



---

# Getting to Know the AT&T Speech API

---





# Table of Contents

---

|  |    |
|--|----|
| Introduction .....   | 2  |
| Prerequisites .....  | 3  |
| Conventions Used .....                                       | 4  |
| Setting Up Your Computer for the Speech Lab Exercises .....  | 5  |
| Join the Developer Program .....                             | 5  |
| Register Your First App .....                                | 5  |
| Lab #1: Obtaining an Access Token without User Consent ..... | 8  |
| API Details .....  | 8  |
| Exercise Variant: REST Client (hurl.it) .....                | 10 |
| Exercise Variant: Command Line Tool (curl) .....             | 12 |
| Summary .....  | 13 |
| Lab #2: Basic Speech-to-Text API .....                       | 14 |
| API Details .....  | 14 |
| Exercise: Command Line Tool (curl) .....                     | 15 |
| Summary .....  | 17 |
| Lab #3: Speech-to-Text with Custom Grammar API .....         | 18 |
| API Details .....  | 18 |
| Exercise: Command Line Tool (curl) .....                     | 20 |
| Summary .....  | 22 |
| Lab #4: Text-to-Speech API .....                             | 23 |
| API Details .....  | 23 |
| Exercise: Command Line Tool (curl) .....                     | 25 |
| Summary .....  | 26 |
| Lab #5: Speech-to-Text with Prefix Grammar API .....         | 27 |
| API Details .....  | 27 |
| Exercise: Command Line Tool (curl) .....                     | 29 |
| Summary .....  | 30 |
| Lab #6: Speech-to-Text with Alternate Grammar API .....      | 31 |
| API Details .....  | 31 |
| Exercise: Command Line Tool (curl) .....                     | 32 |
| Summary .....  | 35 |



|  |    |
|--|----|
| Appendix A: Installing curl for Windows .....            | 36 |
| Appendix B: SDKs and Tools .....                         | 38 |
| Appendix C: Running the iOS Speech Sample Apps .....     | 39 |
| Download the Sample Apps .....                           | 39 |
| SimpleSpeech App.....                                    | 40 |
| SimpleGrammar App .....                                  | 42 |
| Simple TTS App .....                                     | 43 |
| Appendix D: Running the Android Speech Sample Apps ..... | 45 |
| Download the Sample Apps .....                           | 45 |
| Opening the Sample Apps .....                            | 46 |
| SimpleSpeech App.....                                    | 47 |
| SimpleSpeechUI.....                                      | 50 |
| SimpleTTS App .....                                      | 52 |



© 2012 AT&T Intellectual Property. All rights reserved. AT&T, AT&T logo and all other AT&T marks contained herein are trademarks of AT&T Intellectual Property and/or AT&T affiliated companies. All other trademarks are the property of their owners. Actual results and your experience may vary from those described in this case study. Information and offers subject to change.



# Introduction

---



**AT&T's Speech API**, powered by the Watson<sup>SM</sup> speech engine, enables you to provide speech to text and text to speech capabilities to your app. You can even customize our speech recognizer for your app, providing information specific to your app that will improve recognition. The following features are available in the Speech API:

## Speech to Text

The speech to text functionality supports speech-enabled apps that run on virtually any cellular or non-mobile network in the United States—it just needs an Internet connection. Provide audio via the API, and it will return text of what the user said.

## Custom Speech to Text

Through custom grammar lists or hints you can improve the results when the Speech API converts speech to text. This is particularly useful if your app uses uncommon or industry-specific terminology.

## Text to Speech

If you need your app to “talk”, then just use the API to send the text and one of the English or Spanish speaking characters – male or female – will “say” what you sent. You can even control the speed and volume that the text is read.

The intent of this booklet is to guide you through a series of simple lab exercises to familiarize you with using the AT&T APIs and to greatly simplify the integration of one or more of them into your own applications.

These lab exercises are targeted toward developers and will be shown in curl and, where possible, a web-based REST client ([www.hurl.it](http://www.hurl.it)). You can use any programming language or REST client that you prefer, however.

Additional information can be found online at:

- [AT&T Developer Program](#)
- [APIs Overview](#)
- [Speech Overview](#)
- [Speech API documentation](#)

If you have any questions, please contact us at [developer.program@att.com](mailto:developer.program@att.com).



## Prerequisites

---

In order to complete the exercises in this workbook, you will need:

1. A command line tool called curl (officially known as cURL). See [Appendix A](#) on how to download and install curl.
2. Optionally: For Lab Exercise #1, with a browser and internet connection you can make the API calls by using the website <http://hurl.it>.



## Conventions Used

---

The following formatting conventions are used in this workbook:

| Type                    | Style         | Example   |
|-------------------------|---------------|---|
| URL                     | Monospace     | <code>https://api.att.com/speech/v3/speechToText</code>         |
| Parameter               | <i>Italic</i> | <i>client_id</i>  |
| Header                  | <b>Bold</b>   | <b>application/json</b>   |
| Command line            | Monospace     | <code>curl -X POST --data-binary...</code>                      |
| JSON or XML             | Monospace     | <pre>{<br/>  "Recognition": {<br/>    "Status": "OK", ...</pre> |
| User interface elements | <b>Bold</b>   | <b>Join Now</b>   |
| File names              | <b>Bold</b>   | <b>homeBy6.wav</b>  |





# Setting Up Your Computer for the Speech Lab Exercises

---

Pre-requisites: browser

Length: 10 minutes

**In this exercise:**

## Join the Developer Program Register Your First App

In order to call the AT&T APIs, you will first need to join the AT&T Developer Program. Once you have done so, you can register applications, which will each have their own app key, secret, and sometimes a unique shortcode. The app key acts as the OAuth client\_id, the secret acts as the OAuth client\_secret, and the shortcode is the number used as an address for SMS/MMS messages. (Shortcodes are only assigned if the app is going to use SMS/MMS, which is not the case for these exercises.)

For these exercises, you can register for a trial account, which will let you register up to three applications.

See the [Prerequisites](#) section on what you need before starting the exercises.

You will need to register an app in the AT&T Developer Program website to use in the labs.

## Join the Developer Program

Follow these steps to join the developer program.

**Note:** If you already have either a full or trial developer account, simply log in and skip to **Register Your First App**.

1. Launch your browser and go to <http://developer.att.com>.
2. Click on the **Join Now** link in the upper right hand corner.
3. Close the pop up.
4. Enter your name, username, email, and password.

Congratulations! You're part of the AT&T developer program.

## Register Your First App

When you register an app, you will have the app key and secret code generated for you automatically. These codes can be viewed any time by logging into your Developer Program account.

**Notes:**

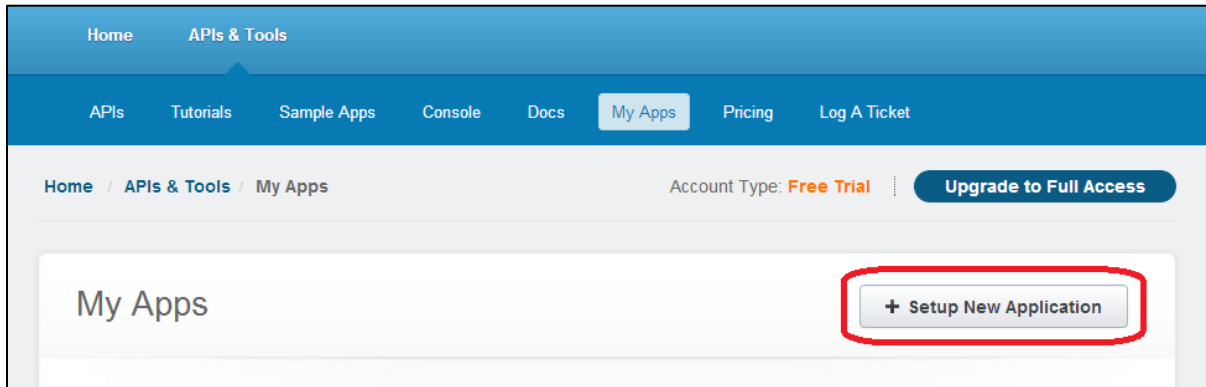
- If you already have a trial or full developer account, then sign in and click on the **My Apps** button.



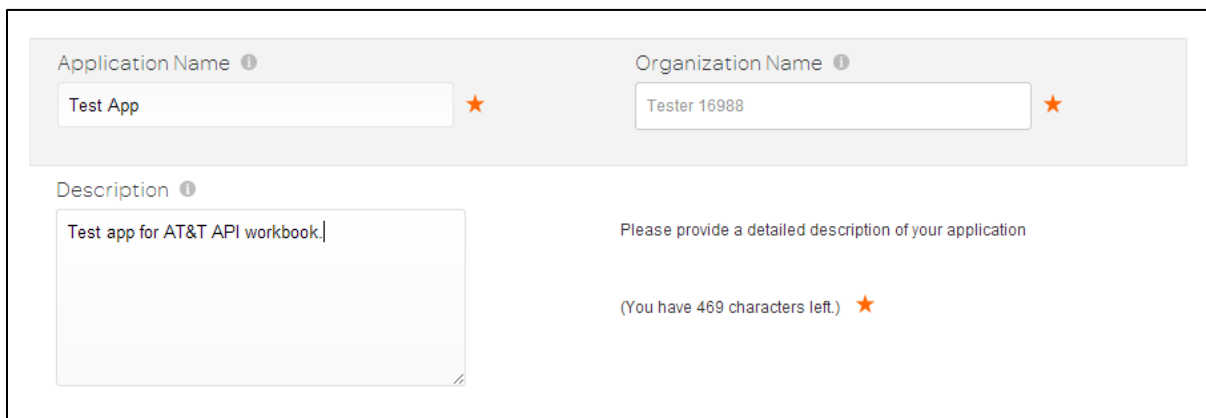
- If you already have an app that you want to use for this exercise, make note of its client ID and secret, and edit the settings so that the Speech checkboxes are checked, as shown in Step 3.

Follow these steps:

1. Click **Setup a New Application**.



2. Type in an Application name and short description.



3. Check the following boxes: **Speech To Text (SPEECH)**, **Speech to Text Custom (STTC)**, and **Text to Speech (TTS)**. (See screenshot that follows.) This will allow you to use these codes for creating apps that convert speech to text and text to speech.



API Services

- ☐ MIM ⓘ
- ☒ Speech To Text (SPEECH) ⓘ
- ☒ Speech To Text Custom (STTC) ⓘ
- ☒ Text to Speech (TTS) ⓘ
- ☐ Terminal Location ⓘ
- ☐ Short Messaging Service (SMS) ⓘ

4. Click **Setup Application**.
5. Now you should see information about your application.

DESCRIPTION: Test app for AT&T API workbook.

ORGANIZATION: Tester 16988

CONNECTED APIs: Speech To Text Custom (STTC) Speech To Text (SPEECH) Text to Speech (TTS)

ACTIVE ⓘ

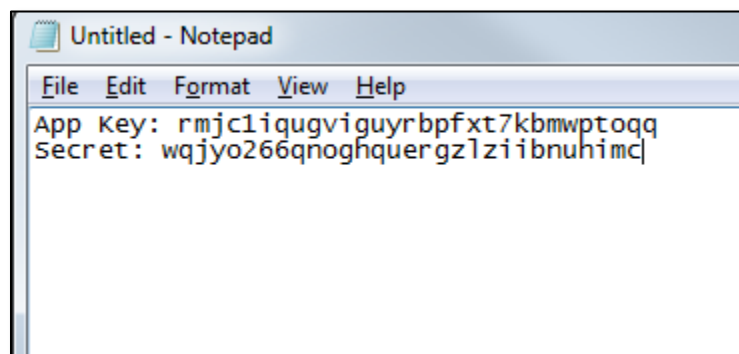
Sandbox ⓘ Upgrade to Full Access

Details

App Key: rmjc1iqugviguyrbpft7kbnwptqq ⓘ

Secret: wqjyo266qnoghquergzlziibnuhimc ⓘ

6. Open up a text editor and copy and paste the App Key and Secret codes into it so you can easily access them later.



# Lab #1: Obtaining an Access Token without User Consent

Pre-requisites: app key, secret, curl or browser

Length: 5 minutes

## In this exercise:

### API Details

**Exercise Variant: REST Client (hurl.it)**

**Exercise Variant: Command Line Tool (curl)**

### Summary

There are two ways to use OAuth to obtain an access token: with or without user consent. API requests only require user consent if they involve user-specific data. For example, sending a message from the user's phone number requires user consent. However, sending messages to or from a shortcode does not require user-specific data, and therefore does not require user consent. For this exercise, the Speech API does not require consent.

In this exercise, you will learn how to use OAuth to get an access token without consent. Once you have the access token, you can use it to make other API requests. The API requests that you can use are determined by the *scope* parameter, as explained below.

To make HTTP requests, we'll be using the curl command line tool (see [Appendix A: Installing curl for Windows](#)).

## API Details

To obtain a token without consent, make an HTTP POST request to this URL:

```
https://api.att.com/oauth/token
```

The POST body contains the following parameters in the x-www-form-urlencoded format:

| Parameter Name | Description   |
|----------------|---|
| client_id      | Your app's app key  |
| client_secret  | Your app's secret   |
| grant_type     | Set to <b>client_credentials</b> for OAuth requests without consent   |
| scope          | Which API requests will be used with this token. We want to use this with speechToText, speechToTextCustom, and textToSpeech, so we'll set it to <b>SPEECH,STTC,TTS</b> . |

**Note:** The scope indicates which groups of API requests your app has permission to use. You can specify more than one scope by using a comma-separated list. The scope values must be a subset of the scope values that you selected using check boxes when editing the app settings in the Developer Program website. If you want to choose a scope that was not checked in the app settings, then you can edit the app settings and add it, but only if the app is in the Sandbox realm. If the app has been promoted to production, these settings cannot be changed.



The return value can be in either JSON or XML format. To set it, add a header called **Accept** with a value of **application/json** or **application/xml**.

More information can be found in the documentation at: [OAuth 2.0 Authentication Management](#).



## Exercise Variant: REST Client (hurl.it)

Follow these steps to obtain an access token without consent using the website **hurl.it**:

1. Open <http://www.hurl.it> in a browser.
2. In the URL box, type `https://api.att.com/oauth/token`
3. Change the dropdown to **POST**.
4. Click on **set post body**.
5. In the large text box for the POST body, copy and paste the following:

```
client_id={appkey}&client_secret={secret}&grant_type=client_credentials&scope=SPEECH,STTC,TTS
```

6. Replace {appkey} with your app key and {secret} with your secret.
7. Click the **+ add header** link to add new headers, if necessary. Add a header with name of **Accept** and value of **application/json**.
8. Add a header with name of **Content-Type** and value of **application/x-www-form-urlencoded**.
9. Click the **Send** button.

The screenshot below shows how to add the information into the hurl.it website.

The screenshot shows the hurl.it website interface for configuring a REST client request. The URL box contains `https://api.att.com/oauth/token`. The method dropdown is set to **POST**, and the **follow redirects** checkbox is checked. Below the URL box, there are links for **+ add param** and **- set post body**. The **set post body** link is active, and the large text area contains the following JSON payload: `client_id=f3a6c8c585e3606a5a2f750e99d78adb&client_secret=6f1d7ee6d357a2122&grant_type=client_credentials&scope=SPEECH,STTC,TTS`. At the bottom, there are radio buttons for **no auth** (selected) and **HTTP basic**. Below these, there is a link for **+ add header**. Two headers are listed: **Accept** with value `application/xml` and **Content-Type** with value `application/x-www-form-urlencoded`. Each header has a small 'x' icon to its right.



You should see something like the following returned:

```
{
  "access_token": "4a0e9c2b14a0548a5ddb29eb77e06a8b",
  "token_type": "bearer",
  "expires_in": 157679999,
  "refresh_token": "f40caf9ef812618275218602111e874a6f60"
}
```

Copy the value of the **access\_token** key and paste it into a text editor for later use. In this case, the token will expire in 157679999 seconds (5 years).

**Note:** If you see the following returned:

```
{
  "error" : "invalid_scope",
  "error_description" : "The requested scope is invalid, unknown, malformed, or exceeds the previously granted scope."
}
```

This means that you set the scope to a value that was not specified in the app settings. Go back to the app settings in the Developer Program (see [Register Your First App](#)) and make sure that the SPEECH, STTC, and TTS checkboxes are checked.



## Exercise Variant: Command Line Tool (curl)

**Note:** The curl exercises are designed for Unix-based terminals, such as Unix, MacOS, and Cygwin. They use the backslash (\) in order to continue onto more than one line. To run them on a Windows command prompt, substitute each backslash for a caret (^), or remove the backslashes and put it all on one line.

**Note:** All curl commands in this workbook have the **--insecure** flag. This allows curl to perform "insecure" SSL connections so that you don't need the correct certification bundles installed. You may choose instead to install the correct certification bundles and to not use that flag.

Follow these steps to obtain an access token without consent using curl:

1. Paste the following into a text editor:

```
curl --request POST --insecure \
--header "Accept: application/json" \
--header "Content-Type: application/x-www-form-urlencoded" \
--data "client_id={appkey}&client_secret={secret}&\
grant_type=client_credentials&scope=SPEECH,STTC,TTS" \
https://api.att.com/oauth/token
```

2. Replace {appkey} with your app key and {secret} with your secret.
3. Open up a terminal (command prompt in Windows) that has curl.
4. Paste the edited curl command into the terminal and hit Enter.

You should see something like the following returned:

```
{
  "access_token": "4a0e9c2b14a0548a5edb29eb77e06a8b",
  "token_type": "bearer",
  "expires_in": 157679999,
  "refresh_token": "f40caf9ef812612275218602111e874a6f60"
}
```

Copy the value of the **access\_token** key and paste it into a text editor for later use. In this case, the token will expire in 157679999 seconds (5 years).

**Note:** If you see the following returned:

```
{
  "error" : "invalid_scope",
  "error_description" : "The requested scope is invalid, unknown,
malformed, or exceeds the previously granted scope."
}
```

This means that you set the scope to a value that was not specified in the app settings. Go back to the app settings in the developer program and make sure that the SPEECH, STTC, and TTS checkboxes are checked.





## Summary

*You've learned how to obtain an access token that can be used for the speech API. In the next exercise, you will use this token to make the actual API request.*



## Lab #2: Basic Speech-to-Text API

Pre-requisites: access token with consent, curl or browser

Length: 10 minutes

### In this exercise:

#### API Details

#### Exercise: Command Line Tool (curl)

#### Summary

In this exercise, you will learn how to use the Speech API that converts speech to text. You'll use the access token you obtained in Lab Exercise #1 to send a sound file and receive text.

Note that there are several ways that you can convert speech to text. This exercise covers the most basic procedure, where a sound file is sent over and speech is returned. Other speech capabilities include:

- Streaming
- Speech to text using a grammar
- Speech to text using hints
- Text to speech

### API Details

To convert speech to text using the generic recognizer, make a call to this URL:

```
POST https://api.att.com/speech/v3/speechToText
```

**Note:** The URL is case sensitive.

The headers will have the follow values:

| Header Name     | Description  |
|-----------------|--|
| Authorization   | Bearer followed by your access token   |
| Accept          | The format for the returned data. Either <b>application/json</b> , <b>application/xml</b> , or <b>application/emma+xml</b> . |
| Content-Type    | The type of audio file you are sending. See the <a href="#">Speech API documentation</a> for valid values.                   |
| Content-Length  | The length of the sound file.  |
| X-SpeechContext | The context for speech recognition, such as gaming, business, etc. For this example, we will use <b>SMS</b> .                |

The POST body will contain the sound file.

**Note:** There are restrictions on what sound files are accepted. For example, they must have only a single channel. See the Speech API documentation for more information.

More information can be found in the documentation at: [Speech API](#).



## Exercise: Command Line Tool (curl)

**Note:** The curl exercises are designed for Unix-based terminals, such as Unix, MacOS, and Cygwin. They use the backslash (\) in order to continue onto more than one line. To run them on a Windows command prompt, substitute each backslash for a caret (^), or remove the backslashes and put it all on one line.

For this exercise, you will send a file in WAV format. Imagine you are creating some kind of app that converts speech into an SMS message. This sound file says, "I will be home by six."

Follow these steps to convert the sound file into text:

1. Download the sound file **homeBy6.wav**, and change directory in your terminal to be in the same directory as the file. Try playing the file if you have a media player that can play WAV files.
2. Paste the following into a text editor. Note that by using the **data-binary** flag, you are adding the file into the POST body as well as setting the **Content-Length** header to the length of the file.

```
curl --request POST --insecure \
--header "Authorization: Bearer {token}" \
--header "Accept: application/json" \
--header "Content-Type: audio/wav" \
--header "X-SpeechContext: SMS" \
--data-binary @homeBy6.wav \
https://api.att.com/speech/v3/speechToText
```

3. Replace {token} with the access\_token you obtained in Lab Exercise #1.
4. Open up a terminal that has curl.
5. Paste the edited curl command into the terminal and hit Enter. You should see a return value in JSON. Once it is formatted, it will look like this:

```
{
  "Recognition": {
    "Status": "OK",
    "ResponseId": "f723e1dddcc5f14d52e952292fdfe499",
    "Info": {
      "metrics": {
        "audioTime": 5.3,
        "audioBytes": 467968
      }
    }
  },
  "NBest": [
    {
      "wordScores": [
        0.61,
        1,
        1,
        1,
        1,
        0.93
      ]
    }
  ]
}
```



```

    "Confidence": 0.92333335,
    "Grade": "accept",
    "ResultText": "I will be home by six",
    "Words": [
        "I",
        "will",
        "be",
        "home",
        "by",
        "six"
    ],
    "LanguageId": "en-US",
    "Hypothesis": "I will be home by six"
  }
}
}

```

Here's how to interpret the various pieces of the response:

- **ResultText.** The result of the recognition process. In other words, it's the best guess of what the sound file is saying.
- **Confidence.** If you need more granularity than **Grade**, then use **Confidence**, which provides a value between 0 and 1, where 1 is the highest confidence.
- **Words.** An array of the individual words that make up the **ResultText**.
- **WordScores.** An array of confidence values for each individual word. This array matches up with the **Words** array so that you know which confidence value applies to which word.
- **Status.** The status for the request. "OK" means a successful request. "Speech Not Recognized" means that recognition was not successful.
- **LanguageId.** The language that was used to recognize the sound file.
- **Hypothesis.** The result from the recognizer before formatting to make more readable. It is generally the same as **ResultText**, but **ResultText** may have some words altered.
- **Grade.** An indication that is supposed to assist in deciding whether you should consider the **ResultText** as accurate. Values are "accept", "confirm", and "reject". However, for many contexts, the grade will be always be "accept". Use **Confidence** instead to determine whether the **ResultText** is accurate.

Note that in our example, the API has correctly recognized the sound file as "I will be home by six".



## Summary

*You've learned how to use curl to convert speech to text, providing a sound file and a context. The interpreted text and confidence values are returned. Read the [Speech API documentation](#) to learn more about how to fine-tune the parameters to get the best results, and how to use streaming instead of sending a single sound file.*



## Lab #3: Speech-to-Text with Custom Grammar API

Pre-requisites: access token, curl, media files (medications.grxml, loratadine.wav)

Length: 15 minutes

### In this exercise:

#### API Details

#### Exercise: Command Line Tool (curl)

#### Summary

In this exercise, you will learn how to use grammars to improve the results of the Speech API that converts speech to text. You'll use the access token that you acquired in the previous exercise.

Note that you can provide either grammar ("Grammar List" context) or hints ("Generic with Hints" context) to the speech recognizer. Grammar List limits the responses to those specified in the grammar file. Generic with Hints uses a grammar file just like the Grammar List context; however, in addition, if end users say a word that is not in the grammar file, the API tries to recognize the word by using the Generic speech-to-text recognizer. As with basic speech, you can either use streaming or non-streaming mode. This exercise focuses on use of the Grammar List context.

### API Details

To convert speech to text using a grammar list, make a call to this URL:

```
POST https://api.att.com/speech/v3/speechToTextCustom
```

**Note:** The URL is case sensitive.

The headers will have the follow values:

| Header Name     | Description  |
|-----------------|--|
| Authorization   | Bearer followed by your access token   |
| Accept          | The format for the returned data. Either <b>application/json</b> , <b>application/xml</b> , or <b>application/emma+xml</b> .   |
| Content-Type    | Set to <b>multipart/x-srgs-audio</b> to indicate that we are sending both the audio file and the grammar (srgs).   |
| X-SpeechContext | The context for speech recognition, which can be either <b>GrammarList</b> for grammar or <b>GenericHints</b> for hints. (Optional, with a default of <b>GenericHints</b> .) |

**Note.** Each MIME part has a content disposition, where the type of file to be uploaded is specified. These file types include pronunciation, grammar, grammar prefix hints, grammar altgram hints, and audio. In this case, the content-disposition will be **x-grammar** for using grammars and **x-voice** for the audio file. For more information, see the custom speech documentation and the WC3 specification.



**Note.** There are restrictions on what sound files are accepted. For example, they must have a single channel. See the speech documentation for more information.

More information can be found in the documentation at: [Speech API](#).



## Exercise: Command Line Tool (curl)

### Step 1. Download the Sound File and Grammar

Imagine we are developing a speech application for doctors, and we expect them to verbally choose from a list of medications. We can use a grammar to narrow the list down to set number of choices. Download the following files and place them in your terminal's current directory.

- medications.grxml
- loratadine.wav

medications.grxml is shown below. It follows the W3C Speech Recognition Grammar Specification Version 1.0, which can be found at <http://www.w3.org/TR/speech-grammar/>. As you can see, it is an XML file with just one rule, which is to choose one of a list of medications.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE grammar PUBLIC "-//W3C//DTD GRAMMAR 1.0//EN"
    "http://www.w3.org/TR/speech-grammar/grammar.dtd">
<grammar version="1.0" mode="voice" root="OTC" tag-
format="semantics/1.0" xmlns="http://www.w3.org/2001/06/grammar">
  <rule id="OTC" scope="public">
    <one-of>
      <item>fexofenadine</item>
      <item>diphenhydramine</item>
      <item>brompheniramine</item>
      <item>loratadine</item>
      <item>clemastine</item>
      <item>chlorpheniramine</item>
      <item>certirizine</item>
    </one-of>
  </rule>
</grammar>
```

**Note.** This is a very simple grammar file. Grammar files can be quite complex and provide many options for how speech can be recognized.

loratadine.wav is a sound file that says the word "loratadine".

### Step 2. Call the Speech Recognizer with the Grammar

**Note:** The curl exercises are designed for Unix-based terminals, such as Unix, MacOS, and Cygwin. They use the backslash (\) in order to continue onto more than one line. To run them on a Windows command prompt, substitute each backslash for a caret (^), or remove the backslashes and put it all on one line.

Now you will make a POST request to the URL: /speech/v3/speechToTextCustom. Note that it is case sensitive.

Follow these steps to convert the sound file into text:

1. Paste the following into a text editor. Note that by using the **form** flag, you are adding both the sound file and the grammar file into the POST body.





```
curl --request POST --insecure \
--header "Authorization: Bearer {token}" \
--header "Accept: application/json" \
--header "Content-type: multipart/x-srgs-audio" \
--form "x-grammar=@medications.grxml;type=application/srgs+xml" \
--form "x-voice=@loratadine.wav;type=audio/wav" \
https://api.att.com/speech/v3/speechToTextCustom
```

2. Replace {token} with your access\_token you got in Lab Exercise #1.
3. Open up a console that has curl.
4. Paste the edited curl command into the console and hit Enter. You should see a return value in JSON. Once it is formatted, it will look like this:

```
{
  "Recognition": {
    "Status": "OK",
    "ResponseId": "4cf4b40b9aa4cab0911269b6f7387272",
    "Info": {
      "metrics": {
        "audioTime": 1.17,
        "audioBytes": 103424
      }
    },
    "NBest": [
      {
        "wordScores": [
          0.639
        ],
        "Confidence": 0.53,
        "Grade": "accept",
        "ResultText": "loratadine",
        "words": [
          "loratadine"
        ],
        "LanguageId": "en-US",
        "Hypothesis": "loratadine"
      }
    ]
  }
}
```

The API has correctly recognized the sound file as "loratadine", and it provides some confidence information.



## Summary

*You've used the custom version of the speech-to-text API and shown how you can use a grammar to more accurately convert speech to text. Read the [Speech API documentation](#) to learn how to use hints instead of a grammar, how to use streaming, and how to use the 19 languages that the API supports.*



## Lab #4: Text-to-Speech API

---

Pre-requisites: access token, curl, sound player

Length: 10 minutes

**In this exercise:**

### API Details

#### Exercise: Command Line Tool (curl)

#### Summary

In this exercise, you will learn how to use the Speech API to convert text to speech. You'll use the access token you obtained in Lab Exercise #1 to send text and receive a sound file of the text being spoken.

Converting text to speech is useful for any kind of application where it is useful to have information in an audio form. Some use cases include:

- Navigation apps where the user is expected to be driving
- Apps for users with visual impairment
- Verbally confirm what the speech-to-text API recognized

The text-to-speech API has several parameters that can be used to alter how the words are spoken. They include:

- Volume
- How fast the text is read
- Male or female voices
- English or Spanish voices

See the [Speech API documentation](#) for more details.

### API Details

To convert text to speech, make a call to this URL:

```
POST https://api.att.com/speech/v3/textToSpeech
```

Note that it is case sensitive.

The text to convert is placed in the body of the POST. The headers will have the follow values:



| Header Name    | Description  |
|----------------|--|
| Authorization  | Bearer followed by your access token   |
| Accept         | The format for the returned sound file. Either <b>audio/amr</b> (Adaptive Multi-Rate codec), <b>audio/amr-wb</b> (Adaptive Multi-Rate Wideband codec), or <b>audio/x-wav</b> (Microsoft Waveform Audio File Format).   |
| Content-Type   | Set to <b>text/plain</b> .   |
| Content-Length | The length of the text.  |
| X-Arg          | Contains multiple parameters in one header. Examples include: <ul style="list-style-type: none"><li>• <b>VoiceName</b>, which can be "crystal" or "mike" for English-language voices or "rosa" or "alberto" for Spanish-language voices. Default is crystal.</li><li>• <b>Tempo</b>, which affects the speed, and has values from -18 (slowest) to 18 (fastest). Default is 0.</li><li>• <b>Volume</b>, which affects how loud the sound is, and has values for 0 (quietest) to 500 (loudest). Default is 100.</li></ul> |

More information can be found in the documentation at: [Speech API](#).



## Exercise: Command Line Tool (curl)

**Note:** The curl exercises are designed for Unix-based terminals, such as Unix, MacOS, and Cygwin. They use the backslash (\) in order to continue onto more than one line. To run them on a Windows command prompt, substitute each backslash for a caret (^), or remove the backslashes and put it all on one line.

Follow these steps to convert the text into a sound file:

1. Paste the following into a text editor. Note that the **Content-Length** header will be set automatically. The output is directed into a file called **speech.wav**.

**Note:** The **X-Arg** header is used to modify the sound file, and contains parameters within it. See [API Details](#) for more information.

```
curl --request POST --insecure \  
--header "Authorization: Bearer {token}" \  
--header "Accept: audio/x-wav" \  
--header "Content-Type: text/plain" \  
--header "X-Arg: VoiceName=crystal, Tempo=-3, Volume=80" \  
--data "Say it loud and proud" \  
--output speech.wav \  
https://api.att.com/speech/v3/textToSpeech
```

2. Replace {token} with the access\_token you obtained in Lab Exercise #1.
3. Open up a terminal that has curl.
4. Paste the edited curl command into the terminal and hit Enter.
5. A file called **speech.wav** should have been created. Open it in a media player, such as Quicktime or Windows Media Player. You should hear a voice say, "Say it loud and proud".

**Note:** if you do not have a media player that can play WAV files, then change the **Content-Type** to **audio/amr** and replace speech.wav with speech.amr. This will create an Adaptive Multi-Rate that hopefully your media player can play.



## Summary

*You've learned how to convert text to speech, which takes text in the POST body and returns a sound file. The interpreted text and confidence values are returned. Read the [Speech API documentation](#) to learn more about how to fine-tune the parameters to get the best results, and how to use streaming instead of sending a single sound file.*



## Lab #5: Speech-to-Text with Prefix Grammar API

Pre-requisites: access token, curl

Length: 15 minutes

**In this exercise:**

**API Details**

**Exercise: Command Line Tool (curl)**

**Summary**

In a previous exercise, we showed you how to constrain speech recognition to a list of words by using a simple grammar file. In this exercise, we'll show you how to use a prefix grammar file.

Often, speech recognizers need to recognize phrases where the beginning of the total phrase is expected to be a common set of phrases. For example, in an app that allows users to update social media sites verbally, you might have phrases that begin with "Update Facebook status with...", "Add tweet that says...", etc. What comes after that phrase can be completely open. So for this scenario, we use a prefix grammar file.

### API Details

To convert speech to text using the custom recognizer, make a call to this URL:

```
POST https://api.att.com/speech/v3/speechToTextCustom
```

**Note:** The URL is case sensitive.

The headers will have the follow values:

| Header Name     | Description  |
|-----------------|--|
| Authorization   | Bearer followed by your access token   |
| Accept          | The format for the returned data. Either <b>application/json</b> , <b>application/xml</b> , or <b>application/emma+xml</b> . |
| Content-Type    | Set to <b>multipart/x-srgs-audio</b> to indicate that we are sending both the audio file and the grammar (srgs).             |
| X-SpeechContext | The context for speech recognition, such as gaming, business, etc. (Optional)  |

**Note.** Each MIME part has a content disposition, where the type of file to be uploaded is specified. These file types include pronunciation, grammar, grammar prefix hints, grammar altgram hints, and audio. In this case, the content-disposition will be **x-grammar-prefix** for using prefixes and **x-voice** for the audio file. For more information, see the custom speech documentation and the WC3 specification.

**Note.** There are restrictions on what sound files are accepted. For example, they must have a single channel. See the speech documentation for more information.



More information can be found in the documentation at: [Speech API](#).





## Exercise: Command Line Tool (curl)

### Step 1. Create the Prefix Grammar File

Open a text editor and paste in the following XML to create a grammar that has social media phrases:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE grammar PUBLIC "-//W3C//DTD GRAMMAR 1.0//EN"
    "http://www.w3.org/TR/speech-grammar/grammar.dtd">
<grammar version="1.0" mode="voice" root="prefixes" tag-
format="semantics/1.0" xmlns="http://www.w3.org/2001/06/grammar">
  <rule id="prefixes" scope="public">
    <one-of>
      <item>Update Facebook status with</item>
      <item>Update Facebook status to</item>
      <item>Send Facebook message to</item>
      <item>Tweet</item>
      <item>Add tweet that says</item>
      <item>Add comment to picture</item>
      <item></item>
    </one-of>
  </rule>
</grammar>
```

This creates several social media app phrases that could be used as prefixes. The empty tag at the end allows the generic recognizer to be used if none of the prefixes match. Save the file as **social.grxml**.

### Step 2. Record or download a sound file

Create a sound file that has you saying a phrase that begins with one of the phrases in the prefix file. Save it as a .wav file with one mono track.

Or, if you don't have sound editing software, download the file **facebook.wav**. It says, "Update Facebook status with waiting in line".

Place the file in the location where you are making the curl command.

### Step 3. Call the Speech Recognizer with the Prefix Grammar

Edit the curl statement you used in the Speech API with Custom Grammar exercise. Make the following changes:

1. Change `x-grammar=@medications.grxml` to

```
x-grammar-prefix=@social.grxml
```

2. Change `x-voice=@loratadine.wav` to `x-voice=@{soundfile.wav}` where `{soundfile.wav}` is the name of your sound file.

Run your curl statement. It should recognize your words. See Lab #2 for a discussion on how to interpret the results.



## Summary

*You've used the custom version of the speech-to-text API and shown how you can use a prefix grammar to more accurately convert speech to text where the beginning of a phrase follows a grammar, but the ending of the phrase does not follow any grammar.*



## Lab #6: Speech-to-Text with Alternate Grammar API

Pre-requisites: access token, curl, media files (nauticalTasks.grxml, berthInspection.wav)

Length: 15 minutes

**In this exercise:**

**API Details**

**Exercise: Command Line Tool (curl)**

**Summary**

### API Details

To convert speech to text using the custom recognizer, make a call to this URL:

```
POST https://api.att.com/speech/v3/speechToTextCustom
```

**Note:** The URL is case sensitive.

The headers will have the follow values:

| Header Name     | Description  |
|-----------------|--|
| Authorization   | Bearer followed by your access token   |
| Accept          | The format for the returned data. Either <b>application/json</b> , <b>application/xml</b> , or <b>application/emma+xml</b> . |
| Content-Type    | Set to <b>multipart/x-srgs-audio</b> to indicate that we are sending both the audio file and the grammar (srgs).             |
| X-SpeechContext | The context for speech recognition, such as gaming, business, etc. (Optional)  |

**Note.** Each MIME part has a content disposition, where the type of file to be uploaded is specified. These file types include pronunciation, grammar, grammar prefix hints, grammar altgram hints, and audio. In this case, the content-disposition will be **x-grammar-altgram** for using alternate grammar and **x-voice** for the audio file. For more information, see the custom speech documentation and the WC3 specification.

**Note.** There are restrictions on what sound files are accepted. For example, they must have a single channel. See the speech documentation for more information.

More information can be found in the documentation at: [Speech API](#).



## Exercise: Command Line Tool (curl)

### Step 1. Download the Alternate Grammar File

Obtain the file **nauticalTasks.grxml** and place it the location where you are calling curl. Open it in a text editor. As you can see, it consists of a list of tasks:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE grammar PUBLIC "-//W3C//DTD GRAMMAR 1.0//EN"
    "http://www.w3.org/TR/speech-grammar/grammar.dtd">
<grammar version="1.0" mode="voice" root="nautical" tag-
format="semantics/1.0" xmlns="http://www.w3.org/2001/06/grammar">
  <rule id="nautical" scope="public">
    <one-of>
      <item>Berth inspection</item>
      <item>Bilge inspection</item>
      <item>Record heading</item>
      <item>Monitor fuel</item>
      <!-- etc... -->
    </one-of>
  </rule>
</grammar>
```

### Step 2. Download the sound file

Obtain the file **berthInspection.wav** and place it in the same directory. It says, "Berth inspection". The word "berth" is unlikely to be chosen by the generic recognizer, but by using the **nauticalTasks.grxml** file, we can have "Berth inspection" be something it expects to recognize.

### Step 3. Call the Simple Speech Recognizer

**Note:** The curl exercises are designed for Unix-based terminals, such as Unix, MacOS, and Cygwin. They use the backslash (\) in order to continue onto more than one line. To run them on a Windows command prompt, substitute each backslash for a caret (^), or remove the backslashes and put it all on one line.

Let's start by seeing how well the generic recognizer works. Follow these steps:

1. Paste the following into a text editor. Note that by using the **data-binary** flag, you are adding the file into the POST body as well as setting the **Content-Length** header to the length of the file.

```
curl --request POST --insecure \
--header "Authorization: Bearer {token}" \
--header "Accept: application/json" \
--header "Content-Type: audio/wav" \
--data-binary @berthInspection.wav \
https://api.att.com/speech/v3/speechToText
```

2. Replace {token} with the access\_token you obtained in Lab Exercise #1.
3. Open up a terminal that has curl.



- Paste the edited curl command into the terminal and hit Enter. You should see a return value in JSON. Once it is formatted, it will look like this:

```
{
  "Recognition":{
    "Status":"OK",
    "ResponseId":"f96f13466f71a0aa53f209b904672448",
    "Info":{
      "metrics":{
        "audioTime":1.5,
        "audioBytes":133120
      }
    },
    "NBest":[
      {
        "wordScores":[
          0.1,
          0.1
        ],
        "Confidence":0.11,
        "Grade":"accept",
        "ResultText":"First inspection.",
        "words":[
          "First",
          "inspection."
        ],
        "LanguageId":"en-US",
        "Hypothesis":"first inspection"
      }
    ]
  }
}
```

It correctly recognized “inspection”, but because “berth” is an unusual word, it chose “first” instead. Note that the confidence value is quite low.

#### Step 4. Call the Speech Recognizer with the Alternate Grammar

Now let’s give it the alternate grammar hints. Follow these steps:

- Paste the following into a text editor. Note that we are using **x-grammar-altgram** for the part that contains the srgs file.

```
curl --request POST --insecure \
--header "Authorization: Bearer {token}" \
--header "Accept: application/json" \
--header "Content-type: multipart/x-srgs-audio" \
--header "X-SpeechContext: GenericHints" \
--form "x-grammar-altgram=@nauticalTasks.grxml;\
type=application/srgs+xml" \
--form "x-voice=@berthInspection.wav;type=audio/wav" \
https://api.att.com/speech/v3/speechToTextCustom
```

- Replace {token} with the access\_token you obtained in Lab Exercise #1.



3. Open up a terminal that has curl.
4. Paste the edited curl command into the terminal and hit Enter. You should see a return value in JSON. Once it is formatted, it will look like this:

```
{
  "Recognition": {
    "Status": "OK",
    "ResponseId": "ae01811980fc494156688a665753a551",
    "Info": {
      "metrics": {
        "audioTime": 1.5,
        "audioBytes": 133120
      }
    },
    "NBest": [
      {
        "wordScores": [
          0.569,
          0.769
        ],
        "NluHypothesis": {
          "OutComposite": [
            {
              "Out": "Berth inspection",
              "Grammar": "altgram"
            }
          ]
        },
        "Confidence": 0.67,
        "Grade": "accept",
        "ResultText": "Berth inspection",
        "Words": [
          "Berth",
          "inspection"
        ],
        "LanguageId": "en-US",
        "Hypothesis": "Berth inspection"
      }
    ]
  }
}
```

Now it has correctly recognized the phrase as "Berth inspection", and the confidence value is much higher.

**Note:** See Lab #2 for a discussion on how to interpret the results.



## Summary

*You've used the custom version of the speech-to-text API and shown how you can use an alternate grammar to more accurately convert speech to text where certain phrases are expected that the generic recognizer may not choose as its recognition result.*



# Appendix A: Installing curl for Windows

Pre-requisites: Windows 7 or later

curl (or cURL) is a powerful command-line tool for making HTTP calls. It is already installed on Mac computers.

## Downloading curl

Use the following steps to install curl:

1. Open <http://curl.haxx.se/dlwiz?type=bin> in a browser.
2. Select your operating system in the dropdown box: either Windows /Win32 or Win 64. Click **Select!**
3. For Win 32, choose whether you will use curl in a Windows Command Prompt (**Generic**) or in a Cygwin terminal (**cygwin**). For Win 64, choose whether you will use curl in a Windows Command Prompt (**Generic**) or MinGW (**MinGW64**). Click **Select!**
4. If required, choose your Windows operating system. **Finish.**
5. Click **Download** for the version which has SSL enabled.
6. Choose a version with support for SSL.
7. Open the downloaded zip file. Extract the files to an easy-to-find place, such as C:\Program Files.

## Testing curl

1. Open up the Windows Command Prompt terminal. (From the Start menu, click Run, then type **cmd**.)
2. Set the path to include the directory where you put **curl.exe**. For example, if you put it in **C:\Program Files\curl**, then you would type the following command:

```
set path=%path%; "c:\Program Files\curl"
```

3. Type *curl*.

You should see the following message:

```
curl: try 'curl -help' or 'curl -message' for more information
```

This means that curl is installed and the path is correct.

4. Type:

```
curl https://api.att.com
```

You should see JSON returned:

```
{
  "error": "invalid API request"
}
```





## Troubleshooting

### SSL certification error

If you see an SSL certification error, add a -k flag into your curl command.

### https not supported error

If you get an error that says, "Protocol http not supported or disabled in libcurl", then you need a different version of curl. Make sure you have downloaded one that says SSL enabled. If you have downloaded the Win64 version, try the Win32 version instead, even if you are on a 64 bit machine.



## Appendix B: SDKs and Tools

---

Several Software Development Kits (SDKs) and other tools are available to make it easier to integrate the AT&T APIs into your apps. Rather than requiring you to make low-level HTTP calls, these SDKs and tools provide classes and methods that reduce the amount of code to call the underlying API requests. The following tools and SDKs are available from the Developer Platform website at [SDKs and Tools](#):

- Microsoft® SDK (certified for Windows® 2008 Server)
- Android SDK
- iOS SDK
- AT&T API SDK for Adobe® PhoneGap®
- AT&T API Module for Appcelerator® Titanium®
- AT&T API SDKs for Windows® and Visual Studio® Extensions
- AT&T Toolkit for Salesforce® Platform
- AT&T API Adapters for IBM® Worklight®
- DeviceAnywhere Virtual Developer Lab Emulator
- Appery.io Plugin
- Viafo Plugin
- StackMob Enterprise Marketplace Custom Integration

Also, CodeKits (source code for classes providing wrappers to the APIs) in the following languages will be available soon:

- Java
- C#
- PHP
- Ruby

Finally, you can view sample code for several languages and platforms at the [Sample Apps](#) page.



## Appendix C: Running the iOS Speech Sample Apps

Pre-requisites: MacOS, XCode

Length: 20 minutes

**In this exercise:**

**Download the Sample Apps**

**SimpleSpeech App**

**Simple Speech UI**

**SimpleTTS App**

### Download the Sample Apps

AT&T provides iOS sample apps that illustrate many aspects of the Speech API. The sample app code uses the iOS Speech SDK, which makes it easy to make Speech API calls.

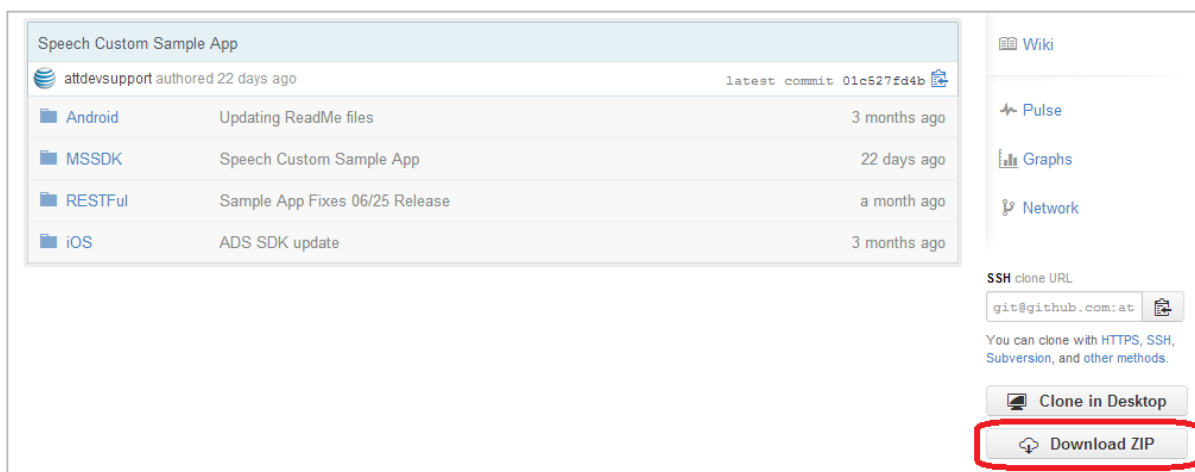
To download the sample code, follow these steps:

1. In a browser, navigate to:

[https://github.com/attdevsupport/ATT\\_APIPlatform\\_SampleApps](https://github.com/attdevsupport/ATT_APIPlatform_SampleApps)

**Note:** You can also get to this site from the Developer Program site. From the **APIs and Tools** menu, choose **Sample Apps**. For the **Language** dropdown, select **AT&T SDK – iOS**. Next to **github source**, click **Download**. Click the Code button (<>) on the right hand navigation bar.

2. Click on the **Download ZIP** button.



3. Unzip the downloaded file into a location.



## SimpleSpeech App

The SimpleSpeech App shows how to make a Basic Speech API call using the Speech SDK. It uses the WebSearch context, and then performs a web search on the phrase that was spoken.

First you will need the Speech SDK:

1. In a browser, navigate to:  
<http://developer.att.com/developer/forward.jsp?passedItemId=12500061>
2. In the **iOS SDK** row, click **Download**.

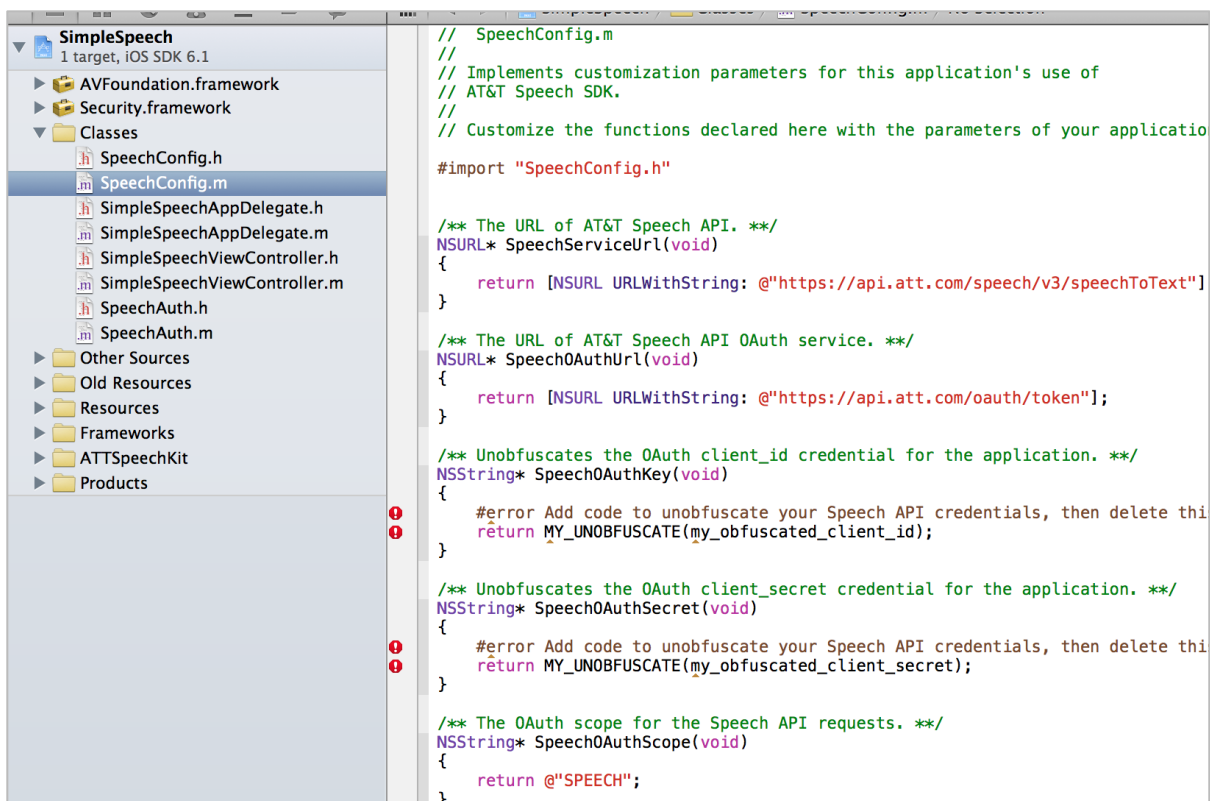
|                 |     |  |  |
|-----------------|-----|--|--|
| Android SDK     | SDK | Speech Sample Code on <a href="#">Github</a> )                                 | <a href="#">Download</a>                         |
| iOS SDK         | SDK | Advertising and Speech (Text to Speech Sample Code on <a href="#">Github</a> ) | <a href="#">View</a><br><a href="#">Download</a> |
| AT&T API Plugin |     | All APIs are supported except AT&T   | <a href="#">Github</a>                           |

3. Open the downloaded file **iOS\_SDK.zip**.
4. Within the file, open the folder that begins with **ATTSpeechKit**.
5. Copy the files **ATTSpeechKit.a** and **ATTSpeechKit.h** into the **ATTSpeechKit** subfolder of the **SimpleSpeech** folder.

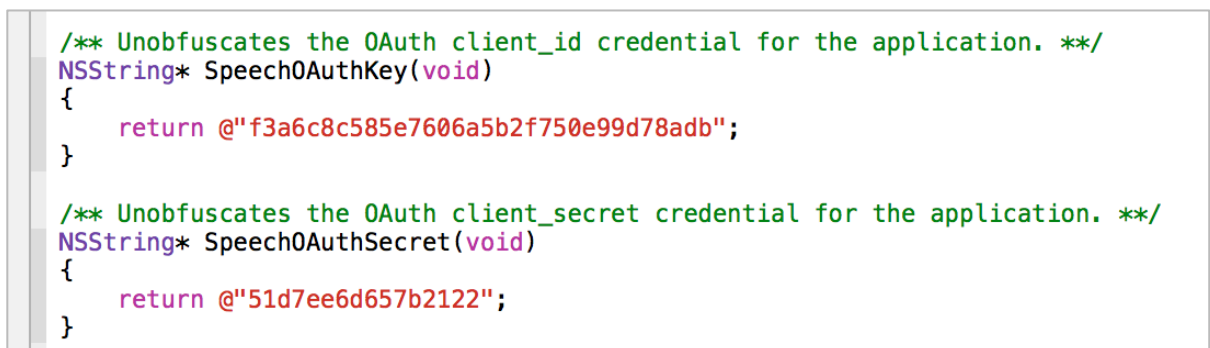
To run the sample app, follow these steps:

1. From the unzipped downloaded sample files, open the following folders: **iOS**, **Speech**, and then **SimpleSpeech**.
2. Double-click the file **SimpleSpeech.xcodeproj**.
3. On the top left, open the **SimpleSpeech** project, and the the **Classes** folder.
4. Click on **SpeechConfig.m**. Note the places where an error is marked by an exclamation point.

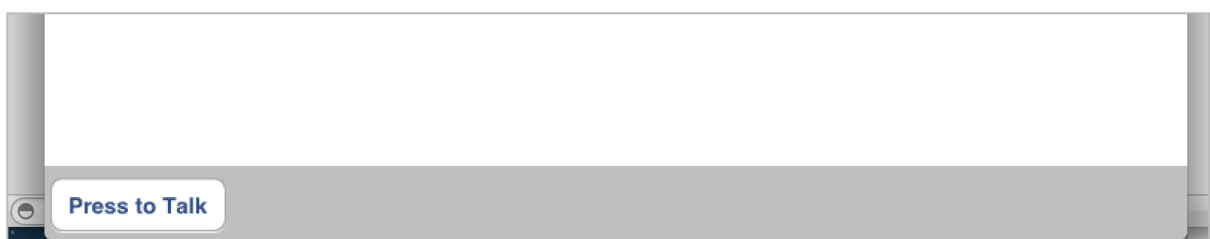




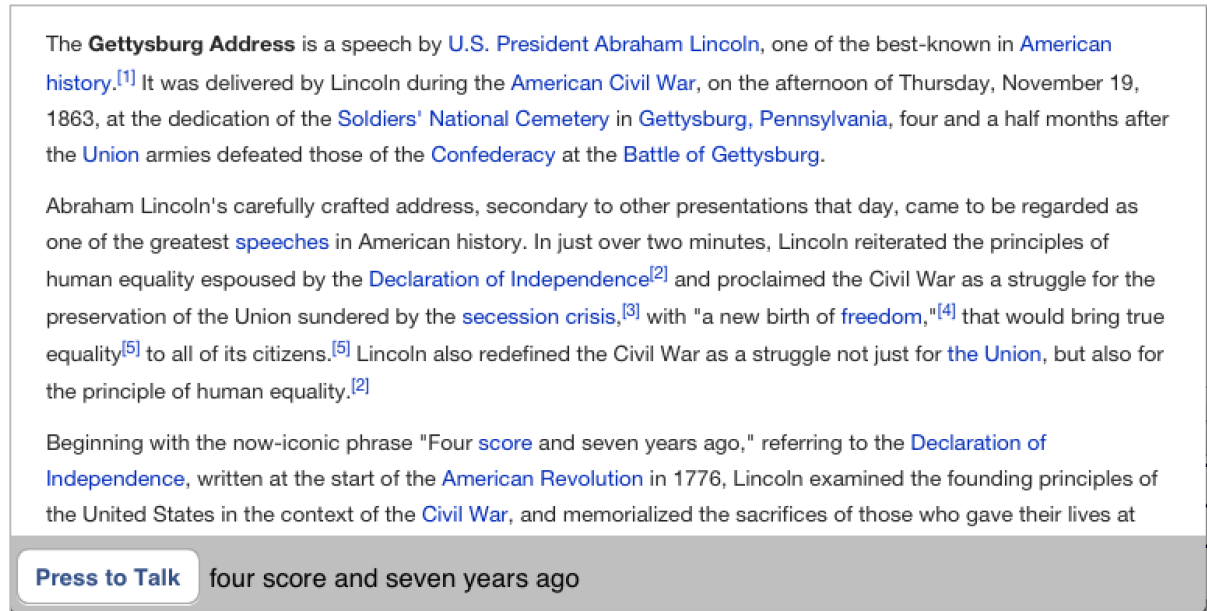
- Replace the **#error** and **MY\_UNOBFUSCATE** lines with lines that return your app key and your secret. Read the **Note** below that explains how you use obfuscation in a real application.



- Click the **Run** button at the top left.
- After a while, the iOS Simulator window will appear. Click the **Press to Talk** button at the bottom.



8. Immediately speak a phrase, followed by a second of silence. The phrase will appear at the bottom of the screen, and the app will navigate to a website based on the phrase.



9. Click the **Stop** button to stop the application.

**Note:** You should never have your app key and secret hardcoded as strings in a production application. It is too easy for someone to examine the compiled code and to extract the values. Instead, they should be obfuscated in some fashion and the MY\_UNOBFUSCATE line should reverse the obfuscation.

Some important parts of the code to examine:

- In the **SpeechAuth.m** file, you will find the method **authenticationForService**, which makes the call to obtain an access token without consent.
- In the **SimpleSpeechViewController.m** file, you will find:
  - The method **prepareSpeech**, which creates an **ATTSpeechService** object and sets the context to "WebSearch". It then obtains the access token.
  - The **listen** method, which sets some **XArg** parameters, and then calls the **startListening** method of the **ATTSpeechService** object.
  - The **speechServiceSucceeded** method, which uses the **responseStrings** property of the **ATTSpeechService** object to find the best recognized text.

More information is available in the Developer Guide for the SDK at:  
[http://developer.att.com/home/api/sdk\\_tools/SpeechSDK-DeveloperGuideiOS-2.1.0.pdf](http://developer.att.com/home/api/sdk_tools/SpeechSDK-DeveloperGuideiOS-2.1.0.pdf)

## SimpleGrammar App

The SimpleGrammar App shows how to make a Custom Speech to Text API call using the Speech SDK. It uses a custom grammar that is set up to accept pizza orders, recognizing size, crust type, type of pizza, and ingredients.

Follow these steps, similar to getting the SimpleGrammar app running:



1. In the **SimpleGrammar** folder, open the **AttSpeechKit** folder and copy in the **ATTSpeechKit.a** and **ATTSpeechKit.h** files, just as you did for the SimpleSpeech app.
2. In the **SimpleGrammar** folder, open the file **SimpleGrammar.xcodeproject**.
3. On the top left, open the **SimpleGrammar** project, the **SimpleGrammar** folder, and then the **Classes** folder.
4. Make the same changes to **SpeechConfig.m** as you did for the **SimpleSpeech** project.
5. Click the **Run** button.
6. Click the **Push and Talk to Order a Pizza** button. Try saying the phrase, "Large thin crust Hawaiian pizza with extra cheese". It should be able to sort out the kind, size, crust type, and ingredients. (You may have to scroll down to see them all.)
7. Try other phrases. The grammar takes the form:

```
[personal|small|medium|large] [(thin|thick) crust]
(Hawaiian|meat lovers|vegetarian|pepperoni|barbeque chicken|supreme)
pizza
[with (#ingredients [and #ingredients])]+
```

8. Click **Stop** in XCode to stop the program.

Some important parts of the code to examine:

- In the **Resources** folder, there is a file called **pizza.srgs**. This file defines the grammar, including the types of crust, sizes, and ingredients.
- The **SimpleGrammarData.m** file loads the srgs file (**loadGrammarData** method) and parses the response into slots for kind of pizza, size, crust and ingredients (**parseResponse** method). It uses the **NluHypothesis** element for easy parsing.
- In the **SimpleGrammarViewController.m** file note that:
  - The **AttSpeechService** object's **delegate** is set to **self** so that certain **SimpleGrammarViewController** will be called during the recognition process.
  - The **AttSpeechService** object's **speechContext** is set to "GrammarList", so that the custom grammar will be used.
  - The **AttSpeechService** object's **startListening** method is called when it's time to record speech.
  - As part of the **speechServiceSendingPartsBeforeAudio** method, it adds the necessary srgs file to the **AttSpeechService** object.

## Simple TTS App

The SimpleTTS App shows how to make a Text to Speech API call using the Speech SDK. You say a phrase, and it recognizes it using speech to text, and then it repeats it back to you using text to speech.

Follow these steps, similar to getting the SimpleTTS app running:

1. In the **SimpleTTS** folder, open the **AttSpeechKit** folder and copy in the **ATTSpeechKit.a** and **ATTSpeechKit.h** files, just as you did for the SimpleSpeech app.
2. In the **SimpleTTS** folder, open the file **SimpleTTS.xcodeproject**.
3. On the top left, open the **SimpleTTS** project and then the **Classes** folder.
4. Make the same changes to **SpeechConfig.m** as you did for the **SimpleSpeech** project.
5. Click the **Run** button.
6. Click the **Press to Talk** button. Try asking a question. The application will repeat your question back to you, followed by the phrase, "I wish I knew the answer."



7. Click **Stop** on XCode to stop the program.

The important parts of the code to examine are all in **TTSRequest.m**:

- In the **TTSRequest.m** file, the **forService:withOAuth:** factory creates the request.
- In the **TTSRequest.m** file, the **postText:forClient:** method adds the text to be spoken to the body and adds a Context-Type header of "text/plain".
- In the **SimpleTTSViewController.m** file, the **startTTS** method creates the **TTSRequest** object and then calls its **postText:forClient:** method to convert the text to audio data, which it then plays.





## Appendix D: Running the Android Speech Sample Apps

Pre-requisites: Eclipse from the Android Developer Tools bundle.

Length: 20 minutes

### In this exercise:

**Download the Sample Apps**

**SimpleSpeech App**

**SimpleGrammar App**

**SimpleTTS App**

### Compatibility Note

Because Android hardware and software has more versions, the sample applications may not run on all Android devices, especially older devices.

### Download the Sample Apps

AT&T provides Android sample apps that illustrate many aspects of the Speech API. The sample app code uses the Android Speech SDK, which makes it easy to make Speech API calls.

To download the sample code, follow these steps:

1. In a browser, navigate to:

[https://github.com/attdevsupport/ATT\\_APIPlatform\\_SampleApps](https://github.com/attdevsupport/ATT_APIPlatform_SampleApps)

**Note:** You can also get to this site from the Developer Program site. From the **APIs and Tools** menu, choose **Sample Apps**. For the **Language** dropdown, select **AT&T SDK – Android**. Next to **github source**, click **Download**. Click the Code button (<>) on the right hand navigation bar.

2. Click on the **Download ZIP** button.

Speech Custom Sample App

attdevsupport authored 22 days ago latest commit 01c527fd4b

| File    | Description                    | Time         |
|---------|--------------------------------|--------------|
| Android | Updating ReadMe files          | 3 months ago |
| MSSDK   | Speech Custom Sample App       | 22 days ago  |
| RESTful | Sample App Fixes 06/25 Release | a month ago  |
| iOS     | ADS SDK update                 | 3 months ago |

Wiki

Pulse

Graphs

Network

SSH clone URL

git@github.com:at:

You can clone with HTTPS, SSH, Subversion, and other methods.

Clone in Desktop

**Download ZIP**

3. Unzip the downloaded file into a location.



## Opening the Sample Apps

The SimpleSpeech App shows how to make a Basic Speech API call using the Speech SDK. It uses the WebSearch context, and then performs a web search on the phrase that was spoken.

First you will need the Speech SDK:

1. In a browser, navigate to:  
<http://developer.att.com/developer/forward.jsp?passedItemId=12500061>
2. In the **Speech SDK for Android** row, click **Download**.

|                        |     |        |  |
|------------------------|-----|--------|--|
| for iOS                |     |        | <a href="#">Download</a>                         |
| Speech SDK for Android | SDK | Speech | <a href="#">View</a><br><a href="#">Download</a> |
| Speech SDK for Java    |     |        | <a href="#">View</a>                             |

3. Open the downloaded file **SpeechKit\_Android\_X.X.X.zip**, where X.X.X is the version number.
4. Within the file, open the folder that begins with **ATTSpeechKit-Android**.
5. Copy the file **ATTSpeechKit.jar** into the **libs** subfolder of the **SimpleSpeech**, **SimpleSpeechUI**, and **SimpleTTS** folders.

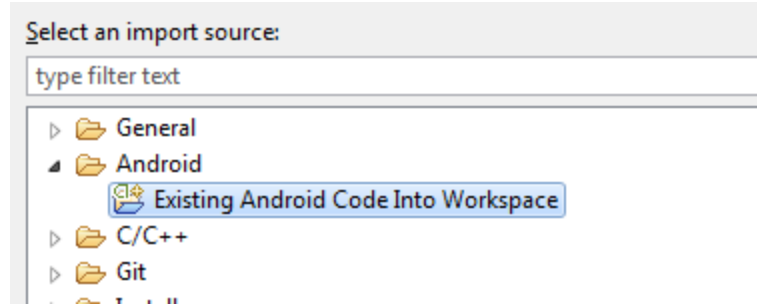
To run the sample app, follow these steps:

1. Open the Android SDK Manager. In the Android Eclipse bundle folder, this is an executable at the top level called **SDK Manager.exe**.
2. AT&T Speech SDK for Android supports applications that target version 2.2 and higher of the Android Platform. Make sure that, at a minimum, you have Android 2.2 (API 8) installed and shown in Android SDK Manager. More information is available in the Developer Guide for the SDK at:  
[http://developer.att.com/home/api/sdk\\_tools/SpeechSDK\\_DeveloperGuideAndroid.pdf](http://developer.att.com/home/api/sdk_tools/SpeechSDK_DeveloperGuideAndroid.pdf)

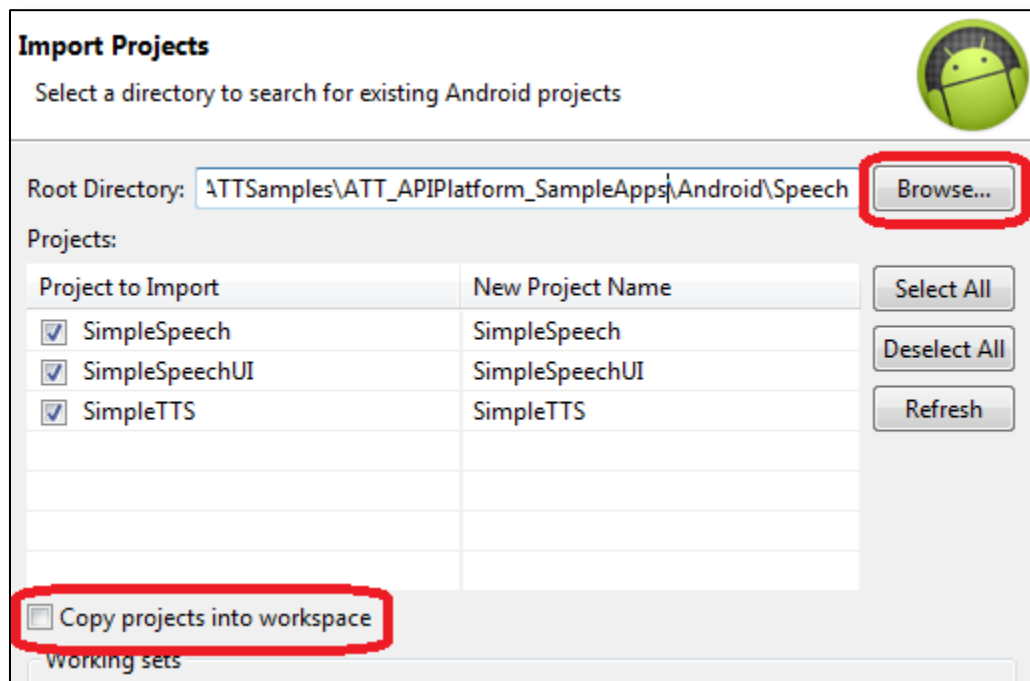
| Packages |                        |     |      |           |
|----------|------------------------|-----|------|-----------|
|          | Name                   | API | Rev. | Status    |
| ▶        | Android 3.2 (API 13)   |     |      |           |
| ▶        | Android 3.1 (API 12)   |     |      |           |
| ▶        | Android 3.0 (API 11)   |     |      |           |
| ▶        | Android 2.3.2 (API 10) |     |      |           |
| ▶        | Android 2.2 (API 8)    |     |      |           |
|          | SDK Platform           | 8   | 3    | Installed |
|          | Samples for SDK        | 8   | 1    | Installed |
|          | Google APIs            | 8   | 2    | Installed |
| ▶        | Android 2.1 (API 7)    |     |      |           |
| ▶        | Android 1.6 (API 4)    |     |      |           |

3. Close the Android SDK Manager.
4. Open eclipse from the Android Developer Tools bundle. (**eclipse.exe** in the **eclipse** folder.)
5. From the **File** menu, choose **Import...**
6. Open the **Android** folder and choose **Existing Android Code Into Workspace**.





- Click the **Browse...** button and navigate to the folder where you unzipped the sample applications. Navigate to the **Android** folder, then the **Speech** folder. If you wish to copy the source files into your workspace folder, check the **Copy projects into workspace** checkbox.



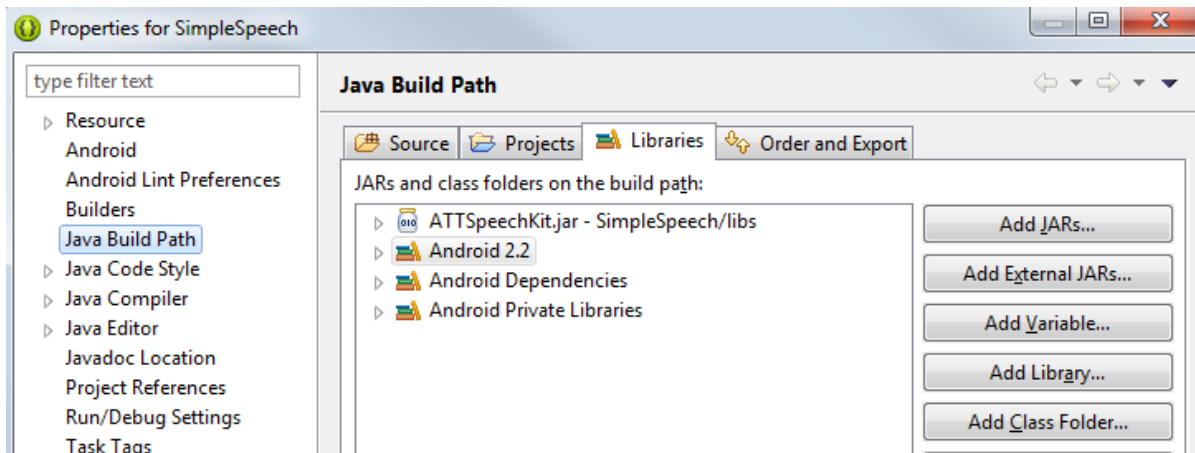
- Click **Finish**.

## SimpleSpeech App

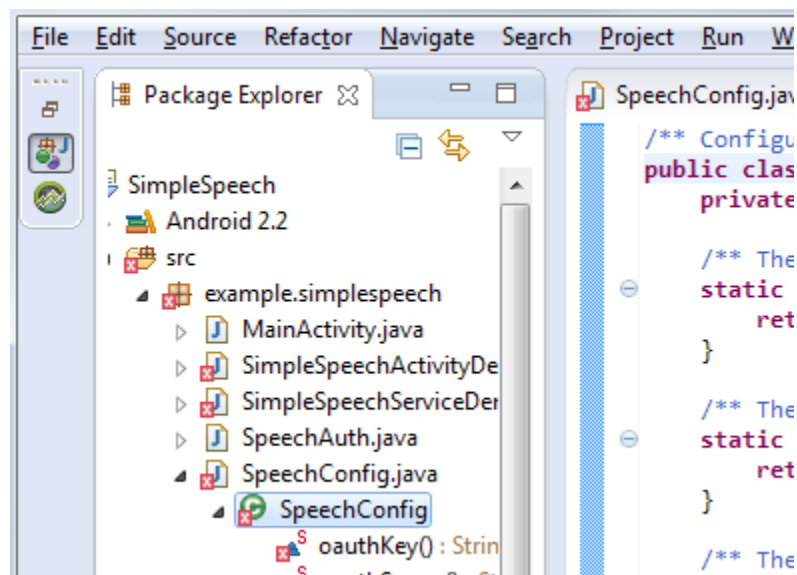
The SimpleSpeech App shows how to make a Basic Speech API call using the Speech SDK. It uses the WebSearch context, and then performs a web search on the phrase that was spoken. Follow these steps to run the SimpleSpeech app.

- Click on **SimpleSpeech** in the Package Explorer.
- Under the **Project** menu, click **Properties**. On the left-hand navigator bar, click **Java Build Path**. Click the **Libraries** tab. Click the **Add JARs...** button and navigate to **ATTSpeechKit.jar**.

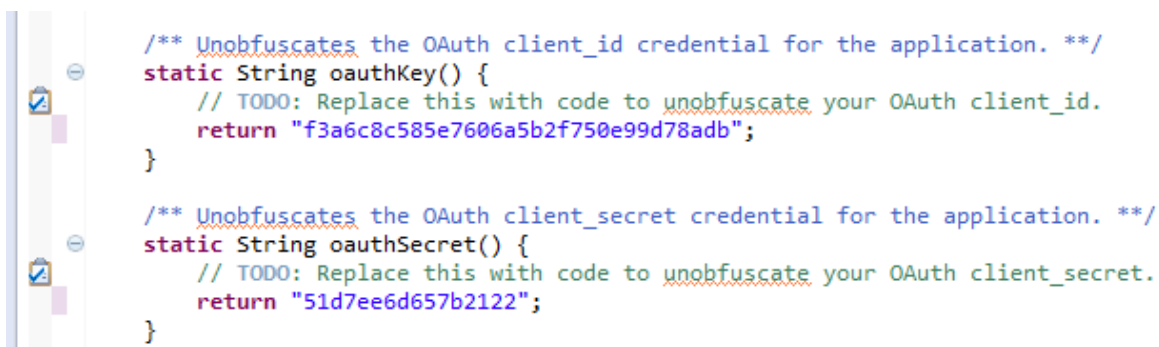




3. Now click the **Order and Export** tab and check the checkbox next to **ATTSpeechKit.jar**. Click **OK** to close the project properties dialog.
4. In the Package Explorer, open the folders **SimpleSpeech**, **src**, and then **example.simplespeech**. Open **SpeechConfig.java** and then **SpeechConfig**.

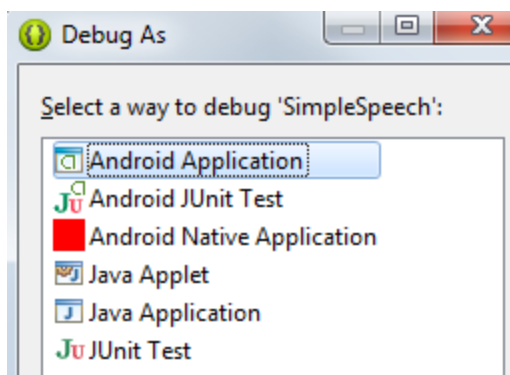


5. In the **oauthKey** and **oauthSecret** methods, replace the **MY\_UNOBFUSCATE** lines with lines that return your app key and your secret. Read the **Note** below that explains how you use obfuscation in a real application.

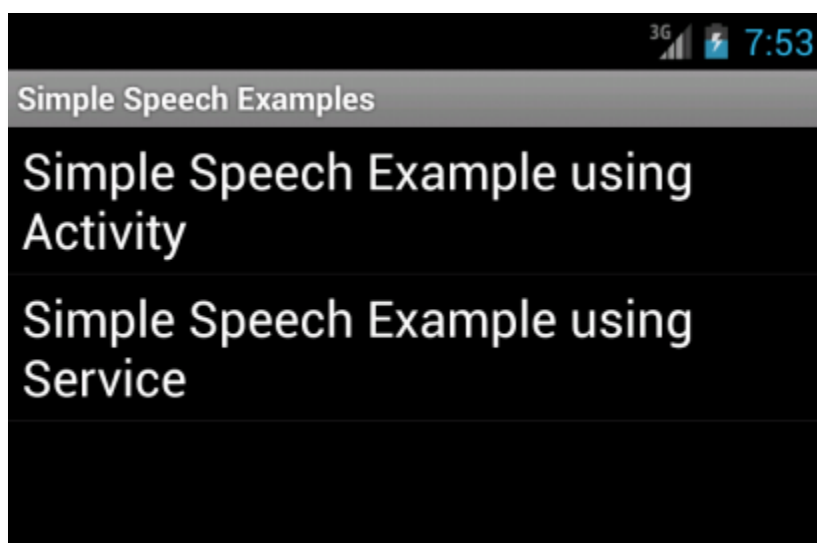


6. From the **Run** menu, choose **Debug**.
7. Select **Android Application**.





8. After a while, the app will appear in either the simulator or on a connected Android device. We recommend using an external device, as the simulator is often so slow that it interferes with the quality of the recorded speech.

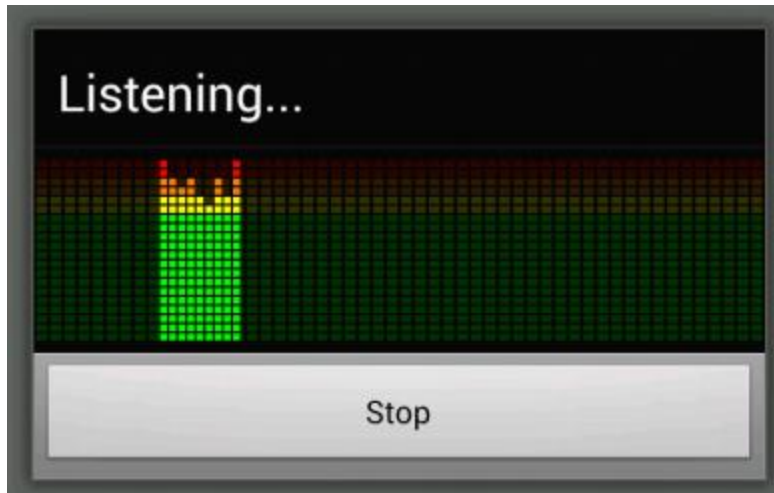


9. Tap **Simple Speech Example using Activity**. The app will appear.
10. Tap the **Press to Talk** button at the bottom.



11. When the text "Listening" appears, speak a phrase, followed by a second of silence. The app will display the recorded sound as a graphic.





12. Once the phrase is recognized, it will do a web search on that phrase and show the results.

**Note:** You should never have your app key and secret hardcoded as strings in a production application. It is too easy for someone to examine the compiled code and to extract the values. Instead, they should be obfuscated in some fashion and the **myUnobfuscate** method should reverse the obfuscation.

**Note:** You can also try the `ATTSpeechService`, but the behavior of the app is identical. `ATTSpeechActivity` is modeled after the **android.speech** package. If you are already using the built-in Google speech recognition, it's easy to use the `ATTSpeechActivity` code to port your app from **android.speech**. However, if you are starting from scratch, `ATTSpeechService` provides simpler code. It also supports callback interfaces that give the developer more feedback during speech interaction, so it's also a better choice for advanced speech applications.

The important parts of the code to examine are the **SimpleSpeechActivityDemo** and **SimpleSpeechServiceDemo** classes. Look in the `startSpeechActivity` and `startSpeechService` methods for examples of setting up and starting a speech interaction. Look in the `onActivityResult` method and `ResultListener/ErrorListener` inner classes for handling recognition responses and errors.

- The **startSpeechActivity** and **startSpeechService** methods set up and start speech interactions.
- The **onActivityResult** method and the **ResultListener** and **ErrorListener** inner classes handle recognition responses and errors.

## SimpleSpeechUI

The `SimpleSpeechUI` sample app includes Android source code and an Eclipse project to show how to call the AT&T Speech SDK from an application that makes use of a custom speech user interface. The app displays a simple client-side mashup. Pressing a button initiates a speech interaction, a text area shows the recognition result, and a web view uses the recognized speech to search a website. The text area also shows the progress of the speech interaction, and the button changes to "stop" or "cancel" during the phases of the interaction.

Follow the instructions for the `SimpleSpeech` app to:

1. Add the **ATTSpeechKit.jar** library.
2. Add your app key and client codes to the code in the **SpeechConfig.java** file.
3. Run the app in debug mode.

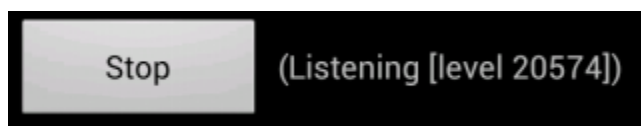


Once the app is running, follow these steps:

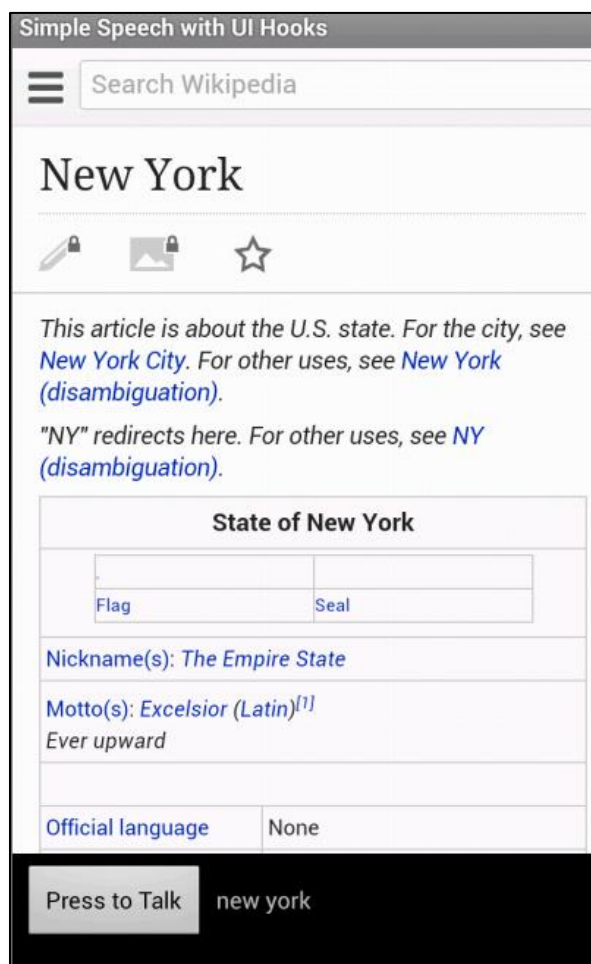
1. Tap the **Press to Talk** button at the bottom.



2. When the button text changes to "Stop", speak a phrase, and tap the button when you are finished. The app will display the volume level to the right of the button. Once you click "Stop", it will change to "Processing".



3. Once processed, the recognized text will be displayed to the right of the button, and the web search will be displayed in the main section of the app.



Some important features of the code:

- The main code of the sample app is in the **SimpleSpeechUIDemo** class. The **setupSpeechService** method and **SpeechButtonListener** inner class contain examples of setting up and starting a speech interaction.
- The **SpeechResultListener** and **SpeechErrorListener** inner classes handle recognition responses and errors.
- The **SpeechStateListener** and **SpeechLevelListener** inner classes handle SpeechKit callbacks for a custom UI.

## SimpleTTS App

The SimpleTTS App is a variation on the SimpleSpeech app, but it uses text-to-speech so that it can tell you to press the talk button and it reads back to the text that it recognized.

Follow the instructions for the SimpleSpeech app to:

1. Add the **ATTSpeechKit.jar** library.
2. Add your app key and client codes to the code in the **SpeechConfig.java** file.
3. Run the app in debug mode.

You will hear spoken instructions. Tap the **Press to Talk** button and say a phrase, just as you did for the SimpleSpeech app.

The important parts of the code to examine are all in **SimpleTTSDemo.java**:

- The **startSpeechService** method sets up and starts speech recognition.
- The **readyForSpeech** method starts text-to-speech playback.
- The **TTSCient** inner class has an example of handling text-to-speech responses.
- The **ResultListener** and **ErrorListener** classes handle recognition responses and errors.

