

Speech Using C#

Copyright © Mar 2005 Emant Pte Ltd
www.emant.com

Table of Contents

Introduction.....	3
Check out your system.....	3
Analyse Source Code.....	4
SpVoice Class.....	4
Using SpeechLib;.....	5
Referencing Assemblies.....	5
Program 1.1 Voice Source Code.....	5
Create the SpVoice object.....	5
Speak method.....	5
SpeechVoiceSpeakFlags Enum.....	6
WaitUntilDone method.....	6
Speech Class.....	6
Using the Speech Control.....	7
AddItem Method.....	8
RemoveItem Method.....	8
SpeechEnabled Property.....	8
Text Property.....	8
Recognized Event.....	8
Install Speech Control.....	9

Introduction

This paper describes object-oriented access to the speech recognition and text-to-speech capabilities of SAPI.

The Microsoft® Speech SDK 5.1 is the developer kit for the Microsoft® Windows environment. Tools, information, and sample engines and applications are provided to help you integrate and optimize your speech recognition and speech synthesis engines with the new Microsoft Speech API 5 (SAPI 5). The Speech SDK also includes updated releases of the Microsoft advanced speech recognition engine and Microsoft concatenated speech synthesis engine.

The SDK and information can be downloaded here.

<http://www.microsoft.com/speech/download/sdk51/>

Microsoft Visual Studio .NET 2003 must already be installed. Copy the entire Speech folder to your PC.

The program has been tested only on a Windows XP computer. Please refer to the SDK for information on the other Windows operating systems.

Check out your system

1. From the Speech folder, browse to `Testspeech\bin\Debug`. Run `Testspeech.exe` by clicking on the program.

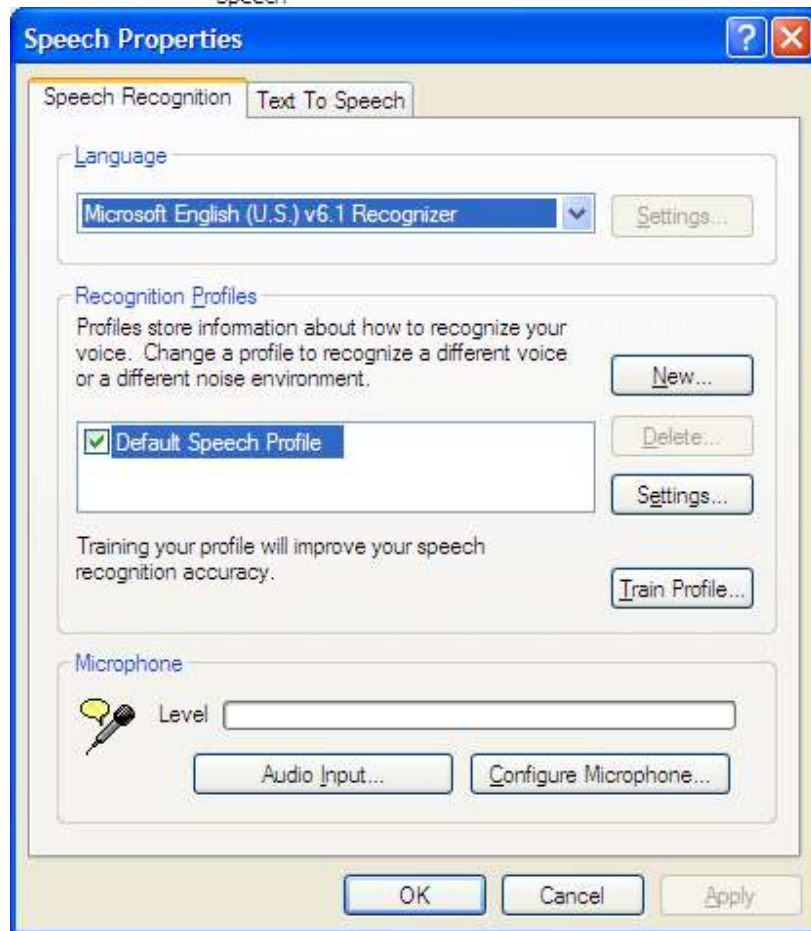


2. Click on the `Speak` button to read the text in the textbox `Hello World`. If you cannot hear the words, check on your audio setting. Goto the control panel and configure the Sound and Audio Devices



3. Enable Speech Recognition by clicking the Check Box.
4. Speak into the microphone `Select Singapore` , the list should highlight `Singapore`
5. Say `Select Malaysia`, the list should highlight `Malaysia`

6. If the words are not recognised, end the Testspeech program, goto the Control Panel and click on the Speech Icon



7. Perform the following steps (and follow the instructions given in the dialog windows that are opened)
8. Click on Audio Input
9. Click on Configure Microphone
10. Click on Train Profile.
11. Rerun Testspeech.exe
12. Enable Speech Recognition by clicking the Check Box.
13. Speak into the microphone **Select Singapore** , the list should highlight Singapore
14. Say **Select Malaysia**, the list should highlight Malaysia
15. If the words are recognised, we can now analyse the C# program.

Analyse Source Code

Goto the Testspeech folder and start the IDE by clicking on the TestSpeech solution.

SpVoice Class

The SpVoice object brings the text-to-speech (TTS) engine capabilities to applications using SAPI automation. An application can create numerous SpVoice objects, each independent of and

capable of interacting with the others. An `SpVoice` object, usually referred to simply as a voice, is created with default property settings so that it is ready to speak immediately.

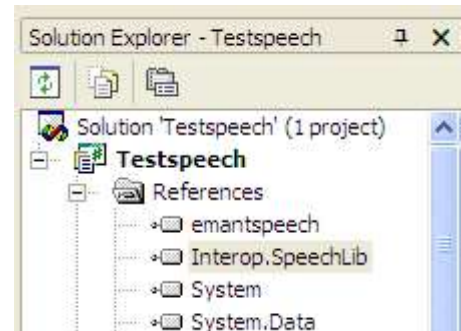
Using SpeechLib;

```
using SpeechLib;
```

Allow unqualified reference to `SpVoice`. If you don't include this using directive, all references to the `SpVoice` object will have to full

Referencing Assemblies

Every project contains a **References** folder for identifying physical assemblies the code in the project uses. In order to use the `SpVoice` class, the program must reference the assembly `Interop.SpeechLib.dll`.



Program 1.1 Voice Source Code

```
SpVoice Voice = new SpVoice();  
Voice.Speak(textBox1.Text, SpeechVoiceSpeakFlags.SVSFlagsAsync);  
Voice.WaitUntilDone( -1 );
```

Create the SpVoice object

```
SpVoice Voice = new SpVoice();
```

An instance of `SpVoice` is created and called `Voice`. All future references to this object will use this name. Full description of the `SpVoice` Class can be found from the SDK website

<http://download.microsoft.com/download/speechSDK/SDK/5.1/WXP/EN-US/sapi.chm>

Speak method

```
Voice.Speak(textBox1.Text, SpeechVoiceSpeakFlags.SVSFlagsAsync);
```

The `Speak` method places a text stream in the TTS engine's input queue and returns a stream number. It can be called synchronously or asynchronously. When called synchronously, the `Speak` method does not return until the text has been spoken; when called asynchronously, it returns immediately, and the voice speaks as a background process.

When synchronous speech is used in an application, the application's execution is blocked while the voice speaks, and the user is effectively locked out. This may be acceptable for simple applications, or those with no graphical user interface (GUI), but when sophisticated user interaction is intended, asynchronous speaking will generally be more appropriate.

int SpVoice.Speak(String Text, [Flags As SpeechVoiceSpeakFlags = SVSFDefault])

Parameters

Text - The text to be spoken

Flags - Default value is `SVSFDefault`.

Return Value

An `int` variable containing the stream number. When a voice enqueues more than one stream by speaking asynchronously, the stream number is necessary to associate events with the appropriate stream.

SpeechVoiceSpeakFlags Enum

The `SpeechVoiceSpeakFlags` enumeration lists flags that control the `SpVoice.Speak` method.

Enum `SpeechVoiceSpeakFlags`

```
'SpVoice flags
SVSFDefault = 0
SVSFlagsAsync = 1
SVSFPurgeBeforeSpeak = 2
SVSFIIsFilename = 4
SVSFIIsXML = 8
SVSFIIsNotXML = 16
SVSFPersistXML = 32
```

```
'Normalizer flags
SVSFNLPSpeakPunc = 64
```

```
'Masks
SVSFNLPMask = 64
SVSFVoiceMask = 127
SVSFUnusedFlags = -128
```

End Enum

Description of some of the Elements (see SDK for full description)

SVSFDefault

Specifies that the default settings should be used. The defaults are:

- To speak the given text string synchronously (override with `SVSFlagsAsync`),
- Not to purge pending speak requests (override with `SVSFPurgeBeforeSpeak`),
- To parse the text as XML only if the first character is a left-angle-bracket (override with `SVSFIIsXML` or `SVSFIIsNotXML`),
- Not to persist global XML state changes across speak calls (override with `SVSFPersistXML`), and
- Not to expand punctuation characters into words (override with `SVSFNLPSpeakPunc`).

SVSFlagsAsync

Specifies that the `Speak` call should be asynchronous. That is, it will return immediately after the speak request is queued.

WaitUntilDone method

```
Voice.WaitUntilDone( 2000 );
```

The `WaitUntilDone` method is used to block an application's forward progress while allowing user interaction with the mouse or keyboard.

bool SpVoice.WaitUntilDone(int msTimeout)

Parameters

`msTimeout` - Specifies the timeout in milliseconds. If -1, the time interval is ignored and the method simply waits for the voice to finish speaking.

Return Value

A Boolean variable indicating which case terminated the call. If true, the voice finished speaking; if false, the time interval elapsed.

Speech Class

The `Speech` control is built on the `Speech List Box for C#` from the SDK.

`Speech List Box for C#` is an elementary application showcasing speech recognition (SR) and dynamic grammars. The source code for the `Speech List Box for C#` is available from the SDK.

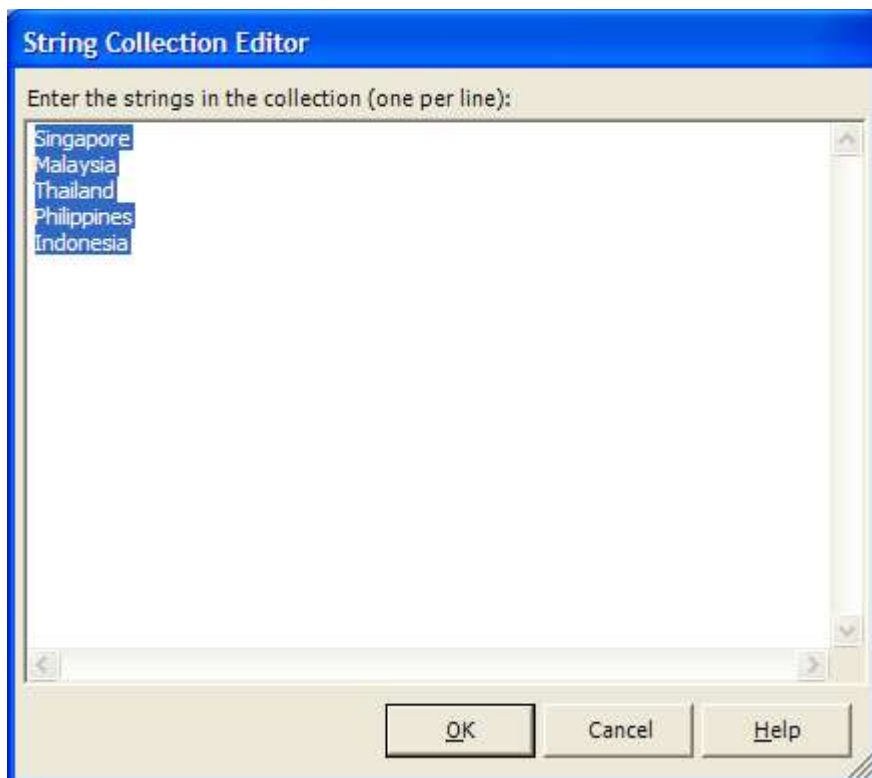
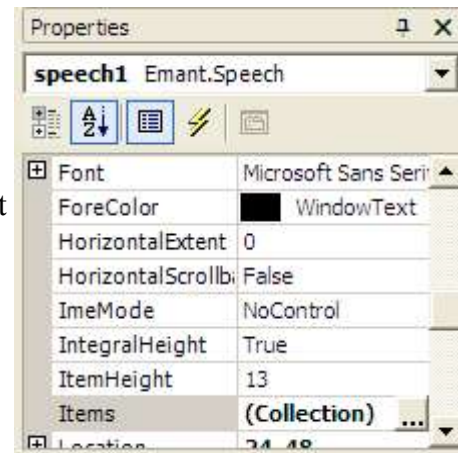
In general, a grammar is the set of words that the engine can recognize. In the case of dictation, all words are in the grammar and no limitation is imposed on the user. In the case of command and control applications, the list of words is much more restricted.

For example, a grammar containing commands to operate a menu system, might include less than a dozen words, each word corresponding to a menu or menu item. Often, this list of words is generated ahead of time and represents a static or fixed grammar. However, dynamic grammars allow users to add words while running the application. This permits users to customize the grammar according to their needs. The list box sample demonstrated making and maintaining a dynamic grammar.

Using the Speech Control

Speech Control is a modified List Box and it contains no words in the dynamic grammar. You can add words or phrases at the design phase by selecting **Properties>Items**.

Enter the text in the string collection editor. After adding at least one item, individual items may be highlighted by saying, "select" followed by the text of the item. For example, if one of the items a list of countries were "Singapore" saying "Select Singapore" will choose the Singapore item.



AddItem Method

Use AddItem to move the text in the edit box to the display area. The text can be either a single word or a phrase.

```
speech1.AddItem(txtNewItem.Text);
```

```
public int AddItem(object newItem)
```

Parameters

newItem - The new item to be added to the list

Return Value

The index of the newly added item.

RemoveItem Method

Use RemoveItem to remove the selected item(s) in the display area from the dynamic grammar.

```
speech1.RemoveItem(speech1.SelectedIndex);
```

```
public void RemoveItem(int index)
```

Parameters

Index of the item to be removed from the list

SpeechEnabled Property

Use SpeechEnabled to enable the speech recognizer. By default it is on and the speech engine will attempt to recognize voice commands. Disabling this option prohibits recognition attempts. Regardless of the recognizer state, if no items have been added to the display area, no recognition will occur.

```
speech1.SpeechEnabled = checkBox1.Checked;
```

```
public boolean SpeechEnabled {get; set;}
```

Property Value

true to recognize, false otherwise

Text Property

The recognized words are assigned to the Text property. Gets the string representing the recognized phrase or word.

```
public string Text {get;}
```

Property Value

string of the recognized words.

Recognized Event

The event is fired whenever words in the grammar are recognized.

```
private void speech1_Recognized(object sender, System.EventArgs e)
{
    label1.Text = speech1.Text;
}
```


Install Speech Control

To integrate the **Speech Control** in Visual Studio .NET:

1. Select the **General** tab in the **Toolbox**
2. Right-click on the **Toolbox** background
3. Select **Add/Remove Items...**
4. On the **Customize Toolbox Dialog**, select the **NET Framework Components** tab
5. Press the **Browse** Button
6. Locate and select the `emantspeech.dll` assembly in the `Testspeech` folder
7. Click **OK**

