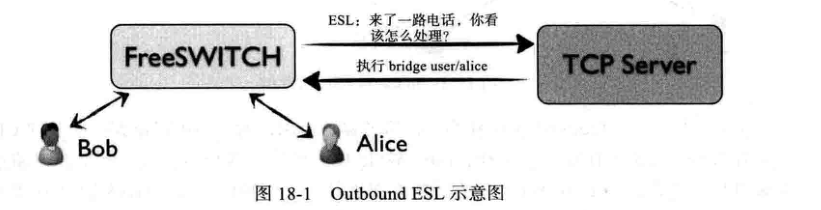
# Event Socket

Event Socket通过Socket的方式使用FreeSwitch提供的所有的App程序和API命令，它只是提供了一个开发接口，所有的通道处理和媒体控制都是由FreeSwitch内部提供的App和API来完成的

## 1架构

### 1.1 外连模式

FreeSwitch作为一个TCP客户端连接到一个TCP Server上，这个TCP Server需要自己实现，这个服务器里，可以实现自己的业务逻辑



当有电话到达FreeSwitch,FreeSwitch查找Dialplan，然后可以通过



连接到TCP Server，并将当前的呼叫相关的信息告诉它，并获悉下一步的指令，这个沟通的过程使用ESL来实现，与SIP和RTP无关。TCP Server只是发布控制指令,并不处理语音数据

TCP Server收到消息后，进行决策，如果确定可以通话，就发一个执行bridge App的消息，告诉FreeSwitch继续呼叫

在挂机之前，FreeSWITCH都会一直向TCP Server汇报Channel的相关消息，从而TCP Server就可以随时了解这路电话的详细消息，也可以在任何时间对它们发号施令

TCP Server会持续监听这个指定的端口等待连接，如果需要支持多个连接，可以使用Socket的Select机制做成多线程

最终TCP Server通过socket这个App来连接，可以认为TCP Server作为socket功能的一部分，在Channel的内部工作

### 1.2 内连模式(Inbound)



在内连模式下，FreeSWITCH作为一个服务器，而用户的程序可以作为一个TCP

Client 主动连接到FreeSWITCH，FreeSWITCH通过EventSocket向外部发送信息， 这些信息以事件的形式体现，用户TCP Client收到事件后，就可以通过App和API来 控制FreeSWITCH的行为

内连模式中，外部的TCP Client是一个第三者，它通常不是Channel的一部分， 而是监听到一个感兴趣的事件以后，通过API(uuid\_ 一族的API)来对Channel进行操 作

### 1.3 Event Socket命令详解

### 1 auth

Inbound模式中，auth是第一个需要发送的命令，用于向FreeSwitch认证，、

auth <password>

例如

auth ClueCon(这个密码在event\_socket.conf.xml中配置)

2 api

用于执行FreeSWITCH的API

api <command> <args>

3 bgapi

api命令是阻塞执行，因此对于执行时间较长的API命令(如originate)会有一 段时间得不到结果，可以用bgapi将这些命令放到后面执行

bgapi <command> <args>

这个命令会立即执行,在后台建立一个任务，并返回一个JobUUID,当真正需要 执行的Api命令返回后，FreeSWITCH会产生一个BACKGROUND\_JOB事 件，该事件带了原先的Job-UUID以及命令的执行结果，客户端可以通过匹配 Job-UUID获取上次命令的执行结果

具体的执行时机？？？

4 linger和nolinger

## 2 Event Socket库

c语言客户端

#include <stdio.h>

#include <stdlib.h>

#include <esl.h>

int main(void)

{

//初始化客户端

esl\_handle\_t handle = {{0}};

//连接服务器

esl\_connect(&handle, "localhost", 8021, NULL, "ClueCon");

//发送一个命令，并接收返回值

esl\_send\_recv(&handle, "api status\n\n");

//freeswitch对api status的响应

if (handle.last\_sr\_event && handle.last\_sr\_event->body) {

printf("%s\n", handle.last\_sr\_event->body);

} else {

// this is unlikely to happen with api or bgapi (which is hardcoded above) but prefix but may be true for other commands

printf("%s\n", handle.last\_sr\_reply);

}

//断开连接

esl\_disconnect(&handle);

return 0;

}