Raspi Asterisk PBX Google Assistant Interface

version 1.0 – 2017-05-02 – initial

Overview

This project is a proof-of-concept using Asterisk PBX, running on a Raspberry Pi Zero or better, interfaced to Google Assistant™ Voice Service SDK & API. Using a SIP Phone or SoftPhone, the user dials into their Raspberry Asterisk PBX extension and follows the prompts to speak questions which are sent to Google Assistant Voice Service. Responses are spoken back over the phone. No Google Home™ required.

> *The Google Assistant SDK lets you add voice control, natural language understanding and Google's smarts to your devices. Your device captures an utterance (a spoken audio request, such as* What's on my calendar?*), sends it to the Google Assistant, and receives a spoken audio response in addition to the raw text of the utterance.*

I used a Perl AGI script and SOX to convert the audio formats and send & receive the audio files via Google's python Google Assistant SDK via the Internet.

Google offers a free developer preview of this service but does require signing up for a Google Developer Account (at no charge).

As this is only a proof-of-concept, it does meet their developer requirements for private, non-commercial use. The primary limitation is that you cannot connect an external telephone number to

this for use as an inbound service currently, unless you can meet their terms of service (i.e. paid service).

Importantly, this proof-of-concept has not been optimized to reduce overhead and improve response time. But for a single user demo, the Raspi Pi and Google Assistant does work, if not quite as good as Amazon's Alexa™ voice service for Raspberry. See my RaspiAsteriskAlexa app on Github. https://github.com/rgrokett/RaspiAsteriskAlexa

If you already have the RaspiAsteriskAlexa project running, this Google Assistant project can run on the same Raspberry Pi at the same time.

Requirements

- Raspberry Pi Zero, B+, 2 or 3
- Raspian Jessie installed and up to date
- Pi ethernet or Wifi connected (direct Ethernet is best) with Internet access
- SSH access to the Pi
- Software from GitHub:
  https://github.com/rgrokett/RaspiAsteriskGoogle
- A SIP/VOIP Phone (Such as Grandstream GXP1620) or SoftPhone such as:
  http://www.asteriskguru.com/tutorials/xlite_softphone.html
  http://www.zoiper.com/en/voip-softphone/download/zoiper-classic

This project assumes some knowledge of Raspian command-line, SSH and some beginning knowledge of Asterisk PBX. It also assumes you have a Raspberry Pi configured with Raspian Jessie (Lite version is fine!) and connected to your network. This does not require a GUI on the Pi, so assumes headless operation and a login prompt, terminal or SSH access.

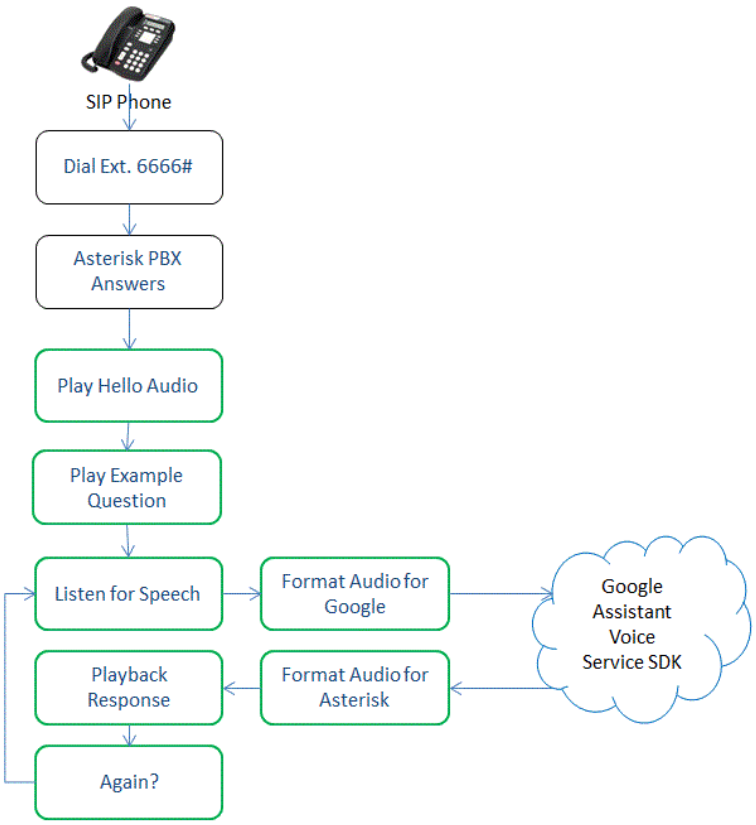

Fig. 1 – Call Flow for Asterisk-to-Google Assistant

Asterisk PBX Configuration

The Pi needs outbound Internet access but the Asterisk PBX does NOT need any external SIP Voice Provider. This project does not use inbound or outbound voice calls.

If you have never used Asterisk, you probably should look up some Asterisk tutorials. Though, most of the information isn't needed here. http://www.asterisk.org/
Note that this does NOT use the "FreePBX for Raspberry Pi" package, but instead just uses the Raspian command line install version.

This project should be done on a CLEAN RASPIAN JESSIE installation. Using an existing installation may have had modifications that impact Asterisk or the Google SDK.

1. Install an Asterisk PBX on your Raspberry Pi:
   ```
   $ sudo apt-get update
   $ sudo apt-get upgrade
   $ sudo apt-get install asterisk git
   ```
   This installs quite a few dependencies, so can take a while!

2. Retrieve the project files from Github: (https://github.com/rgrokett/RaspiAsteriskGoogle)

```
$ cd /home/pi
$ git clone https://github.com/rgrokett/RaspiAsteriskGoogle.git
```

3. Update the Asterisk PBX to add the SIP and Extension entries used by this project.

```
$ cd /home/pi/RaspiAsteriskGoogle
$ bash ./install.sh
```

NOTE:

- You will see "*No client_secret.json…*" and "*No Google API OAUTH…*" error messages, as these are set up later. Just ignore them for the moment.
- It is OK to execute the install.sh script more than once. It will verify installation.

Below is a table showing what is updated in Asterisk using the **install.sh** script.
(Don't worry if you don't know Asterisk PBX configs, yet!)

Appended to /etc/asterisk/sip.conf

```
[5310]
type=friend
username=5310
fromuser=5310
host=dynamic
context=local
insecure=port
qualify=500
dtmfmode=rfc2833
disallow=all
allow=ulaw
obtained
progressinband=no
nat=no
mailbox=5310
callerid=5310
```

Appended to /etc/asterisk/extensions.conf

```
include => phone
include => google_api

; Basic SIP Phone
[phone]
exten => 5310,1,Dial(SIP/5310,15)
exten => 5310,2,Voicemail(5310,u)
exten => 5310,3,Hangup
exten => 5310,102,Voicemail(5310,b)
exten => 5310,103,Hangup

; GOOGLE ASSISTANT VOICE
```

```
[google_api]
exten => 6666,1,Answer()
; Play prompts
exten => 6666,n,Playback(./custom/google_hello)
exten => 6666,n,Playback(./custom/google_example)
; Google Assistant SDK API integration
exten => 6666,n(record),agi(google.agi,en-us)
; Loop
exten => 6666,n,Playback(./custom/google_another)
exten => 6666,n,goto(record)
```

4. Add a SIP Phone to your Asterisk. Since there are many ways to do this, I am showing a generic configuration. Each phone has different options, but this should be all that is needed.
   See the "Requirements" section, above, for SIP (VOIP) softphones. (PC based software. Requires headset with microphone) or use a physical VOIP telephone, such as the Grandstreams avail from Amazon. The cheap ones under $50 are fine for this.

   This is adding a SIP phone with extension number 5310 to Asterisk. Remember, this does NOT include inbound or outbound calling. Only internal calls.

   Configure your SIP Phone with the options below. Most other phone options are not used, so can usually be ignored.

SIP Phone options:

```
SIP Proxy Server: 192.168.1.XX      <- The IP Address of your Raspi  $ hostname -I
Domain/Realm/Registration: <same IP as above>
Username: 5310
Password:  <blank>
```

   Note that you enter the *SIP Proxy IP* (i.e. your Pi's IP address) into the **EXTENSION** portion of your phone. (Some phones have multiple extensions, but this project needs only 1.)

Fig. 2 – Example of a VOIP Extension configuration screen

5. Go off-hook with your SIP/VOIP Phone and see if you can get dial tone.
   If so, try dialing **1000# on your SIP Phone**. You should get a built-in demo.

Debugging:

If you don't get dial tone, your SIP Phone isn't registering to your Asterisk server. Try the following:

a. Reboot the Raspberry and then the SIP phone to see if they reconnect.
b. Verify the files above are edited correctly.
c. Access Asterisk from command line:
```
$ sudo asterisk -r
CLI>  sip show peers
```
This should show your sip phone's IP and status
Watch out for firewalls, particularly if using a softphone!
d. Verify IP addresses of the Sip Phone and Raspi and be sure you can ping the phone from the Raspi.
e. If still an issue, turn on SIP Debugging:
```
CLI> sip set debug on
```
This will display SIP connection attempt messages. If you see nothing after a few minutes, then your SIP Phone isn't even trying to talk to Asterisk. You need to dig into the SIP Phone's setup instructions.
f. Asterisk log messages go to `/var/log/asterisk/messages`. But note there can be lots of nasty looking messages, most are for unused/unneeded features. So it can be difficult to decipher.
g. Google is your friend. Cut/paste any error messages you see into it to learn more!

If you have successfully gotten dial tone and the “Congratulations” demo, then you now have a working Asterisk PBX!

Google Assistant SDK Install

Now you need to create a (free) account on Google Developers site and install their Python Google Assistant SDK for Raspberry Pi.

> *Excerpted from Google Assistant SDK Getting Started with the Raspberry Pi and Python:*
> https://developers.google.com/assistant/sdk/prototype/getting-started-pi-python/config-dev-project-and-account

Configure a Google Developer Project

1) Go to Google Projects Page: https://console.cloud.google.com/project
   You may need to create a Google Developers account or use your existing Google (gmail) account credentials to log in.
2) Create a new project called “AsteriskSpeechAPI” (or any desired name you wish)
3) Enable the Google Assistant API for the project you selected:
   https://console.developers.google.com/apis/api/embeddedassistant.googleapis.com/overview
   Click **Enable**
4) Create an Oauth Client ID with following steps:
   a) Create the client ID: https://console.developers.google.com/apis/credentials/oauthclient
   b) If asked, give the same project name above for the product name on the *OAuth consent screen* tab and click **Save**.
   c) Click **Other** and give the client ID a name (again, same as above is fine).
   d) Click **Create**. A dialog box will appear that shows you a client ID and secret.
   No need to remember or save this, just close the dialog
   e) Click the down arrow on the far right of the screen to download the client ID and secret as a JSON file called (client_secret_XXXX.json)
   (To re-retrieve the credentials again, go to the Google Assistant API Manager, step 3 above, and select Credentials and click the download icon for your project.)
   f) Copy the **client_secret_XXXX.json** file to **/home/pi/client_secret.json** on your Raspi.
   (Rename it to just **client_secret.json)**
   You can use cut/paste to do this, if you like:
   ```
   $ cd /home/pi
   $ nano /home/pi/client_secret.json
   ```
   Paste into the editor and save

5) Google requires you to share certain activity data with Google to use their SDK. The Assistant needs this information to function properly. (You can turn off again once you no longer want to

use the Assistant SDK.)

a) Go to Activity Controls page: https://myaccount.google.com/activitycontrols

b) Set the following ON (blue):

- Web & App Activity
- Location History
- Device Information
- Voice & Audio Activity

Install & Compile Google Assistant SDK for Raspi

1) Get the Google Assistant Python SDK software and install:

   On your Raspi, execute the following:

```
sudo apt-get update
sudo apt-get install python-dev
sudo apt-get install portaudio19-dev libffi-dev libssl-dev
sudo apt-get install python-pip
sudo python -m pip install PySocks
sudo python -m pip install google-assistant-sdk[samples]
```

   NOTE: On a Pi Zero, this compiles for about an hour or more with lots of messages!

   Once finished, reboot your Pi:

```
$ sudo reboot
```

Set up a Google OAUTH authorization:

1) On your Pi, execute following:

```
$ python -m googlesamples.assistant.auth_helpers --client-secrets
/home/pi/client_secret.json
```

You should see something like:

```
Please go to this URL: https://... Enter the authorization code:
```

Copy the long URL it returns and paste it into your browser (any browser on any PC).

Select the account you used to set up your project.

After you approve, a code will appear in your browser, such as "**4/XXXX**". Copy this and paste this code into the Raspi terminal. It should display as shown below:

```
credentials saved (and a path)
```

If instead it displays: `InvalidGrantError` then an invalid code was entered. Try again, taking care to copy and paste the entire code.

This saves your oauth token into

`/home/pi/.config/googlesamples-assistant/assistant_credentials.json`

Test the Google Assistant SDK

1) Execute the test program:

```
$ cd /home/pi/RaspiAsteriskGoogle
$ bash ./test.sh
```

You should see something like:

```
Found client_secret.json OK
Found OAUTH assistant_credentials.json OK
Sending sample to Google Assistant API...

DEBUG:google.auth.transport.requests:Making request: POST
https://accounts.google.com/o/oauth2/token
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1):
accounts.google.com
DEBUG:urllib3.connectionpool:https://accounts.google.com:443 "POST /o/oauth2/token
HTTP/1.1" 200 None
INFO:root:Connecting to embeddedassistant.googleapis.com
INFO:root:Recording audio request.
DEBUG:root:ConverseRequest: config {
  audio_in_config {
    encoding: LINEAR16
    sample_rate_hertz: 16000
  }
  audio_out_config {
    encoding: LINEAR16
    sample_rate_hertz: 16000
    volume_percentage: 50
  }
}

DEBUG:root:ConverseRequest: audio_in (6400 bytes)
DEBUG:root:ConverseRequest: audio_in (6400 bytes)
DEBUG:root:ConverseRequest: audio_in (6400 bytes)
DEBUG:root:ConverseRequest: audio_in (6400 bytes)
DEBUG:root:ConverseRequest: audio_in (6400 bytes)
DEBUG:root:ConverseRequest: audio_in (6400 bytes)
DEBUG:root:ConverseRequest: audio_in (6400 bytes)
DEBUG:root:ConverseRequest: audio_in (6400 bytes)
DEBUG:root:ConverseRequest: audio_in (4096 bytes)
DEBUG:root:ConverseRequest: audio_in (3200 bytes)
…(more)

DEBUG:root:ConverseResponse: event_type: END_OF_UTTERANCE

INFO:root:End of audio request detected
DEBUG:root:ConverseResponse: result {
  spoken_request_text: "what is the current weather"
}

DEBUG:root:ConverseRequest: audio_in (3200 bytes)
INFO:root:Transcript of user request: "what is the current weather".
INFO:root:Playing assistant response.
DEBUG:root:ConverseResponse: result {
  conversation_state: "\n&C#590…6f7-0000-2231-932c-
94e…3c9ba\022\270\001Kj93Nlg1eVpmWWMGlLSURpYUpWdkpXb2hRUW43YnNIcmdBQUFDQ1RjV0… "
  microphone_mode: CLOSE_MICROPHONE
}

DEBUG:root:ConverseResponse: audio_data (1600 bytes)
DEBUG:root:ConverseResponse: audio_data (1600 bytes)
DEBUG:root:ConverseResponse: audio_data (1600 bytes)
DEBUG:root:ConverseResponse: audio_data (1600 bytes)
DEBUG:root:ConverseResponse: audio_data (1600 bytes)
DEBUG:root:ConverseResponse: audio_data (1600 bytes)
DEBUG:root:ConverseResponse: audio_data (1600 bytes)
DEBUG:root:ConverseResponse: audio_data (1600 bytes)
DEBUG:root:ConverseResponse: audio_data (1600 bytes)
DEBUG:root:ConverseResponse: audio_data (1600 bytes)
…(more)
DEBUG:root:ConverseResponse: audio_data (1044 bytes)
INFO:root:Finished playing assistant response.
```

```
AUDIO FILES:
-rw-r--r-- 1 pi   pi    55340 Apr 30 17:02 in.wav
-rw-r--r-- 1 root root 252288 May  1 14:51 out.wav

If you have AUDIO on this Pi:
Verify Audio with:
   $ speaker-test -t wav
Playback audio from Google:
   $ aplay --format=S16_LE --rate=16k --file-type=wav out.wav
```

If successful, this would have sent a pre-recorded audio file in.wav to Google Assistant and received back an audio file out.wav response. You can copy these to a PC to play or play them on a Raspi that has speakers attached.

You are now ready to try out your Asterisk PBX with Google Assistant.

1. Rerun the install.sh script again to be sure the OAUTH and client JSON are OK:

```
$ cd /home/pi/RaspiAsteriskGoogle
$ bash ./install.sh
```

```
$ bash ./install.sh

Copied AGI and audio files to Asterisk
Found client_secret.json OK
Found OAUTH assistant_credentials.json OK
Found Google Extension 6666# already installed OK
Found SIP Extension 5310# already installed OK
```

2. Reboot your Raspberry Pi.

3. Using your SIP Phone, dial 6666#
   You should dial tone, ringing and then audio prompts.
   If you do, GREAT! Try any questions to Google:
   - What time is it?
   - What is the weather in Atlanta, Georgia?
   - How big is the Earth?
   - What time is it in London, England?
   - Tell me a Joke
   - Etc…

NOTE: On the Pi Zero, there is a 5 to 7 second delay for responses due in part to the python SDK API overhead. A faster Pi will reduce this.

Just hang up when done. It will reset automatically.

TROUBLESHOOTING

1) You did get dial tone and able to call the Asterisk demo (1000#) didn't you?

2) Turn on Debug mode in AGI program:

```
$ cd /home/pi/RaspiAsteriskGoogle
$ nano google.agi
```

Change `my $debug = 0;`

To `my $debug = 1;`

Save

```
$ bash ./install.sh
```

3) Access Asterisk CLI to debug:

```
$ sudo asterisk –r
CLI> agi set debug on
CLI> core set debug 4
```

4) Make a call to 6666# again and watch the messages.
5) You can look at **/home/pi/RaspiAsteriskGoogle/google.agi** for matching debug messages to code. Note that Asterisk displays lots of extra debug info beyond the messages

   If you make changes to the Asterisk extensions.conf or sip.conf, you should restart Asterisk

   `CLI> core restart now`

   If you make changes to **google.agi**, execute the install script again to install

   `$ bash ./install.sh`

6) If the phone rings w/o answer, then **extensions.conf** has a problem in the 6666 section.
   If you manually edited extensions.conf, you must reload using `CLI> dialplan reload`

7) If the phone answers but no audio, verify that the files in the asterisk directories are owned by asterisk userid:

```
sudo chown asterisk:asterisk /usr/share/asterisk/agi-bin/google.agi
sudo chown asterisk:asterisk /usr/share/asterisk/sounds/custom/google*.sln
```

8) Look for audio files created in "/tmp" directory:

```
$ ls –l /tmp
-rw-rw---- 1 asterisk asterisk 32266 May  3 02:01 google_audio3424_8k.sln
-rw-rw---- 1 asterisk asterisk 73004 May  3 02:01 google_audio3424_in.wav
-rw-rw---- 1 asterisk asterisk 64576 May  3 02:01 google_audio3424_out.wav
-rw-rw---- 1 asterisk asterisk  6124 May  3 02:01 google_audio3441.wav

google_audio3441.wav      -- Audio of your question
google_audio3424_in.wav   -- Audio formatted for Google
google_audio3424_out.wav  -- Audio response from Google
google_audio3424_8k.sln   -- Audio formatted for Asterisk
```

9) If you get no response back from Google for your question, run the **test.sh** script again to verify the OAUTH token and SDK are good.

```
$ bash ./test.sh
```

If all else fails, step back thru the commands above to be sure you didn't miss a step! There are many steps, so it's easy to accidentally skip one.

Results

I have found that the speech recognition of Google Assistant is excellent when used with a SIP telephone and Asterisk PBX. The recognition appears to be quite good whether on a handset or speakerphone. The Raspberry Pi Zero is able to handle the Asterisk audio processing but is slow in running the SDK. Of course, the wide range of question/answers that Google Assistant can handle is equivalent to Google Home™ , without the Home control or Wake Word functions.

Watch the Google Assistant SDK developers website for future updates!
https://developers.google.com/assistant/sdk/