Speech-to-Text Service  
Programming Guide

Version 2.1 – September 26, 2012

Abstract

By using the Project Hawaii Speech-to-Text (STT) service, a mobile application can translate audio speech to text. An application can deliver up to 10 seconds of audio speech to the service for translation in a single call. The service supports the use of multiple grammars.

This document provides a brief introduction to the managed interface to the STT service and walks you through a simple application that uses it.

Contents

[Introduction 2](#_Toc335305919)

[Prerequisites 2](#_Toc335305920)

[The Speech Recognition Client Library 2](#_Toc335305921)

[Walkthough: SpeechToTextSample Application 2](#_Toc335305922)

[Querying for Grammars 4](#_Toc335305923)

[Converting Speech to Text 5](#_Toc335305924)

[Using the STT Service in an Application 7](#_Toc335305925)

[Add Required Assemblies 7](#_Toc335305926)

[Reference the Namespace 7](#_Toc335305927)

[Set Up Your Authentication Credentials 7](#_Toc335305928)

[Tips and Guidelines 7](#_Toc335305929)

[Resources 8](#_Toc335305930)

**Disclaimer:** This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2012 Microsoft Corporation. All rights reserved.

Microsoft, Visual Studio, Silverlight, XNA, Windows Azure, and Windows are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.

# Introduction

By using the Project Hawaii Speech-to-Text (STT) service, a mobile application can translate audio speech to text. An application can deliver up to 10 seconds of audio speech to the service for translation in a single call. The service supports the use of multiple grammars.

This document provides a brief introduction to the managed interface to the STT service and walks you through a simple application that uses it.

# Prerequisites

Before you can build an application that uses the STT service, you must:

* Install the Project Hawaii SDK.
* Build the Project Hawaii SDK.
* Obtain Project Hawaii authentication credentials.

For information about installation, build procedures, and credentials, see “Hawaii Installation Guide,” which is installed with the SDK and is available on the web, as listed in “Resources” at the end of this document.

In addition, you should be familiar with the following:

* Windows Communication Foundation (WCF)
* Microsoft Silverlight®
* Windows Phone 7 SDK

# The Speech Recognition Client Library

The simplest way to communicate with the Hawaii STT service is to use the Speech Recognition Client Library. This library implements an interface that enables a mobile application to communicate with the Hawaii Rendezvous service. The source code for this library is installed with the Project Hawaii SDK in the following location:

* Source\ServiceClients\SpeechToText

Applications access the Speech Recognition Client library through the **Microsoft.Hawaii.Speech.Client** namespace, which defines the following classes:

|  |  |
| --- | --- |
| Class | Description |
| **SpeechResult** | Describes the result of a Hawaii Speech-to-Text call. |
| **SpeechService** | Helper class that provides access to the Speech-to-Text service. |
| **SpeechServiceResult** | Represents the result of the Speech-to-Text processing. |

# Walkthough: SpeechToTextSample Application

The Project Hawaii SDK includes the SpeechToTextSample application, which demonstrates the features of the STT service. The application is installed in the Samples\SpeechToText subfolder of the Hawaii SDK installation directory.

The sample application implements a simple interface that looks up the available grammars and lets a user record speech and then send it to the STT service for translation. Although the sample runs successfully on the Windows Phone emulator, its features are limited, because the emulator does not support voice input. The sample uses the microphone support from the Microsoft XNA framework.

This brief walkthrough describes how the sample uses the STT service.

To compile and run the sample:

1. In Visual Studio, open SpeechToTextSampleApp.csproj.

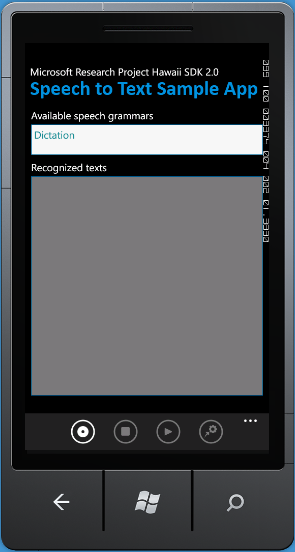
2. Open the HawaiiClient.cs file and set credential string(s) to your application credentials

3. Save the HawaiiClient.cs file.

4. Build the solution.

5. Run the sample with or without the debugger, as you prefer.

The following figure shows the initial emulator window for the sample:



To use the sample program

1. To record a speech sample, tap the **Record** button on the far left at the bottom of the screen. Speak into the microphone and then tap **Stop** (second button from left) to end the sample.

2. To play back the sample, tap **Play** (second button from right). Tap **Stop** to end playback.

3. To send the recorded audio to the STT service for recognition, tap the **Recognize** button on the far right.

## Querying for Grammars

As part of initialization, the sample application queries the server for a list of available grammars. The user can then select a grammar to use as the context in which to convert the speech to text. The **SpeechService.GetGrammarsAsync** method returns the list of grammars; the sample calls it from the MainPage.xaml.cs file, as follows:

SpeechService.GetGrammarsAsync(

HawaiiClient.HawaiiApplicationId,

(result) =>

{

this.Dispatcher.BeginInvoke(() =>

this.OnSpeechGrammarsReceived(result));

});

The method has the following parameters:

* The Hawaii Application ID, which the sample stores in the HawaiiClient object.
* A callback function that the STT service calls when the **GetGrammarsAsync** method completes.

The sample passes the callback as an inline delegate because the OnSpeechGrammarsReceived method displays the list of grammars in the user interface (UI). In Silverlight, you can access UI elements only on the main UI thread, but the STT service by default invokes the callback on a worker thread. By using the **Dispatcher.BeginInvoke** method, the sample ensures that the callback function executes on the main thread.

The following shows the code for the callback function:

private void OnSpeechGrammarsReceived(SpeechServiceResult result)

{

Debug.Assert(result != null, "result is null");

this.RecognizingProgress.Visibility = Visibility.Collapsed;

this.RetrievingGrammarsLabel.Visibility = Visibility.Collapsed;

if (result.Status == Status.Success)

{

this.SetButtonStates(true, false, false, false);

this.SpeechDomainsList.Visibility = Visibility.Visible;

this.availableGrammars = result.SpeechResult.Items;

if (this.availableGrammars == null)

{

return;

}

this.SpeechDomainsList.Items.Clear();

if (this.availableGrammars != null)

{

this.availableGrammars.ForEach((item) =>

this.SpeechDomainsList.Items.Add(item));

}

}

else

{

MessageBox.Show("Error receiving available speech grammars.",

"Error", MessageBoxButton.OK);

this.NoGrammarsLabel.Visibility = Visibility.Visible;

}

}

The STT service returns the list of grammars in the **SpeechServiceResult.SpeechResult.Items** property. If the list is not null, the callback function adds each item to the **Available speech grammars** list in the UI.

## Converting Speech to Text

When the user taps **Recognize**, the sample sends the contents of the current audio stream to the STT service for processing. A grammar must be available. The following shows the code from MainPage.xaml.cs that calls the STT:

private void RecognizeButton\_Click(object sender, EventArgs e)

{

if (this.availableGrammars == null ||

this.availableGrammars.Count == 0)

{

return;

}

this.RecognizingProgress.Visibility = Visibility.Visible;

if (this.AudioStream != null && this.AudioStream.Length != 0)

{

SpeechService.RecognizeSpeechAsync(

HawaiiClient.HawaiiApplicationId,

"Dictation",

this.AudioStream.ToArray(),

(result) =>

{

this.Dispatcher.BeginInvoke(() =>

this.OnSpeechRecognitionCompleted(result));

});

}

else

{

MessageBox.Show(

"Invalid speech buffer found. Record speech and try again.",

"Error", MessageBoxButton.OK);

}

}

The call to **SpeechService.RecognizeSpeechAsync** has the following parameters:

* The Hawaii Application ID, which the sample stores in the HawaiiClient object.
* A string that specifies the name of a grammar.
* A buffer that contains 10 seconds or less of audio data. The audio buffer should have the following characteristics:
* SamplesPerSecond=16000
* AudioBitsPerSample=16
* AudioChannel=Mono
* A callback function that the STT service calls when the **RecognizeSpeechAsync** method completes. Like the callback function for **GetGrammarsAsync**, this callback displays text in the UI, so it must run on the main UI thread.

The following shows the code for the OnSpeechRecognitionCompleted callback:

private void OnSpeechRecognitionCompleted(SpeechServiceResult speechResult)

{

Debug.Assert(speechResult != null, "speechResult is null");

this.RecognizingProgress.Visibility = Visibility.Collapsed;

if (speechResult.Status == Status.Success)

{

this.SetRecognizedTextListBox(speechResult.SpeechResult.Items);

}

else

{

if (speechResult.Exception == null)

{

MessageBox.Show("Error recognizing the speech.", "Error",

MessageBoxButton.OK);

}

else

{

MessageBox.Show(speechResult.Exception.Message, "Error",

MessageBoxButton.OK);

}

}

}

The **SpeechServiceResult.SpeechResult.Items** member contains the returned text. The **Items** member is a list of 10 strings, each of which represents a possible text string for the speech in the buffer. The strings are listed in descending order of their recognition confidence level; that is, the first string in the list has the highest confidence level. The confidence level is internal to the service; the application does not have access to this value.

# Using the STT Service in an Application

To use the STT service in your own application, you must:

* Add required assemblies to the Visual Studio project.
* Reference the namespace in your source code.
* Set up your authentication credentials.

## Add Required Assemblies

Applications that use the STT service depend on the following libraries, which are built as part of the Project Hawaii SDK:

* Microsoft.Hawaii.ClientBase.dll
* Microsoft.Hawaii.Speech.Client.dll

To add the libraries to your application

* Build the Hawaii SDK, as described in “Getting Started with the Project Hawaii SDK.”
* Add references to the following DLLs to your Visual Studio project:
* Microsoft.Hawaii.ClientBase.dll
* Microsoft.Hawaii.Speech.Client.dll

## Reference the Namespace

The STT client library service is defined in the **Microsoft.Hawaii.Speech.Client** namespace. For ease of reference, include the following in your code:

using Microsoft.Hawaii;

using Microsoft.Hawaii.Speech.Client;

## Set Up Your Authentication Credentials

* Your application authenticates itself with the service by using a Hawaii Application ID. If you do not already have a Hawaii Application ID, obtain one as described in “Getting Started with the Project Hawaii SDK.”
* The easiest way to use the Hawaii Application ID in your code is to copy the HawaiiClient.cs file from one of the sample applications, set the **HawaiiApplicationId** string to your Hawaii Application ID, and add the source file to your project. You can then use **HawaiiClient.HawaiiApplicationId** wherever the service requires the Application ID.

## Tips and Guidelines

The following guidelines apply to the STT service:

* Limit speech input to a maximum of 10 seconds. The STT service supports a maximum of 10 seconds of speech. Audio streams longer than this result in the error **Null/Invalid response object from server**.
* You may experience lower-quality results on STT services with the Dell Venue phone.
* Currently, English is the only supported language and Dictation is the only supported grammar.

# Resources

This section provides links to additional information about Project Hawaii and related topics.

Microsoft Research Project Hawaii

<http://research.microsoft.com/en-us/projects/hawaii/default.aspx>

Getting Started with the Project Hawaii SDK

<http://research.microsoft.com/en-US/projects/hawaii/docs.aspx>

Microsoft Research Project Hawaii on Facebook

<http://www.facebook.com/pages/Microsoft-Research-Project-Hawaii/164295863611699>

MSDN

Programming Windows Phone 7  
<http://blogs.msdn.com/b/microsoft_press/archive/2010/10/28/free-ebook-programming-windows-phone-7-by-charles-petzold.aspx>

How to: Create Your First Silverlight Application for Windows Phone  
<http://msdn.microsoft.com/library/ff402526(v=VS.92).aspx>

How to: Create Your First XNA Framework Application for Windows Phone  
<http://msdn.microsoft.com/en-us/library/ff472340(v=vs.92).aspx>