Always on “Ok Google” Technology Stack Integration

GOOGLE CONFIDENTIAL AND PROPRIETARY

Contact: **hotword-partnerships@google.com**

Table of contents:

Revision History

11/04/2015 - Added information about whitelisting the account and updated contact information.
12/08/2014 - Added details to DSP triggering and also more integration steps.
11/25/2014 - First version of the document

Background

The purpose of this document is to outline the work items that are needed in order to implement Google’s Always-On hotword technology stack for Android devices. The following diagram is a high level representation of the multi-stage architecture of the solution:

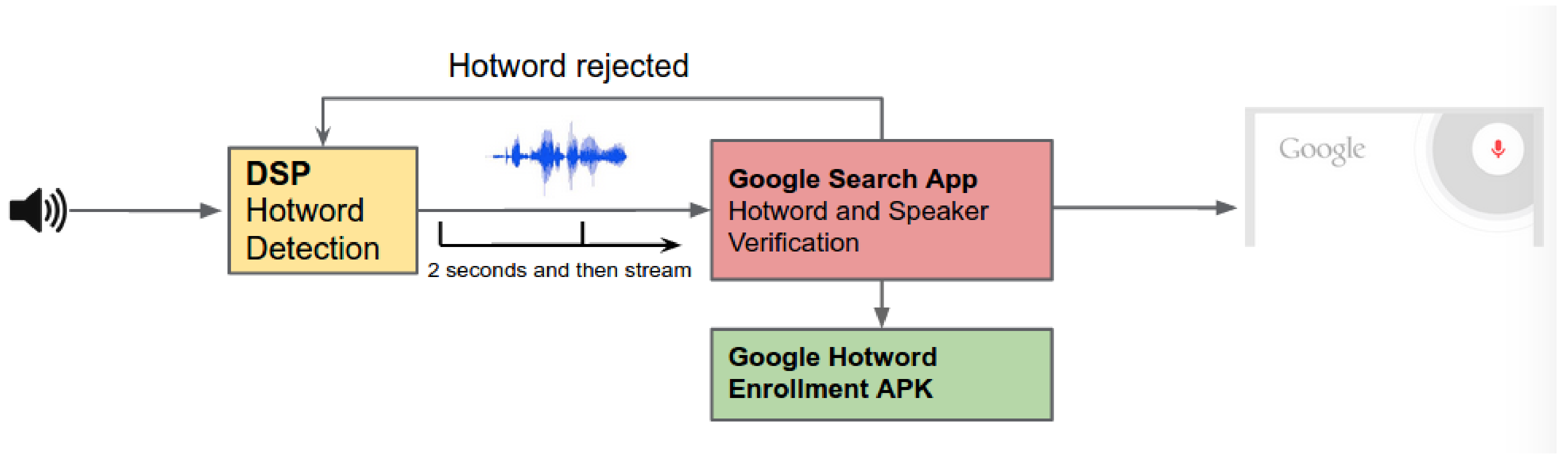
Hotword rejected
DSP Hotword Detection
2 seconds and then stream
Google Search App Hotword and Speaker Verification
Google Hotword Enrollment APK
Google

Figure 1: Google’s Two Stage Hotword Architecture

System Requirements

Android OS

System should be running at least Android OS version 5.0 (Lollipop).

Google Search App

System should include the Google App 4.0 or higher in the system partition.

Google Hotword Enrollment APK

The Google Hotword Enrollment APK will be provided by Google and should be placed in the system partition under `/system/priv-app/`, or in the vendor partition such that it has system privileges.

DSP

Google has ported its “Ok Google” detector to several architectures/manufacturers. Please contact us for a list of supported DSP and requirements in terms of memory and processing power.

- Google needs the DSP to be able to buffer the 2 seconds of 16kHz audio before the trigger happened.
- After a trigger, the DSP driver, kernel and the overall system should function as follows:
  - The DSP should keep running the buffer until the AP wakes up. The propagation should happen fast, preferably with <100ms of latency (example: via SPI).
  - The DSP should start streaming the captured audio as an AudioRecord to the Google app, which will get a hold of it using `getCaptureSession()` of AlwaysOnHotwordDetector. This results in the Audio HAL receiving a request for an AUDIO_SOURCE_HOTWORD.
  - Once the Google app confirms the hotword in software, it will continue to use the same audio stream to perform the voice search.
- Trigger-to-beep/vibrate latency (screen off) must be in the order of 250-350ms, measured as the difference between the end of “Ok Google” and the Google Search App waking up.

Kernel Driver

Google will work with the DSP vendors to provide driver code to the OEM that will integrate into their kernel to support transferring the audio buffer/loading the sound models. The OEM will need to integrate this driver to properly wake up the AP and begin either streaming audio to the Google Search App, or to the OEM’s application.

SoundTrigger HAL

An implementation of the SoundTrigger HAL interface must be put in place in the system since “Ok Google” relies on it. A sample implementation can be found at:

- **sound_trigger_hw.c sample implementation**

Note that there needs to be some coordination with the audio hal, to process the AUDIO_SOURCE_HOTWORD to fetch from the streaming DSP buffer (sample change).

Suggested Integration Steps and Testing

The following steps should serve just as guidance of what the expected behavior of the system should be as the development moves forward.

1. Make Google Search App v4.4 or higher and the provided Google Hotword Enrollment APK part of the system image of the device.
    a. **Result:** Google App should be default Assistant App under Assist & Voice input settings.
    b. **Result:** Clicking on the settings button should go straight to the Google Search App

2. Implement dummy version of SoundTrigger HAL on the Java framework end that makes the system aware of the existence of a DSP that supports "Ok Google":
    a. During device bringup, you must be using a whitelisted google account on the device for this step to work, and you must be connected to the internet. Google can provide you with such an account. Once a device is ready to ship, the device itself will be whitelisted.
    b. onAvailabilityChanged should be called at least once with the initial availability
    c. getSupportedRecognitionModes should return RECOGNITION_MODE_VOICE_TRIGGER
    d. **Result:** "*Always on"* option shows up under **`Google Settings -> Search & Now -> Voice -> "Ok Google" detection.`** Google Search App acknowledges the existence of Always On Hotword.

3. Test that enrolling in the Google app pushes the hotword model down to the SoundTrigger HAL
    a. Google App will call createAlwaysOnHotwordDetector with **`keyphrase = "Ok Google"`** and with the BCP47 locale currently selected at **`Google Settings -> Search & Now -> Voice -> Languages.`** For English (US) this is **`"en-US"`**
    b. Go to **`Google Settings -> Search & Now -> Voice -> "Ok Google" detection`** and enable always-on.
        i. Please contact hotword-partnerships@google.com to obtain a testing account.
    c. Say "Ok Google" 3 times. Once completed, Google app will talk to the hotword enrollment APK to make it push a model to the SoundTrigger HAL
    d. **Result:** SoundTrigger HAL should receive the exact same bytes that were pushed through the Java framework layer.

4. Implement a dummy version of the firmware running on the DSP that pretends to trigger a hotword.
    a. Suggest to implement a simple loop that triggers a hotword every 10 seconds, for example.
    b. **Result:** Once pushed and manually started, saying "Ok Google" should raise a kernel interrupt every 10 seconds.

5. Implement a version of SoundTrigger HAL that is able to start the dummy DSP firmware and trigger a voice search every 10 seconds.
    a. Ignore audio buffering at this point by pretending that the Google firmware of the DSP supports user identification. To do this, getSupportedRecognitionModes should return RECOGNITION_MODE_USER_IDENTIFICATION
    b. **Result:** After enrolling using the Google app (**`Google Settings -> Search & Now -> Voice -> "Ok Google" detection`** and enable always-on.) a Google search starts every 10 seconds.

6. Implement a dummy version of the firmware that pretends to trigger a hotword **and passes canned audio up**
    a. Modify implementation in (4) to pass a canned audio file containing "Ok Google" in streaming fashion up to the HAL.
    b. Google App needs to be able to read this buffer via `getCaptureSession()` of AlwaysOnHotwordDetector (hidden API)
    c. **Result:** Every 10 seconds, the Google app is able to trigger on the Ok Google audio. If any other audio is passed, it should not trigger.

7. Support Audio Buffering: DSP now can stream the audio up to the application layer via AudioRecord
    a. Add ability for DSP to stream the audio input stream up whenever it triggers as expressed in requirements. it.
    b. Google App needs to be able to read this buffer via `getCaptureSession()` of AlwaysOnHotwordDetector (hidden API)
    c. Can still use dummy DSP firmware that simply triggers manually and passes the audio up.
    d. **Result:** User identification is now conducted in the Google App. Recognition mode can now be switched to RECOGNITION_MODE_VOICE_TRIGGER

8. Get "Ok Google" firmware running on the DSP chip (not hooked up to the HAL yet - can be done in parallel with steps 1-7)
    a. Google can provide the OEM and DSP manufacturer with a self-contained binary for testing that will trigger at the hardware level and can be started manually using adb commands.
    b. This allows screen-off power consumption to be measured and verified on the device. Any power optimizations should be done at this stage.
    c. **Result:** Once pushed and manually started, saying "Ok Google" should raise a kernel interrupt that can be logged in a way that can be viewed through the kernel log. Remember that this is a coarse model, so it is expected that it over triggers.