Voice Commands in Windows Phone 8

This article explains how to use voice commands in an application to perform actions, tasks or any initiative work in Windows Phone 8

Introduction

Windows Phone 7.5 had minimal speech functionality. In Windows Phone 8 gives developers a lot more options, including:

- Inbuilt Text-to-speech (TTS) [offline]
- Improvised Speech-to-Text (STT) [online processing for quality] and
- Registration of applications with OS for opening them with voice and in-app voice commands

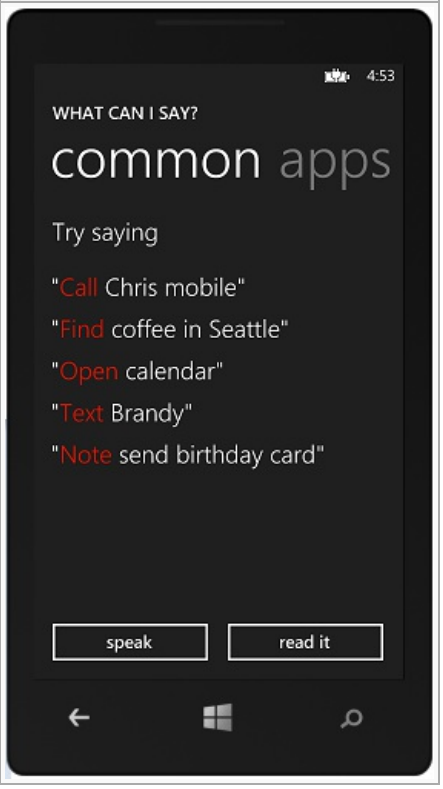
Windows Phone 8 speech functionality includes:

1. voice commands
2. speech recognition
3. text-to-speech, means now developer is able to add above mentioned functionality in his app.

Using voice command functionality user is able to make his application responsible to respond voice commands.

With the help of phrase link to the specific application page application can perform specific task or any initiative work. You can simply check voice command functionality in your windows phone 7.5 and above by just holding start key.

Refer following screen shots

Basic voice command listening UI	Basic what can i say UI
	

How to achieve

The following steps need to be performed in order to add voice command functionality in an application:

1. Create voice command file
2. Install voice command file in OS
3. Write code to handle navigation handling and command execution.

Capabilities required

(C) Copyright Nokia 2014. All rights reserved.

```
ID_CAP_NETWORKING
ID_CAP_SPEECH_RECOGNITION
ID_CAP_MICROPHONE
```

To add capabilities in your project

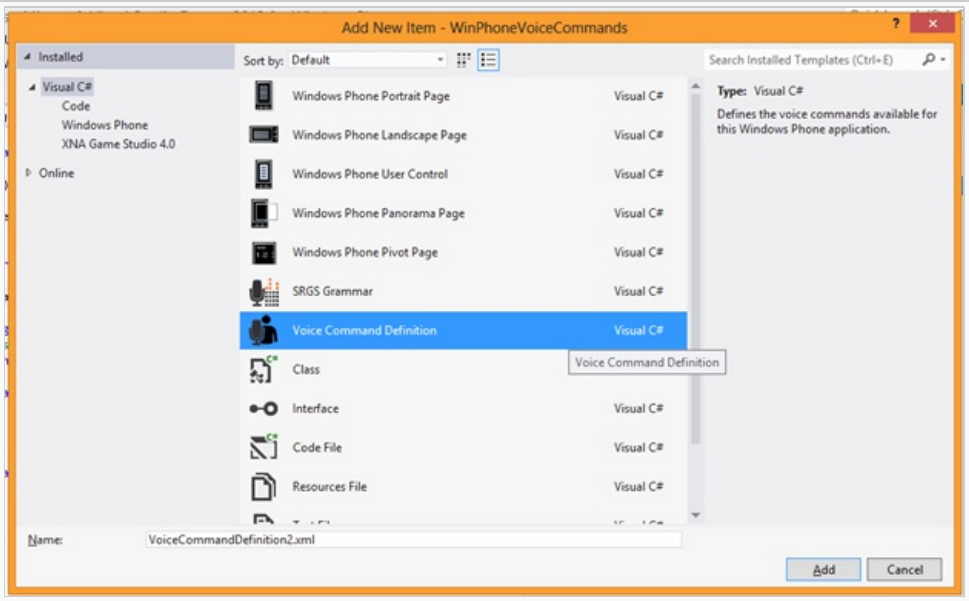
1. Go to your project from Solution Explorer and open WMAppManifest.xml file from Properties section.
2. Click Capabilities tab and check above mentioned capabilities and save the file.

Create voice command definition file

The following steps describe how to create Voice command definition file:

1. Right click the windows phone project in visual studio
2. Go to Add New Item
3. Select Voice Command Definition from list
4. Give proper name to the VCD file.

Refer following screen shot.



Add new VCD file in project

By default visual studio provides basic template of VCD file which you need to update with your own voice commands and their intended actions. Typical voice command file contains

1. Example phrase to show how user can invoke command
2. The words or phrase that can be recognized to invoke command
3. Action taken after the particular command is invoked with the help of page navigation.
4. The text to display or speak to respond user after command recognized(note it uses TTS functionality to speak text)

Voice Command Definition file in detail

A typical Voice Command is made up of 3 parts:

1. Application Name (not required for in-app commands)
2. Command
3. Phrase

Let us consider the following example of VCD file to understand the voice command in detail:

```
<?xml version="1.0" encoding="utf-8"?>

<VoiceCommands xmlns="http://schemas.microsoft.com/voicecommands/1.0">
  <CommandSet xml:lang="en-US" Name="MyVoiceCommands">
    <CommandPrefix>hello</CommandPrefix>
    <Example>hello what is current date</Example>
```


(C) Copyright Nokia 2014. All rights reserved.

```
    <Command Name="Util">
      <Example> What is current time </Example>
      <ListenFor> what [is] current {options}</ListenFor>
      <Feedback>showing current {options}</Feedback>
      <Navigate Target="Target.xaml"/>
    </Command>

    <Command Name="Calculator">
      <Example> 2 plus 3 </Example>
      <ListenFor>{operand1} {operation} {operand2} </ListenFor>
      <Feedback> Showing {operand1} {operation} {operand2}</Feedback>
      <Navigate Target="Target.xaml"/>
    </Command>


    <PhraseList Label="options">
      <Item>time</Item>
      <Item>date</Item>
      <Item>battery percentage</Item>
    </PhraseList>

    <PhraseList Label="operation">
      <Item>plus</Item>
      <Item>minus</Item>
    </PhraseList>

    <PhraseList Label="operand1">
      <Item>1</Item>
      <Item>2</Item>
    </PhraseList>

    <PhraseList Label="operand2">
      <Item>1</Item>
      <Item>2</Item>
    </PhraseList>

  </CommandSet>
</VoiceCommands>
```

VoiceCommands: This is a root element of voice command definition file which can contain one or more CommandSet elements each of which contains voice commands.

CommandSet: This element serves as container for all voice command. User can define actual voice commands under it for a single language defined for xml:lang attribute of CommandSet. User can also specify name for CommandSet element but it is optional. But it becomes useful when user wants to modify PhraseList pragmatically as you can see this in **Modify Voice Command Definition file** section of this article.

CommandPrefix: This is a child element of VoiceCommands which can be optional, but if present, must be the first child element of CommandSet. This element is used for giving name for application which user can speak in order to give voice command hence it is useful for the applications which have long and difficult names to pronounce. In above example, command prefix is “hello” that means if user wants to give command to this application then he must use “hello” word at the start of any command.

Command: This is compulsory child element for CommandSet Element. This element defines action which needs to be taken after receiving voice command. Each command corresponds to specific page in your application, you can have multiple commands.

Example: This is compulsory child element of command which can be used to help user for what he can say as in “What can I Say” screen. You can view this in supported command list of application, please refer following screen shot, it shows Did you

(C) Copyright Nokia 2014. All rights reserved.

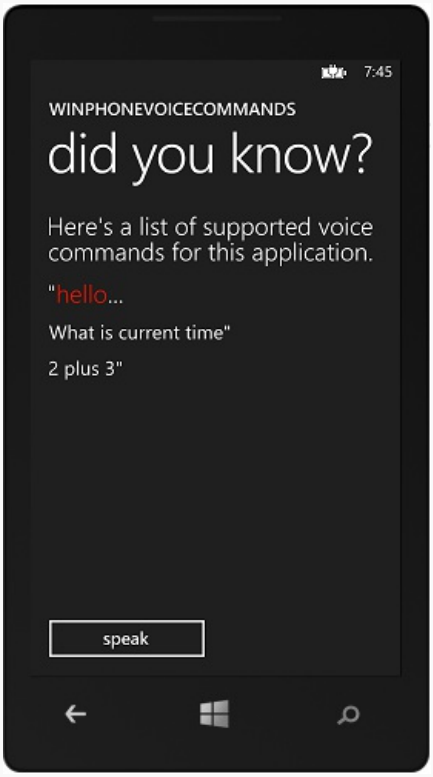
know? screen after successfully registering above vcd file, this screen clearly shows example which you mentioned in VCD file.
To get "did you know?" screen of your application.

1. Make sure that VCD file of your application is successfully installed, Hold start key until "Listening" popup appears.
2. Now click question mark icon on "Listening" popup it will open "WHAT CAN I SAY?" pivot item.
3. Now go to the "apps" section where you can see list of applications which supports voice commands, clicking on application you want leads to "did you know?" screen.

You can launch "WHAT CAN I SAY?" by another way also

1. Hold start button
2. Speak "what can I say?".

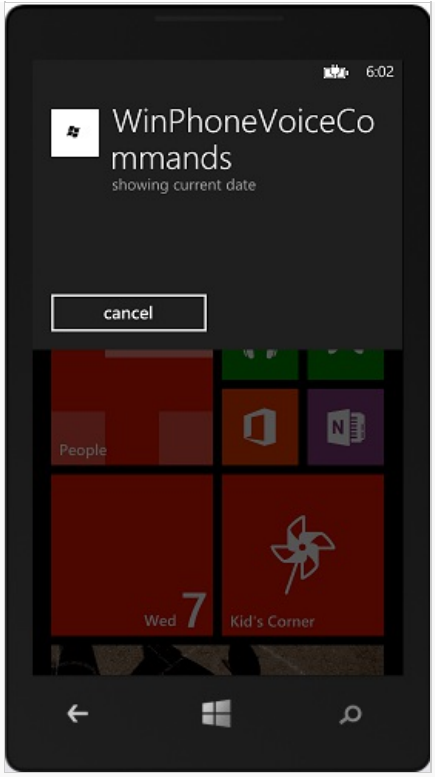
Refer the following screen shots.

WHAT CAN I SAY?	did you know?
	

Listenfor: Compulsory child element of Command element, which defines text that Application recognizes for command. As shown in above VCD file you can use curly and square braces these are special characters. Curly braces represents Phrase list and square braces represent optional text means user does not have any compulsion for speaking text defined in square bracket. Single ListenFor element may refer to one or more PhraseList as shown in above example of vcd file Calculator command refers to three PhraseList namely operand1, operation, operand2 whereas Util command refers single PhraseList namely options.
You can also achieve wildcard functionality by embedding "*" in ListenFor tag.
for eg.`<ListenFor>what [is] current {*}</ListenFor>`, here the voice command gets recognized as long as user speaks 'what is current' followed by any phrase or word.
Note you cant nest the special characters.

Feedback: Compulsory child element for Command element, it contains text which is displayed and spoken using text to speech when command is recognized. Feedback element can contain reference to the PhraseList element but in that case every ListenFor element in the containing Command element must also reference to the same PhraseList element.
Refer following screen shot, which shows feedback element after command gets recognized.

(C) Copyright Nokia 2014. All rights reserved.



FeedBack UI

Navigate: Compulsory child element for Command element, The Target attribute defined contains page which loads after command is recognized. If you omit Target element then app launches with default Mainpage, you can also specify query string for the page. Eg. <Navigate Target="Target.xaml?key=value"/>

PhraseList: Optional child of the CommandSet element ,It require Label value which Listenfor, Feedback element use to reference the PhraseList means those which are enclosed in curly braces. It contains Item elements each of which defines word that can be used to initiate command that references PhraseList.

Item: Optional child element of CommandSet element. It contains one or more words that can be recognized to initiate voice command.

Installing/Registering voice command definition file

Installing VCD file registers voice commands defined in the VCD file to the speech system of phone, You should install the VCD file during the first run of your application. In case your application re installs due to Phone Backup no data related to your voice commands is preserved so it is better practice to install VCD file during each time your application launches.
You can use `InstalledCommandSets` property of `VoiceCommandService` class to check existence of your voice commands.

```
private async void RegisterVoiceCommands()
{
     await VoiceCommandService.InstallCommandSetsFromFileAsync( new Uri("ms-
appx:///vcd.xml", UriKind.RelativeOrAbsolute));
}
```

When voice command for app is recognized the target page gets launched and URI for that page contains recognized commands in the form of QueryString.

You can use `OnNavaigatedTo()` method to extract parameter from QueryString using NavigationContext.

If the page is launched from voice command then Query String contains `voiceCommandName` parameter in it.

`voiceCommandName` parameter is useful in case when more than one commands targets same page, in this case `voiceCommandName` is used to figure out which command user launched.

Handling navigation

When application recognizes voice command then target page is launched and recognized parameters are included in query

(C) Copyright Nokia 2014. All rights reserved.

string.

OnNavigatedTo() method in target page is used to extract parameters, if application is launched from voice command then a parameter named voiceCommandName exist in query string so checking its existence is better way to identify whether page is launched from voice command.

Suppose user says “hello what is current date” then the query string for the same is:

```
[0]={[voiceCommandName,Util]}
[1]={[reco,hello what is current date]}
[2]={[options,date]}
```

And suppose user says “hello two plus two” then query string is

```
[0]={[voiceCommandName,Calculator]}
[1]={[reco,hello 2 plus 2]}
[2]={[operand2,2]}
[3]={[operation,plus]}
[4]={[operand1,2]}
```

`voiceCommandName` parameter contains command name.

`reco` parameter contains whole text which is expected command .

`options`, `operand2`, `operation`, `operand1` contain recognized words from phrase list.

Refer following code snippet to know how to retrieve parameters from query string for above mentioned vcd file.

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
            if (NavigationContext.QueryString.ContainsKey("voiceCommandName"))
            {
                string voicecommandname =
NavigationContext.QueryString["voiceCommandName"];
                switch (voicecommandname)
                {
                    case "Util":
                        string options = NavigationContext.QueryString["options"];
                        //do something..refer attached source code for complete
implementation
                        break;

                    case "Calculator":
                        string number1 = NavigationContext.QueryString["operand1"];
                        string number2 = NavigationContext.QueryString["operand2"];
                        string oprator = NavigationContext.QueryString["operation"];
                        try
                        {
                            //do something..refer attached source code for complete
implementation
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine(ex.Message+"\n"+ex.StackTrace);
                        }
                        break;

                    default:
                        break;
```


(C) Copyright Nokia 2014. All rights reserved.

```
                }
            }
}
```

Note switch case is used because application may have number of commands which targets same page, so it become useful to know which command user has given. In each case i mentioned do something but you can refer attached example which covers proper implementation of each voice command like showing date adding two numbers, showing current battery percentage etc.

Modify Voice Command Definition file

In most of the cases we want PhraseList's content to be populated at run time using webservice or some other source, To achieve this functionality SDK gives you ability to modify voice command definition file Below is code snippets showing how to modify VCD file.

Suppose we want to add more items to `operand1` and `operand2`, so we just called `ModifyPhraseList` function twice from `Button` click event. refer the following code

```
private void Button_Click_1(object sender, RoutedEventArgs e)
{
            ModifyPhraseList("MyVoiceCommands", "operand1");
            ModifyPhraseList("MyVoiceCommands", "operand2");
}
private async void ModifyPhraseList(string commandsetname,string phraselistname)
{
            string[] numbers = new string[]{"1","2","3","4","5","6","7","8","9","10"};
            var vcs = VoiceCommandService.InstalledCommandSets[commandsetname];
            await vcs.UpdatePhraseListAsync(phraselistname, numbers);
}
```

Sample Voice Commands

Following is list of some commands which user can invoke using above VCD file, make sure that before invoking below voice commands the voice commands from VCD file must be install/register with OS.
To start recognizer hold start key from within the application or from outside the application.

1. hello what is current date
2. hello what is current battery percentage.
3. hello two plus two.
4. hello two minus two.

Command	KeyWord	Specification
hello what is current date	hello	CommandPrefix
	current date	PhraseList “options”
hello two plus two	hello	CommandPrefix
	two	PhraseList “operand1”
	plus	PhraseList “operation”
	two	PhraseList “operand2”

Localization of voice commands

You can specify commands in voice command definition file for various languages.
You can create multiple `CommandSets` and set `xml:lang` to the language you want, to test application for particular language make sure that `CommandSet` language must match with language selected on phone.

How to change language on emulator/device.

(C) Copyright Nokia 2014. All rights reserved.

1. Go to the “SETTINGS”
2. Select “Language+Region” , Now set the language you want for eg espanol for Spanish and restart emulator.

Suppose you want to support above commands for Spanish language also then you have to define same `CommandSet` for Spanish language.
Now newly created VCD file will look like as follow.

```
<?xml version="1.0" encoding="utf-8"?>
<VoiceCommands xmlns="http://schemas.microsoft.com/voicecommands/1.0">
  <CommandSet xml:lang="en-US" Name="MyVoiceCommands">
    <CommandPrefix>hello</CommandPrefix>
    <Example>play what is current date</Example>

    <Command Name="Util">
      <Example> What is current time </Example>
      <ListenFor> what [is] current {options}</ListenFor>
      <Feedback>showing current {options}</Feedback>
      <Navigate Target="Target.xaml"/>
    </Command>

    <Command Name="Calculator">
      <Example> 2 plus 3 </Example>
      <ListenFor>{operand1} {operation} {operand2} </ListenFor>
      <Feedback> Showing {operand1} {operation} {operand2}</Feedback>
      <Navigate Target="Target.xaml"/>
    </Command>

    <PhraseList Label="options">
      <Item>time</Item>
      <Item>date</Item>
      <Item>battery percentage</Item>
    </PhraseList>

    <PhraseList Label="operation">
      <Item>plus</Item>
      <Item>minus</Item>
    </PhraseList>

    <PhraseList Label="operand1">
      <Item>1</Item>
      <Item>2</Item>
    </PhraseList>

    <PhraseList Label="operand2">
      <Item>1</Item>
      <Item>2</Item>
    </PhraseList>
   </CommandSet>

  <CommandSet xml:lang="es-ES" Name="MyVoiceCommands">
    <CommandPrefix>¡hola</CommandPrefix>
    <Example>jugar a lo que es la fecha actual</Example>

    <Command Name="Util">
      <Example>¿qué es la hora actual</Example>
```


(C) Copyright Nokia 2014. All rights reserved.

```
      <ListenFor>¿qué [es] corriente {options}</ListenFor>
      <Feedback>mostrando actual {options}</Feedback>
      <Navigate Target="Target.xaml"/>
    </Command>

    <Command Name="Calculator">
      <Example> 2 más 3 </Example>
      <ListenFor>{operand1} {operation} {operand2} </ListenFor>
      <Feedback> demostración {operand1} {operation} {operand2}</Feedback>
      <Navigate Target="Target.xaml"/>
    </Command>

    <PhraseList Label="options">
      <Item>tiempo</Item>
      <Item>fecha</Item>
      <Item>batería porcentaje</Item>
    </PhraseList>

    <PhraseList Label="operation">
      <Item>más</Item>
      <Item>menos</Item>
    </PhraseList>

    <PhraseList Label="operand1">
      <Item>1</Item>  <!--in spanish it is pronounced as "uno"-->
      <Item>2</Item>  <!--in spanish it is pronounced as "dos"-->
    </PhraseList>

    <PhraseList Label="operand2">
      <Item>1</Item>  <!--in spanish it is pronounced as "uno"-->
      <Item>2</Item>  <!--in spanish it is pronounced as "dos"-->
    </PhraseList>

  </CommandSet>
</VoiceCommands>
```

Note everything in new `CommandSet` is same except localized text. There is no major implementation required for this newly added language, you just need to handle new `PharaseList` item. For eg. If user gives command “1 más 1” you have to just do the same operation as you done for “1 plus 1” for more information refer source code of attached example.

Example

Find the complete working example here File:WinPhoneVoiceCommands.zip

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.