Amazon Alexa Voice Service (AVS): Run and Test Sample app on Mac – Java Client and iOS Simulator

Hi Reader,

Earlier I talked about how to create and develop Alexa Skills from scratch and to use it around AWS Lambda function, SSML and Amazon S3 storage. Also, I recommended using echoism.io cloud service for testing the Alexa skills. Here you can refer to learn and start with. Today, I am delighted to share the very straightforward practical approach of using Alexa Skills and Alexa Voice Service (AVS) together – even we could find better accuracy and performance with that especially into the client section side. For this post, I restricted or tested app over Java Client and iOS simulator into Mac machine. More importantly, with this post, we can learn to build or enabled the Alexa skills without dependency of real Amazon Echo hardware (i.e., personal assistant). Let's start:

Important Links to refer for implementation the AVS (Bring your capabilities to connected devices) skill app:

- https://github.com/alexa
- https://github.com/alexa/alexa-avs-sample-app/wiki/Mac
- https://github.com/alexa/alexa-avs-sample-app/wiki/Create-Security-Profile

With this post, I'll highlight the gaping experience takeaway points – should be nice to include into the content of above links for developer comfort and understanding. Though the mentioned links are straightforward and have complete steps for implementation, running or testing the AVS sample app, but at the certain point being a developer I found little difficulty with that:

We need to first follow environment set-up checklist into our development machine before jumping into the coding section:

Install New Software:

- Ensure you have JDK version 8 installed. If JDK is not installed or you need to update from a previous version, you can download version 8 here. **Note:** As mentioned earlier, please ensure that your JDK's architecture matches the architecture for VLC. The sample worked only in JDK 1.8.
- Download VLC media player here.
- Download and install Node.js.
- Download Maven and follow the instructions for Installing Apache Maven.
- If you plan to authenticate using an iOS companion app, download Xcode for Mac from the Mac App Store.
- Also install or update Homebrew (brew) package manager to install OpenSSL (openssl), NVM, Maven (mvn) package via Terminal.

Once finished with the installations – check version of latest tools/software into the Mac-Terminal and update all if it's already been installed earlier in system.

5. After you check the version of installed tools/software, set the following into the environment variables via your favourite editor:

$ open ~/.bash_profile

```
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home

##

# Your previous /Users/ranbijaykumar/.bash_profile file was backed up as
/Users/ranbijaykumar/.bash_profile.macports-saved_2016-12-28_at_14:12:48

##

# MacPorts Installer addition on 2016-12-28_at_14:12:48: adding an appropriate PATH variable
for use with MacPorts.

export PATH="/opt/local/bin:/opt/local/sbin:$PATH"

# Finished adapting your PATH environment variable for use with MacPorts.

export VLC_PATH=/Applications/VLC.app/Contents/MacOS/lib

export VLC_PLUGIN_PATH=/Applications/VLC.app/Contents/MacOS/plugins
```

Once you set the path, save the file and close it. Come back to the terminal:

Once you set the path, save the file and close it. Come back to the terminal:

$ source ~/.bash_profile

Getting Started with the Alexa Voice Service (AVS):

Download the sample app into the project directory:

$ mkdir /Users/ranbijaykumar/**Alexa-AVS-Sample**

Note: /Users/ranbijaykumar/Alexa-AVS-Sample/. Further instructions will reference this location as <REFERENCE_IMPLEMENTATION>

$ cd /Users/ranbijaykumar/**Alexa-AVS-Sample**

$ git clone https://github.com/alexa/alexa-avs-sample-app.git (Hope you have already installed github package)

*By downloading this package, you agree to the Alexa Voice Service Agreement.

Register for a free Amazon Developer Account:

Get a free Amazon developer account if you do not already have one:
https://developer.amazon.com/login.html

Register your product and create a security profile.

1. Login to Amazon Developer Portal - developer.amazon.com

2. Click on Apps & Services tab -> Alexa -> Alexa Voice Service -> Get Started

3. In the Register a Product Type menu, select **Device**

4. Fill in and save the following values:

Device Type Info:

- Device Type ID: **my_device**
- Display Name: **My Device** Then click **Next**

Security Profile

5. Click on the Security Profile dropdown and choose "**Create a new profile**":

6. **General Tab**

- **Security Profile Name**: Alexa Voice Service Sample App Security Profile
- **Security Profile Description**: Alexa Voice Service Sample App Security Profile Description
- Click **Next**

The **Client ID** and **Client Secret** will be generated for you.

7. Now click on the **Web Settings Tab**

- Make sure the security profile you just created is selected in the drop-down menu, then click the **"Edit"** button.
- **Allowed Origins**: Click "**Add Another**" and then enter **https://localhost:3000** in the text field that appears.
- **Allowed Return URLs**: Click "Add Another" and then enter **https://localhost:3000/authresponse** in the text field that appears.
- Click **Next**

8. **Device Details**

1. [Optional] Image: Pick the test image to your computer, then upload it:
2. Category: **Other**
3. Description: **Alexa Voice Service sample app test**
4. Do you have plans to make your product available to the general public? - **No**
5. Click **Next**

9. **We are now ready to generate self-signed certificates**

1. Open a web browser, and visit https://developer.amazon.com/lwa/sp/overview.html.
2. Near the top of the page, select the security profile you created earlier from the drop down menu and click **Confirm**.
3. Enter a privacy policy URL beginning with http:// or https://. For this example, you can enter a fake URL such as http://example.com.
4. [Optional] You may upload an image as well. The image will be shown on the Login with Amazon consent page to give your users context.
5. Click Save.
6. Next to the Alexa Voice Service Sample App Security Profile, click Show Client ID and Client Secret. This will display your client ID and client secret. Save these values. You'll need these.

*Also refer here for above steps with screenshot - Create a device and security profile

Generate self-signed certificates

Change directories to <REFERENCE_IMPLEMENTATION>/samples/javaclient. In this instance, the <REFERENCE_IMPLEMENTATION> is "/Users/ranbijaykumar/Alexa-AVS-Sample/alexa-avs-sample-app/".

Step 1: Edit the text file **ssl.cnf**, which is an SSL configuration file. Fill in appropriate values in place of the placeholder text that starts with **YOUR_**.

$cd <REFERENCE_IMPLEMENTATION>/samples/javaclient - //your sample apps location

$ edit ssl.cnf , will open file in your desired editor

Note: countryName must be two characters (e.g. US). If it is not two characters, certificate creation will fail. Additionally, if you will be accessing your device from any IP or DNS entry besides localhost (127.0.0.1 or 10.0.2.2), you must add the additional IP or or DNS entries to [alt_names]. One situation where you will need to add entries to [alt_names] is if you are going to authenticate using an Android or iOS companion app from a device instead of from the Android or iOS emulators on the same machine as the Node.js server and sample app. Here's what the **ssl.cnf** file would look like, replacing country, state, locality with your respective info.

```
[req]
distinguished_name      = req_distinguished_name
prompt                  = no

[v3_req]
subjectAltName          = @alt_names

[alt_names]
DNS.1                   = localhost
IP.1                    = 127.0.0.1
IP.2                    = 10.0.2.2

[req_distinguished_name]
commonName              = $ENV::COMMON_NAME                 # CN=
countryName             = US                                # C=
stateOrProvinceName     = CA                                # ST=
localityName            = San Diego                         # L=
organizationName        = SaturnMob                         # O=
organizationalUnitName  = None                              # OU=
```

Save the file.

Step 2: Make the certificate generation script executable by typing:

```
$ chmod +x generate.sh
```

Step 3: Run the certificate generation script:

```
$ ./generate.sh
```

Step 4: You will be prompted for some information:

- When prompted for a product ID, enter the `productID` (or Device Type ID) listed under "Device Type Info" in the Amazon developer portal. For this prototype enter **my_device**
- When prompted for a serial number, enter your product's serial number. For this prototype enter **123456**
- When prompted for a password, enter any password and remember what you entered (you can even leave it blank). For this prototype leaving password field blank.

Step 5: Edit the configuration file for the Node.js server

The configuration file is located at: <REFERENCE_IMPLEMENTATION>/samples/companionService/config.js. Make the following changes:

```
$ edit config.js
```

- Set **clientID:** paste in the client ID as a string that we noted in the earlier step while creating "security profile" from here.
- Set **clientSecret**: paste in the client ID as a string that we noted in the earlier step while creating "security profile" from here.
- Set **sslKey** to <REFERENCE_IMPLEMENTATION>/samples/javaclient/certs/server/node.key
- Set **sslCert** to <REFERENCE_IMPLEMENTATION>/samples/javaclient/certs/server/node.crt
- Set **sslCaCert** to <REFERENCE_IMPLEMENTATION>/samples/javaclient/certs/ca/ca.crt
- `Set` **`products`**: The product's object consists of a key that should be the same as the product type ID (also referred as Device Type ID) that we set up in the developer portal and a value that is an array of unique product identifiers. If we followed the instructions above, the product type ID should be my_device. The unique product identifier can be any alphanumeric string, such as 123456. Example products JSON is: products: {"my_device": ["123456"]}. For this project, the array should be a single value, and match the serial number you entered while generating certificates. Please be noted the password field is blank.

Here's what **config.js** file look like:

```
/**
 * @module
 * This module defines the settings that need to be configured for a new
 * environment.
 * The clientId and clientSecret are provided when you create
 * a new security profile in Login with Amazon.
 *
 * You will also need to specify
 * the redirect url under allowed settings as the return url that LWA
 * will call back to with the authorization code.  The authresponse endpoint
 * is setup in app.js, and should not be changed.
 *
 * lwaRedirectHost and lwaApiHost are setup for login with Amazon, and you should
 * not need to modify those elements.
 */
var config = {
    clientId: 'amzn1.application-oa2-client.96f185e06bc74b87942efcfd678db4c7',
    clientSecret: '91f52b503376eaadd56d5589acdf45c885b458086cf0f54e52149415ac8fcea7',
    redirectUrl: 'https://localhost:3000/authresponse',
    lwaRedirectHost: 'amazon.com',
    lwaApiHost: 'api.amazon.com',
    validateCertChain: true,
    sslKey: '/Users/ranbijaykumar/Alexa-AVS-Sample/alexa-avs-sample-app/samples/javaclient/certs/server/node.key',
    sslCert: '/Users/ranbijaykumar/Alexa-AVS-Sample/alexa-avs-sample-app/samples/javaclient/certs/server/node.crt',
    sslCaCert: '/Users/ranbijaykumar/Alexa-AVS-Sample/alexa-avs-sample-app/samples/javaclient/certs/ca/ca.crt',
    products: {
        "my_device": ["123456"], // Fill in with valid device values, eg: "testdevice1": ["DSN1234", "DSN5678"]
    },
};

module.exports = config;
```

IMPORTANT: **Do not** use ~ to denote the home directory. Use the absolute path instead. So in this case, use "/Users/ranbijaykumar/Alexa-AVS-Sample/alexa-avs-sample-app/…"Save the file.

Step 6: Edit the configuration file for the Java client The configuration file is located at: <REFERENCE_IMPLEMENTATION>/samples/javaclient/config.json.

$ edit config.json

Make the following changes:

- Set **productId**: Enter my_device as a string.
- Set **dsn**: Enter the alphanumeric string that you used for the unique product identifier in the products object in the server's config.js. For example: 123456.
- Set **provisioningMethod**: If you want to use either the Android or iOS sample app, enter **companionApp**. Otherwise, enter **companionService**
- Set **companionApp.sslKeyStore** to <REFERENCE_IMPLEMENTATION>/samples/javaclient/certs/server/jetty.pkcs12
- Set **companionApp.sslKeyStorePassphrase** to the passphrase entered in the certificate generation script in step 4 above. For this prototype password field is blank. So leave as it is.

- Set **companionService.sslClientKeyStore** to <REFERENCE_IMPLEMENTATION>/samples/javaclient/certs/client/client.pkcs12
- Set **companionService.sslClientKeyStorePassphrase** to the passphrase entered in the certificate generation script in step 4 above. For this prototype password field is blank. So leave as it is.
- Set **companionService.sslCaCert** to <REFERENCE_IMPLEMENTATION>/samples/javaclient/certs/ca/ca.crt

Here's what **config.json** file look like (this file created for iOS Companion i.e., provisioningMethod: "**companionApp**")


```
{
    "productId":"my_device",
    "dsn":"123456",
    "provisioningMethod":"companionApp",
    "wakeWordAgentEnabled":false,
    "locale":"en-US",
    "avsHost":"https://avs-alexa-na.amazon.com",
    "companionApp":{
        "localPort":8443,
        "lwaUrl":"https://api.amazon.com",
        "sslKeyStore":"/Users/ranbijaykumar/Alexa-AVS-Sample/alexa-avs-sample-app/samples/javaclient/certs/server/jetty.pkcs12",
        "sslKeyStorePassphrase":""
    },
    "companionService":{
        "serviceUrl":"https://localhost:3000",
        "sslClientKeyStore":"/Users/ranbijaykumar/Alexa-AVS-Sample/alexa-avs-sample-app/samples/javaclient/certs/client/client.pkcs12",
        "sslClientKeyStorePassphrase":"",
        "sslCaCert":"/Users/ranbijaykumar/Alexa-AVS-Sample/alexa-avs-sample-app/samples/javaclient/certs/ca/ca.crt",
        "sessionId":"eafacd34-c431-43ef-89b4-a7c14d61bf25"
    }
}
```

Choose your authentication method

You should now decide if you want your users to authenticate using a website or mobile app. You can only authenticate using one of the following methods at a given time:

- Build and Run a Node.js Server Authentication, **or**
- Build and Run an iOS Companion App for Authentication
- **IMPORTANT:** Only one method can be used at a time. If you want to test authentication using a web site **and** a mobile **companion** app (iOS or Android), you can do so by testing the first provisioning method, then clearing your access token and switching to the other provisioning method. For instructions see Clearing Your Access Token to Change Provisioning Methods.

1) Build and Run a Node.js Server Authentication

Install the Companion Service dependencies

Change directories to <REFERENCE_IMPLEMENTATION>/samples/companionService

- $ cd <REFERENCE_IMPLEMENTATION>/samples/companionService

Install the dependencies by typing:

- $ npm install

Run the following:

- $ npm start

The server is running. You are now ready to run the sample app. The server is now running on **port 3000** and you are ready to start the client.

2) Build and Run an iOS Companion App for Authentication:

To see an example of how to obtain a Login with Amazon authorization code from an iOS app, build and run the iOS sample project.

To build and run the iOS mobile app, follow these instructions:

1. Open the iOS sample app in Xcode. To do this, double click `<REFERENCE_IMPLEMENTATION>/samples/iOSCompanionApp/AlexaCcompanionAppSample.xcodeproj`
2. Log in to the Amazon developer portal.
3. In the top navigation, click **Alexa** then select **Alexa Voice Service**. Find your project, and click **Edit**, then navigate to the **Security Profile** tab.
4. From the drop down, select the security profile you created in Getting Started.
5. Click the **iOS Settings** tab.
6. Enter the following information:
    - **API Key Name:** "Alexa Voice Service test iOS app"
    - **Bundle ID:** com.amazon.AlexaCompanionAppSample
7. Click **Add an API Key**
8. Copy the key to your computer's clipboard.
9. Add the API key to the Xcode project.
    - In the Xcode project, make sure the left navigation pane is set to the project navigator by clicking the left-most icon that looks like a file folder at the top of the navigation pane.
    - Expand *AlexaCompanionAppSample* → *Application* → *Supporting Files*, then select `AlexaCompanionAppSample-Info.plist`.
    - In the middle editing pane, make sure none of the entries are selected.
    - In the menu bar, select *Editor* → *Add Item*.
    - In the combo drop down that appears, type `APIKey` and press return.
    - Double click within the **Value** column and paste the API key that you copied in point 8. Make sure there is no extra whitespace before and after the value, then press **return**.
10. Build and run the app in the simulator by clicking the black right-pointing **play** button at the top left of the Xcode window.

You are now ready to run the sample app.

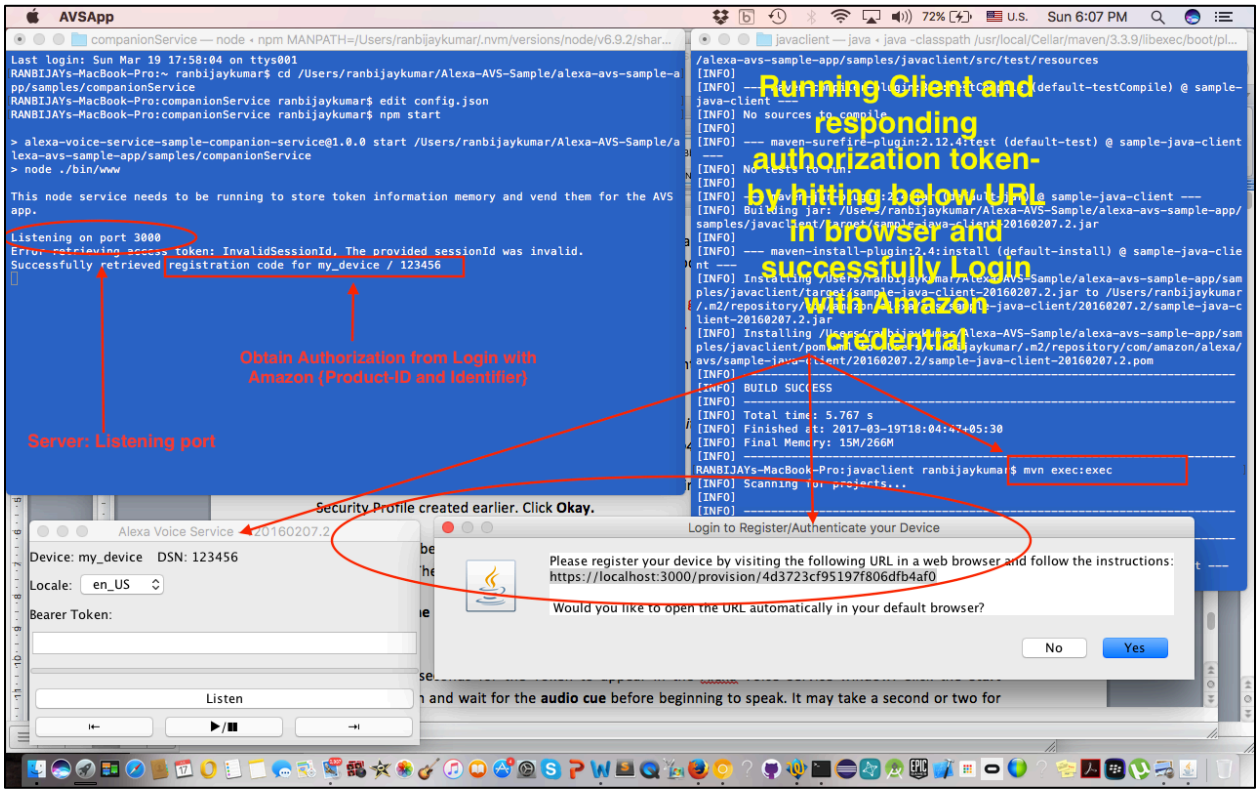
Run the Java Client Sample App:

Depending on whether you specified **`companionApp`** or **`companionService`** as the **`provisioningMethod`** in the `config.json`, follow the instructions below for -

- Using an Android or iOS App to Obtain Authorization from Login with Amazon, or
- Using the Java Client to Obtain Authorization from Login with Amazon.

In text editor, open `<REFERENCE_IMPLEMENTATION>/samples/javaclient/`**`pom.xml`** and locate `{alpn-boot.version}xxx{/alpn-boot.version}`. Confirm that the ALPN version matches your JDK version using the table located here. If the versions match no further action is necessary. If the versions do not match, update the `pom.xml` file with the correct ALPN version.
IMPORTANT: If your JDK version is newer than what's listed in the table, please use the latest ALPN version listed.

1. Open a terminal window.
2. Change directories to `<REFERENCE_IMPLEMENTATION>/samples/javaclient`
3. Before you build the app, let's validate to make sure the project is correct and that all necessary information is available. You do that by running: Run `mvn validate` to ensure the project is correct and that all necessary information is available.
    - $ mvn validate
4. Download dependencies and build the app by typing: (you might want to close any unnecessary apps or you might run into an error due to insufficient memory) .Run `mvn install` to download dependencies and build the sample app.
    - $ mvn install
5. When the installation is completed, we will see a "Build Success" message in the terminal. Run `mvn exec:exec` to run the client sample app.
    - $ mvn exec:exec



Obtain Authorization from Login with Amazon

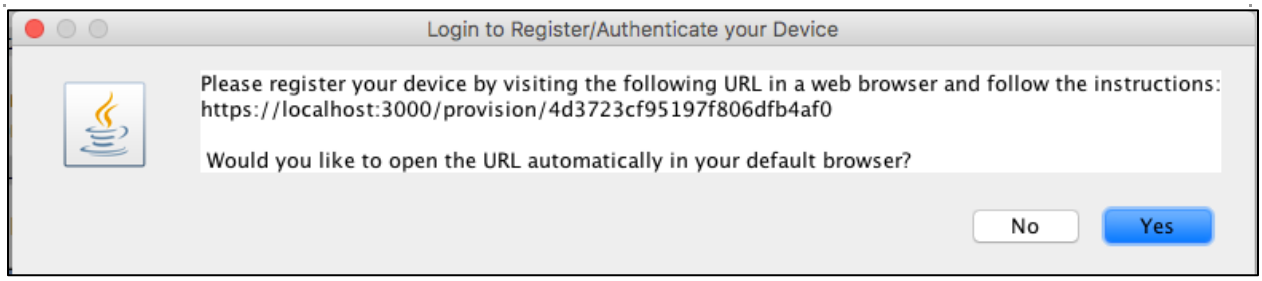
1. When you run the client, a window should pop up with a message that says something similar to:

Copy the URL from the popup window and **paste** it into a **web browser**. In this example, the URL

to copy and paste is https://localhost:3000/provision/*4d3723cf95197f806dfb4af0*.

NOTE: Due to the use of a self-signed certificate, we will get a warning about an insecure website. This is expected. It is safe to ignore the warnings during testing.

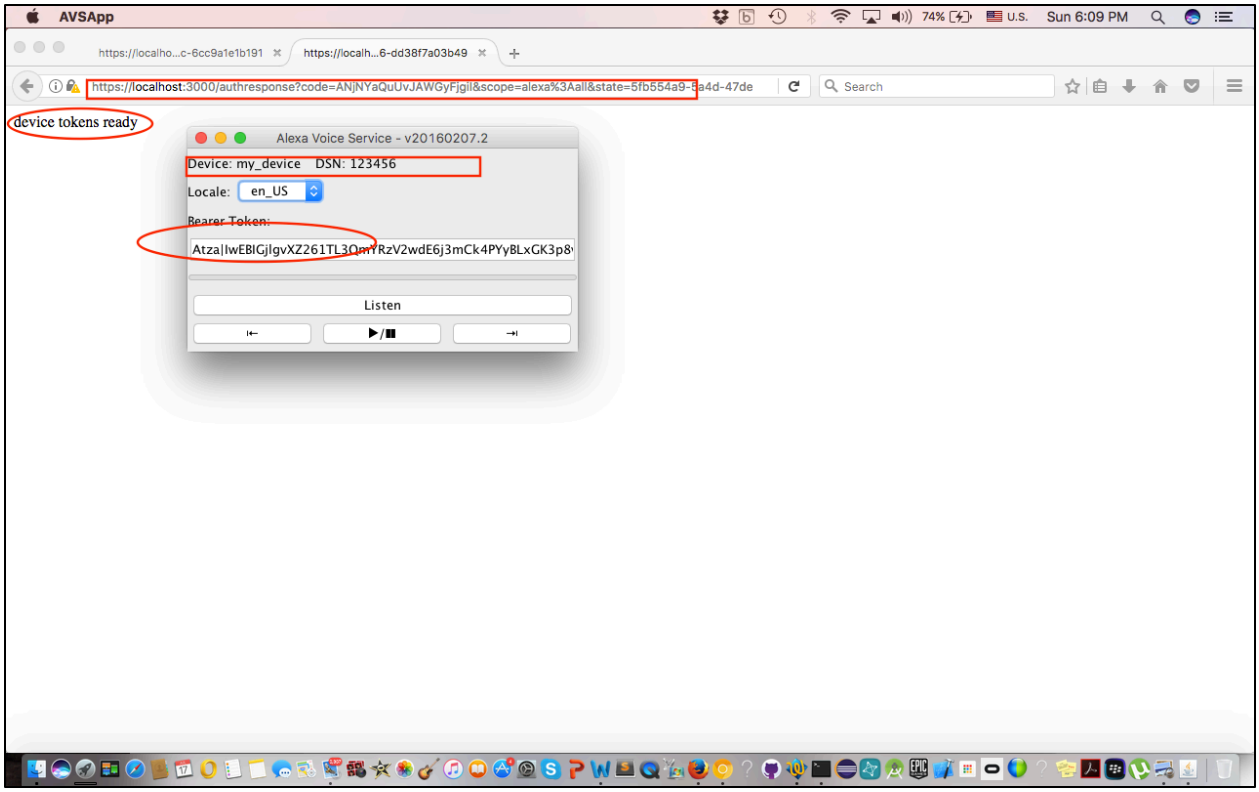
2. We will be taken to a Login with Amazon web page. Enter Amazon credentials.



Please register your device by visiting the following website on any system and following the instructions: https://localhost:3000/provision/4d3723cf95197f806dfb4af0 Hit OK once completed.

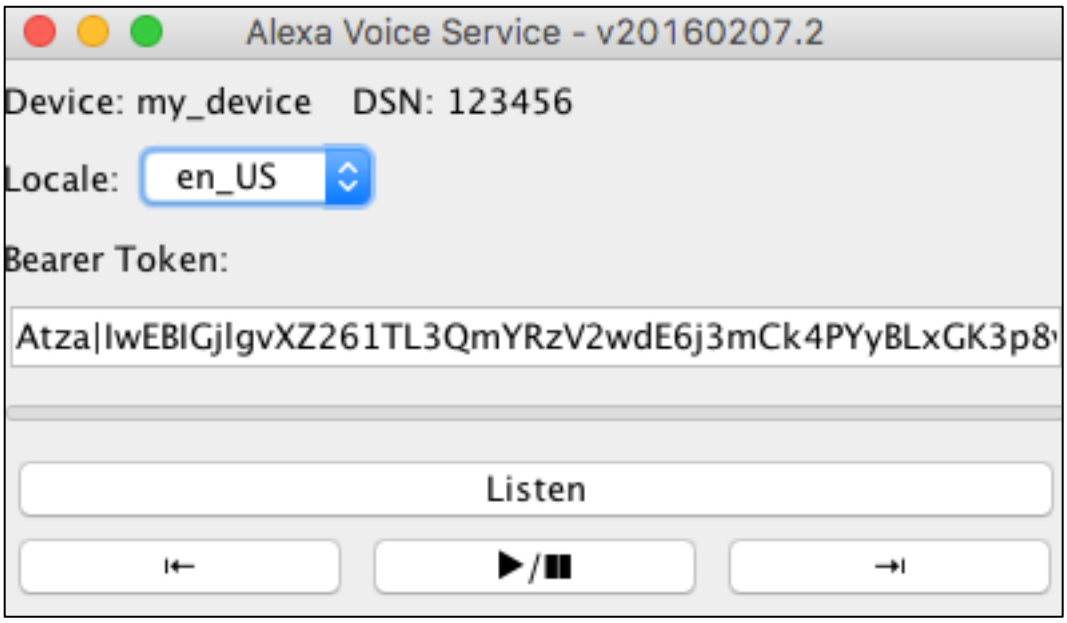
3. We will be taken to a Dev Authorization page, confirming that we'd like your device to access the Security Profile created earlier. Click **Okay.**

4. We will now be redirected to a URL beginning with https://localhost:3000/authresponse followed by a query string. The body of the web page will say **device tokens ready**.



5. **Return to the Java application** and click the OK button. The client is now ready to accept Alexa requests.

6. Wait a few seconds for the Token to appear in the Alexa Voice Service window. Click the **Start Listening** button and wait for the **audio cue** before beginning to speak. It may take a second or two for the connection to be made before you hear the audio cue.



7. Press the **Stop Listening** button when we are done speaking.

Let's talk to Alexa

Ask for Weather: *Click the Start Listening button*. **You**: What's the weather in Seattle? *Click the Stop Listening button*.**Alexa**: Current weather report for Seattle

Some other fun questions we can ask to Alexa

Once we hear the audio cue after clicking "Start Listening" button, here are a few things available we can try saying -

Request Music Playback: Play Bruce Springsteen

General Knowledge: What's the mass of the sun in grams?

Geek: What are the three laws of robotics?

Fun: Can you rap?

Set a Timer: Set the timer for 2 minutes.

Set Alarm: Set the alarm for 7:30 a.m. **More on Music Playback** The "previous", "play/pause", and "next" buttons at the bottom of the Java client UI are to demonstrate the music button events. Music button events allow you to initiate changes in the playback stream without having to speak to Alexa. For example, you can press the "play/pause" button to pause and restart a track of music. To demonstrate the "play/pause" button, you can speak the following command: Play DC101 on iHeartRadio, then press the "play/pause" button. The music will pause in response to the button click. Press the "play/pause" button again to restart the music.

Run the iOS Companion App:

If our provisioning method is **companionApp** and you followed the instructions above to build and run the iOS app, follow these instructions:

1. The iOS app should be running in the iOS simulator on the same computer where we will run the sample app.
2. In the simulator, enter the following address for the device address, then click **Connect**:
    - **iOS:** https://127.0.0.1:8443
3. A gold Login with Amazon button should appear on the screen. Click the button.
4. Enter Amazon login information and click **Sign in**.
5. In the mobile app, we will see a message that says *Your device is now successfully provisioned and ready to use.* In the sample app, the `Bearer Token` text field should now be filled-in with access token. The sample app is now ready to accept requests.

Note: we should only attempt to log in to Login with Amazon from either the Android app or the iOS app, not both at the same time. If we want to test out each app separately, once we have successfully tested one of the apps, follow the instructions below for clearing your access token and then test the other app.

We're now ready to talk to Alexa via iOS Simulator.

Thanks for holding patience to learn a big or elaborated most blog content, hope you find it useful for your development usage.