OCR Service Programming Guide

Version 2.1 – September 13, 2012

Abstract

The Project Hawaii Optical Character Recognition (OCR) service enables a mobile application to retrieve the text that appears in a photograph. This document describes the OCR service and walks you through a Windows Phone application that uses it.

Contents

[Introduction 2](#_Toc335307159)

[Prerequisites 2](#_Toc335307160)

[Overview of the OCR Client Library 2](#_Toc335307161)

[OCR Service Samples 3](#_Toc335307162)

[OcrSample Program 4](#_Toc335307163)

[OcrSampleLite Program 4](#_Toc335307164)

[Using the OCR Service in an Application 6](#_Toc335307165)

[Include the Required Libraries 6](#_Toc335307166)

[Reference the Namespace 6](#_Toc335307167)

[Use Your Credentials 6](#_Toc335307168)

[Initiate the OCR Request 6](#_Toc335307169)

[Implement the Callback Method 8](#_Toc335307170)

[Tips for Using the OCR Service 9](#_Toc335307171)

[Image Guidelines 9](#_Toc335307172)

[Default OCR Processing Actions 9](#_Toc335307173)

[Resources 9](#_Toc335307174)

**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, Windows Azure, and Windows are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.

# Introduction

The Hawaii OCR service takes a photographic image that contains some text and returns the text. For example, given a JPEG image of a road sign, the service returns the text of the sign. This document provides an introduction to the managed code interface for the Hawaii OCR service and walks you through a simple application that uses the service.

# Prerequisites

Before you can build an application that uses the OCR 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

# Overview of the OCR Client Library

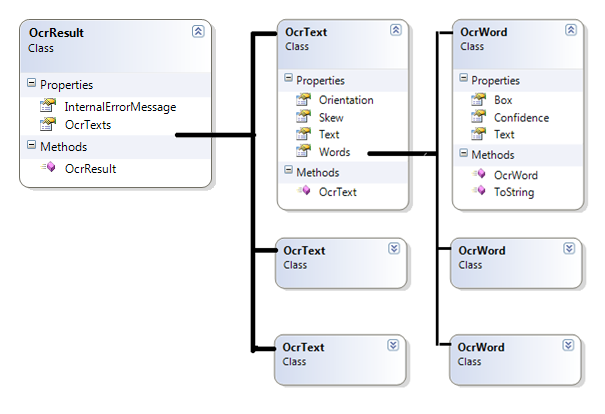
The OCR client library provides a managed interface for applications that communicate with the Hawaii OCR service. The source code for this library is installed with the Project Hawaii SDK in the following folder:

* Source\ServiceClients\Ocr

The library defines the following classes that a client application can use to request OCR processing and parse the results:

|  |  |
| --- | --- |
| Class | Description |
| **OcrResult** | Describes the result obtained from a Hawaii OCR call. The **InternalErrorMessage** and **OcrTexts** properties of this class supply an error message or the returned OCR data, respectively. |
| **OcrService** | Helper class that provides access to the OCR service. Applications call the **RecognizeImageAsync** method in this class to request OCR services. |
| **OcrServiceResult** | Represents the result of the OCR processing. Upon completion of OCR processing, the OCR service calls the application’s callback function and passes an instance of this class. The application then inspects the **OcrResult** member to determine whether the call succeeded. |
| **OcrText** | Contains one text item, its orientation and skew in the processed image. |
| **OcrWord** | Contains one word, its bounding box in the image, and a confidence value. |

The following class diagram shows the OCR service results and associated details.



# OCR Service Samples

The Project Hawaii SDK includes two sample programs that use the OCR service:

* OcrSample
* OcrSampleLite

The samples are installed in the Samples\Ocr subfolder of the Project Hawaii SDK installation folder.

To build the samples

1. In Visual Studio, open OcrSampleApp.csproj or OcrSampleAppLite.csproj.

2. Open the HawaiiClient.cs file and set the Hawaii credentials you have obtained.

3. Save the HawaiiClient.cs file.

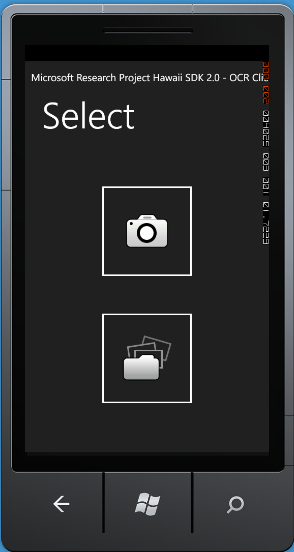
4. Build the solution.

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

## OcrSample Program

The OcrSample program can either pass an image from the camera or use an image that is already present on the device. For convenience of those who are using the Windows Phone emulator, the OcrSample program is installed with a sample photo that includes text.

The following shows the initial display for OcrSample:



To use OcrSample

* Tap the camera icon to take a picture to send to the application.

—OR—

* Tap the file icon to see a list of pictures.
* Tap Saved Pictures to use an existing picture.
* Tap a picture to send the image to the OCR service.

## OcrSampleLite Program

The OcrSampleLite program passes an image stream from the camera in the mobile device to the OCR service. The following shows the initial display for this sample:



To use OcrSampleLite

* Tap the camera icon to take a picture to send to the application.
* Take the photograph.
* Tap Accept to send the image to the OCR service.

The following sections outline the actions that the OcrSampleLite performs.

### Initialization

When you start the application, the constructor for the App class establishes an exception handler, sets up profiling information for debugging, initializes Silverlight, and initializes the phone application. This code appears in App.xaml and its code-behind file.

### Image Generation

The sample then creates an instance of the **Microsoft.Phone.Tasks.CameraCaptureTask** class to enable a user of the application to take a photograph. When the user has chosen a photo, the application calls the internal StartOcrConversion method. This method determines the size of the image and scales it down if necessary. For more information about image size guidelines, see “Image Guidelines” later in this document.

StartOcrConversion then converts the photo stream to a byte array and calls the **OcrService.RecognizeImageAsync** method to request OCR processing. This code appears in MainPage.xaml and its code-behind file.

### Completion

The **OcrService.RecognizeImageAsync** method runs asynchronously. When it is complete, the OCR service calls the OnOcrCompleted callback and passes an instance of the **OcrServiceResult** class that contains the results of OCR processing. The sample application checks the returned status. If OCR processing completed successfully, the OnOcrCompleted callback parses the returned text from the **OcrResult.OcrTexts** field of the result and prepares to display the text. This code also appears in MainPage.xaml and its code-behind file.

# Using the OCR Service in an Application

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

* Include the required libraries.
* Supply your application credentials.
* Initiate the OCR request.
* Implement a callback method.

## Include the Required Libraries

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

* Microsoft.Hawaii.ClientBase.dll
* Microsoft.Hawaii.Ocr.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 project:
* Microsoft.Hawaii.ClientBase.dll
* Microsoft.Hawaii.Ocr.Client.dll

## Reference the Namespace

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

using Microsoft.Hawaii;

using Microsoft.Hawaii.Ocr.Client;

## Use Your Credentials

* Your application authenticates itself with the OCR 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.

## Initiate the OCR Request

To start the OCR process, an application calls **RecognizeImageAsync**. This static method is declared on the **OcrService** class. The following code snippet shows how to call this method:

OcrService.RecognizeImageAsync(

HawaiiClient.HawaiiApplicationId,

OcrClientUtils.GetPhotoBits(this.ocrData.PhotoStream),

OnOcrCompleted);

The method takes the following parameters:

* *hawaiiAppId, a string that specifies the Hawaii Application ID, as described in the previous section*
* *photoBuffer*, a byte array that contains a JPEG image. In the OcrSampleLite sample this is the content of an image as returned by **CameraCaptureTask**.
* *OnOcrCompleted*, a delegate to a callback method that the service invokes when the asynchronous OCR process completes. The callback method has a return type of **void** and takes one parameter, which provides the result of the OCR conversion.

The call to **RecognizeImageAsync** returns immediately because the method runs asynchronously. Execution of the client application continues in parallel with execution of the asynchronous Hawaii OCR service call. When **RecognizeImageAsync** completes, the OCR client library calls the *OnOcrCompleted* callback on a worker thread.

In Silverlight, you can access user interface (UI) elements only on the main UI thread. If your *OnOcrCompleted* method directly or indirectly sets elements of the UI, you must ensure that this method is executed in the main UI thread. One simple solution is to use the **Dispatcher.BeginInvoke** method, which executes the delegate on the current thread. The following excerpt from the sample code in OcrSampleLite shows how to use this method:

private void StartOcrConversion()

{

OcrService.RecognizeImageAsync(

HawaiiClient.HawaiiApplicationId,

OcrClientUtils.GetPhotoBits(this.ocrData.PhotoStream),

(result) =>

{

// This section defines the body of an anonymous method.

this.Dispatcher.BeginInvoke(() => OnOcrCompleted(result));

});

...

}

private void OnOcrCompleted(OcrServiceResult result)

{

...

}

The following syntax, which appears in the sample code, is a simple example of a lambda expression.

(result) => { statement ;}

Although this construction might seem confusing at first, it is a simple way to write an inline delegate. You can think of the content inside the braces as the body of a method. Such a statement is called an anonymous method because it lacks a declaration that provides a name. It is equivalent to the following code:

OcrService.RecognizeImageAsync(

HawaiiClient.HawaiiApplicationId,

OcrClientUtils.GetPhotoBits(this.ocrData.PhotoStream),

OnOcrCompletedDispatcher);

...

private void OnOcrCompletedDispatcher(OcrServiceResult result)

{

this.Dispatcher.BeginInvoke(() => OnOcrCompleted(result));

}

private void OnOcrCompleted(OcrServiceResult result)

{

...

}

## Implement the Callback Method

The *OnOcrCompleted* callback method must do the following:

1. Check whether the call completed successfully or returned an error.

2. If the call completed successfully, perform the appropriate processing, which might simply be to update the content of a text box in the UI.

3. If an error occurred, handle it appropriately. The simplest approach is to display an error message

The following code shows a simple implementation of the *OnOcrCompleted* method:

private void OnOcrCompleted(OcrServiceResult result)

{

if (result.Status == Status.Success)

{

// Use the response from the service. In our case the useful

// data is in result.OcrResult

}

else

{

// Display the error state.

}

}

If OCR processing succeeds, the result of the OCR process is returned in the parameter of the *OnOcrCompleted* callback. This parameter is an instance of type **OcrServiceResult**. The **OcrServiceResult.OcrResult** field contains the OCR information. The **OcrResult.OcrTexts** member of the result contains a list of **OcrText** elements, each of which corresponds to an individual section of text that the OCR service identified. Each **OcrText** item contains information such as **Orientation**, **Skew**, and a list of **Words**. Each word contains the text of the word, a confidence score that the OCR service generates, and the coordinates and size of the enclosing box. For more details on these classes and properties, see the Project Hawaii SDK help, which is installed with the SDK.

# Tips for Using the OCR Service

The following sections provide additional information that can help you use the OCR service successfully:

* Guidelines for images
* Default OCR processing actions

## Image Guidelines

The following constraints apply to images and the text within them.

* Maximum size for uploaded images is 1.5MB.
* Optimal image size is 640x480 pixels, at about 80% to 70% JPEG compression quality.
* Maximum text height is 100 pixels.
* Ideal text height is 30 pixels.
* Text that is less than 12 pixels high is unlikely to be recognized.
* Text should be crisp, printed text of approximately uniform color with a uniform background.
* Text should be horizontal or vertical but a small skew angle of -5 to 5 degrees is acceptable.

To work around text size limitations or to reduce the duration of the transmission, consider scaling down the image before sending it to the OCR service.

## Default OCR Processing Actions

By default, the OCR service performs the following actions, which can affect the results that it returns:

* Removes words for which the confidence value is 150 or less, which indicates low confidence in recognition.
* Removes invalid words. The service considers invalid words that include many non-alphanumeric characters.
* Removes invalid lines removed. The service considers a line invalid if it is composed entirely of words with two or fewer characters.
* Removes all non-alphanumeric characters from the start and end of words, except currency signs and punctuation marks that are valid at the beginning or end of a word or line.

# Resources

This section provides links to additional information about the Project Hawaii OCR Service 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

Dispatcher.BeginInvoke  
<http://msdn.microsoft.com/en-us/library/ms591206.aspx>

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

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>