FreeSWITCH IP PBX with Twilio Elastic SIP Trunking

This guide shows how to configure FreeSWITCH to work with Twilio Elastic SIP trunking by walking you through how I got a test instance up and running. There is a lot of great documentation on both twilio.com and freeswitch.org and I have tried not to recreate that but rather to give a brief and pointed example. We at Twilio are always very keen for feedback on our documentation so do not hesitate to email us at sip.interconnectionguides@twilio.com.

Installing FreeSWITCH
Setting up your Twilio Elastic SIP Trunk
  Termination URI
  Origination URI
  ACL
Configuring your FreeSWITCH PBX
  Updating vars.xml
  Updating the external SIP profile
  Build the SIP trunk to Twilio
  Creating a dialplan

Installing FreeSWITCH
I installed FreeSWITCH on an Amazon EC2 instance running Debian Jessie. FreeSWITCH have an excellent installation guide for FreeSWITCH 1.6 at https://freeswitch.org/confluence/display/FREESWITCH/FreeSWITCH+1.6+Video.

Setting up your Twilio Elastic SIP Trunk
We have a pretty comprehensive guide on how to configure a n Elastic SIP Trunk through your Twilio account portal at https://www.twilio.com/docs/sip-trunking/getting-started. I will highlight 3 aspects here.
Termination URI

This is where you configure a unique URI that identifies your trunk. You will need to remember this when configuring your new FreeSWITCH PBX because we need the PBX to reference this URI in its SIP requests.

Origination URI

The easiest way to configure the Origination URI is using “sip:” followed by the public IP address of your FreeSWITCH. In my case, that would be “sip:52.21.106.113”. I chose to add a user part to my Origination URI (“5555”) to make configuring the PBX easier. You don’t have to do this and it is better not to if you have multiple phone numbers on the same trunk.

ACL

Create an ACL that contains the public IP address of your FreeSWITCH PBX.

Note: You can also optionally configure SIP Authentication Credentials
Configuring your FreeSWITCH PBX

Updating vars.xml

All new FreeSWITCH instances come with the same set of users and the same SIP password preconfigured. Hackers know this and will scour the internet for new instances, register with them and try to make calls. The first thing you should do is change the default password on your system.

```
root@ip-172-31-54-222: # cd /usr/local/freeswitch/conf
root@ip-172-31-54-222: /usr/local/freeswitch/conf # vi vars.xml
```

Find the line that starts with "<X-PRE-PROCESS cmd="set" data="default_password="", change the default password.

You then need to find the variables for "external_rtp_ip" and "external_sip_ip" and set them to the public IP address of your FreeSWITCH. In my case, they ended up like this.

```
<X-PRE-PROCESS cmd="set" data="external_rtp_ip=52.21.106.113"/>
<X-PRE-PROCESS cmd="set" data="external_sip_ip=52.21.106.113"/>
```

FreeSWITCH uses different signaling ports for each SIP profile. Because, in my setup, all other SIP devices were on the WAN side of my FreeSWITCH, I made 5060 (the default SIP port) the signaling port for my external SIP profile.

```
<X-PRE-PROCESS cmd="set" data="internal_sip_port=5080"/>
<X-PRE-PROCESS cmd="set" data="external_sip_port=5060"/>
```

If you don't do this, you will need to tell Twilio and any other device on the WAN side of your PBX to send SIP to port 5080.

Updating the external SIP profile

```
...# cd sip_profiles
.../sip_profiles# vi external.xml
```

Uncomment the lines for "ext-rtp-ip" and "ext-sip-ip" and set them to reference the variables in vars.xml.

```
<param name="ext-rtp-ip" value="${external_rtp_ip}"/>
<param name="ext-sip-ip" value="${external_sip_ip}"/>
```

Build the SIP trunk to Twilio

There are several ways to do this. I built the trunk by creating a gateway under the external SIP profile.

```
.../sip_profiles# cd external
.../sip_profiles/external# vi twilio.xml
```

This is what twilio.xml looked like for me
Using SIP Authentication
If you decide to use SIP authentication you can configure Freeswitch Profile to use the credentials configured in the Twilio SIP trunk credential settings:

```xml
<include>
  <gateway name="twilio">
    <param name="proxy" value="dspfree.pstn.twilio.com"/>
    <param name="register" value="false"/>
  </gateway>
</include>
```

Creating a dialplan
This is how I created the dialplan for my test PBX. It is highly unlikely that you will want an identical dialplan so you will need to expand on this example. FreeSWITCH have some great documentation about how to set up dialplans at https://wiki.freeswitch.org/wiki/Dialplan_XML

```bash
...# cd /usr/local/freeswitch/conf/dialplan/public
remove all of the files that are already there
vi 00_inbound_did.xml
and this is what my 00_inbound_did.xml looks like.
<include>
```
The first extension handles calls coming in across the trunk. Because I configured my trunk the way I did (see Origination URI), all calls coming into the FreeSWITCH across my trunk are for “5555”. I also created a user called “5555” on the FreeSWITCH and registered an x-lite client on
the WAN against it. The “data” label in the example below references the external SIP profile because my x-lite client is on the WAN side.

The second through fourth extensions handle calls from my external x-lite client and direct them across the trunk to Twilio. The important things to remember when sending calls to Twilio are

1. The called number has to be in e.164 format: +<country code><full phone number>

2. The invite that your PBX sends to Twilio has to reference the Termination URI you configured. Here, that is handled by referencing the gateway I provisioned earlier in my extensions. That gateway, in turn, referenced my Termination URI.

And that’s it. I was then able to make calls from my x-lite client to the PSTN and vice-versa.