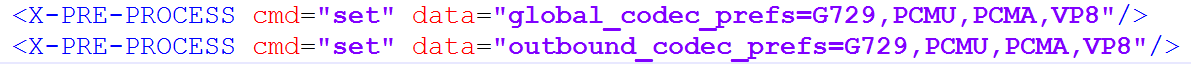
## 一.配置文件

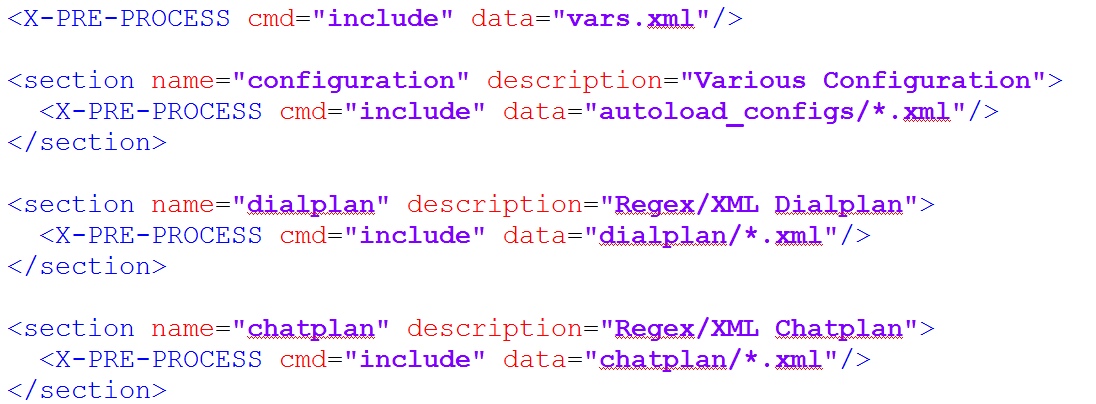
conf/varsxml 常用变量

如，可以配置G729

 当有多个网卡时，需要指定local\_ip\_v4

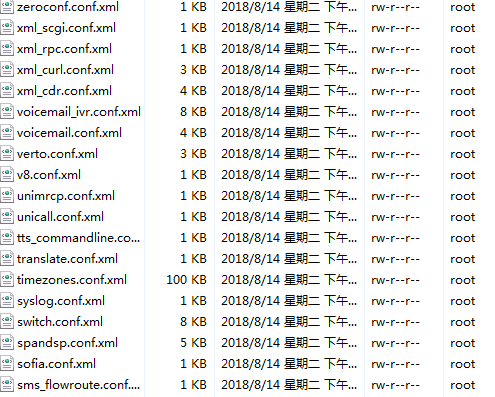


conf/freeswitch.xml 主配置文件，会使用include语句装入其它文件



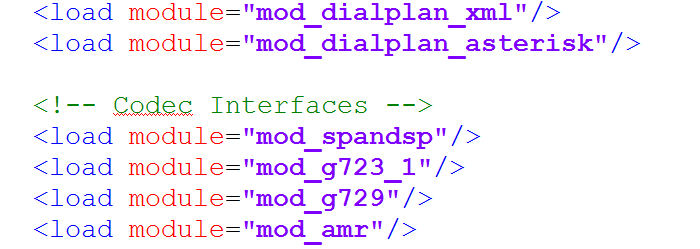
conf/autoload\_configs 目录存放自动加载的配置文件

每个模块都有一个配置文件



modules.conf.xml 配置当FreeSwitch启动时自动装载哪些模块，

如加载g729模块



chatplan 聊天计划

dialplan 拨号计划

default.xml 默认的拨号计划配置，一般用于内部用户路由

public.xml 默认的拨号计划配置，一般用于外部来电路由

directory 用户目录

default 默认的用户目录配置

\*.xml SIP用户，每个用户一个文件

ivr\_menus IVR菜单

jingle\_profiles 连接Google Talk的相关配置

lang 多语言支持

mrcp\_profiles MRCP配置，用于跟第三方语音合成和语音识别系统对接

sip\_profiles SIP配置文件

internal.xml 一个SIP-UA，监听在本地IP及端口5060

externa.xml 另外一个SIP-UA，用作外部连接，端口5080

skinny\_profiles 思科SCCP协议话机的配置文件

# 二 FreeSwitch 架构

FreeSwitch由一个稳定的核心与外围模块组成。外围模块可以动态加载或卸载，在实际应用中可以只加载用到的模块，外围模块通过核心提供的public api与核心通信，而核心通过回调(钩子)机制执行外围模块中的代码。如当Endpoint模块收到呼入请求时，该模块就可以调用核心的switch\_core\_session\_request函数将该Session释放

与session相关的函数都是与呼叫相关的，可以认为它主要与信令层相关

## 1 核心

### 1.1 数据库

freeswitch的核心除了使用内部的队列，哈希表存储数据外，还使用外部的关系型数据库存储数据，默认使用SQLite

freeswitch使用一个核心的数据库(默认位置是/usr/local/freeswitch/db/core.db)来记录系统的接口，任务，当前的通道，通话等实时数据

模块也有自己的数据库，可以通过相关API用于从这些表里查询数据，必要时可以直接查询这些数据库来获取

### 1.2 公共应用程序接口

FreeSwitch的public API可以被任意外围模块调用，来实现和内核的通信，同时还提供了工具函数，如字符串处理等

### 1.3 接口

freeswitch提供了很多抽象接口，但没有具体实现，具体的实现由外围的模块负责，核心层通过回调方式调用具体的实现代码或函数

typedef enum {

SWITCH\_ENDPOINT\_INTERFACE, //终点

SWITCH\_TIMER\_INTERFACE, //定时器

SWITCH\_DIALPLAN\_INTERFACE, //拨号计划

SWITCH\_CODEC\_INTERFACE, //编解码

SWITCH\_APPLICATION\_INTERFACE, //应用程序

SWITCH\_API\_INTERFACE, //命令

SWITCH\_FILE\_INTERFACE, //文件

SWITCH\_SPEECH\_INTERFACE, //语音合成

SWITCH\_DIRECTORY\_INTERFACE, //用户目录

SWITCH\_CHAT\_INTERFACE, //聊天计划

SWITCH\_SAY\_INTERFACE, //分词短语

SWITCH\_ASR\_INTERFACE, //语音识别

SWITCH\_MANAGEMENT\_INTERFACE,//网管接口

SWITCH\_LIMIT\_INTERFACE, //资源限制接口

SWITCH\_CHAT\_APPLICATION\_INTERFACE,//聊天应用程序接口

SWITCH\_JSON\_API\_INTERFACE,//JSON接口

} switch\_module\_interface\_name\_t;

外围模块可以选择实现其中一个或多个接口，并向核心层"注册"这些接口，核心层需要这些时就回调接口中设定的函数

### 1.4 事件

FreeSwitch也使用消息和事件机制来进行进程间和模块间通信，事件的机制是"生产者--消费者"模型，即事件的产生和处理是异步的。事件可以在freeswitch内部通过绑定一定的回调函数进行捕获，freeswitch的核心事件系统会依次回调这些回调函数。在嵌入式脚本中也可以订阅相关的事件并进行处理。

在外围模块，可以通过Event Socket等接口订阅相关的事件。

freeswitch的事件主要有两大类:

一类是主事件，可根据事件的名字来区分，如

CHANNEL\_ANSWER（应答） CHANNEL\_HANGUP（挂机）

一类是自定义事件,事件名永远是CUSTOM.

## 2 模块

### 2.1 Endpoint

它包含了不同呼叫控制协议的接口，如SIP,H323,Google Talk等，从而可以与众多不同的电话系统进行通信。如，可以用mod\_skypopen与Skype网络通信，可以通过mod\_portaudio驱动本地声卡，用作一个软电话

### 2.2 拨号计划(Dialplan)

Dialplan主要提供查找电话路由功能，以XML描述，默认的Dialplan由mod\_dialplan\_xml提供

### 2.3 聊天计划(Chatplan)

Chatplan主要对文本消息进行路由，如SIP SIMPLE,Skype Message,XMPP Message等，它在mod\_sms中实现

### 2.4 应用程序(Application,APP)

FreeSwitch提供了许多App使复杂的任务变得简单，如mod\_vicemail可以实现语音留言，mod\_conference模块则可以实现高质量的多方会议

所有与FreeSwitch的通话都是在一个与多个App在交互，mod\_dptools模块提供了系统中大部分App.

### 2.5 命令接口(FSAPI)

是一种对外的命令接口，它的原理非常简单，输入一个简单的字符串(以空格分隔)，然后由内部函数处理，然后得到一个输出，一个模块通过命令接口可以用命令字符串的方式得到另外一个模块提供的功能，而不用连接其他模块的函数，系统中的大部分API都是由mod\_commands模块提供

### 2.6 XML接口

对XML的解析和访问都是在核心中实现，但对XML的应用和扩展都在外部模块中完成，如mod\_xml\_rpc,mod\_xml\_curl,mode\_xml\_cdr等

### 2.7 编解码器

支持G711,G722,G729,GSM，还支持iLBC,BV16/32,SILK,iSAC,CELT,OPUS等

### 2.8 语音识别和语音合成(ASR/TTS)

支持语音自动识别(ASR)及文本/语音转换(TTS)，可以通过本地模块实现,也可以通过MRCP协议与其他语音产品对接(mod\_unimrcp）实现

### 2.9 格式，文件接口

支持不同格式的声音文件回放,录音,如wav，MP3等，mod\_sndfile实现了对大部分音频格式的支持，mp3在mod\_shout中实现

### 2.10日志

日志可以写到控制台，日志文件，系统日志和远程的日志服务器，实现日志功能的模块有mod\_console,mod\_logfile,mod\_syslog等

### 2.11 定时器

可以使用软时钟或内核的时钟来定时，Freeswitch最理想的工作时钟频率是1000Hz

### 2.12 嵌入式语言

通过swig可以支持多种嵌入式语言控制呼叫流程,如Lua,JavaScript,Perl等

### 2.13 事件套接字

可以通过Event Socket使用其他任何语言，通过TCP Socket可控制呼叫流程，扩展FreeSwitch的功能

## 3 呼叫的概念

呼叫流程一般有这样两种：

1） A向FreeSwitch发起呼叫，FreeSwitch接着启动一个UA呼叫B，两者通话

A到FreeSwitch的通话称为来话，freeswitch呼叫B就称为去话

2） FreeSwitch同时呼叫A和B，两者接电话后将a-leg和b-leg桥接到一起

两路通话都是去话

无论来话还是去话，每次呼叫，freeswitch都会启动一个Session(会话,包含SIP会话，SIP会在每对UAC-USA之间生成一个SIP Session)，用于控制整个呼叫，它会一直持续到通话结束

每个Session控制着一Channel(通道，又称信道),Channel是一对UA间通信的实体，相当于FreeSwitch的leg，每个Channel都用一个唯一的UUID来标识，称为channel UUID

Channel上可以绑定一些呼叫参数，即Channel Variable(通道变量)

Channel中可能包含媒体(音频或视频流),也可能不包含.通话时，freeswitch的作用是将两个Channel(a-leg,b-leg，通常先创建的或主动的叫a-leg)桥接（bridge)到一起,使双方可以组成一个通话，称为一个Call

在通话中，媒体(音频或视频)数据流在RTP包中传送,一般说来channel双向，因此媒体流会有发送(Send/Write)和接收(Receive/Read)两个方向

### 4 全局变量与局部变量

全局变量仅在预处理阶段(系统启动或重载(reloadxml))被求值，一般用于设置一些系统一旦启动就不会轻易改变的量，而局部变量(通道变量)仅在Channel的生命周期中有效,所以，两者的区别是,$${var}只在加载时求值一次，${var}在每次执行时都求值

### 5 App和API

一个App是一个程序，它作为一个Channel一端与另外一端的UA进行通信，相当于它工作在Channel内部，而一个API则是独立于一个Channel之外的，它只能通过Channel的UUID来控制一个Channel，相当于一个第三者

通常在控制台上输入的命令都是API，而在diaplan中执行的程序都是App(dialplan中也能执行一些特殊API)，大部分公用的API都是在mod\_commands模块中加载，而App则在mod\_dptools中

# 三.拨号计划

拨号计划即是对电话进行路由

## 1 XML Dialplan

XML Dialplan由一系列XML配置文件组成，这些XML可以是静态配置，也可以是使用动态配置方式从其他服务器或脚本中动态获取，Dialplan有特定的结构,freeswitch通过解析相关的结构，可以对Dialplan进行路由的呼叫，决定执行何种动作或流程

### 1.1 配置文件结构

拨号计划配置文件默认在conf/dialplan目录中，它们是在freeswitch.xml中定义的，由以下预处理指令装入:

<X-PRE-PROCESS cmd="include" data="dialplan/\*.xml">

### 1.2 正则表达式

Dialplan使用与Perl兼容的正则表达式算法

^1234$ ^匹配字符串开头，$匹配字符串结尾，严格匹配1234

^1234|5678$ 表示匹配1234或5678

^123[0-9]$ 匹配1230,1231,1232.....1239

^123\d$ \d等同于[0-9]

^123\d+$ +表示匹配1个或多个它前面的字符，这里，由于"+"前面是"\d",

所以它就等于1或多个数字，匹配任意以123开头的任意数字

^123\d\*$ \*和+的不同在于，它匹配0个或多个前面的字符，所以它匹配以

123开头的至少3位数的数字串，123,123343等

^123 匹配以123开头的任意字符串

123$ 匹配任何以123结尾的字符串

^123\d{5}$ {5}表示精确匹配5位,包含它前面一个字符，在这里，它匹配以 123开头的所有8位电话号码 ，如^123{5}$ 匹配1233333

^123(\d+)$ ()不起匹配作用，但是他会将123之外的数字保存到$1这个变 量中

^123(\d)(\d+)$ 如果和12345678匹配，则会匹配成功，结果$1=4,$2=5678

### 1.3 通道变量

每一个通道都有很多属性，用于标识Channel的状态，性能等，这些属性即是通 道变量，可简写为Channel Var,Chan Var或Var

在拨号计划中添加

<extension name="Show Channel Variable">

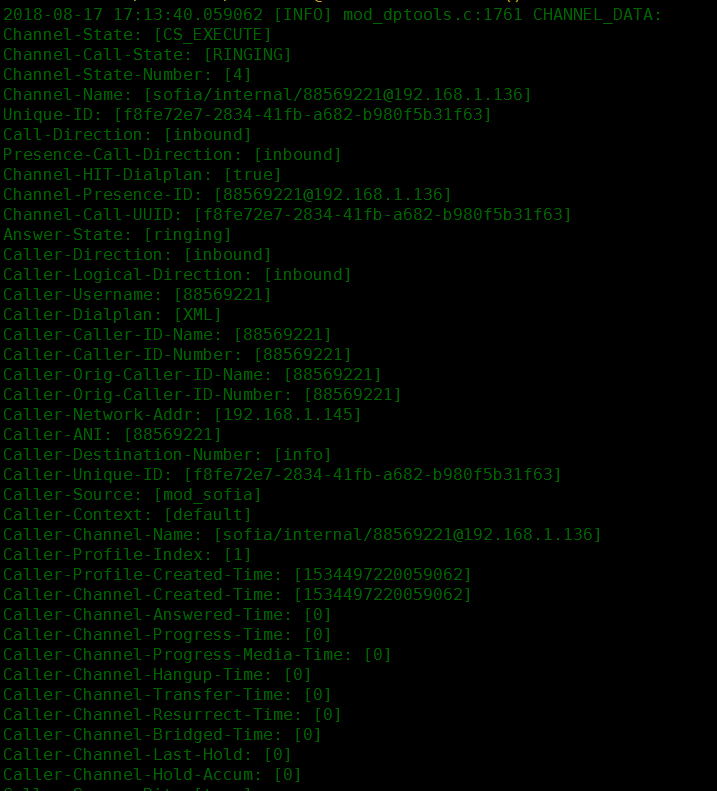
<condition field="destination\_number" expression="^1235|info$">

<action application="info" data=""/>

</condition>

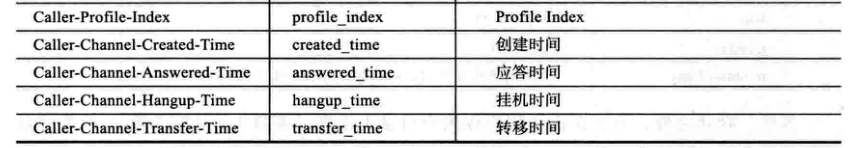
</extension>

就可通过拨打1235或info查看通道信息



其中与通道变量的对应关系





还可以通过${变量名}来访问通道变量，在拨号计划中添加

<extension name="Accessing Channel Variable">

<condition field="destination\_number" expression="^1236(\d+)$">

<action application="log" data="INFO ha，I know you called ${destination\_number}"/>

<action application="log" data="NOTICE The Last few digits $1"/>

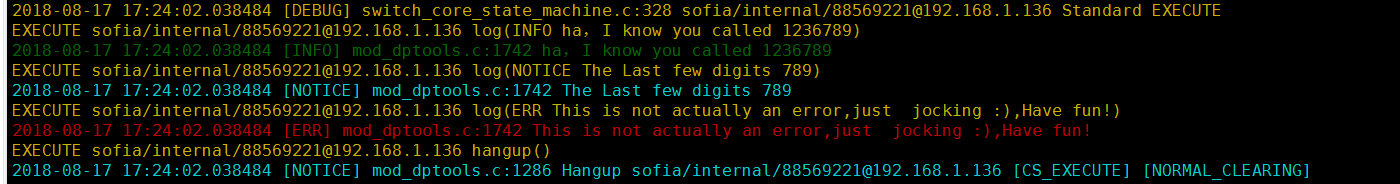
<action application="log" data="ERR This is not actually an error,just jocking :),Have fun!"/>

<action application="hangup"/>

</condition>

</extension>

通过拨打1236789等以1236开头的号码可以查看打印的日志信息



日志有以下几种级别

0 CONSOLE

1 ALERT

2 CRIT

3 ERR

4 WARNING

5 NOTICE

6 INFO

7 DEBUG

### 1.4测试条件

context Dialplan 当前的Context

rdnis 被转接的号码(在呼叫转移中设置)

destination\_number 被叫号码

dialplan Dialplan模块的名字，如XML，YAML，inline

caller\_id\_name 主叫(来电显示)名称

ani 主叫的自动号码识别

aniii 主叫类型，如payphone(投币电话)

uuid 本Channel的唯一标志

source 呼叫源，来自哪一个FreeSwitch模块，如 mod\_portaudio，mod\_sofi等

chan\_name Channel名字，如portaudio/1234

network\_addr 主叫的IP地址

year 当前的年 0~9999

yday 一年中的第几天，1~366

mon 月，1~12

mday 日，1~31

week 一年中的第几周，1~53

mweek 本月中的第几周，1~6

wday 一周中第几天,1~7（周日=1，周1=2）

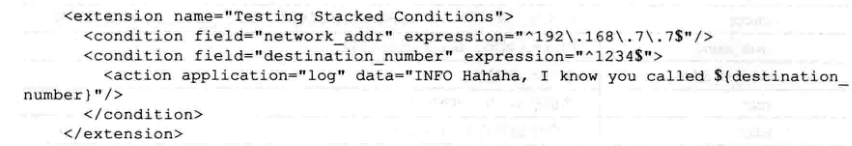
hour 小时(0~23)

minute 分(0~59)

minute-of-day 一天中第几分钟，(1~1440)（午夜=1，1点=60，中午=720）

测试条件接受用户在用户目录中设置的变量,但必须要使用${}对变量进行引用

测试条件一般不嵌套，但可以叠加

如：

还可以用break搭配on-false,on-true,always,never等使用

on-false，在第一次匹配失败时，就停止当前extension的处理，处理下一个extension，默认也是这个配置.

on-true,在第一次匹配成功时才停止当前extension的处理

always，不管是否匹配，都停止

never， 不管是否匹配，都继续

如下:

<extension>

<condition field="destination\_number" expression="^1358$"/>

<condition field="network\_addr" expression="^192\.168\.1\.148$" break="on-true">

<action application="log" data="INFO network is 192.168.148"/>

</condition>

<condition field="network\_addr" expression="^192\.168\.1\.145$">

<action application="log" data="INFO network is 192.168.145"/>

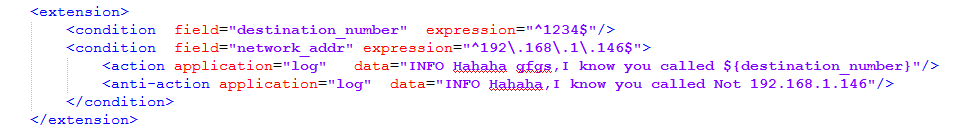
</condition>

</extension>

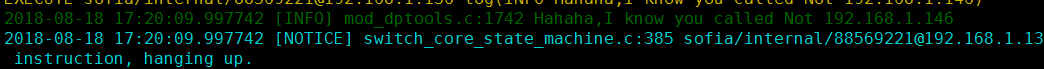
当匹配到192.168.1.148就打印日志192.168.1.148，如果和192.168.1.148不匹配就去和192.168.1.145进行匹配



动作与反动作



因为主机的网址不是192.168.1.146，所以会匹配到反动作，即打印



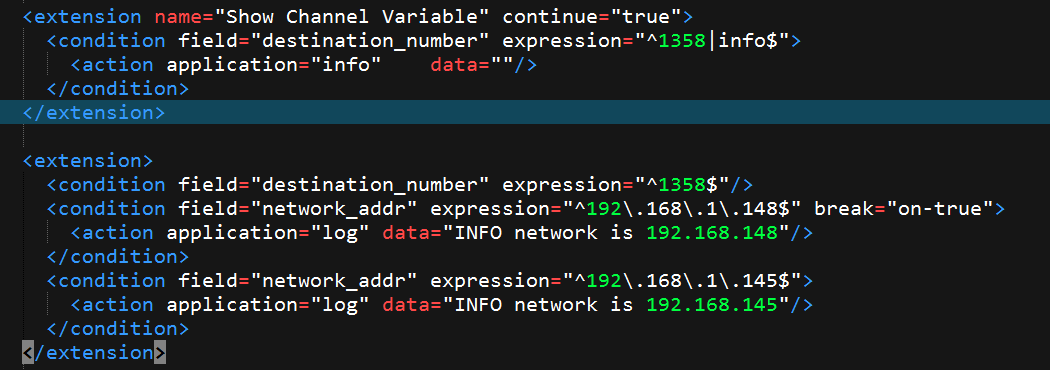
### 1.5 Channel



当新建一个Channel时，首先进行初始化(INIT)，然后进入路由，即查找解析Dialplan阶段，找到合适的路由入口之后，Hunting会执行(EXECUTE)一系列动作，最后无论哪一方挂机,都会进入挂机(HANGUP)阶段，后面的报告(REPORTING)一般用于进行统计，计费等，最后将Channel销毁(DESTROY),释放系统资源

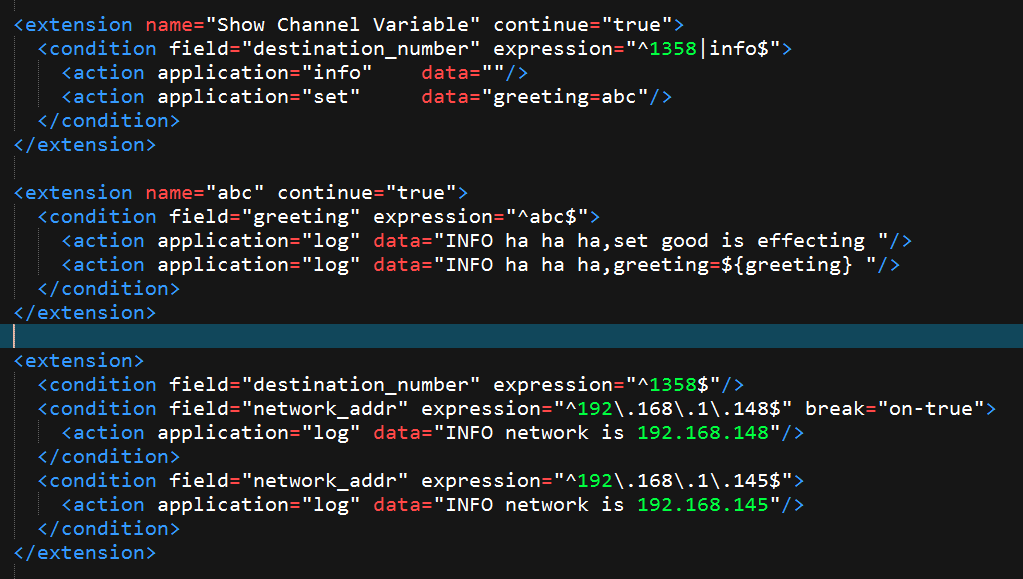
在EXECUTE状态，可能会发生转移(Transfer),会转移到同一context下其他的extension,或者转移到其他context下的extension，无论怎么转移都会重新路由，即进入ROUTING

extension默认只要匹配成功一个就不会再继续执行，除非添加continue="true"，默认这个参数为false



拨号1358时，当第一个行不加continue="true"，后面一个extension就不会执行

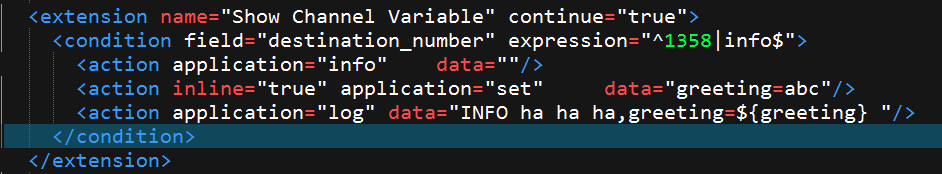
当一个Channel进入ROUTING阶段，首先会到达Dialplan，然后对预设的Dialplan进行解析,解析的过程即Parsing或Hunting，解析成跟成功后，会得到Action，将这个Action指定的动作放入一个队列中，然后进入EXECUTE阶段,依次执行所有的Action，如



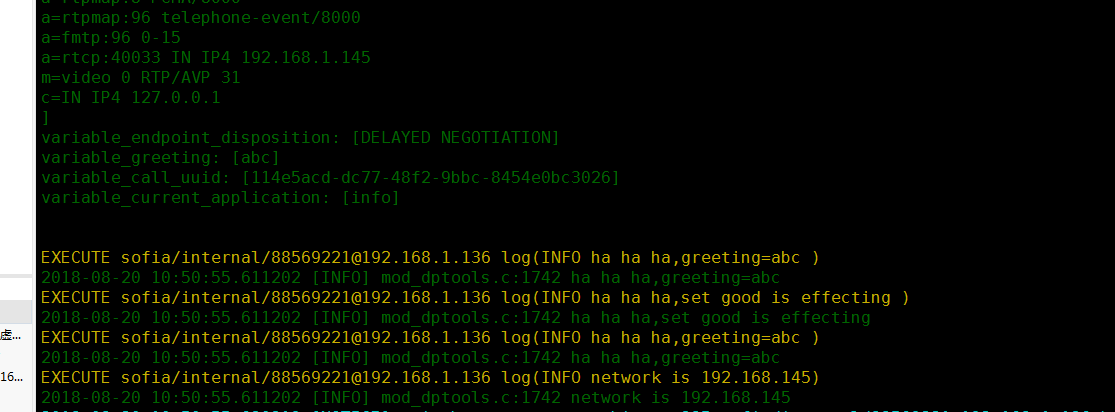
对于上面的拨号规则，不能打印出ha ha ha引导的日志，只能输出前面和后面的日志，因为set是在EXECUTE阶段执行，不是解析一个就执行一个，所以greeting在解析的阶段还不能设置为abc，导致被条件过滤，不能执行

### 1.6 内联执行

上面的问题可以用inline属性来解决，加上inline后，freeswitch就会立即执行它，而不会等到EXECUT阶段



能输出上面未打印的日志



但不是所有的应用地能使用inline执行，一般只是很快地存取某个变量，并且不能改变当前Channel的状态，包括以下应用可以使用

check\_acl,eval,event,export,enum,log,presence,set,set\_global,lcr,set\_profie\_var,set\_user,sleep,unset,nibblebill,verbose\_events,cidlookup,curl,easyroute,odbc\_query

因为inline会干扰执行的顺序，可能会导致非预期的结果

### 1.7 inline Dialplan(内联拨号计划)

通过inline Dialplan可以很方便地在脚本中生成动态的Dialplan而无须使用reloadxml和mod\_xml\_curl

语法：

app1:arg1,app2:arg2,app3:arg3

它没有extension,condition，只是像XML Dialplan那样简单地叠加Action

### 2 常用的Dialplan App

2.1 set

设置一个通道变量

<action application="set" data="my\_var=123456">

2.2 echo

2.3 info

info会在日志中打印全部的通道变量

<action application="info"/>

2.4 answer

用于应答一路呼叫。Freeswitch做被叫时，如果想给主叫放音，必须应答后才可以

<action application="answer"/>

有些App会隐含应答(conference 或fifo)

2.5 bridge

bridge负责桥接另外一条腿，它的参数是一个呼叫字符串

<action application="bridge" data="usr/1000">

2.6 playback

playback用于给Channel放音，比如，如果需要对主叫放音，可以使用playback

<action application="playback" data="/tmp/test.wav">

参数为声音文件的路径,一般的声音文件是.wav格式，如果要播放多个文件可以串联操作

<action application="playback" data="/tmp/test1.wav">

<action application="playback" data="/tmp/test2.wav">

也可以使用file\_string协议来串联

<action application="playback" data="flie\_string:///tmp/test.wav!/tmp/test2.wav"/>

!为文件名分隔符，可以用通道变量的方式指定

<action application="set" data="playback\_delimiter=|">

指定多个文件播放的时间间隔

<action application="set" data="playback\_sleep\_val=500"/>

2.7 sleep

用于设置等待/暂停的一段时间，单位默认为毫秒

2.8 ring\_ready

用于在SIP中给对方回180消息，即通知对方振铃

<action application="ring\_ready" data="1000"/>

2.9 pre\_answer

pre\_answer用于在SIP中给对方回183消息，后续的playback的动作将作为早期的媒体发给对方，如彩铃

2.10 read

read用于实现播放声音，并等待DTMF按键,它的格式

<min> <max> <sound file> <variable name> <timeout> <terminators>

其中

min: 最少收号位数

max:最大收号位数

sound file:要播放的声音文件

variable name:收到用户按键后存到哪个变量

timeout:等待每一位的输入超时毫秒数

terminators：收号小于min位时，该按键可以提前结束，通常是#

## 3 在Dialplan中使用API命令

可以通过变量引用的方式来实现对API命令的引用

<action application="set" data="api\_result=${status()}"/>

<action application="set" data="api\_result=${version()}"/>

<action application="set" data="api\_result=${strftime()}"/>

## 4 通道变量

<action application="transfer" data="1234">

表示切换到1234这个Extension

<action application="set" data="my\_var=my\_value"/>

<action application="export" data="my\_val=my\_value"/>

以上两条命令都可以将my\_var变量的值设置为my\_value，不同的是set程序会作用于当前通道，而export会将变量设置到两个通道上，如果另外一个通道还没有创建,则会在创建时设置，另外export也可以通过nolocal参数将变量限制到b-leg

<action application="set" data="nolocal:my\_var=my\_value">

将存在于a-leg上的变量复制到b-leg上，可以使用

<action application="export" data="my\_val=$my\_val"/>

或

<action application="set" data="export\_vars=var1,var2,var3"/>

freeswitch中提供了截取子字符串的语法，使用格式是 ${var:位置：长度}

位置是以0为参考位：小于0就是倒数

# SIP协议

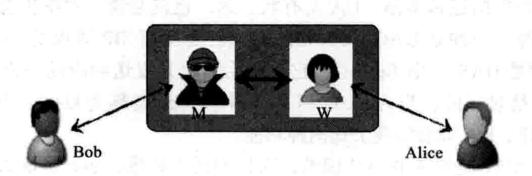
## 一.基础概述

SIP不是C/S,而是UA，即能互做客户端与服务端，UA是在SIP网络中发起或响应SIP处理的逻辑实体.

SIP也是一个基于文本的协议，SIP的请求方式

SIP:INVITE sip:alice@192.168.1.2 SIP/2.0 alice@192.168.1.2为地址，INVITE为建立会话的请求，SIP/2.0为版本号

Freeswitch中常使用B2BUA的方式来建立通信

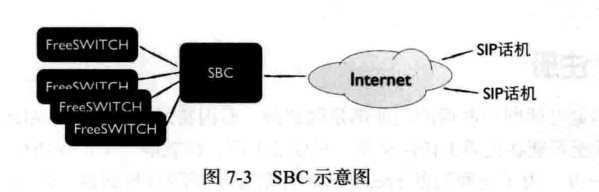


即Bob只能将消息传给M，Alice只能将消息传给W，Bob传给Alice只能通过

Bob-->M--->w--->Alice这样的路径来传递，Alice也是一样

因为Bob的位置可能是不断变更的，每次改动都必须向M报告，相当于M作为注册服务器，然后因为Alice不认识M，Bob又只能发消息给M,所以M只能将消息告知W，即W就作为一个重定向服务器。从而M和W就组成了实现交换功能的交换机，对外使用SIP协议沟通，Bob和Alice就成了软电话或硬件的SIP话机

还有一个边界会话控制器（SBC），它主要位于一堆SIP服务器的边界，用于隐藏内部服务器的拓扑结构，抵御外来攻击。SBC可能是一个代理服务器，也可能是一个B2BUA



一般说来，SBC具有两个或多个网络接口卡(网卡),一边连接互联网，一边连接内部的网络，从服务器上对内部SIP服务器的访问只能经过SBC

### 1 SIP协议的方法和头域

方法：

REGISTER 注册联系消息

INVITE 初始化一个会话(发起一个呼叫)

ACK 对INVITE消息的最终处理

CANCEL 取消一个等待处理或正在处理的请求

BYE 终止一个会话

OPTIONS 查询服务器和能力(也可作ping测试)

头域：

Call-ID 用于区分不同会话的唯一标志

CSeq 顺序号，用于在同一会话中区分事务

From 说明请求来源

To 说明请求接受方

Max-Forwards 限制跳跃点数和最大转发次数

Via 描述请求消息经过的路径

Contact 提供了

### 3 SIP注册

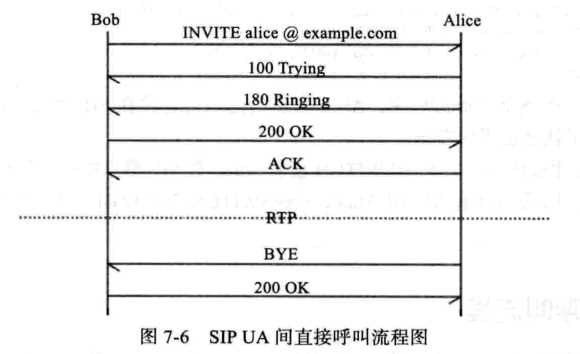


可以解释为 Alice向FreeSwitch发送REGISTER请求，然后FreeSwitch返回一个401消息，提示未认证，Alice将收到的Challenge信息进行计算，并将计算结果附加到下一个REGISTER请求上，重新发起注册，FreeSwitch收到后对本地数据库中保存的Alice信息使用同样的算法进行计算和加密，并将其与Alice发送过来的计算结果进行比较，如结果相匹配，则认证通过，Alice就可以正常注册

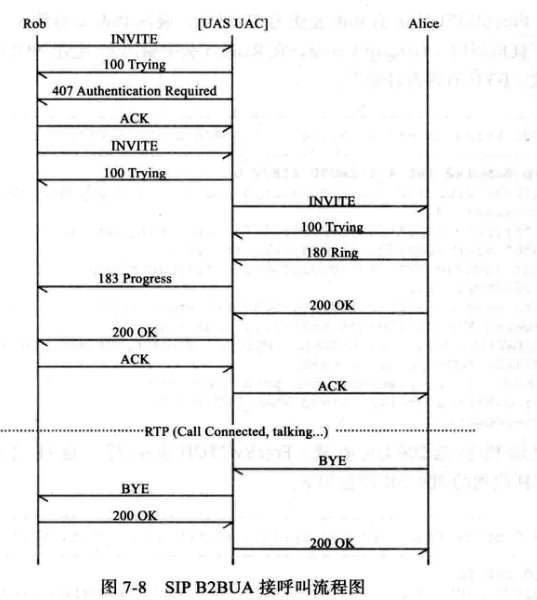
### 4 呼叫流程

4.1 UA间直接呼叫

SIP的UA都是平等的，如果一方知道另外一方的地址，就可以通**信**



### 4.2 B2BUA呼叫



首先Bob向FreeSwitch发起呼叫请求(INVITE),FS回复100 Trying示意已经收到，正在处理，然后发送407(带有Digest验证消息)说明未授权，Bob回复ACK证实收到认证要求，然后重新发送INVITE请求(请求消息中附带了验证消息)，FreeSwitch回复100Trying示意已收到正在处理，认证后发现实授权用户，因而继续运行，这时就建立起了一个通道。

接下来就进入路由阶段，根据路由发现是要拨号给Alice，于是向Alice发送一个请求(INVITE）,同样Alice发送100 Trying，提示已经收到，正在处理，因为被叫不需要验证权限，所以直接回复180Ringing提示正在振铃，此时Freeswitch发送一个183消息，产生一个回铃音。

当Alice发现电话铃响，就接听电话，并向freeswitch发送 200 OK消息报告正在接听，FreeSwitch获悉后就向Bob发送200 OK，并转入Alice的语音，切断振铃的播放，然后Bob向FreeSwitch发送ACK，证实已经收到200 OK消息，然后两者就在RTP协议上通过，直到一方挂机，向FreeSwitch发送Bye，然后FreeSwitch向另一方转发Bye消息

# 媒体

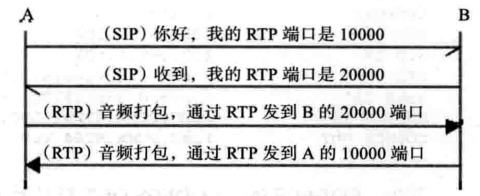
## 1音频编解码

为了能在网络环境中进行传输，需要将模拟的语音信号转变为数字信号，然后在接收端再转换为模拟信号

实现AD/DA的过程需要进行编解码，一般采样频率越高，声音越清晰，保留的细节越多，但同时占用的带宽更大

## 2 媒体工作原理和配置

媒体数据在RTP流中传输，在每个音频包前面加上12字节的RTP包头，就成为了一个RTP包，RTP包中携带了音频的编码类型和时间戳等数据，便于对方收到后进行解码回放以及同步等



A和B首先通过SIP信令协商好双方使用的RTP端口号，然后互相向对方发送RTP媒体流

SIP可使用TCP或UDP承载，但RTP数据一般只使用UDP承载

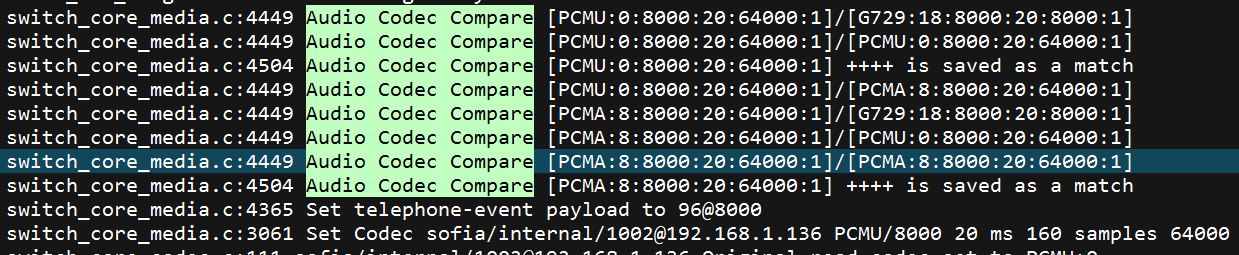
SIP Profile支持的媒体列表在vars.xml中配置



或者可以直接在internal.xml中进行更改



## 3 媒体协商



SIP使用Offer/Answer(请求/应答)机制来协商，请求一方提供自己支持的媒体编码列表，被请求的一方比较自己支持的媒体列表最终选择一种(或几种)编码以应答(Answer)方式通知请求者.

## 4协商时机与策略

FreeSwitch的协商分为早协商和晚协商，当呼叫到达一个SIP Profile时，即某端口收到INVITE请求而未到达路由阶段时，就先行协商，称为早协商，而如果等到路由阶段，到达拨号计划时再进行协商称为晚协商

系统默认晚协商,如果使用早协商，可以在SIP Profile中设置以下参数



将默认的true修改为false

协商策略

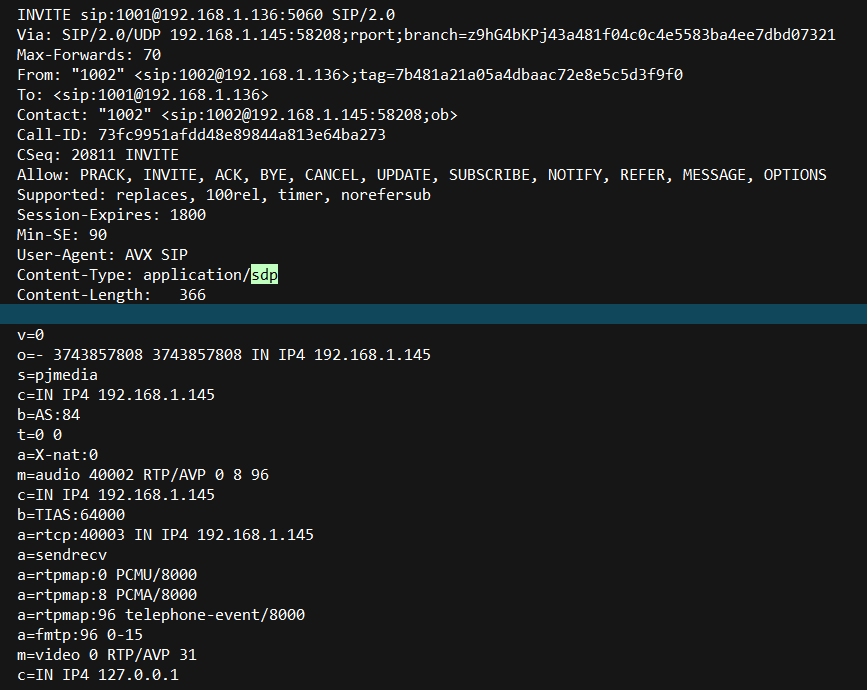


默认为generous，还可以设置为greedy和scrooge

如果设置为greedy(贪婪的),则FreeSwitch会优先考虑自己的编码

如果设置为scrooge(吝啬的),它还会强制采用自己的采用率

## 5 SDP



一般在SIP INVITE消息体都包含SDP

媒体类型由SDP消息(Session Description Protocol 会话描述协议)描述

SDP消息的MIME类型是application/dsp

其中v=0 版本号

C=IN 媒体所在机器的IP地址

m=audio 描述音频数据,40002就是RTP的端口号，用于收发RTP数据，

RTP/AVP表示RTP承载的Audio/Video Profile，0表示PCMU,

8表示PCMA

a=sendrecv 表示双向收发

## 5 RTP

RTP协议(实时传输协议)，它是建立在UDP协议之上，常用于流媒体系统，视频会议和一键通系统

包头文件：

version(V) 1bit 标明RTP版本号，RFC3550中规定版本号为2

padding(P) 1bit 如果该位被设置，则在该packet末尾包含额外的附加信息

extension(X) 1bit 如果该位为1，则在固定的头部后存在一个扩展头部

CSRC count(CC) 4bit 在固定头部后存在多少个CSRC标记

marker(M) 1bit 该位的功能依赖于RTP中实际传输类型的Profile的定义(Profile

可改变该位的长度，视频中marker为1表示一帧图像的结束)

payload type(PT) 7bit 标记着RTP packet所携带信息的类型 sequence number 16bit 序列号，每个RTP包发送后该序列号加1，接收方可以根 据该序列号检测丢包或重新排列数据包顺序

timestamp 32bit 时间戳，反映RTP包中所携带信息

RTP使用一个偶数端口，而RTCP(实时传输控制协议)使用相邻的奇数端口，RTCP本身不传输数据,定期在多媒体会话和参加者之间传输控制数据,RTCP收集相关媒体的连接的统计信息，例如传输字节数，传输分组数，丢失分组数及拉动，单向和双向网络延迟。网络应用程序即可以利用RTCP的统计信息来控制传输的品质，比如当网络拥塞比较严重时，可以限制信息流量或改用压缩率较高的编解码器

## 6 透传，媒体绕过，媒体代理

透传：在不经过转码的情况，将一方收到的媒体流原样转给另外一方

媒体绕过：媒体流使用点对点传输，不经过Freeswitch，只要是通话双方都支持的编码，FreeSwitch也能正常建立通话

媒体代理: 不管FreeSwitch是否支持对该种编码转码，它都对RTP数据在不进行任何处理的情况下发给另外一方.

# SIP模块

## 1基本概念：

Sofia-SIP FreeSwitch的SIP功能是在mod\_sofia模块中实现.

Endpoint 实现互联协议接口的模块，不同的Endpoint主要是使用不同的控制协议

跟其它Endpoint通话

mod\_sofia mod\_sofia实现了注册服务器，重定向服务器，媒体服务器，呈现服务器，

SBC等各种功能，它的定位是一个B2BUA，不能实现SIP代理服务器

的功能

SIP Profile 相当于一个SIPUA，通过各种配置参数可以配置一个UA的行为,一个系 统中可以有多个SIP Profile，每个SIP Profile都可以监听不同的IP地址 和端口对

Gateway 一个SIP Profile中有多个Gatway,用于定义一个远端的SIP服务器使 FreeSwitch可以与其他服务器通信

本地SIP用户 FreeSwitch可以作为注册服务器，这时候其他的SIP客户端就可以向它 注册，FreeSwitch将通过用户目录中的配置信息对注册用户进行鉴权

## 2 Sofia配置文件