

## Configuring a "Dumb" SBC

What this example hopes to provide is an extremely minimalistic setup of FreeSWITCH which will basically just proxy media based on directives from a pre-existing routing engine. All the routing logic is passed to the to FreeSWITCH via a SIP redirect message (300,302,305).

In the picture on the left, I show an example of the four(4) basic call paths that could take place through FreeSWITCH. In the picture on the right I show the config files and call flow inside of FreeSWITCH that a call would take.

This example assumes that you have completed the basic installation of FreeSWITCH and some sort of SIP proxy (Sonus PSX, Kamailio, OpenSIPS, etc.) that will be controlling your LCR. This is designed for a wholesale model in mind with limited switch based security and no registrations. (see further down for OpenSIPS example config to use with this.)

## SBC\_FreeSWITCH\_Configuration\_Example\_2

The first thing to realize is that we are not registering or authenticating any calls, so the `/conf/directory/default.xml` and `default/*` will be empty except for the default domain setting (which as far as i can tell isn't even needed).

The gateways information in the `sip_profiles` will also be empty since realistically your vendors would not need any kind of gateway registration from you. Typically in the wholesale model, authentications and rejections are done via firewalling or internal systems sending back certain responses.

Now that we have that set up, we want to set up a baseline rejection that will allow the customers to always route advance, so we're going to overwrite any negative responses back with a 503. Since no calls are registered, there is no differentiation between default and public contexts, so you can put this in both, but I believe it's only required in `/conf/dialplan/public.xml` at the very end. (You can also put it at the very end of the `/conf/dialplan/default.xml` as a precaution if you wish).

```
<extension name="nothing_left_private" continue="false">
  <condition break="always">
    <action application="set" data="proto_specific_hangup_cause=sip:503"/>
    <action application="hangup"/>
  </condition>
</extension>
```

The blue in the below examples are what you would replace with your own information.

Now you will need to define your customers. I do this in `/conf/dialplan/public/00_customer_list.xml`.

```
<include>
  <extension name="customer_my_public_desk">
    <condition field="${network_addr}" expression="pu.bl.ic.ip"/>
    <condition field="destination_number" expression="^\+?1?(\d+)$">
      <action application="set" data="hangup_after_bridge=true"/>
      <action application="set" data="continue_on_fail=true"/>
      <!-- THIS WORKS FOR CALL DISTRIBUTION -->
      <!-- <action application="bridge" data="sofia/external/${destination_number}@${distribu
      <!-- THIS WORKS FOR SINGLE PROXY -->
      <action application="bridge" data="sofia/external/${destination_number}@127.0.0.1:5062"/>
    </condition>
  </extension>
  <extension name="customer_my_private_desk">
    <condition field="${network_addr}" expression="pr.iv.ate.ip"/>
    <condition field="destination_number" expression="^\+?1?(\d+)$">
      <action application="set" data="hangup_after_bridge=true"/>
      <action application="set" data="continue_on_fail=true"/>
      <action application="bridge" data="sofia/internal/${destination_number}@127.0.0.1:5062"/>
    </condition>
  </extension>
</include>
```

This will send the request to your proxy LCR engine which will return to you a 30x response that FreeSWITCH automatically dumps into the XML `redirected` context. This context must be defined in `/conf/dialplan/public.xml` if you want it to load properly. You can use a single destination, such as a localhost OpenSIPS proxy as I have shown here, or you can use load balancing to distribute the traffic over multiple proxies (see [Mod\\_distributor](#)). Depending on the ingress IP/location, you should set the `sip_profile` of `sofia` to the external or internal to keep the call on the same side of FreeSWITCH so you don't have to traverse internally unless you have to.

## SBC\_FreeSWITCH\_Configuration\_Example\_2

In your redirected context, you will put as many list checks in as you will be sending from your LCR engine in the 30x.

```
<context name="redirected">
  <extension name="redir_list_contact_0" continue="true">
    <condition field="${sip_redirect_contact_0}" expression="(((192.168.)|(172.24.)|(10.10.))\d+
      <action application="set" data="hangup_after_bridge=true"/>
      <action application="set" data="continue_on_fail=NORMAL_TEMPORARY_FAILURE,TIMEOUT,NO_ROUTE"/>
      <action application="bridge" data="sofia/internal/${sip_redirect_contact_0}"/>
      <anti-action application="bridge" data="sofia/external/${sip_redirect_contact_0}"/>
    </condition>
  </extension>
  <extension name="redir_list_contact_1" continue="true">
    ...
  </extension>
</context>
```

What this does is check the location of the contact to see if the destination resides on the internal or external side of FreeSWITCH. That way it does not try to send the new INVITEs out of the wrong ethernet interface.

## OpenSIPS Example

This is an very minimal example configuration to generate a 300 Multiple Choice to test this FreeSWITCH configuration with if you don't already have some LCR engines in place.

(I WILL INSERT THE EXAMPLE HERE)