 播音举例

```
esl_execute(handle, "play_and_get_digits", "1 12 1 15000 # intelno.wav" , NULL);
```

其中的"1 12 1 10000 # intelno.wav" 分别对应下面参数：

<min> <max> <tries> <timeout ms> <terminators> <file>

表示 最小收 1 个按键 最大收 12 个按键 重复 1 次 等 15000 毫秒，按#结束。提示语音是 intelno.wav。

异步模式下 esl_execute 执行之后马上返回，ivr 可以进入处理收按键环节。

用户开始听见 intelno.wav 的语音内容。

在异步模式下，用户的按键处理条件可以是程序自己来处理。按键的结束条件也不仅仅只有上面几个，比如可以设置 两个按键的间隔时间等参数。这样大大增加了灵活性。

同步模式下，函数 esl_execute 执行就没有返回，除非以下几个条件之一达到：

用户按键 达到 12 个，

用户按了#键

语音播放结束 超过 15 秒时间。

函数才会返回。

表面上看同步模式比较简单，开发流程比较容易。

但是一旦是多线并发，业务复杂的流程，比如使用 esl 开发呼叫中心的流程，或者其他的复杂的流程。同步模式处理起来就吃力。简单就意味着不够灵活。

为了系统将来的业务扩展支持，我们建议使用异步模式。

以下的例子都是以异步模式作为说明。

111. FreeSwitch ESL 开发如何理解 IVR 的播音取按键条件？

Ivr 系统最常用的是播音，然后取按键 的动作。
典型是情况是播语音菜单，或者播提示输入一串号码的语音提示。
该动作 的结束条件很多，初学者往往稀里糊涂的。
讨论如下：
A．ivr 系统先播音，过程可以按键打断，进入收按键后续动作这个 FreeSwitch 本身就是支持的
B．播音正常结束。等待用户输入按键
C．进入收用户输入按键动作之后：有很多条件是并发的
这个些条件通常可以分为：
最大允许输入几个按键
按某个或者某些特殊按键，比如*或者#结束输入
全部按键的最大允许时间，
两个按键之间的允许最大间隔时间
典型的情况是一旦上面的 4 个条件任何一个达到条件，就要完成整个播音取按键动作。上面的这 4 种情况能满足大多数的 ivr 业务需要。
。。。当然还可以自己设计条件，比如包括播音的整个动作的允许时间，但是很少的情况会使用到这些额外的。

112. FreeSwitch ESL 开发如何支持连续播多个语音获取按键？

以异步模式为例子：
主要由以下几个函数组成，说明如下：
//播音取按键
//返回: <0 表示对方挂机，=100 表示按键最大时间结束条件满足 =101 表示两个按键间的间隔时间结束条件条件满足,=3 表示 按键结束条件,比如"#" 满足，=2 表示 最大按键个数结束条件 满足
//参数:handle:会话 handle
//uuid： 会话的 id
//filename:语音文件名称，多个文件以分号";"隔开,文件名称可以带.wav,扩展名,假如没有,默认是.alaw 扩展名.可指定路径,假如没有,默认是语音程序的./data/system 目录下
//EndDtmf:按键结束条件,比如"#"表示按#号结束输入,""表示没有结束按键条件//支持最大 3 个结束按键 比如 EndDtmf="*0#" 表示按 0，* 或者#都可以结束
//MaxDtmf:最大按键个数结束条件,0 表示没有按键个数结束条件
//MaxTimer:按键最大时间结束条件,单位秒,0 表示没有最大时间结束条件
//TwoDtmfTimer:两个按键间的间隔时间结束条件,单位秒,0 表示没有两个按键间的间隔时间结束条件
//dtmf:收到的用户的按键(输出参数),包括结束按键的条件,比如"#"
//说明:假如只有播音,不收取按键 设置：MaxDtmf=0
//member_id 会议的成员 id >0 =0 表示没有进入会议
int play_get_dtmf(esl_handle_t *handle,char*uuid,char*filename0,char*EndDtmf,int MaxDtmf,int MaxTimer,int TwoDtmfTimer,char*outdtmf,int &member_id)

```
{
    int res=0;
    char *p1,*p2,*p,*pan;
    char files[MAXFILES][100];
    int count,i,any_stop=0;
    char
tmp[1024],cmd_tmp[1024],enddtmf[128],outdtmfbuff[128],filename[128],filename1[128];
    if(EndDtmf==NULL || EndDtmf[0]==0)
        strcpy(enddtmf,"q");
    else
        strcpy(enddtmf,EndDtmf);
    if(outdtmf) outdtmf[0]=0;

    p1=filename0;
    p2=p1;
    count=0;
    while ((*p2)!='\0')
    {
        p2++;
        if( *p2!=';') continue;
        strcpy(tmp,p1);
        tmp[p2-p1]='\0';
        strcpy(files[count++],tmp);
        p2++;
        if(*p2==';') p2++;
        p1=p2;
        if(count>=MAXFILES) break;
    }
    if(p2>p1)
    {
        strcpy(files[count++],p1);
        files[count][p2-p1]='\0';
    }
    if(count<=0 || count>=MAXFILES)
    {
        disp_msg("count<2 || count>=MAXFILES _file=[%s]",filename0);
        return -1;
    }

    for(i=0;i<count;i++)
    {
        pan=strstr(files[i],":");
        p=strstr(files[i],"\\");
        if(p==NULL) p=strstr(files[i],"/");
```

```
                if(p==NULL)
                {
                    sprintf(filename1,"%s/%s",_subdir,files[i]);
                }
                else
                    strcpy(filename1,files[i]);

                if(pan==NULL)
                {
                    if(strstr(filename1,".")==0)
                        sprintf(filename,"%s/%s.alaw",playdir,filename1);
                    else
                        sprintf(filename,"%s/%s",playdir,filename1);
                }
                else
                {
                    if(strstr(filename1,".")==0)
                        sprintf(filename,"%s.alaw",filename1);
                    else
                        sprintf(filename,"%s",filename1);
                }
                if(_access(filename,0)==-1)
                {
                    strcpy(tmp,filename);
                    strtok(tmp,".");
                    strcat(tmp,".pcm");
                    if(_access(tmp,0)==-1)
                        sprintf(filename,"%s/%s/default.alaw",playdir,_subdir);
                    else
                        copyfile(tmp,filename);//pcm==>alaw.
                }

    disp_msg("play_get_dtmf  %s,MaxDtmf=%d,MaxTimer=%d,TwoDtmfTimer=%d,EndDtmf=[%
s]",filename,MaxDtmf,MaxTimer,TwoDtmfTimer,EndDtmf);
                //<min> <max> <tries> <timeout ms> <terminators> <file>
                sprintf(cmd_tmp,"1 1 1 10 a %s",filename);//设置让 play_get_dtmf 函数里面的
按键处理条件无效，ivr check_dtmf_event 函数自己处理按键判断
                esl_execute(handle, "play_and_get_digits",cmd_tmp, NULL);
                memset(outdtmfbuff,0,sizeof(outdtmfbuff));
                if(i<count-1)

    res=check_play_dtmf_event_nowaitdtmf(handle,uuid,enddtmf,MaxDtmf,MaxTimer,TwoDtm
fTimer,outdtmfbuff,member_id);
                else
```

```
    res=check_play_dtmf_event(handle,uuid,enddtmf,MaxDtmf,MaxTimer,TwoDtmfTimer,outdtmfbuff,member_id);
                if(outdtmf) strcat(outdtmf,outdtmfbuff);
                disp_msg("play_get_dtmf %s end,dtmf=[%s],res=%d",filename,outdtmfbuff,res);
                if(res!=1)
                {
                    break;
                }
            }
            if(outdtmf) disp_msg("play_get_dtmf %s all end,dtmf=[%s]",filename,outdtmf);
            return res;
}
```

//播音之后的取按键检查函数

//返回: <0 表示对方挂机，=100 表示按键最大时间结束条件满足 =101 表示两个按键间的间隔时间结束条件条件满足,=3 表示 按键结束条件,比如"#" 满足，=2 表示 最大按键个数结束条件 满足

//参数:handle:会话 handle

//filename:语音文件名称，多个文件以分号";"隔开,文件名称可以带.wav,扩展名,假如没有,默认是.alaw 扩展名.可指定路径,假如没有,默认是语音程序的./data/system 目录下

//enddtmf:按键结束条件,比如"#"表示按#号结束输入,""表示没有结束按键条件//支持最大 3 个结束按键 比如 EndDtmf="*0#" 表示按 0，* 或者#都可以结束

//MaxDtmf:最大按键个数结束条件,0 表示没有按键个数结束条件

//MaxTimer:按键最大时间结束条件,单位秒,0 表示没有最大时间结束条件

//TwoDtmfTimer:两个按键间的间隔时间结束条件,单位秒,0 表示没有两个按键间的间隔时间结束条件

//outdtmf:收到的用户的按键(输出参数),包括结束按键的条件,比如"#"

//其他说明:假如只有播音,不收取按键 设置：MaxDtmf=0

//member_id 会议的成员 id >0 =0 表示没有进入会议

int check_play_dtmf_event(esl_handle_t *handle,char*uuid,char*enddtmf,int MaxDtmf,int MaxTimer,int TwoDtmfTimer,char*outdtmf,int &member_id)

//播音之后的取按键检查(不等待)函数

//当多个语音的时候，不是最后一个语音播放的时候使用这个函数。播放完毕就返回，不等待按键，继续播放下一个语音

int check_play_dtmf_event_nowaitdtmf(esl_handle_t *handle,char*uuid,char*enddtmf,int MaxDtmf,int MaxTimer,int TwoDtmfTimer,char*outdtmf,int&member_id)

这个函数和 check_play_dtmf_event（）区别在于一个要等等，一个不要等，实现上基本类似。

//播音录音的事件体内容检查函数：

int check_play_record_body(esl_handle_t *handle,char*uuid,const char*eventbody,char*dtmfbuf,int maxdtmf,char* endchar,int *channel_execute,int &member_id)

{

```
char tmp[128],cmd_tmp[1024],*p;
unsigned int i,len;

if (eventbody==NULL) return 0;
len=strlen(eventbody);
if(len>64) len=64;
strncpy(tmp,eventbody+strlen("Event-Name: "),len);
tmp[len]=0;
for(i=0;i<strlen(tmp);i++) if(tmp[i]=='\n' || tmp[i]=='\r') {tmp[i]=0;break;}
if(strcmp(tmp,"CHANNEL_EXECUTE_COMPLETE")==0)
{
	if(*channel_execute){ disp_msg("Event-Name:[%s]", tmp);return 1;}
	disp_msg("Event-Name:[CHANNEL_EXECUTE_COMPLETE] invalid");
	return 0;
}

disp_msg("Event-Name:[%s]", tmp);
if(strcmp(tmp,"CHANNEL_EXECUTE")==0) *channel_execute=1;
if(strcmp(tmp,"CHANNEL_HANGUP")==0) return -98;
if(tmp[0]='s' && tmp[1]=='s')//strcmp(tmp,"ss: conference%3A%3Amaintenance")==0)
{
	if(member_id==0)
	{
		p=(char*)strstr(eventbody,"Member-ID: ");
		if(p!=NULL)
		{
			strcpy(tmp,p+strlen("Member-ID: "));
			for(i=0;i<strlen(tmp);i++)if(tmp[i]=='\n' || tmp[i]=='\r') {tmp[i]=0;break;}
			member_id=atoi(tmp);
			disp_msg("member_id:%d", member_id);
		}
	}
	return 0;
}
if(_strnicmp(eventbody,"Event-Name: DTMF",strlen("Event-Name: DTMF"))==0)
{
	char*p;
	p=(char*)strstr(eventbody,"DTMF-Digit: ");
	if(p!=NULL)
	{
		*channel_execute=0;
		strcpy(tmp,p+strlen("DTMF-Digit: "));
		for(i=0;i<strlen(tmp);i++)if(tmp[i]=='\n' || tmp[i]=='\r') {tmp[i]=0;break;}
		if(strcmp(tmp,"%23")==0) strcpy(tmp,"#");
```

```
                disp_msg("dtmf:[%s]", tmp);
                if(dtmfbuf && strlen(dtmfbuf)<64) strcat(dtmfbuf,tmp);
                if(strlen(dtmfbuf)>=maxdtmf) return 2;
                len=strlen(endchar);
                //支持最大 3 个结束按键
                if(len>0) if(tmp[0]==endchar[0]) return 3;
                if(len>1) if(tmp[0]==endchar[1]) return 3;
                if(len>2) if(tmp[0]==endchar[2]) return 3;
            }
        }
        return 0;
}
```

使用例子如下 ：

```
//播音，语音播放结束不等按键，
play_get_dtmf(&handle,"hello1;hello2","#",0,0,0,dtmf, member_id);
//播音同时取按键，语音播放结束等用户按键，
play_get_dtmf(&handle,"menu ","*0#",5,20,0,dtmf, member_id);
//不播音，直接等按键，
check_play_dtmf_event (&handle,"#",0,0,0,dtmf, member_id);
```

113. FreeSwitch ESL 开发如何支持录音取按键？

主要有 3 个函数组成，说明如下：

```
//事件体检查内容函数,这个上面已经介绍过
int check_play_record_body()
```

//录音取按键函数

//返回: <0 表示对方挂机，=100 表示按键最大时间结束条件满足 =101 表示两个按键间的间隔时间结束条件条件满足,=3 表示 按键结束条件,比如"#" 满足，=2 表示 最大按键个数结束条件 满足

//参数:handle:会话 handle

//uuid： 会话的 id

//filename:语音文件名称，多个文件以分号";"隔开,文件名称可以带.wav,扩展名,假如没有,默认是.alaw 扩展名.可指定路径,假如没有,默认是语音程序的./data/system 目录下

//EndDtmf:按键结束条件,比如"#"表示按#号结束输入,""表示没有结束按键条件//支持最大 3 个结束按键 比如 EndDtmf="*0#" 表示按 0，* 或者#都可以结束

//MaxDtmf:最大按键个数结束条件,0 表示没有按键个数结束条件

//MaxTimer:按键最大时间结束条件,单位秒,0 表示没有最大时间结束条件

//TwoDtmfTimer:两个按键间的间隔时间结束条件,单位秒,0 表示没有两个按键间的间隔时间结束条件

//outdtmf:收到的用户的按键(输出参数),包括结束按键的条件,比如"#"

//member_id 会议的 id >0 =0 表示没有进入会议

```
int record_get_dtmf(esl_handle_t *handle,char*uuid,char*filename,char*EndDtmf,int MaxDtmf,int MaxTimer,int TwoDtmfTimer,char*outdtmf,int&member_id)
{
    int res;
    char cmd_tmp[1024],enddtmf[128],outdtmfbuff[128];
    if(EndDtmf==NULL || EndDtmf[0]==0)
        strcpy(enddtmf,"q");
    else
        strcpy(enddtmf,EndDtmf);
    if(outdtmf) outdtmf[0]=0;

    char *pan, _file[BUF_SIZE*2],tfile[BUF_SIZE*2];

    strcpy(_file,filename);
    pan=strstr(_file,":");
    if(pan)
    {
        strcpy(tfile, _file);
    }
    else
    {
        strcpy(tfile, recdir);
        strcat(tfile, "/");
        strcat(tfile, _file);
        if(strstr(_file,".")==0)
        {
            strcat(tfile,".alaw");
        }
    }
    CheckCreateDirectorys(tfile);
    if(_access(tfile,0)!=-1) remove(tfile);

    disp_msg("record_get_dtmf:%s",tfile);
    memset(outdtmfbuff,0,sizeof(outdtmfbuff));
    sprintf(cmd_tmp,"api uuid_record %s start %s %d",uuid,tfile,MaxTimer*2);//MaxTimer*2 的条件是为了保证 uuid_record 的条件无效。而是通过自己的 ivr check_record_dtmf_event 函数里面进行条件录音结束判断
    esl_send_recv_timed(handle, cmd_tmp,1000);
    res=check_record_dtmf_event(handle,uuid,enddtmf,MaxDtmf,MaxTimer,TwoDtmfTimer,outdtmfbuff,member_id);
    if(outdtmf) strcpy(outdtmf,outdtmfbuff);
    disp_msg("record_get_dtmf:%s,end,dtmf=[%s]",tfile,outdtmfbuff);
    sprintf(cmd_tmp,"api uuid_record %s stop all",uuid);
    disp_msg("record_get_dtmf:%s stop",tfile);
```

```
        esl_send_recv_timed(handle, cmd_tmp,1000);
        disp_msg("record_get_dtmf:%s stop end",tfile);

        return res;
}

    //录音之后的取按键检查函数
    //返回: <0 表示对方挂机，=100 表示按键最大时间结束条件满足 =101 表示两个按键间的间隔时间结束条件条件满足,=3 表示 按键结束条件,比如"#" 满足，=2 表示 最大按键个数结束条件 满足
    //参数:handle:会话 handle
    //filename:语音文件名称，多个文件以分号";"隔开,文件名称可以带.wav,扩展名,假如没有,默认是.alaw 扩展名.可指定路径,假如没有,默认是语音程序的./data/system 目录下
    //enddtmf:按键结束条件,比如"#"表示按#号结束输入,""表示没有结束按键条件//支持最大 3 个结束按键 比如 EndDtmf="*0#" 表示按 0，* 或者#都可以结束
    //MaxDtmf:最大按键个数结束条件,0 表示没有按键个数结束条件
    //MaxTimer:按键最大时间结束条件,单位秒,0 表示没有最大时间结束条件
    //TwoDtmfTimer:两个按键间的间隔时间结束条件,单位秒,0 表示没有两个按键间的间隔时间结束条件
    //outdtmf:收到的用户的按键(输出参数),包括结束按键的条件,比如"#"
    //其他说明:假如只有播音,不收取按键 设置：MaxDtmf=0
//member_id 会议的 id >0  =0 表示没有进入会议

    int     check_record_dtmf_event(esl_handle_t     *handle,char*uuid,char*enddtmf,int MaxDtmf,int MaxTimer,int TwoDtmfTimer,char*outdtmf,int& member_id)
    {
        int done = 0,c=0,twodtmfc=0;
        esl_status_t status;
        time_t exp = 0;
        char dtmf[128];
        int res=0,press_dtmf;
        int  channel_execute=0;

        press_dtmf=1;
        if(outdtmf) outdtmf[0]=0;
        dtmf[0]=0;
        while((status = esl_recv_timed(handle, 1000)) != ESL_FAIL)
        {
            if(press_dtmf)
            {
                twodtmfc++;
                c++;
            }
            if(MaxTimer>0  &&  c>=MaxTimer){disp_msg("Waiting  alldtmf  %d  seconds
```

```
timout.",c); res=100;break;}
                if(TwoDtmfTimer>0 && twodtmfc>0 && twodtmfc>=TwoDtmfTimer){disp_msg("Waiting twodtmf %d seconds timout.",c); res=101;break;}
                if (status == ESL_SUCCESS)
                {
                    const char *type = esl_event_get_header(handle->last_event, "content-type");
                    if (type)
                    {
                        if(strcasecmp(type, "text/disconnect-notice")==0)
                        {
                            const char *dispo = esl_event_get_header(handle->last_event, "content-disposition");
                            disp_msg("Got a disconnection notice dispostion: [%s]", dispo ? dispo : "");
                            if (dispo && strcmp(dispo, "linger")==0)
                            {
                                res=-99;
                                break;
                            }
                        }
                        if(strcasecmp(type, "text/event-plain")==0)
                        {
                            const char *eventbody=esl_event_get_body(handle->last_event);
                            int oldlen=strlen(outdtmf);

    if((res=check_play_record_body(handle,uuid,eventbody,outdtmf,MaxDtmf,enddtmf,&channel_execute,member_id)))
                            {
                                break;
                            }
                            if(oldlen==1) press_dtmf=1;
                            if(strlen(outdtmf)>oldlen) twodtmfc=0;
                        }
                    }
                }
            }
            return res;
}
```

使用例子如下 ：

```
// 录音到 z:/目录下 .alaw 扩展名表示 录制为 8K8bit 的 alaw 格式语音，.wav 扩展名录制为 8K16bit 的线性 pcm 格式语音
                sprintf(recordfilename,"z:/%s.alaw",uuid);//录制到 z:/目录下 ， 以 uuid 作为文件名次
```

```
record_get_dtmf(&handle,uuid,recordfilename,"#",1,30,0,dtmf,member_id);
```

114. FreeSwitch ESL 开发如何停止正在进行的播音录音等媒体操作？

停止录音要看是否是会议录音，假如是会议录音 停止方法如下：

```
if(member_id>0 && meetno>0 && recordmeet>0)
{
 sprintf(cmd_tmp,"%d recording stop all",meetno);
 disp_msg("conference %s",cmd_tmp);
 sprintf(tmp,"api conference %s\nconsole_execute: true\n\n", cmd_tmp);
 disp_msg(tmp);
 esl_send_recv(handle, tmp);
 if (handle->last_sr_event)
 {
        const char *err = NULL;
        if (handle->last_sr_event->body) {
                disp_msg("%s\n", handle->last_sr_event->body);
        } else if ((err = esl_event_get_header(handle->last_sr_event, "reply-text"))
&& !strncasecmp(err, "-err", 3)) {
                disp_msg("Error: %s!\n", err + 4);
        }
 }
 recordmeet=0;
 break;
}
```

否则假如是通道的播音和录音等媒体操作，停止的方法如下：

```
 sprintf(cmd_tmp,"api uuid_break %s all",uuid);
esl_send_recv_timed(handle, cmd_tmp,1000);
```

115. FreeSwitch ESL 开发如何支持 ivr 电话呼叫通之后的连接和断开？

连接两个通道，在 mycallback 函数使用以下办法实现：

```
sprintf(cmd_tmp,"api uuid_bridge %s %s",uuid,uuid_to);
esl_send_recv_timed(handle, cmd_tmp,1000);
```

某个通道要断开和其他通道的媒体连接，在 mycallback 函数里面使用以下办法：

esl_execute(handle, "park","" , NULL);

116. FreeSwitch ESL 开发如何支持电话会议？

某个通道要参加会议，在 mycallback 函数里面使用以下办法：

```
sprintf(cmd_tmp,"%d@default",meetno);
esl_execute(handle, "conference",cmd_tmp , NULL);
```

其中的会议号码理论上没有任何限制。

会议的配置参照 conf\autoload_configs\conference.conf.xml 配置文件

默认的会议配置文件用户可以按键控制会场很多操作。我们全部屏蔽，修改如下：

```
<caller-controls>
    <group name="default">
  <!--        <control action="mute" digits="0"/>
      <control action="deaf mute" digits="*"/>
      <control action="energy up" digits="9"/>
      <control action="energy equ" digits="8"/>
      <control action="energy dn" digits="7"/>
      <control action="vol talk up" digits="3"/>
      <control action="vol talk zero" digits="2"/>
      <control action="vol talk dn" digits="1"/>
      <control action="vol listen up" digits="6"/>
      <control action="vol listen zero" digits="5"/>
      <control action="vol listen dn" digits="4"/>
      <control action="hangup" digits="#"/>    -->
      <control action="energy equ" digits="a"/>
    </group>
</caller-controls>
```

117. FreeSwitch ESL 开发如何支持退出电话会议？

退出电话会议在 mycallback 函数使用以下办法实现：

```
esl_execute(handle, "park","" , NULL);
```

118. FreeSwitch ESL 开发如何实现系统对会场的某个人静音和去除静音？

设置某个人静音（只能听，不能说）在 mycallback 函数使用以下办法实现：

```
sprintf(cmd_tmp,"%d mute %d",userevent.data,member_id);
sprintf(tmp,"api conference %s\nconsole_execute: true\n\n", cmd_tmp);
```

```
disp_msg(tmp);
esl_send_recv(handle, tmp);
if (handle->last_sr_event)
{
 const char *err = NULL;
 if (handle->last_sr_event->body) {
        disp_msg("%s\n", handle->last_sr_event->body);
 }    else    if    ((err    =    esl_event_get_header(handle->last_sr_event,    "reply-text"))
&& !strncasecmp(err, "-err", 3)) {
        disp_msg("Error: %s!\n", err + 4);
 }
}
```

设置某个人去除静音（能说话）在 mycallback 函数使用以下办法实现：

```
sprintf(cmd_tmp,"%d unmute %d",userevent.data,member_id);
sprintf(tmp,"api conference %s\nconsole_execute: true\n\n", cmd_tmp);
disp_msg(tmp);
esl_send_recv(handle, tmp);
if (handle->last_sr_event)
{
 const char *err = NULL;
 if (handle->last_sr_event->body) {
        disp_msg("%s\n", handle->last_sr_event->body);
 }    else    if    ((err    =    esl_event_get_header(handle->last_sr_event,    "reply-text"))
&& !strncasecmp(err, "-err", 3)) {
        disp_msg("Error: %s!\n", err + 4);
 }
}
```

说明：注意上面 的 api 命令里面的 console_execute: true
经过本人测试 没有这个，会议的设置不会起作用，这个的 console_execute: true 是参考 fs_cli.c 的源码实现的。

119. FreeSwitch ESL 开发如何支持电话会议录音？

电话会议录音，在 mycallback 函数使用以下办法实现：

```
pan=strstr(_file,":");
if(pan)
{
 strcpy(tfile, _file);
}
else
{
```

```
 strcpy(tfile, recdir);
 strcat(tfile, "/");
 strcat(tfile, _file);
 if(strstr(_file,".")==0)
 {
       strcat(tfile,".alaw");
 }
}
CheckCreateDirectorys(tfile);
if(_access(tfile,0)!=-1) remove(tfile);

disp_msg("OP_RECCONF:%s",tfile);
sprintf(cmd_tmp,"%d recording start %s",meetno,tfile);
recordmeet=1;
```

电话会议录音的停止操作 可以参考上面的 停止媒体操作。
另外电话会议结束，会议录音自动结束。

0. 如何实现 FreeSwitch ESL 开发的完整 IVR 演示流程？

系统提供的 demo 比较简单，没有处理基本的 ivr 的工作单元：
播音取按键，录音取按键，电话转接，参加会议
下面的这个文件能实现上面的这些基本功能：
先修改拨号计划：增加下面的内容：

```
<extension name="socket">
<condition field="destination_number" expression="^12396$">
<action application="set" data="call_timeout=30"/>
<action application="set" data="record_sample_rate=8000"/>
<action application="export" data="RECORD_STEREO=false"/>
<action application="set" data="hangup_after_bridge=false"/>
<action application="set" data="continue_on_fail=true"/>
<action application="socket" data="127.0.0.1:8084 async full"/>
</condition>
</extension>
```

然后使用软电话注册上去然后拨打 12396
完整的 main.cpp 代码请参照附录。或者到 ftp 42.121.124.34 上匿名下载 main.cpp

120. 实网环境中如何跟踪和解决崩溃？

非开发环境的实网环境下的 debug
a.使用微软的工具
可以使用微软的安装包,也可以使用安装之后的，免安装版本：

b.需要提供开发环境编译产生的所有工程的 pdb 文件

以 test.ext 为例：非开发环境下的 debug 步骤：

.0 准备提供 test.pdb 在 test.exe，Pdb 文件名次要和编译产生的 exe 名次一样，不能修改。Releae 版本也可参数 pdb 文件。

.1 免安装版本：Windows_debug.rar 解压都某个目录，Windows_debug.rar 可以到 ftp 42.121.124.34 上匿名下载

.2 命令行下执行： "Windows Kits\8.0\Debuggers\x86\adplus_old.vbs" -crash -pn test.exe -o "c:\debug"

Debug 目录是预先建立好的输出 debug 日志的目录

.3 一旦程序崩溃会参数 dump 和 log 文件

比如下面的 code

```
int check_time()
{
 static char *p;
 static int c=1;
        printf("time=%d\n",time(NULL));
        if((c++)%10==0)
        {
              printf("time=%d,%s\n",time(NULL),time(NULL));
              *p=1;
              p++;
        }
        return 0;
}
```

崩溃之后

打开 PID-4324__TEST.EXE__Date_11-14-2012__Time_10-11-3737.log 文件可以找到是哪个函数里面崩溃

```
Faulting stack below ---
 # ChildEBP RetAddr  Args to Child
00 0012fe5c 004022cb 0042aa80 0042818e 0012fe9c test!_output+0x5b3
01 0012fe88 00401ebe 00428184 50a2fde1 50a2fde1 test!printf+0x5b
02 0012fee8 00401f82 00000000 00000000 7ffd4000 test!check_time+0x6e
03 0012ff48 004036a9 00000001 006c0f88 006c0fd0 test!main+0x62
04 0012ff88 77741174 7ffd4000 0012ffd4 77d0b3f5 test!mainCRTStartup+0xe9
WARNING: Stack unwind information not available. Following frames may be wrong.
05 0012ff94 77d0b3f5 7ffd4000 73482437 00000000 kernel32!BaseThreadInitThunk+0x12
06 0012ffd4 77d0b3c8 004035c0 7ffd4000 00000000 ntdll!RtlInitializeExceptionChain+0x63
07 0012ffec 00000000 004035c0 7ffd4000 00000000 ntdll!RtlInitializeExceptionChain+0x36
```

这样对崩溃的跟踪就比较准确。

FreeSwitch 已知 bug

参考资料：

fs1.2.1 使用 flash 的 rtmp_mod 功能会崩溃。fs 运行环境 win7 6G men P950CPU

fs1.2.3 启动崩溃，怀疑是软电话 eyebeam 没有关闭，自动的注册，注册的时候 fs 正在初始化某些东西导致崩溃 fs 运行环境 win7 6G men P950CPU

fs1.2.5.3 启动崩溃，怀疑是软电话 eyebeam 没有关闭，自动的注册，注册的时候 fs 正在初始化某些东西导致崩溃 fs 运行环境 win7 6G men P950CPU

fs1.2.3 使用 flash 测试 会出现 thread failure fs 运行环境 win7 2G 内存，CPU :T7200

fs1.2.5.3 使用 flash 测试 会出现 [CRIT] switch_core_session.c:1644 Thread Failure! fs 运行环境 win7 2G 内存，CPU :T7200

fs1.2.5.3 一直 f 使用 lash 很快呼叫，被叫马上接通，然后主叫马上挂机，会崩溃，fs 运行环境 XP+sp3 2G MEM CPU E3300

附录 1 参考资料：

参考资料：

FreeSwitch VOIP 实战 杜金房著（到 2013-03-10 为止未正式出版，书店买不到的噢！）

http://wiki.freeswitch.org/wiki/Main_Page

http://www.freeswitch.org.cn/

http://www.ppcn.net/n1306c2.aspx

http://midcom-p2p.sourceforge.net/draft-ford-midcom-p2p-01.txt

附录 2 ESL IVR 控制代码下载：

您可以从 ftp 42.120.20.89 匿名下载 main.cpp

附录 3 FLEX flashphone 代码下载：

您可以从 ftp 42.120.20.89 匿名下载 test.mxml