A Step-by-Step guide for Caller-ID based Home or Small OfficeCall Routing

(Using Asterisk and FreePBX to route callers based on Caller ID)

Version 1.0

I have recently noted a significant increase in Robo-calls, despite the fact that I am on the "National Do Not Call" registry ("http://www.donotcall.gov"). I usually simply hang up on them, but on a few occasions I have tried to speak with them in an attempt to determine who they are. Most of the time, I am prompted to simply leave my name and number for them to get back to me. Sometimes I am simply disconnected and a very few times I have spoken to some poor sap who finally hung up on me when he realized I was not going to be a sale. I have not had the patience to string them along as has been suggested many times on the net.

With the upcoming political season I desperately wanted a way to block any callers that were not a member of the "the privileged few".

This is what I envisioned an ideal system would work like…

1) Callers with no caller ID would be blocked ruthlessly.
2) Callers with a "blacklisted" caller ID would be politely told to go away or sent directly to a voicemail
3) Callers with a "whitelisted" Caller ID would ring through
4) All other callers would be told a code that that they needed to enter in order to ring through.

   Example: *"Your Caller ID is not recognized. Please press 7 to be connected. Thank you."*

Something that met all of these would block Robo-Calls, but would be friendly to my real friends. It would not stop actual, live, solicitors, but when was the last time a "real person" called you to sell you something? For those few, I can politely (or not!) tell them that I am not interested.

Vonage, like many others, allows me to block callers without Caller ID and I have heard of other carriers that have whitelist/blacklist abilities. I searched the Internet for a solution, thinking that I can't be the first person who had had this problem. In my searching I found devices for sale that would "whitelist" or "blacklist" based on CID, but did not allow for a caller to bypass the list. I also found devices that force every caller to enter a code to ring through.

Bottom line is that I could not find a complete solution and I wasn't willing to compromise on the key features.

I have the skills and the resources to design and build a dedicated appliance to do the job, but it would be a reasonably large effort. Not worth it unless I wanted to develop it for

sale. (An option I am still toying with in the back of my mind!) In addition, I would not be able to get it in place before the election rolled around this year.

I decided that building an Asterisk server was going to be the best way to achieve my goal in a reasonable amount of time and within a reasonable cost.

A customers' computer started behaving badly in the middle of tax season by suddenly powering-off whenever more then a few applications were running. He wanted it working and working ***right now*****!** So I literally jacked up the operating system and slid a whole new computer underneath. Within hours he was operational again and I knew that every single physical component had been replaced. (No callback so far!)

Now I had a complete, quite decent, computer that had a possible random problem. I figured it would be a good test bed for my concept. If it turned out that there was a real hardware issue, then I could narrow down the culprit at my leisure.

You must understand that the following is not the path I actually took. I went down some dead ends. Installed some things. Removed them. Installed others. Banged my head on the keyboard a few times. Spent hours trolling through forums to solve each and every little thing. I found countless others who wanted to do the same things I was doing. In many places I found people who said "I did that and it works great!". Unfortunately, few of them explained ***what*** they did in enough detail for an Asterisk neophyte to follow. One major problem was that often there would be **too much** information and I would have to carefully sort the wheat from the chaff!

But, compared to other things I have done, it actually went reasonably smoothly.

Note: I tried to use the "Smartroutes" module, but 1) it seemed like it was way more complicated then I needed and 2) I could not get it to work. The problem may have been very simple and probably something I was doing wrong, but I decided not to spend any more time on it.

I call this a "step-by-step guide", but it is *not* a "click-by-click" guide. It assumes that you have installed Linux before and are comfortable with networking and navigating in Linux and Windows, using SSH or Putty and other items. I wrote this guide after I had my system running, so I may have missed some detail or glossed over some steps. If someone feels I need to explain something in more detail, feel free to let me know and I will consider it.

In some cases I selected numbers and names that made sense to me and were appropriate for my setup. None of these are critical and you can change them to suit your setup. Just remember to adjust them everywhere else.

Steps to get it running:

Bought a TDM410P clone with two FXO and 2 FXS modules from a vendor on eBay for $79 (including shipping). (http://www.ebay.com/itm/120769412880) This card does not require a separate power connection and worked perfectly with no extra drivers (Once I figured out how to make it work!)

Installed FreePBX/Asterisk 64-bit distro, using the NetInstall ISO.
(FreePBX-Distro-Net-64bit-1.811.210.57.iso)

Configured the system to recognize the TDM410P card. I had to run a script (dahdi_hardware) from the command-line to get the card and its ports to be fully recognized. Most places I looked neglected to mention this step. It was hinted at and there were some references, but it took some doing to figure it out. My card was configured with FXO modules in positions 1 and 2 and FXS modules in positions 3 and 4. You connect position 1 (Line 1) to your phone service provider (Vonage in my case) and position 3 to your phones.

I first installed the "customerdb" module but found it to be hopelessly overcomplicated for my needs. So I used it as a foundation to write my own module, which I gave the truly original name of "SmartHomeRoute". You may download it here. (http://www.eventhorizons.com/CIDrouting/smarthomeroute-1.0.tgz).

Installed the "Dynamic Route" module. (This was not the first one I tried, but the others were either inadequate, overcomplicated or simply did not work!) You can get it from the original author here (http://www.gufonero.com/asterisk/dynroute.html) or from me here (http://www.eventhorizons.com/CIDrouting/dynroute-2.8.0.0.tar.gz).

Built the following in the FreePBX GUI:

Extension as a destination for the inbound calls
Click "Applications" - "Extensions"
Click "Add Extension"
Select "Generic DAHDI Device" and click "Submit"
User Extension: "10"
Display Name: "HouseLine"
channel: 3 (See notes on the FXO/FXS card)

Click on the Red "Apply Config" on the top bar.
Restart the system to get all the above settings applied.
You should now get dial-tone on a phone plugged into the FXS port.

Recordings needed for our special IVR
Click "Admin" - "System Recordings"
Enter your extension number and click "Go"
Pick up the phone and dial "*77" and record your announcment
Name the Recording "CIDNotRecognized"
Alternate: You can download my announcement and upload it to Asterisk.

IVR to deal with callers with unknown caller ID

Click “Applications” - “IVR”
Click “Add New IVR”
IVR Name: “HouseLineIVR”
IVR Desc: “Menu for unknown CID:
Announcement: “CIDNotRecognized:
Direct Dial: “Disabled”
Timeout: “6” (I don't give them much time)
Invalid Retries: “1” (One chance to try again)
Append Original: “Check” (This will replay the announcement after a fail)
Invalid Recording: “None” (We are just hanging up)
Timeout Retries: “0”
Invalid Destination: “Terminate Call” “Hangup”
Timeout Recording: “None”
Timeout Destination: “Terminate Call” Hangup”
IVR Entries:
Ext: “7” “Extensions” “<10> HouseLine”

Dynamic Route to route the calls based on the database search results

Click “Other” - “Dynamic Routes”
Click “Add Route”
Name: “HouseLineRoute”
Host: “localhost”
Database “asterisk”
Username “root”
Password <left blank>
Query: “select destination from smarthomeroute where cid like '%[NUMBER]'”
destination1: “default” “IVR” “HouseLineMain”
destination2: “whitelist” “Extensions” “<10> HouseLine”
destination3: “blacklist” “Terminate Call” “Hangup”
Click “Save”

Trunk as a connection to the outside world (PSTN or POTS)

Click “Connectivity” - “Trunks”
Click “Add Zap Trunk (DAHDI compatibility Mode)”
Trunk Name: “HouseLine1”
Outbound CID: “<xxx-xxx-xxxx>” (enter your number here)
Maximum Channels: “1”
Zap Identifier: “1” (See notes on the FXO/FXS card)

Inbound route to bring the calls into “Dynamic Route”

Click “Connectivity” - “Inbound Routes”
Click “Add Incoming Route”
Description: “HouseLineInbound”
Set Destination: “Dynamic Routes” “HouseLineRoute”

Click “Submit”

Outbound Route to allow me to call out

Click “Connectivity” - “Outbound Routes”
Click “Add Route”
Route Name: “NormalDialing”
Dial Patterns: In field “match pattern” place a single “period”
(This will cause all numbers to match, so all calls will use this route)
Trunk Sequence 0: “HouseLine1”
Click “Submit Changes”

Click on the Red “Apply Config” on the top bar.

At this time you should be able to make outbound calls and inbound calls will be filtered according to the “SmartHomeRoute” database. You can add numbers to the database manually as follows:

Click “Other” - “SmartHome Routes”
Click “Add SmartHome Route”
Name: <any name that makes sense to you>
Caller ID: <exactly as received>
Destination: “whitelist” or “blacklist”

This is actually a very simple arrangement and only scratches at what Asterisk is capable of.

There are those who will point out the many failings and security holes in this setup. And they are legion. This system is only intended to be operated in a friendly environment (home network behind a firewall). If you want to harden it against assault, then feel free to do so. This is just to get you started.

Future Improvements list:

Feature codes to add/delete from an instrument.
*40 - Add/Delete number on any list
*41 - Add last caller to Whitelist
*42 - Add last caller to Blacklist

Feature code to allow all calls through for a specified time:
*49 - Allow all calls for 30 minutes

Distinctive Ringing: To tell the difference between a Whitelist caller and a caller who entered the correct code.

Individual Selection: Additional “destinations” and IVR selections to choose specific members of my household, including personal voicemail boxes and distinctive rings for each person. I won't even need to get up if it is ringing for my daughter!