AT&T API Platform Advertising and Speech SDK for iOS™

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 iOSTM (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.

iOS is the software platform for the Apple iPhone, iPod touch, and iPad. This SDK supports applications that are developed with the iOS Software Development Kit and are distributed by the iOS App Store. The iOS SDK, which is available at http://developer.apple.com, provides the tools and APIs necessary to develop applications on the iOS platform using the Objective-C programming language. Xcode is available at the Apple developer web site https://developer.apple.com/xcode/index.php. This SDK supports applications that target iOS version 3.1 or higher and Xcode version 4.0 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 *ATTAdKit*. This 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, you must have the following prerequisites.

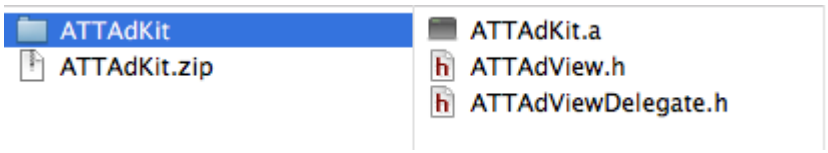
- A Mac OS X computer that is capable of running the Xcode tools. To install the iOS SDK and the Xcode development tools, follow the instructions on the Apple Developer website.
- You must be a member of Apple's iOS Developer Program to run applications on a device and to distribute applications to customers.
- You must understand how to program iOS applications using Objective-C.

1.3 Setting up your iOS Project

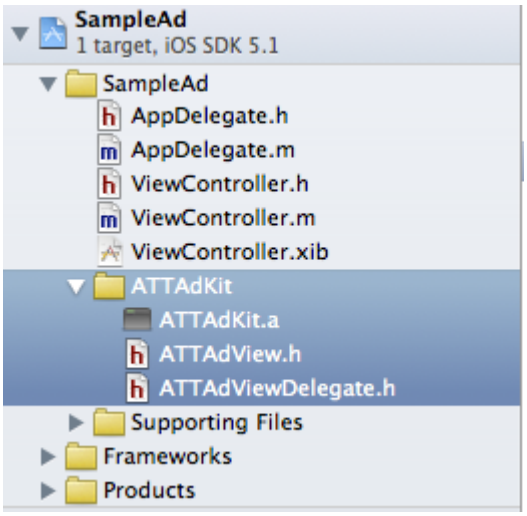
To incorporate the Advertising API into an iOS project, perform the following steps.

1. Open the Xcode project for your application.

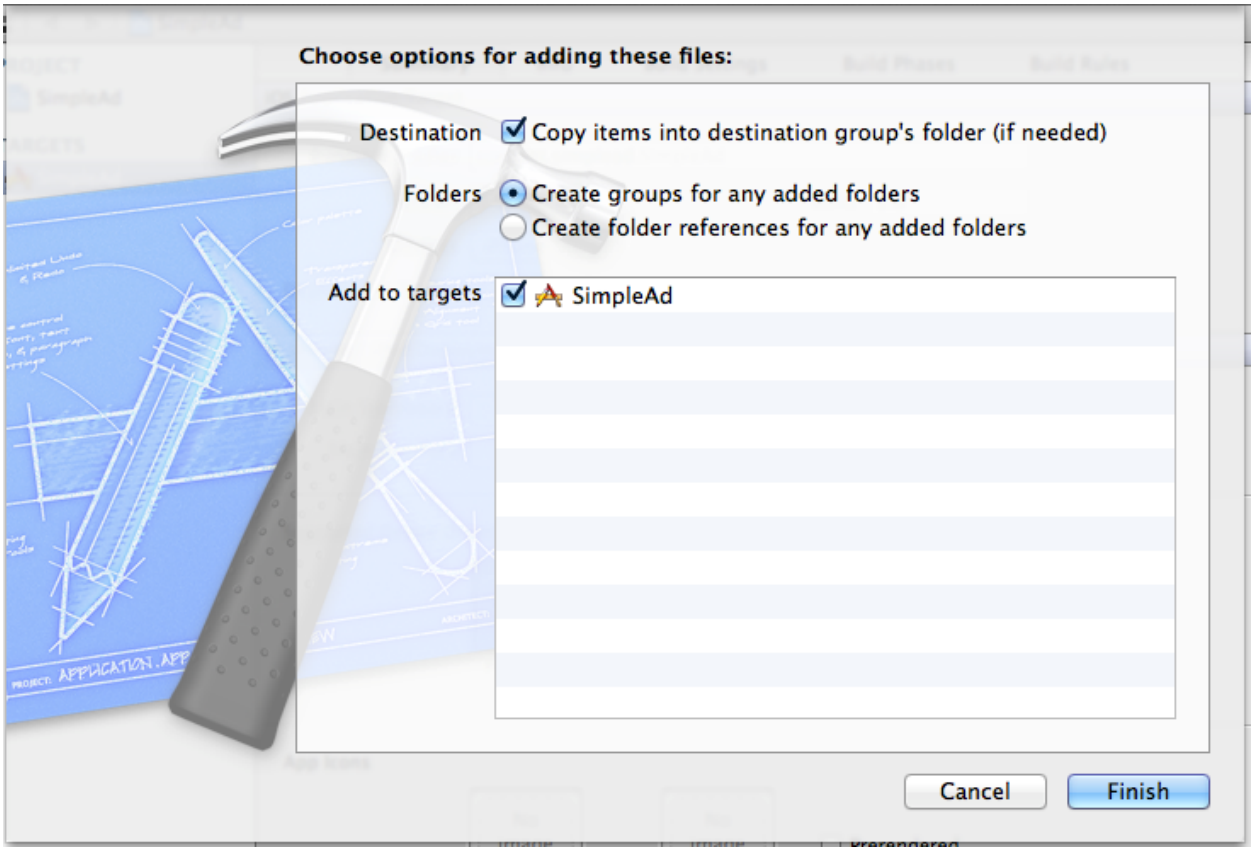
Display the unzipped *ATTAdKit* folder in the Finder, as shown in the following screen shot.



Drag the *ATTAdKit* folder from the Finder to the Xcode project hierarchy, as shown in the following screen shot.



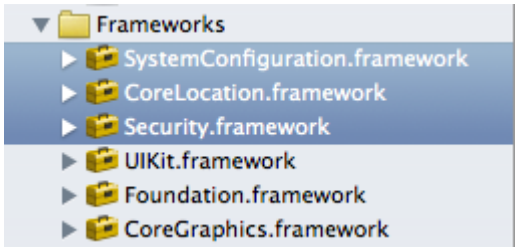
2. In the confirmation dialog, select the **Copy items into destination group's folder (if needed)** check box. Also, ensure that your application target is selected in the **Add To Targets** list. The result should look something like the following screen shot.



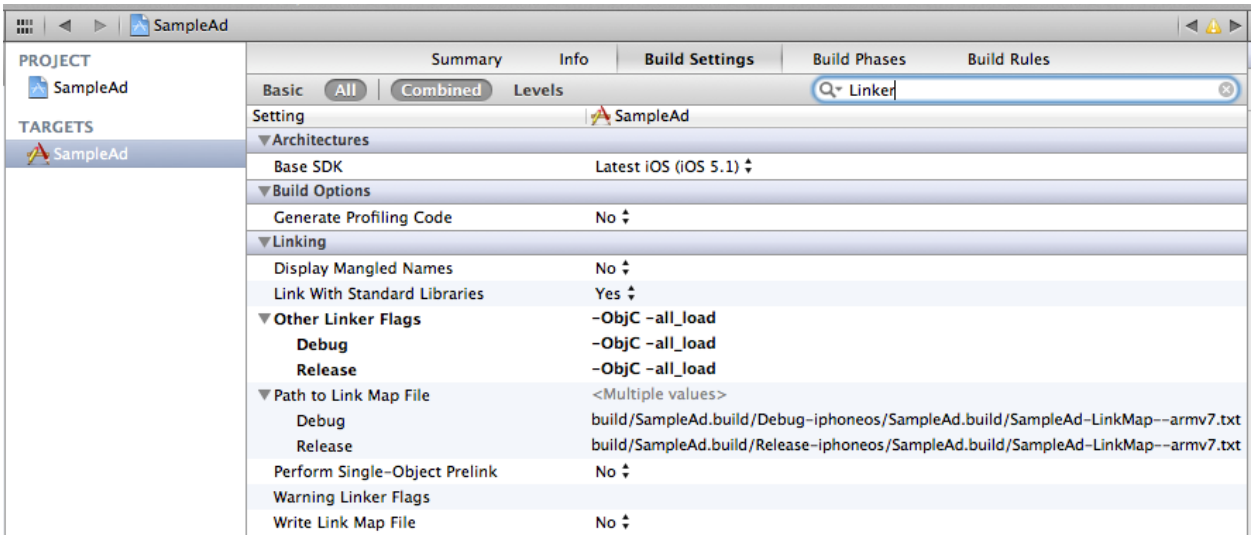
3. Click **Finish**.
4. In your Xcode project, right-click on the **Frameworks** group and select **Add > Existing Frameworks**. Select the following system frameworks.

   - SystemConfiguration.framework
   - CoreLocation.framework
   - Security.framework

   The result should look something like the following screen shot.



5. Click **Add**.
6. Set the **Other Linker Flags** in **Xcode Build Settings** to **ObjC -all_load**. The result should look something like the following screen shot.



7. Follow the instructions in the following section.
8. Build your application that includes the Advertising API.

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 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

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

```
// Create the ATTAdView object
ATTAdView *attAdView = [[ATTAdView alloc]init];

// Set  properties to filter the ad content
[attAdView setZipCode:98101];
[attAdView setPremium:PremiumBoth];
[attAdView setAgeGroup:@"26-35"] ;

// Initialize authentication/authorization, ad service call and draw the
// ad component with received response
attAdView = [attAdView initWithFrame:CGRectMake(0, 0, self.view.frame.size.width, 50)
                    appKey:APP_KEY appsecret:APP_SECRET category:@"other" udid:UDID];

// Add the view
[self.view addSubview:attAdView];
```

You must supply an APP key and secret key value for the APP_KEY and APP_SECRET placeholder in this code, and for the **Category** and **UDID** properties.

Since the app key and secret key values provided by AT&T are confidential, you should obfuscate them before passing them to the Advertising service. The sample app in the SDK demonstrates how to obfuscate and un-obfuscate your code.

1.4.3 Implementing Success and Failure Callback Methods

To handle the success or failure when your application requests an advertisement, implement the **ATTAdViewDelegate** so that it receives the following callbacks.

- **didReceiveAd:jsonResponse**
- **didFailedToReceiveAd:nsError**

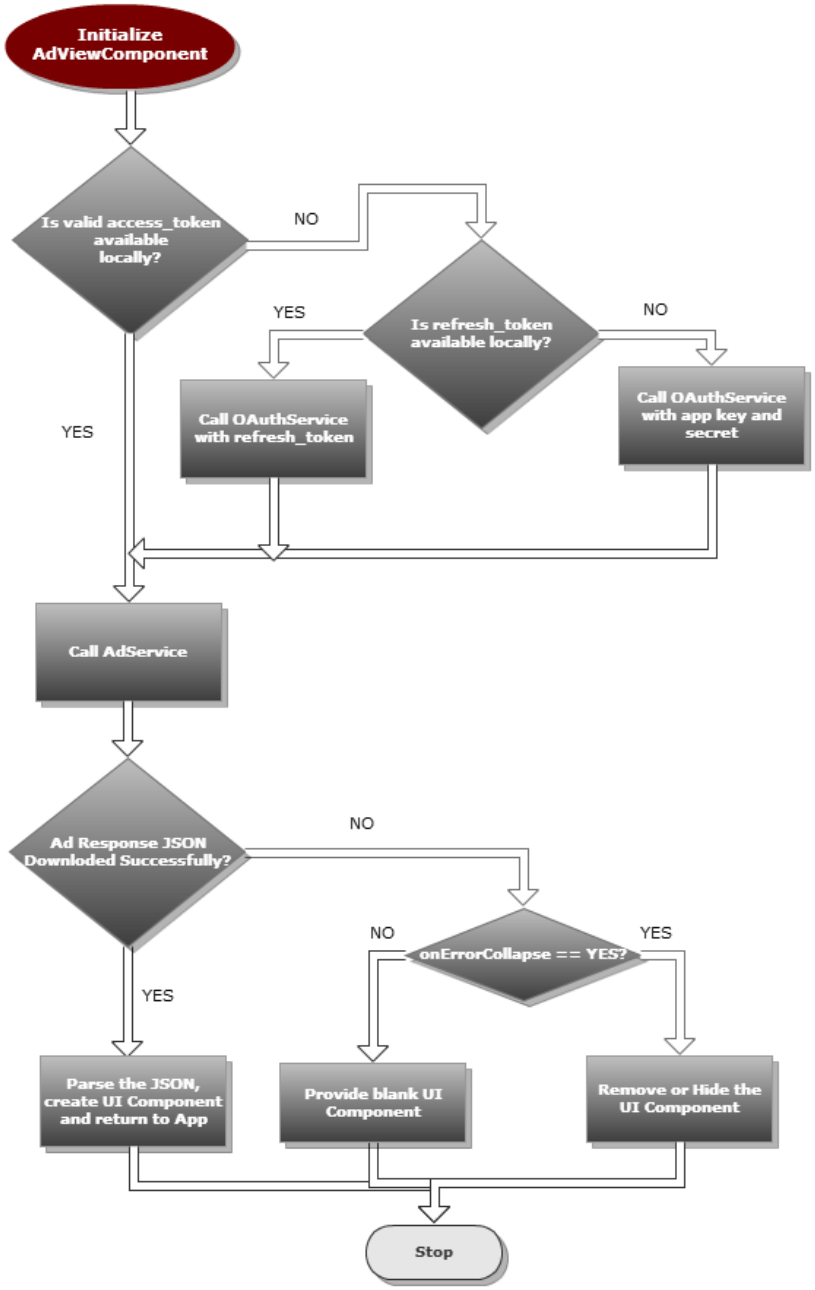
The following code sample shows the signature of these two methods with code comments that describe the values returned by the Advertising service that can be used to implement these two delegates.

```
-(void)didReceiveAd:(NSString *)jsonResponse {
   // The raw JSON response received from the ad service
   // is available in the variable jsonResponse
}

-(void)didFailedToReceiveAd:(NSError *)nsError {
   // The error code and message are available
   // in the variable nsError
```

```
}
```

1.4.4 Advertising API Flow



1.5 API Reference

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

1.5.1 ATTAdView Methods

The following table describes the methods in the **ATTAdView** class.

Method	Result Type	Required?	Description
initWithFrame:appKey: appsecret:category:adCategory	void	Yes	Initializes the advertisement component and renders it in the specified frame.

1.5.2 ATTAdView Properties

The following table describes the properties in the **ATTAdView** method.

Property Name	Default Value	Type	Required?	Description
udid		NSString	Yes	Specifies the UDID that is provided by the developer. Must be at least 30 characters in length or an error is returned.
category		NSString	Yes	Specifies the category of the app. The defined values for this parameter are: • auto • business • chat • communication • community • entertainment • finance • games

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

Property Name	Default Value	Type	Required?	Description
city	nil	NSString	No	Specifies the the USA city and state of the user.
country	nil	NSString	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		NSInteger	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		NSInteger	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		NSInteger	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.

Property Name	Default Value	Type	Required?	Description
minWidth		NSInteger	No	Specifies the minimum width, in pixels, of the advertisement banner. The width of the content may be greater than or equal to, but not less than this value.
type	-1	NSInteger	No	Specifies the type of advertisement. 1. Text 2. Image 3. Text and Image The default value is -1, which indicates any type of advertisement.
timeout	1000	NSInteger	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	nil	NSString	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

Property Name	Default Value	Type	Required?	Description
over18	nil	NSInteger	No	Specifies whether to display adult ads. The defined values for this parameter are: 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	nil	NSString	Optional	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 filter for ads about music, tv, or games, use "music,tv,games".
isSizeRequired	No	BOOL	No	Indicates whether the size of the advertisement must be returned in the response.
premium	0	NSInteger	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.
adRefreshPeriod	2	NSInteger	No	Specifies the interval, in minutes, at which the advertisement should be refreshed. A value of 0 indicates

Property Name	Default Value	Type	Required?	Description
				that the advertisement should not be refreshed.
onErrorCollapse	No	BOOL	No	Indicates whether to collapse the Advertising component, which hides the area from the user, if there is any error getting the advertisement.
autoDetectLocation	No	BOOL	No	Indicates whether the **ATTAdView** object should automatically detect the location of the device. When this value is **YES**, the location is automatically detected and passed to the Advertising service. Note that getting the ad may be delayed if this feature is enabled.

1.5.3 ATTAdViewDelegate Callbacks

Once the advertisement is requested, one of the delegate methods of the **ATTAdViewDelegate** class is called. The delegate may implement any of the two methods that communicate the success or failure of advertisement loading. The following table describes the delegate callback methods in the Advertising API.

Delegate Callback Method	Return	Required?	Description
didReceiveAd:jsonResponse	void	Optional	This method is invoked when an advertisement is successfully delivered and it renders properly. This method returns a reference to the raw JSON response received from Advertising service.

Delegate Callback Method	Return	Required?	Description
didFailedToReceiveAd:error	void	Optional	This method is invoked when authentication fails, an advertisement is not delivered, or the advertisemetn cannot be rendered. It returns a reference to an **NSError** object that contains error information.

1.5.4 ATTAdView Error Codes

When the **adViewDelegate:didFailedToReceiveAd** callback method is invoked, the application can determine what happened by examining the properties of the **code** property of the **NSError** object.

The **adViewDelegate:didFailedToReceiveAd** callback method receives errors from a variety of sources, such as network errors, HTTP error results, and errors reported by the operating system. It follows the standard practice for **NSError** objects, so each source of error corresponds to a value of the **domain** property of the **NSError** object and each distinct error from a source corresponds to a value of the **code** property of the **NSError** object.

For advertising service errors, value of the **domain** property of the **NSError** object is always **ATTAdErrorDomain**.

The following table describes the possible values of the **code** property of the **NSError** object.

NSError Code	Description
ATTAdErrorCodeAuthFailure	The Advertising API was unable to authenticate the app with the supplied APP key and secret key.
ATTAdErrorCodeServiceError	The Advertising API was unable to retrieve an advertisement from the Advertising service.
ATTAdErrorCodeNetworkError	The device was unable to connect to internet.
ATTAdErrorCodeMissingParameter	One or more values for the mandatory properties is either missing or is invalid.

NSError Code	Description
ATTAdErrorCodeOtherError	An error not classified above.

2. Speech SDK

This section describes how to add speech recognition to iOS applications using version 2.3.2 of the AT&T API Platform Speech SDK for iOS™.

iOS is the software platform for the Apple iPhone, iPod touch, and iPad. The AT&T Speech SDK supports applications developed with the iOS Software Development Kit and distributed by the iOS App Store. The iOS SDK is available for iOS Development Program members at the website http://developer.apple.com, which provides the tools and APIs necessary to develop applications on the iOS platform using the Objective-C programming language. The Speech SDK supports applications that target iOS version 4.3 and higher.

2.1 About the Speech SDK

The Speech SDK is packaged as a static library called `ATTSpeechKit` that links to your iOS application code. The library implements a client for the Speech to Text web service. It exports the `ATTSpeechService` class, which has the following functions:

- Capturing audio from the device microphone.
- Showing feedback to the user.
- Streaming audio and custom data 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 an `ATTSpeechService` object and calls a method to start the interaction. The `ATTSpeechService` instance 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 `ATTSpeechService` class returns the result of recognition to your application.

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

2.2 Prerequisites

To develop applications for iOS, you need a Mac OS X computer that runs the Xcode tools, and you should be familiar with programming iOS applications using Objective-C. To get started with the Speech SDK for iOS, prepare your development environment by completing the following steps:

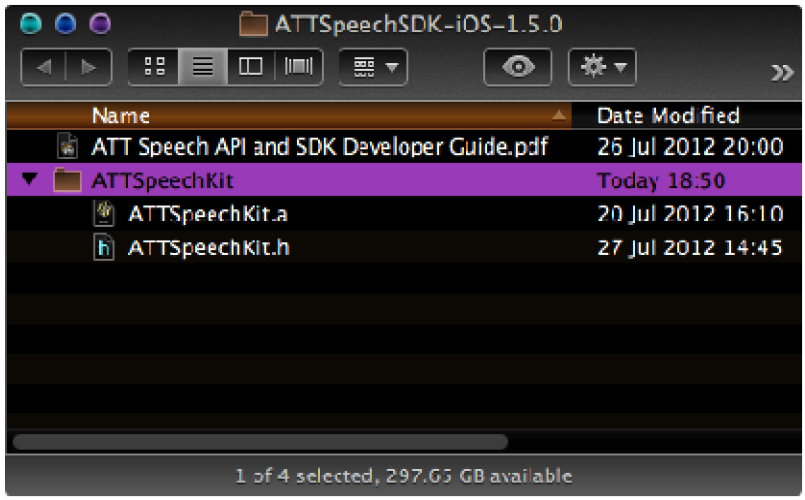
1. **iOS Developer Program** — You must be a member of Apple's developer program to run applications on a device and to distribute applications to customers.
2. **Install the iOS SDK and Xcode** — Follow the instructions on the Apple Developer website to install the SDK and development tools.

2.3 Setting up your iOS Project

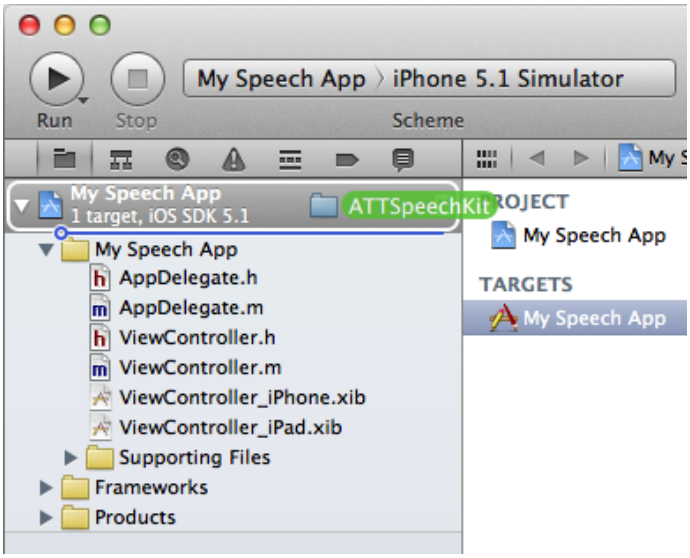
To use the Speech SDK in your application, you must add the `ATTSpeechKit` library to your Xcode project and link it to a set of system frameworks. The `SimpleSpeech` sample code on the AT&T Developer web site includes an Xcode project that is already configured to link with `ATTSpeechKit`. You can use that sample project as the basis for new apps.

To add the Speech SDK to an existing project, perform the following steps:

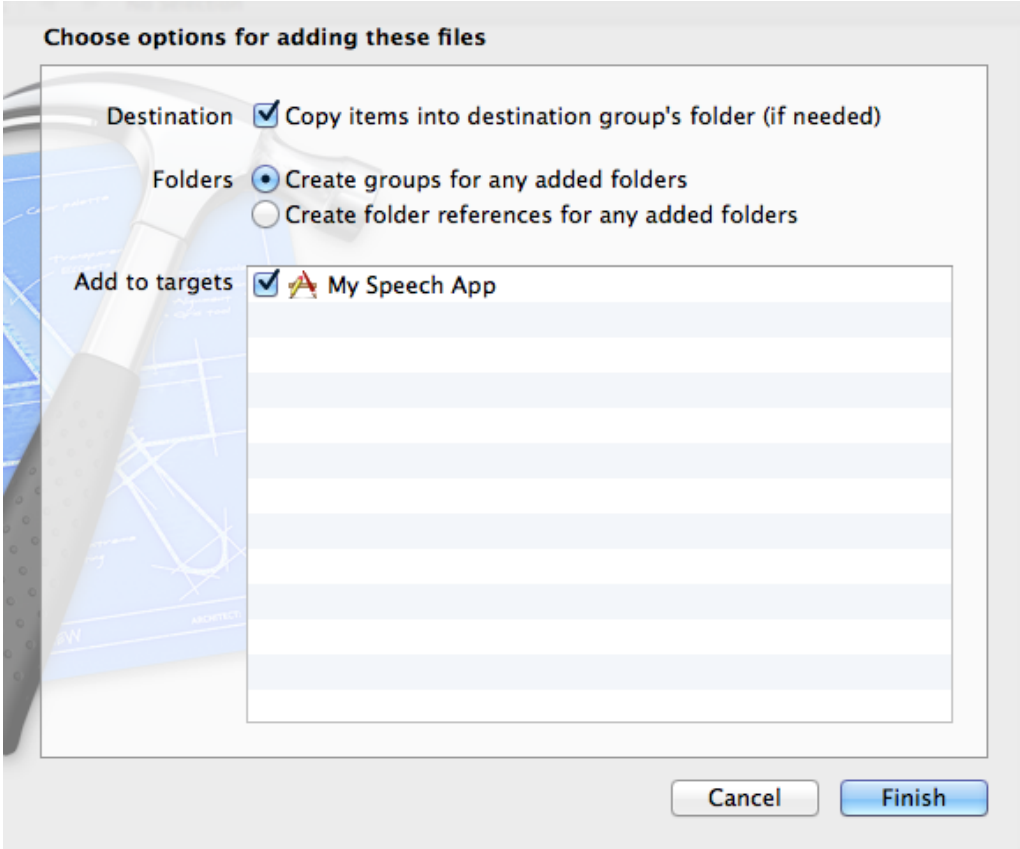
1. Open the Xcode project for your application.
2. Display the unzipped Speech SDK in the Finder and select the `ATTSpeechKit` subfolder.



3. Drag the `ATTSpeechKit` folder from the Finder onto the project icon within your Xcode window.



4. In the confirmation dialog that appears, select the **Copy items into destination group's folder (if needed)** check box and the **Create groups for any added folders** button. Also, ensure that your application target is selected in the **Add To Targets** list. Then click the **Finish** button.



Link to the system frameworks by following these steps:

1. Display the Project Editor in Xcode by selecting your project in theProject Navigator.
2. Select your target in the Project Editor.
3. Select the **Build Phases** tab.
4. Expand the Link Binaries With Libraries section.
5. Press the “+” button, and add the following system frameworks:

```
AudioToolbox.framework
AVFoundation.framework
CFNetwork.framework
Security.framework
```

(Optional) In the Project Navigator, move the newly added frameworks to the Frameworks group.

When complete, your Xcode target should look like the following.

Follow the instructions in the following section to add code that invokes the `ATTSpeechService` class and receives delegate callbacks.

2.4 Speech Recognition Tasks on iOS

To add speech recognition to your application, you need to write code that performs the following functions:

- Configure `ATTSpeechService` runtime properties.
- Acquire an OAuth access token.
- Start speech interaction at the appropriate time (for example, in a button press handler)
- (Optional) Send inline grammar data.
- (Optional) Customize UI during speech interaction.
- (Optional) Customize speech endpointing.
- Handle speech recognition results and errors.

2.4.1 Configure `ATTSpeechService` runtime properties

The AT&T Speech SDK needs to be configured before an application uses it to call the Speech API. The best place to perform this configuration is in the application delegate's `applicationDidBecomeActive:` method. Adding the configuration code to this method ensures that the Speech SDK is configured early during the application lifecycle, which

minimizes latency during a speech interaction. It also ensures that the configuration occurs each time iOS brings the application to the foreground.

Your application will access the Speech SDK though the singleton `ATTSpeechService` object. This object manages the device's microphone input and network connection to the Speech API on behalf of your application. It exposes a number of properties that you can use to specify the parameters of a speech recognition request. See Table 2 for the full list of `ATTSpeechService` request properties.

The `delegate` property is a mandatory property to set. It is an object supplied by your application that implements the `ATTSpeechServiceDelegate` protocol. For more information on the `ATTSpeechServiceDelegate` protocol, see Table 4 . iOS ATTSpeechServiceDelegate Callback Methods.

Another mandatory property is `recognitionURL`. This is the URL of the Speech API `SpeechToText` service that your application uses. Get this URL from your application's on-boarding information on the AT&T Developer web site.

A third mandatory property for Speech API is `speechContext`. This property contains the name of the speech context that your application wants to use for speech recognition. See the **Error! Reference source not found.** section for more information on the available contexts.

When your application configures the runtime properties of the Speech SDK, it can optionally tell `ATTSpeechService` to prepare the audio hardware of the device. This minimizes delays that can occur the first time the microphone is activated.

In addition to setting the Speech SDK properties, your application should also acquire an OAuth access token when the OS activates the application. See section 2.4.2 for more details on acquiring the OAuth token.

In the following example, code is added to the application's `applicationDidBecomeActive:` method to configure the Speech SDK and get an OAuth token.

Example 1. Configure ATTSpeechService in applicationDidBecomeActive

```
- (void) applicationDidBecomeActive: (UIApplication*) app
{
    ATTSpeechService* speechService = [ATTSpeechService sharedSpeechService];
    speechService.delegate = self;
    speechService.recognitionURL =
        [NSURL URLWithString: @"https://api.att.com/rest/1/SpeechToText"]
    speechService.speechContext = @"Generic";
    [self myRenewSpeechOAuth];
    [speechService prepare];
}
```

2.4.2 Acquire OAuth access token

Requests to the Speech to Text service must include a valid OAuth access token. To avoid authentication failures during a speech interaction, your application should obtain 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 can use the token in subsequent speech interactions by setting the `bearerAuthToken` property, as in the following lines:

```
ATTSpeechService* speechService = [ATTSpeechService sharedSpeechService];
speechService.bearerAuthToken = myAccessToken;
```

Since the network I/O 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 is every time your app comes to the foreground. This ensures that the subsequent speech request will have a valid token. The sample code in example 1 includes the line `[self myRenewSpeechOAuth]`. You should replace that line with a call to your application's code that begins the asynchronous token request. If the users of your application will keep it in the foreground longer than an hour, you need to add code to request a token at hourly intervals.

2.4.3 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 application code handling the button press calls the `startListening` method, and the `ATTSpeechService` object will begin a speech interaction. The Speech SDK will turn on the microphone and begin streaming data in compressed AMR format to the Speech API server. Your delegate object is called at various stages through the speech interaction until it receives the recognition result or an error.

In the following example, code is added to the application's action handler method to start a speech interaction.

Example 2. Start speech interaction on iOS

```
- (IBAction) listen: (id) sender
{
    ATTSpeechService* speechService = [ATTSpeechService sharedSpeechService];
    [speechService startListening];
}
```

2.4.4 Send inline grammar data

If your application is using a Speech to Text context with custom grammars, you need to add the grammar data to the multipart document of the web service request. See the **Error! Reference source not found.** section for more information on the multipart format.

There are two steps in adding grammar data to the request. First, your application must set the `isMultipart` and `contentType` properties of the `ATTSpeechService` object as in the following lines. That tells the Speech SDK to send data in the proper multipart format.

```
speechService.isMultipart = YES;
speechService.contentType = @"multipart/x-srgs-audio";
```

Then your delegate object needs to implement the `speechServiceSendingPartsBeforeAudio:` callback method. ATTSpeechService calls this method as it is streaming data on the HTTP request to the web service. Your delegate implementation will insert the inline grammar data at this stage using the `addPart:contentType:disposition:` method. The following example shows adding both lexicon and grammar data to a request.

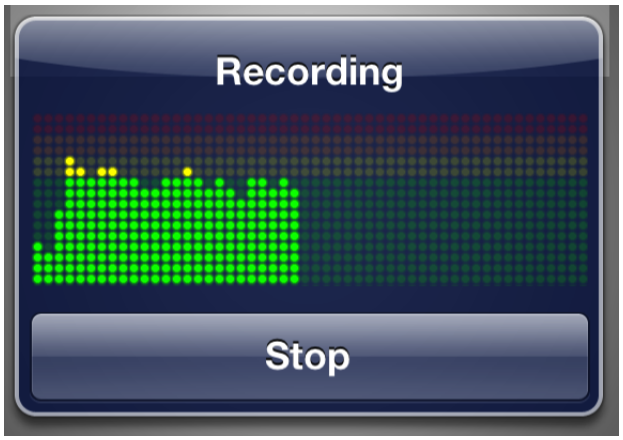
Example 3. Send inline grammar on iOS

```
- (void) speechServiceSendingPartsBeforeAudio: (ATTSpeechService) speechService
{
    [speechService addPart: myLexiconData contentType: @"application/pls+xml"
disposition: @"form-data; name=\"x-dictionary\""];
    [speechService addPart: myGrammarData contentType: @"application/srgs+xml"
disposition: @"form-data; name=\"x-grammar\""];
}
```

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, as shown in Figure 1. 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. Your application can enable this UI by setting the `showUI` property of `ATTSpeechService` to `YES`. It can customize the labels in the alert through several `ATTSpeechService` properties. See Table 2 for more information on the UI properties.

Figure 1. Speech SDK standard UI on iOS



If your application displays its own UI during a speech interaction, it can register delegate callback methods to receive updates during the recording and processing. Implement the delegate method `speechService:audioLevel:` to receive an average input volume level roughly every 1/10 second. Other delegate methods receive notifications when listening starts and ends. See Table 4 for more information on the delegate callbacks.

The Speech SDK automatically ends recording when the user stops talking. However, your application can manually stop capturing audio by calling the `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 `cancel` method to abort the speech interaction. Note that your delegate 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 iOS:

- **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 `endingSilence` property.
- **Timeout Expiration** — An application using the Speech SDK can set the maximum duration of speech input through the `maxRecordingTime` 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 `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 the delegate's `speechServiceSucceeded:` method. The `ATTSpeechService` argument exposes properties for the application to get the recognition response data. An array of transcription hypothesis strings is available in the `responseStrings` property. The parsed JSON data from

the service is in the `responseDictionary` property, and the raw binary data is in the `responseData` property. The full list of response properties is in Table 5.

The following example shows a minimal implementation of the `speechServiceSucceeded:` method that checks the hypothesis array.

Example 4. Successful Transaction Delegate Method

```
-(void) speechServiceSucceeded: (ATTSpeechService*) speechService
{
    NSArray* nbest = speechService.responseStrings;
    NSString* recognizedText = @"";
    if (nbest != nil && nbest.count > 0)
        recognizedText = [nbest objectAtIndex: 0];
    if (recognizedText.length)
        NSLog(@"Recognition result: %@", recognizedText);
    else
        NSLog(@"Speech was not recognized.");
}
```

If an error occurs doing the speech recognition process, the Speech SDK calls the delegate's `speechService:failedWithError:` method. Possible errors include the user canceling, the user not speaking in time, problems with the network connection, and errors reported by the server. The `NSError` argument includes properties that describe the type of error. See section 2.5.5 for more information on interpreting error callbacks.

The following example shows a minimal implementation of the `speechService:failedWithError:` method that checks the error status.

Example 5. Failed Transaction Delegate Method

```
-(void) speechService: (SpeechService*) speechService failedWithError: (NSError*)
error
{
if ([error.domain isEqualToString: ATTSpeechServiceErrorDomain] && (error.code ==
ATTSpeechServiceErrorCodeCanceledByUser))
        NSLog(@"User canceled.");
else
        NSLog(@"Recognition failed with error: %@", [error localizedDescription]);
}
```

2.5 Speech SDK Reference for iOS

This section is a reference for the Speech SDK on iOS. It describes the methods, properties, callbacks, error codes, and state transitions in the `ATTSpeechService` class:

- `ATTSpeechService` Methods
- `ATTSpeechService` Request Properties
- `ATTSpeechService` Delegate Callbacks
- `ATTSpeechService` Response Properties
- `ATTSpeechService` Error Codes

- `ATTSpeechService` State Transitions

2.5.1 `ATTSpeechService` Methods

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

Table 1. iOS `ATTSpeechService` Methods

Method	Required /Optional	Description
`+(ATTSpeechService) sharedSpeechService`	Required	This class method returns the singleton speech service object.
`-(void) prepare`	Optional	Initializes the iOS audio subsystem. Calling this method is optional, but if it is called early during the application lifecycle, it can reduce the latency of the first speech interaction requested by the user.
`-(BOOL) 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 this method is called, the Speech SDK will invoke methods on the delegate object to report interaction status, errors, and the recognition result.
`-(BOOL) startWithAudioData: (NSData*) 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 audio data, so it 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 delegate object to report interaction status, errors, and the recognition result.

Method	Required /Optional	Description
`-(void) stopListening`	Optional	Manually ends speech input. This method is optional because the `ATTSpeechKit` library will automatically end 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 it will report the result or error to the delegate object.
`-(void) extendTimeout: (NSTimeInterval) extraInterval`	Optional	Keep acquiring data for the specified number of seconds. This method is optional in unimodal speech applications. Call this in multimodal applications to prevent `ATTSpeechService` object from ending audio input while the user is supplying input through other means.
`-(void)` addPart: (NSData*) partData contentType: (NSString*) contentType disposition: (NSString*) contentDisposition	Optional	When the Speech SDK is posting multipart data to the Speech to Text service, the application can call this method to add additional parts to the posted data. It may only be called when the delegate is handling a `speechServiceSendingPartsBeforeAudio:` or `speechServiceSendingPartsAfterAudio:` callback. It is an error to call this method in any other callbacks or outside of a speech interaction.
`-(void) 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 delegate callbacks related to the canceled interaction.

Method	Required /Optional	Description
`+(void) cleanupForTermination`	Optional	Call this class method when you are done with all speech interactions to release all resources used by the Speech SDK. This method is intended for non-production builds of applications when investigating memory issues.

2.5.2 `ATTSpeechService` Request Properties

The following table consists of the `ATTSpeechService` properties that control speech recognition. Set these properties before calling the `startListening` or `startWithAudioData:` methods.

Table 2. iOS `ATTSpeechService` Request Properties

Property Name	Default Value	Type	Required /Optional	Description
`delegate`		`ATTSpeechService Delegate`	Required	An application-defined object that receives callbacks during speech recognition. See section 2.5.3 `ATTSpeechService` Delegate Callbacks for more information.
`currentState`	`ATTSpeechService StateIdle`	ATTSpeechService State	n/a	Returns the current state of a speech interaction.
`recognitionURL`		`NSURL`	Required	The URL of the Speech API service. See the **Error! Reference source not found.** section for more information.

Property Name	Default Value	Type	Required /Optional	Description
`isMultipart`	`NO`	`BOOL`	Optional	Whether to wrap the audio data in a MIME multipart container. When true, Speech SDK will post data in multipart format, and the delegate must implement the `speechServiceSendingPartsBeforeAudio:` and/or `speechServiceSendingPartsAfterAudio:` callbacks.
`contentType`	see description	`NSString`	Optional	MIME type of the audio data for the server. It is used in the `Content-Type` HTTP header. This property is needed only when uploading an audio file. When using the microphone, `ATTSpeechService` uses MIME type for the chosen audio format.
`speechContext`		`NSString`	Required	The name of the Speech to Text context to use for recognition. It is used in the `X-SpeechContext` HTTP header.
`xArgs`	`nil`	`NSDictionary` mapping `NSString` to `NSString`	Optional	A collection of key-value argument pairs for a Speech API context. It is used in the `X-Arg` HTTP header. This property will be merged with a header specified in `requestHeaders` and the default values generated by Speech SDK.

Property Name	Default Value	Type	Required /Optional	Description
`sendsDefaultXArgs`	`YES`	`BOOL`	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.
`audioFormat`	ATTSKAudioFormatAMR_NB	`NSString`	Optional	Name of format in which to encode audio from the microphone. Defaults to AMR narrowband format. Use only the constants defined in Table 3.
`bearerAuthToken`		`NSString`	Required	An OAuth access token that validates speech requests from this application. It is used in the `Authentication: Bearer` HTTP header. For more information on OAuth access tokens, see the **Error! Reference source not found.** section.
`requestHeaders`	`nil`	`NSDictionary` mapping `NSString` to `NSString`	Optional	A collection of HTTP headers to add to the request.
`showUI`	`NO`	`BOOL`	Optional	Setting this property to true enables the Speech SDK to display a progress meter and a button for canceling the interaction.

Property Name	Default Value	Type	Required /Optional	Description
`recordingPromptMessage`	"Listening"	`NSString`	Optional	The text that the Speech SDK displays while capturing audio data from the microphone. Use this property only when the `showUI` property is true.
`recordingStopButtonTitle`	"Stop"	`NSString`	Optional	The button label that the Speech SDK displays while capturing audio data from the microphone. The user can press the button to manually endpoint audio input. Use this property only when the `showUI` property is true.
`processingPromptMessage`	"Processing"	`NSString`	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.
`processingCancelButtonTitle`	"Cancel"	`NSString`	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.
`connectionTimeout`	10.0	`float`	Optional	The maximum number of seconds that the Speech SDK waits for a connection to the server when initiating a recognition request.

Property Name	Default Value	Type	Required /Optional	Description
`cancelWhenSilent`	`YES`	`BOOL`	Optional	Controls how the Speech SDK handles silent audio input. When the value is `NO`, the Speech SDK will send silent audio to the server for processing. When the value is `YES`, the Speech SDK will automatically return an error code of `ATTSpeechServiceErrorCodeNoAudio` when it detects silent audio.
`maxInitialSilence`	1.5	`float`	Optional	The maximum number of seconds that a user can remain silent before beginning to talk. To disable the timeout, set the property to `ATTSpeechServiceInfiniteInterval`.
`endingSilence`	0.85	`float`	Optional	The number of seconds of silence after the user stops talking before the Speech SDK performs endpointing. To disable automatic endpointing, set the property to `ATTSpeechServiceInfiniteInterval`.
`minRecordingTime`	1.0	`float`	Optional	The minimum number of seconds of audio to capture from the microphone. This limit is mainly useful for manual endpointing via stopListening. If the user does not meet this limit, Speech SDK will return an error code of `ATTSpeechServiceErrorCodeAudioTooShort`. To disable this limit, set the property to zero.

Property Name	Default Value	Type	Required /Optional	Description
`maxRecordingTime`	25.0	`float`	Optional	The maximum number of seconds of audio to capture from the microphone. When this limit is exceeded, Speech SDK stops listening to the microphone and waits for the Speech API to process the captured audio. To make this duration unlimited, set the property to `ATTSpeechServiceInfiniteInterval`.
`serverResponseTimeout`	30.0	`float`	Optional	The maximum number of seconds that the Speech SDK waits for the server to complete a recognition response.

Some of the values for `ATTSpeechService` properties are special constants declared by the ATTSpeechKit library. The following table describes those constants.

Table 3. iOS `ATTSpeechService` Constants

Constant Name	Type	Description
ATTSKAudioFormatAMR_NB	`NSString`	Assigned to `ATTSpeechService.audioFormat` to specify AMR narrowband format, which has a bitrate of 12.2 kbps.
ATTSKAudioFormatWAV_NB	`NSString`	Assigned to `ATTSpeechService.audioFormat` to specify uncompressed streaming WAV narrowband format, which has a bitrate of 128 kbps.

Constant Name	Type	Description
ATTSKAudioFormat WAV_WB	`NSString`	Assigned to `ATTSpeechService.audioFormat` to specify uncompressed streaming WAVE wideband format, which has a bitrate of 256 kbps.
ATTSpeechService InfiniteInterval	`float`	Represents an infinite time duration for properties such as `ATTSpeechService.maxRecordingTime`.

2.5.3 `ATTSpeechService` Delegate Callbacks

While the `ATTSpeechService` object is performing a speech recognition interaction, it calls methods on its delegate object. The application is required to implement the two delegate methods that communicate the recognition result to the application. The remaining delegate methods are optional, giving the application a chance to customize the behavior of the speech interaction.

The following table describes the Speech SDK delegate callback methods.

Table 4. iOS ATTSpeechServiceDelegate Callback Methods

Delegate Callback Method	Return	Required / Optional	Description
`speechServiceSucceeded:`	void	Required	The Speech SDK calls this method when it returns a recognition result. The `ATTSpeechService` object will contain properties that include the response data and recognized text. For more information on how to interpret the response data, see the ATTSpeechService Response Properties section.

Delegate Callback Method	Return	Required / Optional	Description
`speechService: failedWithError:`	void	Required	The Speech SDK calls this method when speech recognition fails. The reasons for failure can include the user canceling, network errors, or server errors. For more information on the `NSError` argument, see the ATTSpeechService Error Codes section.
`speechServiceWillStartListening:`	void	Optional	The Speech SDK calls this method immediately before capturing audio to give the application a chance to perform a task. For example, the application can display a custom recording UI.
`speechServiceIsListening:`	void	Optional	The Speech SDK calls this method when it has begun capturing audio from the microphone.
`speechServiceShouldEndRecording:`	BOOL	Optional	The Speech SDK calls this method when it is ready to end audio capture because the user is silent. The delegate method should return `NO` if listening should continue.
`speechServiceHasStoppedListening:`	void	Optional	The Speech SDK calls this method when it has finished capturing audio from the microphone.
`speechService:audioLevel:`	void	Optional	The Speech SDK calls this method repeatedly as it captures audio. The callback rate is roughly 1/10 second. An application can use the audio level data to update its UI.

Delegate Callback Method	Return	Required / Optional	Description
`speechService: willEnterState:`	void	Optional	The Speech SDK calls this method when it transitions among states in a recording interaction, for example, from capturing to processing. The method argument contains the state the service will enter, and `speechService.currentState` contains the state the service is leaving. Upon exiting this delegate callback, the service will be in the new state.
speechServiceSendingParts BeforeAudio:	void	Optional	When Speech SDK is posting data in multipart format, it calls this delegate method so the application can add parts to the request. The application should call `addPart:contentType:disposition` to add a part. Parts added during the handling of this delegate callback will be received by the speech recognition service before the audio from the microphone, so this is the appropriate place to add inline grammars and hints to a request.
speechServiceSendingParts AfterAudio:	void	Optional	When Speech SDK is posting data in multipart format, it calls this delegate method so the application can add parts to the request. The application should call `addPart:contentType:disposition` to add a part. Parts added during the handling of this delegate callback will be received by the speech recognition service after the audio from the microphone.

2.5.4 `ATTSpeechService` Response Properties

When a speech interaction is complete, the `ATTSpeechService` object exposes data about the response in its object properties. These response properties are valid during the execution of the `speechServiceSucceeded:` callback.

Table 5. iOS `ATTSpeechService` Response Properties

Property Name	Type	Description
`responseStrings`	`NSArray` of `NSString`	Contains the recognition response as an array of hypothesis strings. It is roughly equivalent to the JSON path `recognition.nbest[*].hypothesis`. Each string in the array is a possible way of interpreting the submitted speech. When the speech cannot be recognized, this property will be `nil` or have zero elements.
`responseDictionary`	`NSDictionary`	Contains the parsed JSON tree of the speech recognition response. For more information on how to interpret the data, see the **Error! Reference source not found.** section. This property is `nil` if the speech service did not return JSON.
`responseData`	`NSData`	Contains the raw bytes of the speech recognition response.
`statusCode`	`int`	Contains the HTTP status code of the speech recognition response.
`responseHeaders`	`NSDictionary` mapping `NSString` to `NSString`	Contains the HTTP headers that were included in the speech recognition response. Headers whose keys appear multiple times in a response (such as `Set-Cookie`) have their values concatenated together.

2.5.5 `ATTSpeechService` Error Codes

When the Speech SDK calls the application's `speechService:failedWithError:` callback, the application can determine what happened by examining the properties of the `NSError` argument. For more information on the callback, see the ATTSpeechService Delegate Callbacks section.

The Speech SDK reports errors from a variety of sources, such as network errors, HTTP error results, and errors reported by the operating system. It follows the standard practices for

`NSError` objects, so that each source of error corresponds to a value of the `NSError domain` property, and each distinct error from a source corresponds to values of the `NSError code` property. The following table describes the error domains and codes reported by the Speech SDK.

Table 6. iOS NSError Values

`NSError` Domain	**`NSError` Code**	**Description**
`ATTSpeechServiceErrorDomain`	`ATTSpeechServiceErrorCodeCanceledByUser`	The user canceled the speech interaction. This occurs, for instance, when the user presses the **Cancel** button or puts the application in the background.
`ATTSpeechServiceErrorDomain`	`ATTSpeechServiceErrorCodeAttemptAtReentrancy`	The application called `[ATTSpeechService startListening]` or `[ATTSpeechService startWithAudioData:]` while a speech interaction was already in process.
`ATTSpeechServiceErrorDomain`	`ATTSpeechServiceErrorCodeInvalidParameter`	One of the properties on the `ATTSpeechService` object was invalid.
`ATTSpeechServiceErrorDomain`	`ATTSpeechServiceErrorCodeInvalidURL`	The URL that was passed to the `recognitionURL` property is invalid.
`ATTSpeechServiceErrorDomain`	`ATTSpeechServiceErrorCodeNoDelegate`	The application did not set the delegate of the `ATTSpeechService` object.
`ATTSpeechServiceErrorDomain`	`ATTSpeechServiceErrorCodeConnectionFailure`	The Speech SDK was unable to make a network connection to the Speech API service.
`ATTSpeechServiceErrorDomain`	`ATTSpeechServiceErrorCodeNoResponseFromServer`	The server did not send a full response because it timed out, or the network connection was dropped.
`ATTSpeechServiceErrorDomain`	`ATTSpeechServiceErrorCodeAudioTooShort`	`[ATTSpeechService stopListening]` was called before enough audio input was collected. Returned only when the `minRecordingTime` property is nonzero.

NSError Domain	NSError Code	Description
ATTSpeechServiceErrorDomain	ATTSpeechServiceErrorCodeNoAudio	The user didn't speak loud enough for the speech to be recognized. Returned only when the cancelWhenSilent property is true.
ATTSpeechServiceErrorDomain	ATTSpeechServiceErrorCodeNoMicrophone	The device doesn't have a microphone.
ATTSpeechServiceHTTPErrorDomain	HTTP error code, for example 400, 404, 500, etc.	The server returned an HTTP error code.
Other domain values	N/A	There was an error propagated from the operating system.

2.5.6 ATTSpeechService State Transitions

An application can test the current stage of a speech interaction by accessing the currentState property of the ATTSpeechService instance. The delegate can also be notified automatically about the transitions among the stages by implementing the speechService:willEnterState: method.

The following table describes the ATTSpeechService states that occur during speech interaction.

Table 7. iOS ATTSpeechServiceState Values

State	Description
ATTSpeechServiceStateIdle	Nothing occurring. ATTSpeechService is in this state before beginning a speech interaction and after delegate callbacks have returned.
ATTSpeechServiceStateConnecting	Attempting to connect to the host before audio capture begins.
ATTSpeechServiceStateRecording	Audio capture is taking place.
ATTSpeechServiceStateSendingAudioData	Audio data are being sent to the server.

State	Description
`ATTSpeechServiceStateProcessing`	The server is processing audio data.
`ATTSpeechServiceStateError`	`ATTSpeechService` is handling an error condition.

2.6 Testing Speech Applications on iOS

The Speech SDK implements audio capture and network I/O as asynchronous activities. However, iOS does not offer easy-to-use testing frameworks that support the event loops required for asynchronous operations. As a consequence, it is difficult to use automated unit testing tools to test speech functionality.

The best procedure for application developers is to verify Speech API functions (like valid credentials and connectivity) outside of their iOS application by using the cURL command-line tool. For more information on the testing with cURL, see the **Error! Reference source not found.** section.

2.7 Deploying Speech Applications on iOS

Developers who create speech-enabled applications on iOS devices using the Speech SDK must link their applications to the `ATTSpeechKit` static library. There is nothing additional that must be added to distribute the application to end-users.