AT&T API Platform Advertising and Speech SDKs for Android™

Revision **2.3.2**
Revision Date **April 2013**

Legal Disclaimer

This document and the information contained herein (collectively, the "**Information**") is provided to you (both the individual receiving this document and any legal entity on behalf of which such individual is acting) ("**You**" and "**Your**") by AT&T, on behalf of itself and its affiliates ("**AT&T**") for informational purposes only. AT&T is providing the Information to You because AT&T believes the Information may be useful to You. The Information is provided to You solely on the basis that You will be responsible for making Your own assessments of the Information and are advised to verify all representations, statements and information before using or relying upon any of the Information. Although AT&T has exercised reasonable care in providing the Information to You, AT&T does not warrant the accuracy of the Information and is not responsible for any damages arising from Your use of or reliance upon the Information. You further understand and agree that AT&T in no way represents, and You in no way rely on a belief, that AT&T is providing the Information in accordance with any standard or service (routine, customary or otherwise) related to the consulting, services, hardware or software industries.

AT&T DOES NOT WARRANT THAT THE INFORMATION IS ERROR-FREE. AT&T IS PROVIDING THE INFORMATION TO YOU "AS IS" AND "WITH ALL FAULTS." AT&T DOES NOT WARRANT, BY VIRTUE OF THIS DOCUMENT, OR BY ANY COURSE OF PERFORMANCE, COURSE OF DEALING, USAGE OF TRADE OR ANY COLLATERAL DOCUMENT HEREUNDER OR OTHERWISE, AND HEREBY EXPRESSLY DISCLAIMS, ANY REPRESENTATION OR WARRANTY OF ANY KIND WITH RESPECT TO THE INFORMATION, INCLUDING, WITHOUT LIMITATION, ANY REPRESENTATION OR WARRANTY OF DESIGN, PERFORMANCE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, OR ANY REPRESENTATION OR WARRANTY THAT THE INFORMATION IS APPLICABLE TO OR INTEROPERABLE WITH ANY SYSTEM, DATA, HARDWARE OR SOFTWARE OF ANY KIND. AT&T DISCLAIMS AND IN NO EVENT SHALL BE LIABLE FOR ANY LOSSES OR DAMAGES OF ANY KIND, WHETHER DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, PUNITIVE, SPECIAL OR EXEMPLARY, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, LOSS OF GOODWILL, COVER, TORTIOUS CONDUCT OR OTHER PECUNIARY LOSS, ARISING OUT OF OR IN ANY WAY RELATED TO THE PROVISION, NON-PROVISION, USE OR NON-USE OF THE INFORMATION, EVEN IF AT&T HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSSES OR DAMAGES.

Table of Contents

1. Advertising SDK

The AT&T API Platform Advertising SDK for AndroidTM (Advertising SDK) provides a library, documentation, and sample code that assist developers in building mobile applications using the AT&T Advertising API. The library contains classes and methods for handling authentication and authorization, requests for ad content, and rendering the advertising component in the designated frame.

The Advertising API enables your app to support advertisements within the app. This allows the developer of the application to collect a revenue share of the advertisement. When users click the advertisement in the app, they are redirected to the web page for the advertisement.

Android is the software platform for the Android Smart Phones, and Tablets. This SDK supports applications that are developed with the Android Software Development Kit. The Android SDK, which is available at http://developer.android.com, provides the tools and APIs necessary to develop applications on the Android platform using the Java programming language. This SDK supports applications that target Android version 2.2 and higher.

Since the advertisements are for mobile devices, only the following sizes of advertisements are supported. See the MMA Universal Mobile Ad Package web page on the Mobile Marketing Association web site for further information.

- 120 pixels by 20 pixels
- 168 pixels by 28 pixels
- 216 pixels by 36 pixels
- 300 pixels by 50 pixels
- 300 pixels by 250 pixels
- 320 pixels by 50 pixels

1.1 About the Advertising API

The Advertising API is packaged in the static library *adsapi.jar*. The library exports the **ATTAdView** class, which contains methods that perform the following tasks.

- Handle authentication and authorization
- Handle click events in the advertising component
- Get the advertisement
- Render the advertisement content to the designated area of your app

The library components require the App key and secret key that were created for your app when you register your app on the AT&T Developer Program web site. The methods in the library contain parameters that allow you to create an advertising component based on filtering criteria and display it in the designated area of your app.

1.2 Prerequisites

To develop Advertising API apps for Android, install the following software on your development machine.

- The Android SDK, version 2.2 or newer.
- A supported IDE, such as Eclipse.

To install the Android SDK and Eclipse IDE, follow the instructions on the Android Developer web site.

In addition to the software required to develop Android applications, the following prerequisites must be met to develop applications using the Advertising API for Android.

- Knowledge of how to program Android applications using Java.
  If you are not familiar with Android development, see the Android developer documentation.
- The Advertising SDK for Android from the AT&T Developer Program web site.
- The permissions and minimum SDK version must be updated in the manifest for the app.
- The project must contain at least one activity and layout containing the UI elements of the app.

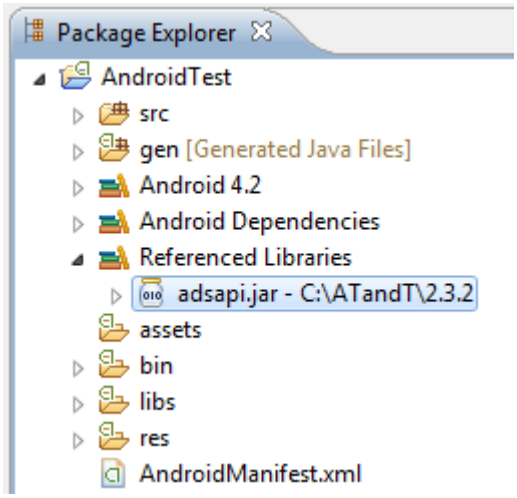
1.3 Setting up your Android Project

To incorporate the Advertising API into an Android project in Eclipse, perform the following steps.

1. Open the Android project for your application in Package Explorer.
2. Select **Project**, **Properties**.
3. Select the **Java Build Path** entry in the navigation bar.
4. Click the **Libraries** tab.
5. Click the **Add External JARs** button on the right side of the properties window.
6. Browse to the folder where you extracted the Advertising files.

Select the library *adsapi.jar*.

7. Click **OK**. The library should appear in Package Explorer, as shown in the following screen shot.



Double-click *AndroidManifest.xml* and add the following permissions.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission. ACCESS_NETWORK_STATE " />
```

8. Open the activity and layout that contain the UI elements of your application.
9. Follow the instructions in the Using the ATTAdView Class section.
10. Build your application that includes the AT&T Ad SDK.

1.4 Using the ATTAdView Class

This section describes how to use the **ATTAdView** class in your app, including implementing the listener callback methods.

1.4.1 Configuring ATTAdView Properties

You must set values for the following properties of the **ATTAdView** class to authenticate and authorize your app, and use the Advertising API.

- **appKey**
- **category**
- **secret**
- **udid**

You can set additional properties in the **ATTAdView** object to filter the ad content. These properties include keywords, city, and zip code. For a full list of properties, see the ATTAdView Properties section.

1.4.2 Initializing the ATTAdView Object

The **ATTAdView** class is a member of the **com.att.ads** namespace; the **ATTAdViewListener** class is a member of the **com.att.listener** namespace. Add the following `import` statements to your app code to use these classes.

```
import com.att.ads.*;
import com.att.ads.listeners.*;
```

To initialize an **ATTAdView** object, insert code like the following in the **onCreate** method.

```
// Create the ATTAdView object
ATTAdView attAdView = new ATTAdView(this, appKey, secret, udid, category);

// Set keywords to filter the ad content.
// This only returns ads for music and cars.
attAdView.setKeywords("music,cars");

// Set the reload interval, in seconds, at which the ad is refreshed. The default is
120 seconds (two minutes).
attAdView.setAdReloadPeriod(60);

// Add the ATTAdView object to your layout.
adFrameLayout.addView(attAdView);

// Set the layout properties for the ATTAdView object
attAdView.setLayoutParams(new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.FILL_PARENT, 100));

// Get the first ad and refresh the ad every 60 seconds.
attAdView.initOrRefresh();
```

The Reload timer, whose interval is set by the **setAdReloadPeriod** method and is initiated by the **initOrRefresh** method, works as a parallel thread to the activity of the **AttAdView** object. The Reload timer only works on foreground activity. To support this functionality, the **ATTAdView** object overrides the **onAttachedToWindow** and **onDetachedFromWindow** methods in the parent **android.webkit.WebView** class. This refresh model works for almost all cases. However, in some of the cases for some platforms or devices, the **onAttachedToWindow** and **onDetachedFromWindow** call back methods might not be reached. To ensure that the refresh timer works properly, override the **onPause** and **onResume** call back methods in the **Activity**, as shown in the following code example.

```
// Ensure that Reload timer is stopped
@Override
protected void onPause() {
   super.onPause();

   if(null != attAdView){
      attAdView.stopRefresh();
```

```
   }
}

// Ensure that Reload timer is running
@Override
protected void onResume() {
   super.onResume();

   if(null != attAdView) {
      attAdView.startRefresh();
   }
}
```

1.4.3 Implementing Success and Failure Callback Methods

To handle the success or failure when your app requests an advertisement, override the **onSuccess** and **onFailure** methods of the **ATTAdViewListener** class, as shown in the following code example.

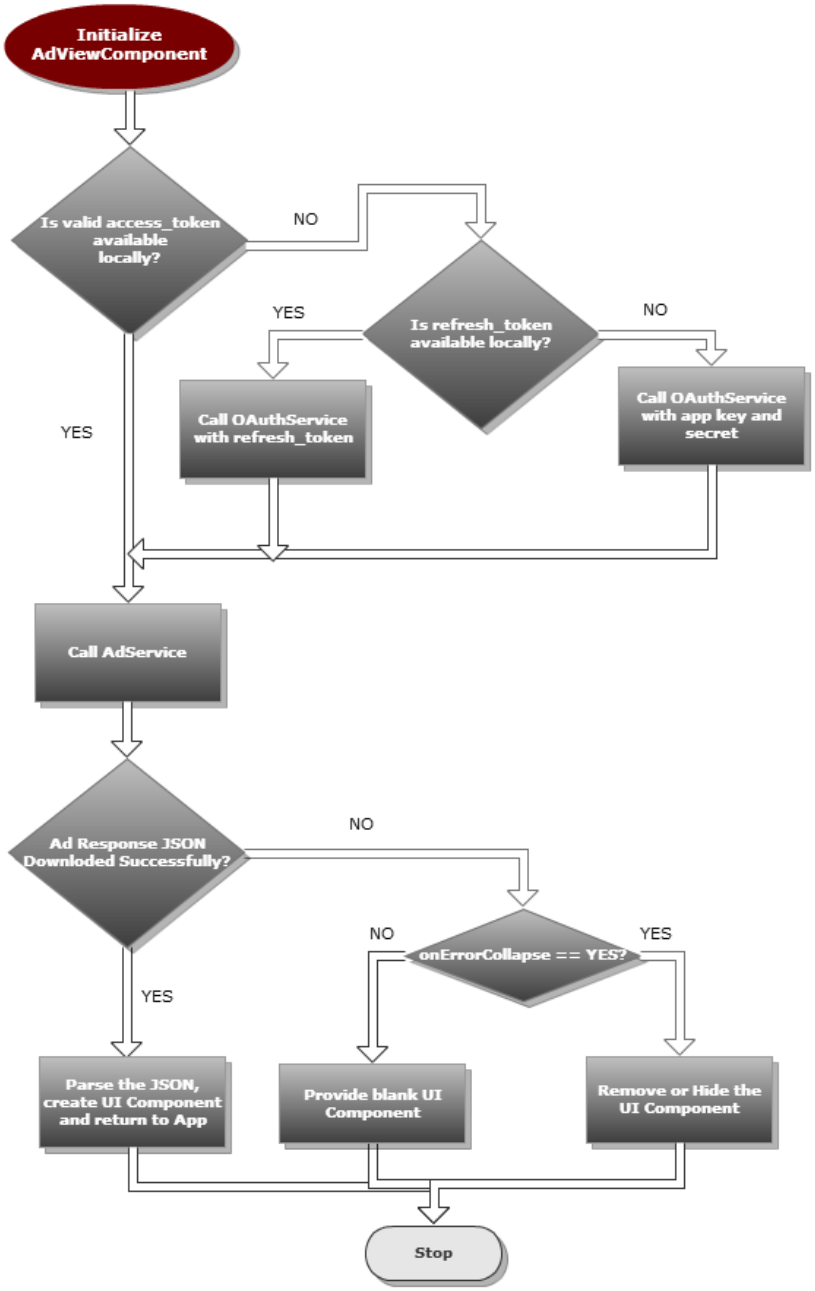
```
adView.setAdViewListener(new ATTAdViewListener() {
   public void onSuccess(String adViewResponse) {
      Log.d("attAdView", "onSuccess() response:" + adViewResponse);
   }

   public void onError(ATTAdViewError error) {
      StringBuffer res = new StringBuffer();
      res.append(error.getType());
      res.append(": ");
      res.append(error.getMessage());
      Exception e = error.getException();

      if(null != e) {
         res.append("\n Exception :\n ");
        res.append(e);
      }

      Log.e(TAG, "onError(): " + res.toString());
   }
});
```

1.4.4 Advertising API Flow



1.5 API Reference

This section provides reference information on the **ATTAdView** and **ATTAdViewListener** classes.

1.5.1 ATTAdView Methods

The following table describes the methods in the ATTAdView class.

Method	Return Type	Required?	Description
ATTAdView(Context context, String appKey, String secret, String udid, String category)	ATTAdView	Yes	Creates and initializes the advertising component using the following parameters. 1. A reference to the context of the Activity. 2. The APP key of the app. 3. The secret key of the app. 4. The unique ID of the publisher. 5. The preferred advertising category.
initOrRefresh()	void	Yes	Initializes or refreshes the advertisement.
setAdViewListener(ATTAdViewListener adViewListener)	void	No	Registers an **AdViewListener** delegate allowing the application to monitor success or failure when an advertisement is loaded.
setAdReloadPeriod(int adReloadPeriod)	void	No	Sets the interval, in seconds, between when the Reload timer refreshes the advertisement.
stopRefresh()	void	Yes	Stops the Reload timer, which stops

Method	Return Type	Required?	Description
			refreshing the advertisement.
startRefresh()	void	Yes	Starts the Reload timer. The Reload timer refreshes the advertisement.

1.5.2 ATTAdView Properties

The following table describes the properties in the ATTAdView method.

Property Name	Default Value	Type	Required?	Description
context		String	Yes	A reference to the context of the Activity.
appKey		String	Yes	The application key of the publisher.
secret		String	Yes	The secret key of the publisher.
udid		String	Yes	Specifies the UDID that is provided by the developer. Must be at least 30 characters in length or an error is returned.
category		String	Yes	Specifies the category of the app. The defined values for this parameter are: • auto • business • chat • communication • community • entertainment

Property Name	Default Value	Type	Required?	Description
				• finance • games • health • local • maps • medical • movies • music • news • personals • photos • shopping • social • sports • technology • tools • travel • tv • video • weather • other
gender		String	No	Specifies the gender of the audience demographic for the app. The defined values for this parameter are: • M • F
zipCode		Integer	No	Specifies the USA zip code of the app user.
areaCode		Integer	No	Specifies the USA area code of the app user

Property Name	Default Value	Type	Required?	Description
city		String	No	Specifies the the USA city and state of the user.
country		String	No	Specifies the three-letter, ISO-3166 country code of the user.
longitude		double	No	Specifies the current longitude, in degrees, of the geographical position for the mobile device.
latitude		double	No	Specifies the current latitude, in degrees, of the geographical position for the mobile device.
maxHeight		Integer	No	Specifies the maximum height, in pixels, of the advertisement banner. The height of the content can be less than or equal to, but not greater than, this value.
maxWidth		Integer	No	Specifies the maximum width, in pixels, of the advertisement banner. The width of the content may be less than or equal to, but not greater than, this value.
minHeight		Integer	No	Specifies the minimum height, in pixels, of the advertisement banner. The height of the content may be greater than or equal to, but not less than, this value.
minWidth		Integer	No	Specifies the minimum width, in pixels,

Property Name	Default Value	Type	Required?	Description
				of the advertisement banner. The width of the content may be greater than or equal to, but not less than this value.
timeout	1000	Integer	No	Specifies the timeout, in milliseconds, of this method. This is the maximum time the user is willing to wait for a response. The maximum value is 3000ms (three seconds) and the default value is 1000ms (one second).
ageGroup		String	No	The age group of the demographic audience of the app. The defined values for this parameter are: • 1-13 • 14-25 • 26-35 • 36-55 • 55-100
over18		Integer	No	Specifies whether to display adult ads. The defined values for this parameter are: 1. Do not display adult ads. 2. Do not display adult ads. 3. Show only adult ads. 4. Show all ads.
keywords		String	No	Specifies the keywords that are used to filter the ads. The values are not case-sensitive and multiple values must be separated by commas. For example, to

Property Name	Default Value	Type	Required?	Description
				filter for ads about music, tv, or games, use "music,tv,games".
isSizeRequired	FALSE	Boolean	No	Indicates whether the image size is returned in the response.
premium	0	Integer	No	Specifies whether to show premium ads. The defined values for this parameter are: 1. Do not show premium ads. 2. Show only premium ads. 3. Show all ads.

1.5.3 ATTAdViewListener Callbacks

By default, the response to an advertisement request is handled by the listener methods of the **ATTAdViewListener** class. You must implement these methods to either handle the returned advertisement or handle an error. The following table describes these methods.

Listener Callback Method	Return	Required?	Description
onSuccess(String adViewResponse)	void	No	This method is invoked when the request for an advertisement is successful. This method gets a reference to the raw JSON response received from AT&T Advertising Service.
onError(ATTAdViewError error)	void	No	This method is invoked when the request for an advertisement is not successful. This method get a reference to the **ATTAdViewError** object, which

Listener Callback Method	Return	Required?	Description
			has the error information.

1.5.4 ATTAdView JSON Response

The following table describes the parameters returned in the JSON response to an **ATTAdView** method.

Parameter Name	Type	Required?	Description
AdsResponse	Object	Yes	Container for the advertisements.
Ads	Object	Yes	Container for the advertisements.
ClickUrl	String	Yes	Specifies the web site to which the user is sent if they click the ad on their device. For SMS advertisements, the URL is shortened to between 35 and 40 characters.
Content	String	No	Contains the content of the ad from the third party.
Text	String	No	Contains any textual representation of the ad.
TrackUrl	String	No	Contains the pixel tracking URL.
Type	String	Yes	Specifies the type of ad.

A JSON response from the **ATTAdView** method might look something like the following.

```
{
   "AdsResponse": {
```

```
      "Ads": {
         "Type": "thirdparty",
         "ClickUrl" : "http://ads.advertising.bf.sl.attcompute.com/1/redir/21707",
         "TrackUrl": "http://bos-tapreq25.jumptap.com/a30/r/bos-tapreq25/13499/L",
         "Text": "",
         "Content": "<a
ref="http://ads.advertising.bf.sl.attcompute.com/1/redir/6dea9/0/221707">
                     <img src="http://i.jumptap.com/img/13450.jpg" alt=""
width="320px" height="50px" /></a>
                     <img src="http://bos-tapreq25.jumptap.com/11468/L" alt=""
width="1px" height="1px" />"
      }
   }
}
```

1.5.5 ATTAdView Error Codes

When the **adViewListener.onError** method is invoked, the application can determine what happened by examining the properties of the **ATTAdViewError** argument.

The **adViewListener.onError** method receives errors from a variety of sources, such as Advertising service downstream errors, OAuth downstream errors, network errors, HTTP error results, and errors reported by the operating system. The different types of errors are encapsulated into an **ATTAdViewError** object, so each source of error corresponds to a value of the **errorCode** property and each distinct error from a source corresponds to the error message of the **errorMessage** property.

The following table describes the possible values of the **errorCode** property the **adViewListener.onError** method.

Error Code	Description
-10	Returned when the OAuth service is unable to make a successful response. The corresponding information for this error is returned in the **getMessage** and **getException** methods of the **ATTAdViewError** object.
-20	Returned when the Advertising API service is unable to make a successful response. The corresponding information for this error is returned in the **getMessage** and **getException** methods of the **ATTAdViewError** object.
-30	Returned when the device was unable to connect to the internet to make service calls. The corresponding information for this error is returned in the **getMessage**

Error Code	Description
	method of the **ATTAdViewError** object.
-40	Returned when any of the parameter values was either not supplied or was invalid. The corresponding information for this error is returned in the **getMessage** method of the **ATTAdViewError** object.
-50	Returned when any other error occurs, such as unable to set or get the values from the SharedPreferences of the device. The corresponding information for this error is returned in the **getMessage** method of the **ATTAdViewError** object.

2. Speech SDK

This section describes how to add speech recognition to Android applications using version 2.3.2 of the AT&T API Platform Speech SDK for AndroidTM. (Speech SDK).

Android is a software platform for mobile devices and tablets. The AT&T Speech SDK supports applications developed using the Android Software Development Kit and distributed by Google Play. The Android SDK is available for download from http://developer.android.com, which provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language. The AT&T Speech SDK supports applications that target version 2.2 and higher of the Android platform.

2.1 About the Speech SDK

The Speech SDK is packaged as a library called `ATTSpeechKit` that links to your Android application code. The library implements a client for the Speech to Text web service. It provides the following classes to perform speech recognition on the Android platform:

- **`ATTSpeechActivity`** — A high-level API that uses a child Activity to perform a speech interaction. It is straightforward to convert code that uses Android's `RecognizerIntent` API to use the `ATTSpeechActivity` API.
- **`ATTSpeechService`** — A lower-level API that gives an application more control over the speech interaction.

Both classes in the `ATTSpeechKit` library perform the following tasks:

- Capturing audio from the device microphone.
- Showing feedback to the user.
- Streaming audio to AT&T Speech to Text web service.
- Waiting for a response from the service.

When your application wants to accept speech input from the user, it configures properties on instances of the `ATTSpeechKit` classes and calls a method to start the interaction. The `ATTSpeechKit` library activates the device's microphone, accepts audio input from the user, and waits for the response from the server. When the recognition is complete, the `ATTSpeechKit` library returns the result of recognition to your application.

Though the `ATTSpeechKit` library does not include classes that interface the Text to Speech web service, the Speech SDK includes sample code that shows how to use standard Android classes to access the Text to Speech service and play the audio it returns.

2.2 Prerequisites

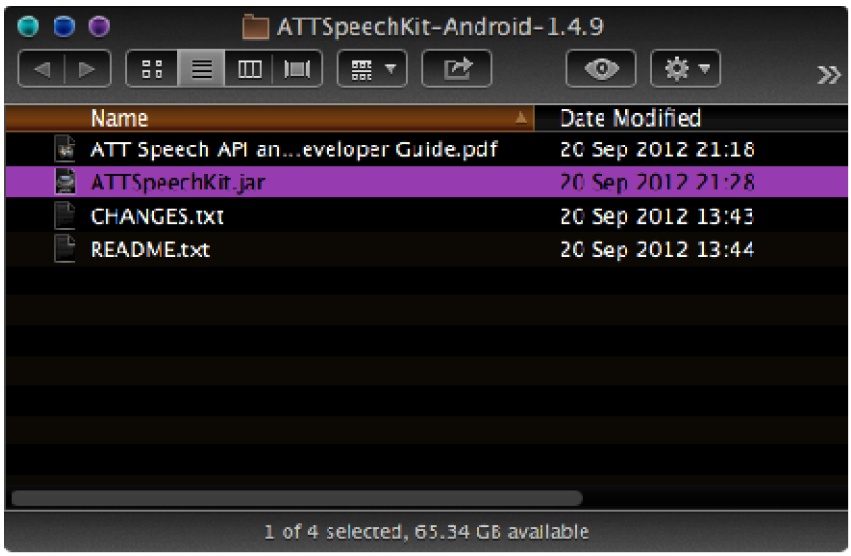
Before using the Speech SDK for Android, you should be familiar with how to program Android activities using Java. To get started with the Speech SDK for Android, prepare your development environment by completing the following steps on the Android Developer web site:

1. Install the Android SDK .
2. Install the Android Developer Tools plug-in for Eclipse (optional). Android has a close integration with Eclipse for building and running applications.

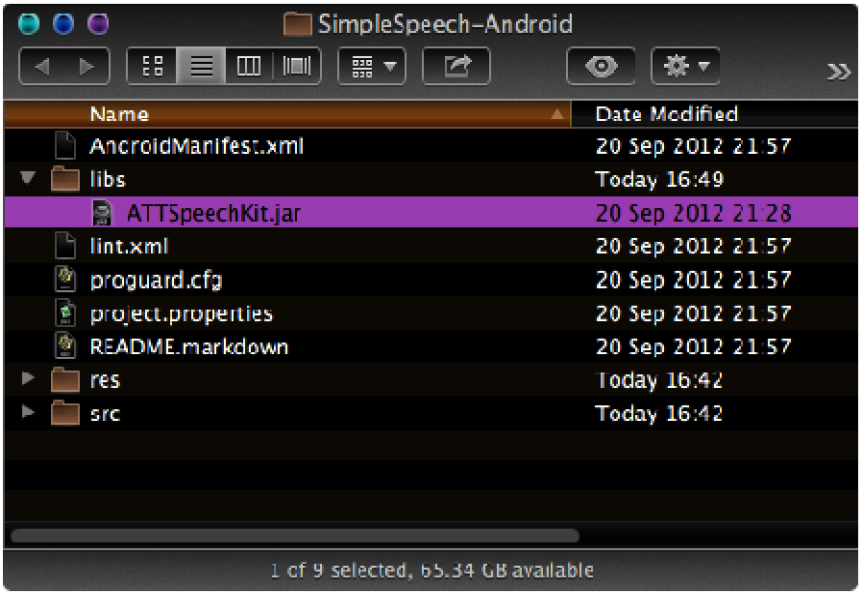
2.3 Setting up your Android Project

Complete the following steps to add the Speech SDK to your Eclipse project and configure it for speech recognition:

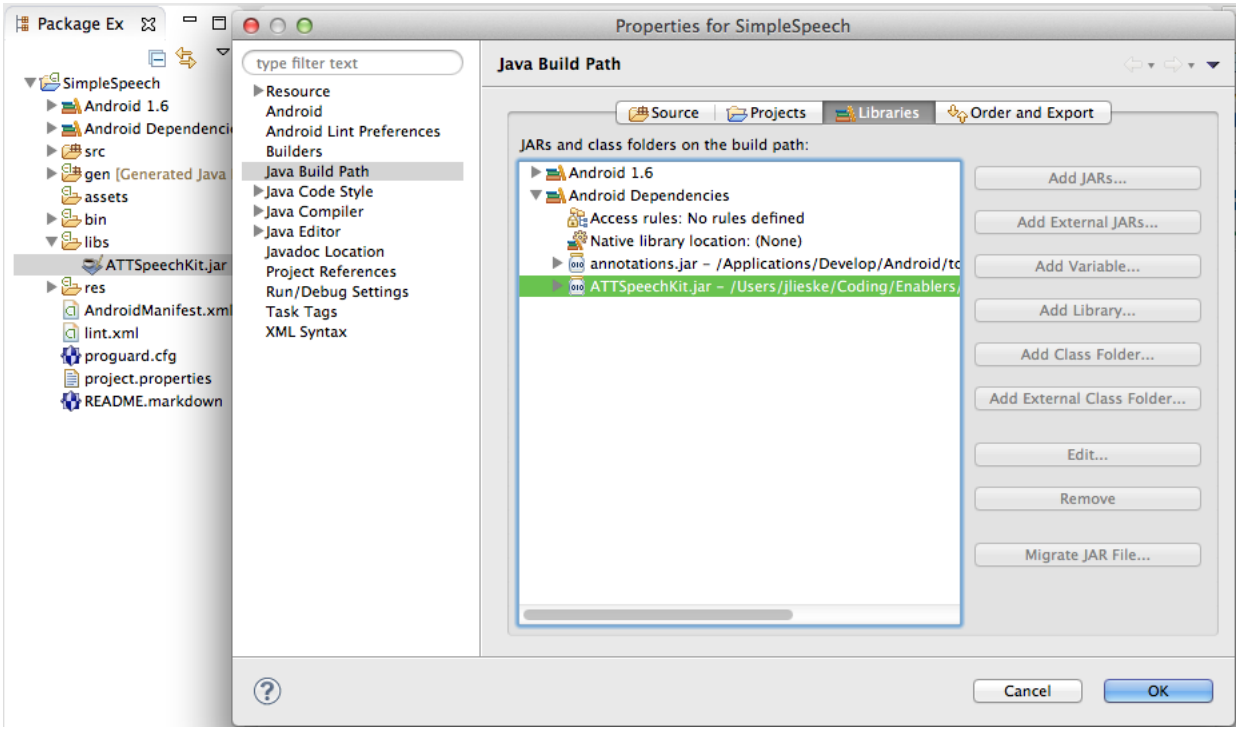
1. Unzip the Speech SDK ZIP file. The resulting folder contains the ATTSpeechKit.jar file, which is the code library that you will link to your application.



2. Add a new folder to your Android project directory called `libs` (if you do not already have one). Copy the `ATTSpeechKit.jar` file into the `libs` folder.



3. Make sure the `ATTSpeechKit.jar` file is on your project's build path. In Eclipse, right-click on the `libs/ATTSpeechKit.jar` file to bring up a context menu. Choose the menu item **Build Path > Configure Build Path**. Look under "Android Dependencies" in the **Libraries** tab. `ATTSpeechKit.jar` should appear in that list. If not, choose **Add JARs** and add `ATTSpeechKit.jar`.



4. Configure your project's Android manifest to enable microphone and network access. In the `AndroidManifest.xml` file, add the following `uses-permission` items before the `<application>` tag (or at the same place where the other `uses-permission` items are located):

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

5. If your application uses the `ATTSpeechActivity` API, configure your project's Android manifest to refer to the `ATTSpeechActivity` class. Add the following activity between the `<application ...>` and `</application>` tags In your `AndroidManifest.xml` file:

```
<activity android:name="com.att.android.speech.ATTSpeechActivity"
android:theme="@android:style/Theme.Translucent.NoTitleBar"
android:configChanges="mcc|mnc|locale|touchscreen|keyboard|keyboardHidden|navig
ation|orientation|fontScale" />
```

2.4 Speech Recognition Tasks on Android

Android applications are composed of activities, which are objects that derive from the `android.activity.Activity` class. In a typical application, each screen of the application

consists of a distinct activity object. The Speech SDK provides the `ATTSpeechActivity` class to let developers leverage this standard Android design pattern. `ATTSpeechActivity` has a programming model similar to Android's built-in `RecognizerIntent` interface.

If an application needs more control over the speech interaction, then it can use the `ATTSpeechService` API in the Speech SDK. This API lets the developer supply listener objects to the Speech SDK, which calls the listeners at various stages of the speech interaction. For example, an application that wants to display a custom audio feedback UI can install an `ATTSpeechAudioLevelListener` to get audio level values while the user speaks.

To add speech recognition to your application using either `ATTSpeechActivity` or `ATTSpeechService`, you need to write code that performs the following functions:

- (`ATTSpeechActivity` only) Configure the Android manifest to refer to the `ATTSpeechActivity` class.
- Configure the Speech SDK runtime properties.
- Acquire an OAuth access token.
- Start speech interaction at the appropriate time (for example, in a button press handler).
- (Optional) Customize UI during speech interaction.
- (Optional) Customize speech endpointing.
- Handle speech recognition results and errors.

2.4.1 Configure the Android manifest

Every Android application's `AndroidManifest.xml` file has a list of Activity classes contained in the application. When your application links the AT&T Speech SDK and calls `ATTSpeechActivity`, Android will use the information in the manifest to run the child activity. To use `ATTSpeechActivity`, ensure that the following appears in the Android manifest:

```
<activity android:name="com.att.android.speech.ATTSpeechActivity"
 android:theme="@android:style/Theme.Translucent.NoTitleBar"
android:configChanges="mcc|mnc|locale|touchscreen|keyboard|keyboardHidden|navigation|o
rientation|fontScale" />
```

2.4.2 Configure Speech SDK runtime properties

The AT&T Speech SDK needs to be configured before an application uses it to call the Speech API. The mechanism is different between `ATTSpeechActivity` and `ATTSpeechService`, but the configuration data is largely the same. Refer to Table 2 Android Request Properties for a full list of the configuration properties.

To perform speech recognition with the `ATTSpeechActivity` API, an application uses the standard Android techniques to start a child activity. Your activity creates an `Intent` object and sets properties (called "extras") on it to configure a speech recognition request.

When using `ATTSpeechService`, instead of using an Intent object, your application will call methods on the `ATTSpeechService` singleton object. Obtain a reference to the

ATTSpeechService singleton object by calling the getSpeechService() method. Note that the singleton is per-Activity; don't save the singleton in a global variable across multiple Activities.

A mandatory configuration property to set is the URL of the Speech to Text service that your application uses. Set this via the ATTSpeechActivity.EXTRA_RECOGNITION_URL Intent extra or the method ATTSpeechService.setRecognitionURL(). Get this URL from your application's on-boarding information on the AT&T Developer web site.

A required property for the Speech API is the name of the speech context that your application wants to use for recognition. Set this via the ATTSpeechActivity.EXTRA_SPEECH_CONTEXT Intent extra or the method ATTSpeechService.setSpeechContext(). Refer to section TO DO more information on the available speech contexts.

In addition to setting the Speech API configuration properties, your application must also acquire an OAuth token as described below. It will add the token string to the configuration via theATTSpeechActivity.EXTRA_BEARER_AUTH_TOKEN Intent extra or by calling ATTSpeechService.setBearerAuthToken().

Programs using ATTSpeechService must also set the listener objects that the Speech SDK will call during a speech interaction. It's required to set the result and error listeners via setSpeechResultListener() and setSpeechErrorListener(). The other listener objects are optional.

2.4.3 Acquire OAuth access token

Requests to AT&T Speech API must include a valid OAuth access token. To avoid authentication failures during a speech interaction, your application should acquire the access token before the user tries to speak. Your application obtains an access token by making an OAuth request to the Speech API. The Speech SDK includes sample code that demonstrates how to obtain an OAuth access token. Your application can re-use the sample code or implement its own technique for obtaining the token.

When your code receives the access token, it should save the token in an instance variable for use in speech interactions.

Since the network I/O required to fetch the OAuth token can take a fraction of a second, it's important that the request be done asynchronously or on a background thread. That prevents the UI of the application from locking up as it waits for a response over the network. Your application should also disable its speech UI briefly until it receives an OAuth access token.

It is best to request an access token every time your app comes to the foreground. This ensures that the subsequent speech request will have a valid token. If the users of your application will keep it in the foreground for longer than an hour, you need to add code to request a token at hourly intervals.

2.4.4 Start speech interaction

Once your application has configured the Speech SDK and obtained an OAuth access token, it can perform speech requests. In a typical application, the user can trigger speech input by pressing a button. The Speech SDK will turn on the microphone and begin streaming data in compressed AMR format to the Speech API server. The application code handling the button press makes different calls depending on whether it uses `ATTSpeechActivity` or `ATTSpeechService`.

2.4.4.1 Starting speech using `ATTSpeechActivity`

An application using `ATTSpeechActivity` will use Android system calls to start a child activity: it passes an `Intent` object to the `startActivityForResult()` method. Android will call your activity's `onActivityResult()` method when speech processing is complete or if there is an error. For more information on the `onActivityResult()` method, refer to Example 3 `onActivityResult` Method.

The following example illustrates the sequence of calls to construct an Intent and initiate a speech interaction. Your application could make these calls, for example, in an action handler that is called when a user taps the **Record** button in your application's UI.

Example 1 Starting ATTSpeechActivity

```
public void myStartSpeechRecognition()
{
    Intent request = new Intent(this, ATTSpeechActivity.class);
    request.putExtra(ATTSpeechActivity.EXTRA_RECOGNITION_URL, MY_SPEECH_URL);
    request.putExtra(ATTSpeechActivity.EXTRA_SPEECH_CONTEXT, "Generic");
    request.putExtra(ATTSpeechActivity.EXTRA_BEARER_AUTH_TOKEN, myOAuthToken);
    startActivityForResult(request, MY_SPEECH_INTERACTION);
}
```

2.4.4.2 Starting speech using `ATTSpeechService`

An application using `ATTSpeechService` calls `startListening()` on the singleton object to begin a speech interaction with microphone input. Your callback interface instance will be called at various stages through the speech interaction until it receives the recognition result or an error.

The following example illustrates the sequence of calls to configure the `ATTSpeechService` singleton and initiate a speech interaction. Your application could make these calls, for example, in an action handler that is called when a user taps the **Record** button in your application's UI.

Example 2 Starting ATTSpeechService

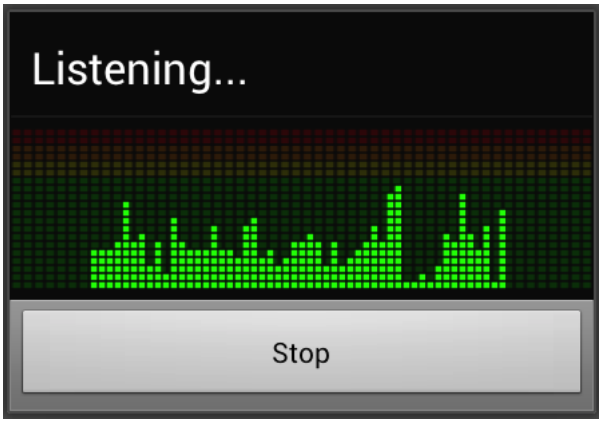
```
public void myStartSpeechRecognition()
{
    ATTSpeechService speechSvc = ATTSpeechService.getSpeechService(this);
```

```
    speechSvc.setSpeechResultListener(mySpeechResultListener);
    speechSvc.setSpeechErrorListener(mySpeechErrorListener);
    speechSvc.setRecognitionURL(MY_SPEECH_URL);
    speechSvc.setSpeechContext("Generic");
    speechSvc.setBearerAuthToken(myOAuthToken);
```

2.4.5 Customize UI

An application should display a user interface that lets the user know that microphone input is being captured during a speech interaction. The Speech SDK can display a standard alert window while listening to the microphone. When listening is complete, the alert window changes to a progress display while the Speech SDK is waiting for a recognition result from the server. Speech SDK will display this UI by default; your application can disable this UI by setting the `ATTSpeechActivity.EXTRA_SHOW_UI` Intent extra to false or calling `ATTSpeechService.setShowUI(false)`. When displaying the standard UI, your application can customize the labels in the alert through several properties. See Table 2 for more information on the UI properties.

Figure 1 Speech SDK standard UI on Android



An application using `ATTSpeechService` has the option of displaying its own UI during a speech interaction. It can register listener objects that receive updates during the recording and processing. Implement the interface `ATTSpeechAudioLevelListener` to receive an average input volume level roughly every 1/10 second. `ATTSpeechStateListener` receives notifications when microphone input starts and ends. See section 2.5.3 for more information on the listener callbacks.

The Speech SDK automatically ends recording when the user stops talking. Additionally, your application can manually stop capturing audio by calling the `ATTSpeechService.stopListening()` method. The Speech SDK will wait for the Speech API services to perform recognition on the speech that was captured up to that point.

Your app can call the `ATTSpeechService.cancel()` method to abort the speech interaction.

Note that your `ATTSpeechErrorListener` will not get a callback from the service if `cancel` is called.

2.4.6 Speech Endpointing

Once the user has begun a speech interaction, the Speech SDK needs to determine when to stop listening, a process called *endpointing*. The Speech SDK supports the following techniques for endpointing on Android:

- **Silence Detection** — When the user is quiet after doing some speaking, the Speech SDK can treat the silence as a signal to stop listening. An application can tune this value by setting the `ATTSpeechActivity.EXTRA_ENDING_SILENCE` Intent extra or calling `ATTSpeechService.setEndingSilence()`.
- **Timeout Expiration** — An application using the Speech SDK can set the maximum duration of speech input through the `ATTSpeechActivity.EXTRA_MAX_RECORDING_TIME` Intent extra or the `ATTSpeechService.setMaxRecordingTime()` property.
- **Manual Endpointing** — The user may press a button in the application's user interface to signal that they are done talking. The standard UI presented by the Speech SDK displays a **Stop** button for the user to perform manual endpointing. An application that presents its own UI while listening can call the `ATTSpeechService.stopListening()` method to do the same thing.

The Speech SDK handles all three techniques during speech interactions. Timeouts and manual endpointing serve to override silence detection. For example, if an application sets the `maxRecordingTime` property to 5 seconds and the user is still talking at the 5 second mark, then the Speech SDK will stop listening, even though the user was not silent.

In addition to using silence to detect that the user has finished talking, the Speech SDK uses silence to determine whether the user spoke at all during an interaction. If the user has not spoken after `maxInitialSilence` seconds, then the Speech SDK can cancel the speech interaction. An application customizes this behavior through the `cancelWhenSilent` property.

2.4.7 Handle speech recognition results and errors

When speech recognition is complete, the Speech SDK calls back to your application. The exact mechanism is different for `ATTSpeechActivity` and `ATTSpeechService`, but similar information is provided by both classes. Your callback methods can obtain an array of transcription hypotheses or the raw JSON response from the speech service. The full list of response properties is in Table 4.

If an error occurs during a speech interaction, Speech SDK will call back to your application with the error information. Possible errors include the user canceling, the user not speaking in time, problems with the network connection, and errors reported by the server. Refer to section 2.5.5 for more information on interpreting error callbacks.

2.4.7.1 Handling ATTSpeechActivity results

When your app uses ATTSpeechActivity for a speech interaction, Speech SDK calls your activity's onActivityResult(requestCode, resultCode, resultIntent) method when processing is complete or an error occurs. The interpretations of the resultCode argument are as follows:

- **RESULT_OK** — Indicates that the processing was successful. The resultIntent argument contains the "extras" properties with the recognition response. An array of transcription hypothesis strings is available in the EXTRA_RESULT_TEXT_LIST extra property, and the raw binary data is in the EXTRA_RESULT_RAW_BYTES property. The full list of response properties is in Table 4.
- **RESULT_CANCELED** — Indicates that the user canceled the interaction.
- **Other values** — Indicate that an error occurred. The resultCode value describes the type of error, and the resultIntent argument includes "extras" properties with diagnostic information. For a description of the error codes and properties, refer to section 2.5.5.

The following example shows a minimal implementation of the onActivityResult() method.

Example 3 onActivityResult Method

```
public void onActivityResult(int requestCode, int resultCode, Intent resultIntent)
{
    if (requestCode == MY_SPEECH_INTERACTION) {
        if (resultCode == RESULT_OK) {
            // Retrieve text recognized from speech
            ArrayList<String> nbest =
               resultIntent.getStringArrayListExtra(
                   ATTSpeechActivity.EXTRA_RESULT_TEXT_LIST);
            if ((nbest != null) && (nbest.size() > 0)) {
                String text = nbest.get(0);
                Log.v(MY_ACTIVITY, "Recognition result: "+text);
            }
            else
                Log.v(MY_ACTIVITY, "Speech was silent or not recognized.");
        }
        else if (resultCode == RESULT_CANCELED)
            Log.v(MY_ACTIVITY, "User canceled.");
        else {
            String error = resultIntent.getStringExtra(
                ATTSpeechActivity.EXTRA_RESULT_ERROR_MESSAGE);
            Log.v(MY_ACTIVITY, "Recognition failed with error: "+error);
        }
    }
}
```

2.4.7.2 Handling ATTSpeechService results

When your app uses ATTSpeechService for a speech interaction, the Speech SDK calls the ATTSpeechServiceResultListener interface when processing is complete. It passes an ATTSpeechResult argument that exposes properties for the application to get the recognition response data. An array of transcription hypothesis strings is available in the getTextStrings() method. The parsed JSON data from the service is in the getJSON() method, and the raw binary data is in the getRawData() method. The full list of response properties is in Table 4.

The following example shows a minimal implementation of ATTSpeechServiceResultListener that examines the array of transcription hypotheses.

Example 4 Successful Transaction Listener

```
class ResultListener implements ATTSpeechResultListener {
    public void onResult(ATTSpeechResult result)
    {
        List<String> nbest = result.getTextStrings();
        if (nbest != null && nbest.size() > 0)
            Log.v("MyApp", "Recognition result: "+nbest.get(0));
        else
            Log.v("MyApp", "Speech was silent or not recognized.");
    }
}
```

If an error occurs doing the speech recognition process, the Speech SDK calls the ATTSpeechErrorListener callback interface. Possible errors include the user canceling, the user not speaking in time, problems with the network connection, and errors reported by the server. The ATTSpeechError argument exposes properties for the application to get the information about the error. Refer to section 2.5.5 for more information on interpreting error callbacks.

The following example shows a minimal implementation of ATTSpeechErrorListener that checks the error status.

Example 5 Failed Transaction Listener

```
class ErrorListener implements ATTSpeechErrorListener {
    public void onError(ATTSpeechError error)
    {
        ErrorType resultCode = error.getType();
        if (resultCode == ErrorType.USER_CANCELED)
            Log.v("MyApp", "User canceled.");
        else
            Log.v("MyApp", "Recognition failed with error "+resultCode+":
"+error.getMessage());
```

```
    }
}
```

2.5 Speech SDK Reference for Android

This section is a reference for the `ATTSpeechKit` library on Android. It describes the classes, methods, properties, callback listeners, error codes, and state transitions in both `ATTSpeechActivity` and `ATTSpeechService`.

- Android `ATTSpeechService` Methods
- Android Request Properties
- Android `ATTSpeechService` Callbacks
- Android Result Properties
- Android Error Properties

2.5.1 Android `ATTSpeechService` Methods

The following table describes the methods that the `ATTSpeechService` class implements.

NOTE: The `ATTSpeechActivity` API does not support these methods. It supports the standard Android `startActivityForResult()` method.

Table 1 Android `ATTSpeechService` Methods

Method	Required / Optional	Description
`getSpeechService``(Activity activity)`	Required	This static method the singleton speech service object for the activity.
`startListening``()`	Required	Starts a speech interaction using the microphone. Before calling this method, set the request properties on the `ATTSpeechService` object to configure the interaction. After the method is called, the Speech SDK will invoke methods on the delegate object to report interaction status, errors, and the recognition result.

Method	Required / Optional	Description
`startWithAudioData``(byte[] audioData)`	Required	Sends audio data to the Speech to Text service for processing instead of using the microphone. Before calling this method, set the request properties on the `ATTSpeechService` object to configure the interaction. The method does no compression of the data, so the file should already be in a format supported by the Speech to Text service. After this method is called, the `ATTSpeechService` instance will invoke methods on the callback objects to report interaction status, errors, and the recognition result.
`stopListening``()`	Optional	Manually ends speech input. This method is optional because the `ATTSpeechKit` library automatically ends speech input when the user stops talking or a timeout expires. After this method is called, the `ATTSpeechService` object will wait for the server to process the submitted speech and then report the result or error to the listener objects.
`cancel``()`	Optional	Cancels a speech interaction. After this method is called, the `ATTSpeechService` object will not attempt to perform speech recognition on the submitted data, and it will not make further listener callbacks related to the canceled interaction.

2.5.2 Android Request Properties

Table 2 describes the Android Speech SDK properties that control a speech interaction.

For developers using the `ATTSpeechActivity` class, the table lists the `Intent` extras that control speech recognition in an `ATTSpeechActivity` request. Your application sets these extra properties on the `Intent` that it passes to the `startActivityForResult()` method. The property names are all static constants on the `ATTSpeechActivity` class.

For developers using the `ATTSpeechService` class, the table describes the methods on that

class that control speech requests.

Table 2 Android Request Properties

ATTSpeechActivity Extra Name and ATTSpeechService Method	**Default Value**	**Type**	**Required /Optional**	**Description**
`EXTRA_RECOGNITION_URL` `setRecognitionURL()`		`String`	Required	The URL of the Speech to Text service. Refer to section **Error! Reference source not found. Error! Reference source not found.** for more information.
`EXTRA_AUDIO_DATA`		`byte[]`	Optional	The audio data to submit to the Speech to Text service for recognition. When this property is omitted, the `ATTSpeechActivity` gets input from the microphone. (`ATTSpeechActivity` only. When using `ATTSpeechService`, use `startWithAudioData()`.)
`EXTRA_CONTENT_TYPE` `setContentType()`	`"audio/amr"`	`String`	Optional	MIME type of the audio data for the server. The value must be a MIME type such as "`audio/amr`". It is used in the `Content-Type` HTTP header. This property is needed only when uploading audio data.
`EXTRA_SPEECH_CONTEXT` `setSpeechContext()`		`String`	Required	The name of the Speech to Text context to use for recognition. It is used in the `X-SpeechContext` HTTP header.
`EXTRA_XARGS`		`String`	Optional	A set of extra arguments for the Speech to Text service. The parameter is a string of comma-separated *name=value* pairs, with the values percent-encoded. It is used in the `X-Arg` HTTP header. (`ATTSpeechActivity` only.)

ATTSpeechActivity Extra Name and ATTSpeechService Method	Default Value	Type	Required /Optional	Description
setXArgs()		Map <String, String>	Optional	A set of extra arguments for the Speech to Text service. The parameter is a mapping of name-value pairs, with the values unencoded. It is used in the X-Arg HTTP header. (ATTSpeechService only.)
EXTRA_SENDS_DEFAULT_XARGS sendsDefaultXArgs()	true	boolean	Optional	Leaving this property true enables the Speech SDK to automatically generate X-Arg values based on the device and application properties: DeviceType, DeviceOS, DeviceTime, ClientSDK, CientApp, and ClientVersion. When the property is set to false, Speech SDK will not add the X-Arg values.
EXTRA_BEARER_AUTH_TOKEN setBearerAuthToken()		String	Required	The OAuth access token that validates speech requests from this application. It is used in the Authentication: Bearer HTTP header.
setRequestHeaders()		Map <String, String>	Optional	A collection of HTTP headers to add to the request. (ATTSpeechService only.)
EXTRA_REQUEST_HEADER_NAMES		String[]	Optional	An array of HTTP header names to add to the request. The EXTRA_REQUEST_HEADER_VALUES property must also be set when this property is set. (ATTSpeechActivity only.)
EXTRA_REQUEST_HEADER_VALUES		String[]	Optional	An array of HTTP header values to add to the request. It is a parallel array to the EXTRA_REQUEST_HEADER_NAMES property. (ATTSpeechActivity only.)

ATTSpeechActivity Extra Name and ATTSpeechService Method	Default Value	Type	Required /Optional	Description
`EXTRA_SHOW_UI` `setShowUI()`	`true`	`boolean`	Optional	Controls the display of a progress meter and a button for canceling the interaction. Set this property to `false` to hide the UI.
`EXTRA_RECORDING_PROMPT` `setRecordingPrompt()`	`"Listening"`	`String`	Optional	The text that the Speech SDK displays while capturing audio data by the microphone. Use this property only when the `showUI` property is true.
`EXTRA_RECORDING_STOP_BUTTON` `setRecordingStopButton()`	`"Stop"`	`String`	Optional	The button label that the Speech SDK displays while capturing audio data by the microphone. The user can press the button to manually endpoint audio input. Use this property only when the `showUI` property is true.
`EXTRA_PROCESSING_PROMPT` `setProcessingPrompt()`	`"Processing"`	`String`	Optional	The text that the Speech SDK displays while waiting for the server to return a recognition result. Use this property only when the `showUI` property is true.
`EXTRA_PROCESSING_CANCEL_BUTTON` `setProcessingCancelButton()`	`"Cancel"`	`String`	Optional	The button label that the Speech SDK displays while waiting for the server to return a recognition result. The user can press the button to cancel the speech interaction. Use this property only when the `showUI` property is true.
`EXTRA_CANCEL_WHEN_SILENT` `setCancelWhenSilent()`	`true`	`boolean`	Optional	Specifies how Speech SDK handles silent audio input. When the value is false, Speech SDK will send silent audio to the server for processing. When true, Speech SDK will automatically cancel the processing when it detects silent audio.

ATTSpeechActivity Extra Name and ATTSpeechService Method	Default Value	Type	Required /Optional	Description
`EXTRA_MAX_INITIAL_SILENCE` `setMaxInitialSilence()`	`1500 ms`	`int`	Optional	The maximum number of milliseconds that a user can remain silent before the Speech SDK aborts the speech interaction. To disable aborting, set the value to `Integer.MAX_VALUE`.
`EXTRA_ENDING_SILENCE` `setEndingSilence()`	`1000 ms`	`int`	Optional	The number of milliseconds of silence after the user stops talking when the Speech SDK performs endpointing. To disable endpointing, set the value to `Integer.MAX_VALUE`.
`EXTRA_MIN_RECORDING_TIME` `setSpeechInputMinimum Length()`	`500 ms`	`int`	Optional	The minimum number of milliseconds of audio to capture from the microphone. To disable aborting, set the value to `0`.
`EXTRA_MAX_RECORDING_TIME` `setRecordingTimeout()`	`25000 ms`	`int`	Optional	The maximum number of milliseconds that the user can speak into the microphone. When the maximum time is reached, Speech SDK automatically stops microphone input and submits the audio for processing. To disable the timeout, set the value to `Integer.MAX_VALUE`.
`EXTRA_CONNECTION_TIMEOUT` `setConnectionTimeout()`	`10000 ms`	`int`	Optional	The maximum number of milliseconds that the Speech SDK waits for a successful connection to the server when initiating a recognition request. To disable the timeout, set the value to `Integer.MAX_VALUE`.
`EXTRA_SERVER_RESPONSE_TIMEOUT` `setServerResponseTimeout()`	`30000 ms`	`long`	Optional	The maximum number of milliseconds that the Speech SDK waits for the server to complete a recognition response. To disable the timeout, set the value to `Integer.MAX_VALUE`.

2.5.3 Android ATTSpeechService Callbacks

While the ATTSpeechService object is performing a speech recognition interaction, it calls methods on the listener objects set by the application. The application is required to set the ATTSpeechResultListener and ATTSpeechErrorListener callbacks that communicate the recognition result to the application. The remaining listeners are optional, giving the application a chance to customize the behavior of the speech interaction.

The following table describes the Speech SDK callback interfaces. (ATTSpeechActivity does not support these interfaces. It supports the standard Android onActivityResult() method.)

Table 3 Android ATTSpeechService Callback Interfaces

Listener Interface	Listener Method	Required/ Optional	Description
ATTSpeechResult Listener	onResult()	Required	The Speech SDK calls this interface when it returns a recognition result. The ATTSpeechResult argument has properties that contain the response data and recognized text. For more information on how to interpret the response data, refer to Section 2.5.4.
ATTSpeechError Listener	onError()	Required	The Speech SDK calls this interface when speech recognition fails. The reasons for failure can include the user canceling, network errors, or server errors. For more information on the ATTSpeechError argument, refer to Section 2.5.5 Android Error Properties.
ATTSpeechState Listener	onStateChanged()	Optional	The Speech SDK calls this interface when it transitions among states in a recording interaction, for example, from capturing to processing.
ATTSpeechAudio LevelListener	onAudioLevel(int)	Optional	The Speech SDK calls this interface repeatedly as it captures audio. An application can use the audio level data to update its UI.

2.5.4 Android Result Properties

When a speech interaction completes successfully, the Speech SDK returns the data described

in this section.

For applications that use the `ATTSpeechActivity` API, the `onActivityResult()` method is called with a result code of `RESULT_OK` to indicate a successful interaction. The `ATTSpeechActivity` API attaches "extras" properties to the `Intent` object that are passed to the `onActivityResult()` method. Table 4 Android Result Properties describes the available extras; the property names are all static constants on the `ATTSpeechActivity` class.

Applications that use the `ATTSpeechService` API will have their `ATTSpeechResultListener` interface called on a successful interaction. The following table describes the data available on the `ATTSpeechResult` object passed to the `onResult()` method.

Table 4 Android Result Properties

`ATTSpeechActvity` Extra/ `ATTSpeechResult` method	**Type**	**Required/ Optional**	**Description**
EXTRA_RESULT_TEXT_STRINGS getTextStrings()	ArrayList <String>	Optional	Contains the recognized text as a collection of hypothesis strings. The collection can be null or zero length when the speech was silent or the service did not recognize the speech. This property is present only when the service returns the standard Speech API JSON format.
EXTRA_RESULT_RAW_DATA getRawData()	byte[]	Required	The full binary data that is returned by the server.
EXTRA_RESULT_STATUS_CODE getStatusCode()	int	Required	The HTTP status code that the speech service returned.
getResponseHeaders()	Map <String, String>	Required	The HTTP headers that the speech service returned. Header names are case-insensitive via `get()` and preserve the case returned by the service. (`ATTSpeechService` only.)

ATTSpeechActvity Extra/ ATTSpeechResult method	Type	Required/ Optional	Description
EXTRA_RESULT_HEADER_NAMES	String[]	Required	The names of the HTTP headers that the speech service returned. EXTRA_RESULT_HTTP_HEADER_VALUES contains the corresponding values. (ATTSpeechActivity only.)
EXTRA_RESULT_HEADER_VALUES	String[]	Required	The values of the HTTP headers that the speech service returned. These correspond to the header names in EXTRA_RESULT_HTTP_HEADER_NAMES. (ATTSpeechActivity only.)
getJSON()	JSONObject	Optional	The parsed JSON data that the speech service returned. This property is present only when the service returns JSON data. (ATTSpeechService only.)

2.5.5 Android Error Properties

When a speech interaction does not complete successfully, the Speech SDK calls back to your application with information about the error. Possible errors include the user canceling, the user not speaking in time, problems with the network connection, and when the server returns a non-200 HTTP response.

Table 5 Android Error Codes describes the error codes that an ATTSpeechActivity or ATTSpeechService request returns. When using ATTSpeechActivity, the result codes are passed in the resultCode parameter of the onActivityResult() method. When using ATTSpeechService, the error codes are ATTSpeechError.ErrorType enumeration constants accessed by calling ATTSpeechError.getType() on the ATTSpeechError object passed to ATTSpeechErrorListener.

Table 5 Android Error Codes

ATTSpeechError getType() Value	ATTSpeechActivity Result Code	Description
ErrorType. USER_CANCELED	RESULT_CANCELED	The user or the application canceled the speech interaction.

ATTSpeechError getType() Value	ATTSpeechActivity Result Code	Description
ErrorType. PARAMETER_ERROR	RESULT_PARAMETER_ ERROR	One of the properties set on ATTSpeechService or on the Intent passed to the ATTSpeechActivity class was invalid.
ErrorType. CAPTURE_FAILED	RESULT_AUDIO_ERROR	There was a problem capturing data from the microphone.
ErrorType. BELOW_MINIMUM_LENGTH	RESULT_AUDIO_LENGTH_ERROR	The user's audio was too short.
ErrorType.INAUDIBLE	RESULT_INAUDIBLE_ ERROR	The user was silent and cancelWhenSilent was set to true.
ErrorType. CONNECTION_ERROR	RESULT_CONNECTION_ ERROR	A problem occurred while connecting to the server.
ErrorType. RESPONSE_ERROR	RESULT_RESPONSE_ ERROR	A problem occurred while reading the response over the network.
ErrorType.SERVER_ERROR	RESULT_SERVER_ERROR	The server reported an HTTP error. See Table 6 Android Error Properties for the data returned by the server.
ErrorType.OTHER_ERROR	RESULT_CLIENT_ERROR	Some other error occurred.

Table 6 Android Error Properties shows the additional data that is returned with an error result. This additional data, known as "extras" properties, are attached to the Intent object that is passed to the onActivityResult()method when using the ATTSpeechActivity API. Note that some of the properties are present only for HTTP errors reported with the result code ATTSpeechActivity.RESULT_SERVER_ERROR. Table 6 Android Error Properties lists the available extras; the property names are all static constants on the ATTSpeechActivity class.

Applications that use the ATTSpeechService API will have their ATTSpeechErrorListener callback called on an error result. The Speech SDK will pass an ATTSpeechError object to the onError() method. Note that some of the properties are present only for HTTP errors reported with the result code ATTSpeechError.ErrorType.SERVER_ERROR; call the

ATTSpeechError.getResult()method to access the server's HTTP response data for such errors. The following table describes the data available on the ATTSpeechError and ATTSpeechResult objects passed to the onError() method.

Table 6 Android Error Properties

ATTSpeechActvity Extra/ ATTSpeechService Class and Method	**Type**	**Required/ Optional**	**Description**
EXTRA_RESULT_ERROR_ MESSAGE ATTSpeechError. getMessage()	String	Required	A string from the Speech SDK that contains diagnostic information for network or server errors. It is unsuitable for display to the user.
EXTRA_RESULT_STATUS_ CODE ATTSpeechResult. getStatusCode()	int	Optional	The HTTP status code returned by the speech server. It is included only for errors of type SERVER_ERROR.
ATTSpeechResult. getResponseHeaders()	Map <String, String>	Optional	The HTTP headers that the speech service returned. Header names are case-insensitive via get() and preserve the case returned by the service. It is included only for errors of type SERVER_ERROR. (ATTSpeechService only.)
EXTRA_RESULT_HEADER _NAMES	String[]	Optional	The names of the HTTP headers that the speech service returned. EXTRA_RESULT_HEADER_VALUES contains the corresponding values. It is included only for errors of type SERVER_ERROR. (ATTSpeechActivity only.)

ATTSpeechActvity Extra/ ATTSpeechService Class and Method	Type	Required/ Optional	Description
EXTRA_RESULT_HEADER_VALUES	String[]	Optional	The values of the HTTP headers that the speech service returned. These correspond to the header names in EXTRA_RESULT_HEADER_NAMES. It is included only for errors of type SERVER_ERROR. (ATTSpeechActivity only.)
EXTRA_RESULT_RAW_DATA ATTSpeechResult. getRawData()	byte[]	Optional	The full binary data that is returned by the. It is included only for errors of type SERVER_ERROR.

2.5.6 Android State Transitions

An application that wants to be notified about the progress of a speech interaction can provide an ATTSpeechStateListener object to ATTSpeechService. Speech SDK will call the listener with values of the ATTSpeechStateListener.SpeechState enumeration when it enters a new stage of the interaction. The following table describes the stages that occur during speech interaction.

Table 7 Android ATTSpeechStateListener Values

State	Description
SpeechState.IDLE	Nothing occurring. ATTSpeechService is in this state before beginning a speech interaction and after delegate callbacks have returned.
SpeechState.INITIALIZING	ATTSpeechService is beginning a speech interaction, but it is not ready to accept audio yet. This stage can be skipped in some situations.

State	Description
`SpeechState.RECORDING`	Audio capture is taking place.
`SpeechState.PROCESSING`	The server is processing audio data.
`SpeechState.ERROR`	`ATTSpeechService` is handling an error condition.

2.6 Android Testing

The Speech SDK supports the tools of the Android platform for running automated tests. For more information on testing the Android application, refer to the topic **Testing** at the following location:

http://developer.android.com/guide/topics/testing/index.html

Note that Android Emulator has limited support for audio input. The emulator's software codecs tend to distort the audio, which causes speech recognition to fail. For best results, use actual Android devices instead of the emulator for testing speech applications.

2.7 Android Deployment

Developers who create speech-enabled applications on Android devices must link their applications to the `ATTSpeechKit` library JAR, so that the library code is embedded in the application's application package (APK) file. There is nothing additional that must be added to distribute the application to end-users.