Scannerfly™ SDK

Developer's Guide

Scannerfly™ SDK 1.0 Beta 1, December 2007

Version

Scannerfly SDK 1.0 Beta 1 Developer's Guide. Released December 2007.

Copyright Information

Trademark Information

License and Patent Information

Acknowledgements

- Special thanks to Ashley Richards (http://www.ashleyrichards.com) for proof-reading and editing this document

Limitations

Contents

1 Introduction

This introduction provides you with an overview of the purpose and organization of this developer’s guide.

In This Chapter

Purpose

This developer’s guide explains how to use the Scannerfly SDK, a software library for integrating image-based barcode recognition into Flash-based applications. The purpose of the developer’s guide is to show Flash developers:

- The organization and functionality of components of the Scannerfly SDK
- How to use the Scannerfly SDK to add barcode recognition to Flash applications, by presenting code fragments for a few of the most frequently needed operations
- How to build the sample applications accompanying the Scannerfly SDK to demonstrate its barcode recognition capabilities

Who Should Read This Guide?

Read this guide if you are a software developer using the Scannerfly SDK and Flex Builder 2 to add barcode recognition to your Flash-based application.

You will need to be an experienced ActionScript programmer to understand and use this material.

Document Overview

This guide is organized as follows:

- *Getting Started* provides an overview of the Scannerfly SDK, the supported platforms, the installation process, and the procedure for including the Scannerfly SDK library in your project settings.
- *The ScannerPanel Component* describes the main Scannerfly SDK component used to incorporate barcode recognition functionality into the user interface of an application.
- *Advanced Programming* details how to interact with the Scannerfly SDK directly via its API to include barcode recognition functionality in applications for which the `ScannerPanel` component is not suitable.
- *Sample Applications* presents simple example applications snippets that demonstrate the functionality of the Scannerfly SDK.

Related Documents

This guide is a companion to the API documentation accompanying the Scannerfly SDK, which documents the classes that comprise the Scannerfly SDK. The API documentation can be found in `docs` directory in the installation zipfile.

Comments and Feedback

The Scannerfly SDK development team appreciates your input. Please send any documentation comments to developer-guide@scannerfly.com.

Documentation Conventions

This developer’s guide uses Notes, Cautions, and Warnings to highlight important information in the following manner:

Note: Notes are extra, but important, information. A note calls your attention to important aspects of the product. You will be able to use the product better if you read the notes.

Caution: Cautions indicate where possible problems could occur unless precautions are taken. Pay attention to cautions.

Warning: Warnings indicate the possibility of malfunction. A warning means serious problems are will occur unless you take appropriate action. Please take warnings very seriously.

2 Getting Started

This chapter provides a high-level overview of the things you need to do to prepare to use the Scannerfly SDK.

In This Chapter

Overview

The Scannerfly SDK provides a library that Flash-based applications can use to perform barcode recognition. Using the library, Flash applications can include functionality to extract barcodes from images, typically captured from a webcam connected to the user's computer.

Barcodes are encoded using a symbology which performs mapping between data values and their visual representation. The Scannerfly SDK supports three of the most popular barcode symbologies:

- **UPC-A:** The Universal Product Code (UPC) is a barcode symbology that is widely used in the United States and Canada for tracking trade items in stores. UPC encodes 12 digits into a bit pattern consisting of parallel vertical bars and spaces.
- **EAN-13:** A European Article Number (EAN) is a standard barcode representation which is a superset of the original UPC system developed in North America. EAN-13 barcodes are used worldwide for marking retail goods. The EAN-13 symbology supersedes UPC-A, but provides backwards compatibility; a UPC barcode is also an EAN-13 barcode with the first digit set to zero.
- **JAN-13:** A Japanese Article Number (JAN) is a standard barcode representation which is a subset of the EAN-13 barcode representation. A JAN-13 barcode is an EAN-13 barcode whose system or country code is set to 49, the country code for Japan.

While barcode recognition functionality can be added programmatically using the Scannerfly SDK library's API, the library also provides a component called `ScannerPanel`. The `ScannerPanel` component can be integrated easily into existing applications' user interface to capture barcode data using a webcam connected to the user's computer.

Supported Platforms

The Scannerfly SDK can be used on any of the operating systems and browsers supported by Adobe® Flash® Player 9, as detailed below. For more information on minimum supported operating systems, browsers, and system requirements, please see:

http://www.adobe.com/products/flashplayer/productinfo/systemreqs/

Windows

Platform	Browser
Microsoft® Windows® Vista	Microsoft Internet Explorer 7 Firefox 2.0 AOL 9
Microsoft Windows XP	Microsoft Internet Explorer 6.0 or later Firefox 1.x and 2.x Mozilla 1.x or later Netscape 7.x or later AOL 9 Opera 7.11 or later
Windows Server® 2003	Microsoft Internet Explorer 6.0 or later Firefox 1.x and 2.x
Windows 2000	Microsoft Internet Explorer 5.x Firefox 1.x and 2.x Mozilla 1.x Netscape 7.x or later AOL 9 Opera 7.11 or later
Windows Me	Microsoft Internet Explorer 5.5 Firefox 1.x Mozilla 1.x Netscape 7.x or later AOL 9 Opera 7.11 or later
Windows 98	Microsoft Internet Explorer 6.0 or later Firefox 1.x Mozilla 1.x Netscape 7.x or later CompuServe 7 AOL 9 Opera 7.11 or later

Macintosh

Platform	Browser
Mac OS X v.10.1.x, 10.2.x, 10.3.x, or 10.4.x (PowerPC)	Firefox 1.5.0.7 and higher Mozilla 1.7.x and higher SeaMonkey 1.0.5 and higher
Mac OS X v.10.4.x (Intel)	Firefox 1.5.0.7 and higher Mozilla 1.7.x and higher SeaMonkey 1.0.5 and higher

Linux

Platform	Browser
Red Hat® Enterprise Linux® (RHEL) 3 update 8, RHEL 4 update 4 (AS/ES/WS)	Firefox 1.x Mozilla 1.x Netscape 7.x or later AOL for Mac OS X Opera 6 Safari 1.x or later
Novell SUSE 9.x or 10.1	Firefox 1.5.0.3 or later Opera 6 Safari 2.x or later

Solaris

Platform	Browser
Solaris 10	Firefox 1.5.x and higher Mozilla 1.7.x and higher

Installation Procedure

The Scannerfly SDK installation consists of a single zipfile (scannerfly.zip) containing the following directories:

- **docs:** The auto-generated API documentation for the Scannerfly SDK, which provides additional detail on using the API directly. For more information on programming using the API, see the *Advanced Programming* section.
- **lib:** The SWC library for the Scannerfly SDK that needs to be added to Flash projects to make the barcode recognition functionality accessible to the compiler for inclusion in a Flash application.
- **samples:** A set of sample applications demonstrating how to include the Scannerfly SDK in a Flash application, as well as how to incorporate it into web-based applications driven by JavaScript.

Installing the Scannerfly SDK requires you to copy the contents of the zipfile to a folder on your system. For details on how to include the SWC library in your own project, see the *Including the Scannerfly SDK Library in a Project* section.

Including the Scannerfly SDK Library in a Project

Before attempting to use the Scannerfly SDK in your application, you need to include the Scannerfly SDK library in your project. Including the library makes it available to the Flash compiler when building your application.

▸ **Include the Scannerfly SDK library in your project**

1 Copy the `scannerfly.swc` from the installation zipfile's `lib` directory to your project's working directory.

2. Open your project in Flex Builder 2.
3. Verify that Build Automatically is selected in the Project menu.
4. Select Project -> Properties.
5. Select the Flex Build Path item.
6. Navigate to the Library path tab.
7. Choose the Add SWC option.
8. Browse for `scannerfly.swc`, select it, and click Open.
9. Click OK to add the SWC to the project.
10. Click OK to apply the changes to your project.

After applying this change, Flex Builder 2 will automatically build your project for you. You are now ready to include `ScannerPanel` in your application’s user interface.

3 The ScannerPanel Component

This chapter describes the `ScannerPanel` component, a simple Flash component that allows you to integrate barcode recognition without a lot of programming. The `ScannerPanel` component grabs video from a camera connected to the user's computer, performs barcode recognition, and dispatches an event when a barcode is extracted from the video stream.

In This Chapter

About the Component

Many applications require the ability to perform barcode recognition on a video stream obtained from a camera connected to the user's computer. The Scannerfly SDK packages this functionality in an easy-to-integrate component called `ScannerPanel`. `ScannerPanel` is a subclass of the `mx.controls.VideoDisplay` object, and displays a video stream captured from a web camera connected to the user's computer. As a result, adding barcode recognition is as simple as including `ScannerPanel` in your application's user interface and writing a function to handle barcode detection events.

Note: For applications where the user does not have a camera connected to their computer and is expected to input images by other means, the Scannerfly SDK can perform barcode recognition directly on a `BitmapData` object. See the *Advanced Programming* section for more details.

Add `ScannerPanel` to your application

1. Include the Scannerfly SDK library in your project settings (see the *Including the Scannerfly SDK Library in a Project* section for details).
2. Add the `ScannerPanel` component to your user interface.

3 Create a function to handle barcode detection events.
4 Add code to configure `ScannerPanel`'s operation.
5 Start the scanning process.

Warning: If you don't include the library in your project, Flex Builder 2 will be unable to find the classes associated with the Scannerfly SDK when attempting to build the application. The Problems view will display the error "Type was not found or was not a compile-time constant" indicating that the compiler is unable to resolve references to the Scannerfly SDK. If you encounter these errors, see the *Including the Scannerfly SDK Library in a Project* section for instructions on including the Scannerfly SDK library properly.

The following sections provide detailed instructions on how to perform each of these steps.

Adding ScannerPanel to the User Interface

The following instructions detail each of the three methods of adding the `ScannerPanel` component to your application's user interface.

Add `ScannerPanel` using the MXML editor's Design Mode

1 Open your application's MXML file.
2 Switch the MXML editor into Design Mode.
3 Open the Custom folder in the Components palette.
4 Drag and drop the `ScannerPanel` component onto the design area.

Add `ScannerPanel` to your application's MXML directly

1 Open your application's MXML file.
2 Switch the MXML editor into Source Mode.
3 Add `com.scannerfly.barcode.util.*` package to your application's namespace using the `xmlns:mx` attribute.
4 Add a `ScannerPanel` tag to the MXML.

Regardless of whether you add `ScannerPanel` in Design Mode or Source Mode, the final MXML should look like this:

```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
     xmlns:Barcode="com.scannerfly.barcode.util.*" width="640"
     height="480" layout="absolute">
     <Barcode:ScannerPanel width="640" height="480" x="0" y="0"
            id="panel"/>
```

Add `ScannerPanel` using ActionScript

1 Import the `com.scannerfly.barcode.util.*` package inside your application's `<mx:Script>` tag or in another ActionScript class.

2. Create an instance of the `ScannerPanel` class.
3. Add the `ScannerPanel` instance as a child of another user interface container instance.

For example, the following ActionScript code demonstrates how to add the `ScannerPanel` component programmatically to the `parent` container object:

```
import com.scannerfly.barcode.util.*

…

var parent:Panel = new Panel();
var scannerPanel:ScannerPanel = new ScannerPanel();
scannerPanel.width = 640;
scannerPanel.height = 480;
parent.addChild(scannerPanel);
```

Handling Barcode Detection Events

When `ScannerPanel` successfully detects a barcode in the video stream, it will dispatch a `BarcodeEvent` describing the barcode it has detected. The barcode information is represented as a `Barcode` object, allowing you to easily access the value and symbology of the detected barcode. To be notified when a barcode is detected, your application needs to register a handler function to process the `BarcodeEvent`.

Handling a barcode detection event

1. Import the `com.scannerfly.barcode.events.*` and the `com.scannerfly.barcode.*` packages inside your application's `<mx:Script>` tag or in another ActionScript class.
2. Define a handler function that accepts a single `BarcodeEvent` instance as a parameter and processes it according to your application's needs.
3. Register the handler function with the `ScannerPanel` instance, either by setting the `ScannerPanel` tag's `barcodeDetected` attribute, or by calling the `ScannerPanel` instance's `addEventListener` method.

For example, the following ActionScript code demonstrates a simple handler function that displays the detected barcode using an `Alert` window:

```
private function myHandler(event:BarcodeEvent):void
{
     var barcode:Barcode = event.barcode;

     Alert.show(barcode.getString(),"Found barcode!");
}
```

To register this handler function in MXML, simply set the `barcodeDetected` attribute on the `ScannerPanel` tag to the name of the handler function:

```
<Barcode:ScannerPanel width="640" height="480" x="0" y="0"
     barcodeDetected="myHandler(event);"/>
```

Alternatively, you may choose to add the handler function as an event listener to the `ScannerPanel` instance programmatically:

```
panel.addEventListener(BarcodeEvent.BARCODE_DETECTED, myHandler);
```

Either of these methods will trigger the `myHandler` function whenever `ScannerPanel` detects a barcode, and pass a `BarcodeEvent` object containing a `Barcode` object detailing the detected barcode value and symbology.

Configuring ScannerPanel

`ScannerPanel` has two properties that control the barcode recognition process:

- **hotspot:** The `hotspot` property is a `Rectangle` which limits where the `Scanner` object will look in images for a potential barcode. The purpose of this property is to eliminate portions of the image, decreasing the computational cost of performing barcode recognition. `ScannerPanel` will outline the hotspot area as an overlay on top of incoming video stream.
- **readers:** The `readers` property is an `Array` of `BarcodeReader` objects that will be used to perform barcode recognition. A `BarcodeReader` object is responsible for attempting to extract a barcode from a region of the image that has been identified by a `Scanner` object as possibly containing a barcode. By default, `readers` is an `Array` that only contains a `BarcodeReader` object capable of recognizing EAN-13 barcodes.

Changing the `hotspot` property is as simple as creating a new `Rectangle` object and updating the property:

```
var panel:ScannerPanel = new ScannerPanel();
var bounds:Rectangle = new Rectangle(10, 10, 500, 500);


panel.hotspot = bounds;
```

Updating the `readers` property requires you to create an `Array` containing the `BarcodeReader` objects you want the `ScannerPanel` instance to use recognize barcodes. Creating a `BarcodeReader` concrete instance requires you to use the `BarcodeReaderFactory` factory class' `getBarcodeReader` method, and the symbology constants defined by the `Barcode` class:

```
var panel:ScannerPanel = new ScannerPanel();
var readers:Array = new Array();
var reader:BarcodeReader =
     BarcodeReaderFactory.getBarcodeReader(Barcode.EAN_13);


readers.push(reader);
panel.readers = readers;
```

Starting the Scanning Process

To start the barcode recognition process, your application must call the `ScannerPanel` object's `start` method. Once `start` is called, `ScannerPanel` will begin capturing video from the user's camera and performing barcode recognition.

Caution: As `ScannerPanel` relies on the `Camera` object to obtain the video stream, it is subject to Flash's security restrictions. The first time your application calls `start`, the user will be presented with a security dialog requesting permission to access the camera and microphone. If the user does not agree to providing access to the camera and microphone, `ScannerPanel` will display the Flash Player Settings UI.

For more information on Flash's security restrictions, see the Adobe Flash Player 9 Security Whitepaper available at:

http://www.adobe.com/devnet/flashplayer/articles/flash_player_9_security.pdf

Once the scanning process has started, `ScannerPanel` will dispatch a `BarcodeEvent` to your registered handler function whenever it detects a barcode. To stop `ScannerPanel` processing the video stream, simply call the `stop` method; a subsequent call to `start` will re-start the barcode recognition process.

4 Advanced Programming

The `ScannerPanel` component is not appropriate for applications that do not use Flash's built-in `Camera` object to capture images. Alternatively, an application can perform recognition on images the user loads into the Flash application. In this case, a developer will need to create code that configures the barcode recognition engine directly and passes a `BitmapData` object to the Scannerfly SDK for processing.

In This Chapter

The Barcode Recognition Process

Before detailing how to use the barcode recognition directly, it's important to understand the process used by the Scannerfly SDK to extract barcodes from an image.

Attempting to extract a barcode from an image is broken down into the following steps:

- **Scan the image to identify candidate barcode areas:** This process may use edge detection and other techniques to identify areas of the image that potentially contain a barcode. This process reduces the amount of area that will be subjected to subsequent analysis, reducing the computation required to extract a barcode.
- **Analyze each candidate area for a valid barcode:** The areas identified in the first phase are now analyzed in depth to extract a barcode value, as well as verify any extracted barcode's checksum. Extracted barcodes without a valid checksum are eliminated.
- **Choose the best barcode:** The results of the previous phase are now evaluated for each candidate area or areas. An algorithm may use a majority voting mechanism or other scoring mechanism to choose which barcode value is the best estimate of the actual barcode value.

Important Interfaces and Classes

From this description, it's clear that there are only three classes that a programmer needs to deal with: an object to scan the image for candidate areas containing barcodes, an object that attempts to extracts barcodes for a specific barcode symbology from a candidate area, and an object to represent any extracted barcode.

In the Scannerfly SDK, the following interfaces and classes define the core barcode recognition functionality:

- **Scanner:** The `Scanner` interface defines a `scan` method that identifies areas of a given `BitmapData` object that may contain a barcode.
- **BarcodeReader:** Once the `Scanner` has identified candidate areas, it uses one or more concrete implementations of the `BarcodeReader` interface to analyze each candidate area for a valid barcode. A `BarcodeReader` implementation is specific to a barcode symbology.
- **Barcode:** A class that defines a data structure to hold the details of an extracted barcode, including the data encoded by the barcode and the symbology used to encode the data.

To insulate the developer from the details of concrete implementations of the `Scanner` and `BarcodeReader` interfaces, the Scannerfly SDK relies on the following two factory classes to handle creating concrete instances:

- **ScannerFactory:** Provides a mechanism to create a Scanner instance by calling the static `getScanner` method with a scanner type constant defined by the `ScannerFactory` class.
- **BarcodeFactory:** Provides a mechanism to create a `BarcodeReader` instance by calling the static method `getBarcodeReader` with a barcode reader symbology constant defined by the `BarcodeReaderFactory` class.

Note: The Scannerfly SDK currently only provides a one-dimensional scanner based on analyzing a subset of horizontal lines from the video stream. In the future, Scannerfly SDK may add other scanning methodologies that make it necessary to alter the `scanner` property from its default value.

Using the Scannerfly SDK API

Using the interfaces and classes described in the previous section makes using the Scannerfly SDK API simple.

Use the Scannerfly SDK API directly

1 Create a `Scanner` instance by calling `ScannerFactory`'s `getScanner` method with one of the scanner type constants defined by `ScannerFactory`.

2. Create one or more `BarcodeReader` instances by calling `BarcodeReaderFactory.getBarcodeReader` and passing a barcode symbology constant defined by `BarcodeReaderFactory`.
3. Call the `Scanner` instance's scan method with the `BitmapData` object containing the image to be analyzed, an array of the BarcodeReader instances to use to analyze candidate areas, and a `Rectangle` object defining the boundary in which to search for barcodes.

The following example code shows how this should be done in your application:

```
import flash.display.BitmapData;
import flash.geom.Rectangle;

import com.scannerfly.barcode.*;

…

var image:BitmapData;
var scanner:Scanner;
var readers:Array;
var reader:BarcodeReader;
var bounds:Rectangle;
var barcodes:Array;


// Initialize the scanner to use to identify areas of the
// image that might contain barcodes and reader that will
// attempt to extract a barcode from a candidate area.
scanner =
     ScannerFactory.getScanner(ScannerFactory.ONE_DIMENSIONAL);
reader =
     BarcodeReaderFactory.getBarcodeReader(Barcode.EAN_13);
readers = new Array();
readers.push(reader);

…

// Population the image source here

…

// Scan the image and report the results for detected barcodes.
bounds = new Rectangle(0, 0, image.width, image.height);
barcodes = scanner.scan(image, readers, bounds);
for (var i:Number; i < barcodes.length; i++)
{
     Alert.show(barcodes[i].getString(),"Found barcode!");
}
```

5 Sample Applications

The Scannerfly SDK installation zipfile includes three sample applications to help you see its capabilities in action.

In This Chapter

Using the Sample Applications

The `Samples` directory in the Scannerfly SDK installation zipfile contains three directories, one for each sample application:

- **SimpleScan:** A simple Flash application that uses the `ScannerPanel` component to scan the video stream from a camera connected to the user's computer, and displays detected barcodes using `Alert.show`.
- **JavascriptScan:** A variant of the SimpleScan application that displays detected barcodes using a Javascript function that gets registered by the HTML page container. This sample shows how to easily integrate barcode recognition with an existing web application using the Scannerfly SDK.
- **AmazonScan:** A more complete Flash application that uses the `ScannerPanel` to scan book barcodes and then display details about the book. Details about the book are retrieved from Amazon.com using Amazon Web Services.

Each sample provides the Flex Builder 2 project files for the application, as well as the source code for the sample. The samples are completely self-contained and independent of each other.

Note: Running the sample applications require a webcam and the Adobe Flash Player 9 plugin. Adobe Flash Player 9 is available as a free download from Adobe's corporate website at:

http://www.adobe.com/products/flashplayer/

Run a sample application

1. Navigate to the `bin` directory for the sample you want to run.
2. Double-click the HTML file with the same name as the sample (for example: SimpleScan.html) to launch it in your web browser.

Warning: Unlike the other sample applications, AmazonScan does not contain a `bin` directory. You will need to build the sample application before you will be able to run the sample application. See the *Building the AmazonScan Application* section for instructions on building the AmazonScan sample application.

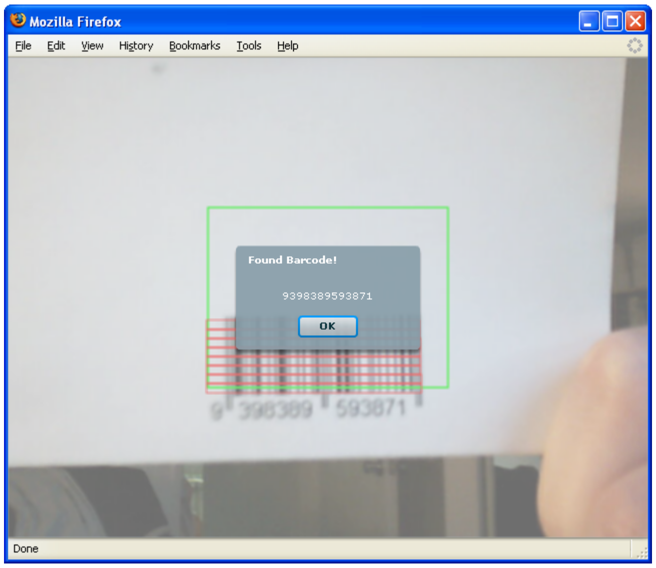
Once you understand a sample application's functionality, you can view or modify the source code. Editing the code requires that you have Flex Builder 2 already installed on your machine.

Open the sample applications in Flex Builder 2

1. Start Flex Builder 2.
2. Select File, and click Import.
3. Choose to import Existing Projects into Workspace and click Next.
4. Check the Select root directory option.
5. Click Browse, select the location of the `samples` directory, and click OK.
6. Ensure that all of the sample applications are checked and click Finish.

The SimpleScan Application

The SimpleScan sample application provides the simplest example of embedding barcode recognition in an application. The sample uses the `ScannerPanel` component to scan the video stream from a camera connected to the user's computer, and displays detected barcodes using `Alert.show`:



Found Barcode!

9398389593871

Run the SimpleScan sample application

1. Navigate to the `simplescan\bin` directory.
2. Double-click `SimpleScan.html` to launch it in your web browser.

Once the application has started, hold a barcode in front of your computer's webcam. When a barcode is detected, the SimpleScan application will display the detected barcode's value in a dialog box within the application.

Components of the Application

There is only one component of the SimpleScan sample application:

- **SimpleScan.mxml:** This is the source code file that defines the application's user interface and functionality.

All other files in the project, such as the HTML container page, are automatically generated by the project and have not been altered from their defaults.

How It Works

The code for the SimpleScan sample is quite simple. First, the application includes the namespace for the `ScannerPanel` component:

```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
     xmlns:Barcode="com.scannerfly.barcode.util.*" width="640"
     height="480" layout="absolute">
```

Next, the application includes the `ScannerPanel` component, configures the event that will handle barcode detection events, and sets the handler function for the `creationComplete` event to trigger `ScannerPanel` to start scanning for barcodes. Note that in this sample, the application is relying on `ScannerPanel`'s default behavior; by default, `ScannerPanel` will scan only for EAN-13 barcodes in this sample application.

```
<Barcode:ScannerPanel width="640" height="480" x="0" y="0"
     barcodeDetected="displayBarcode(event);" id="scannerPanel"
     creationComplete="scannerPanel.start();"/>
```

Finally, the application defines the handler function that will be called when `ScannerPanel` detects a barcode:

```
import mx.controls.Alert;

import com.scannerfly.barcode.Barcode;

import com.scannerfly.barcode.events.BarcodeEvent;


/**
 * Displays the barcode information detected in the video stream.
 *
 * @param event The event signaling a barcode has been found.
 */
```

```
private function displayBarcode(event:BarcodeEvent):void
{
      if (event.barcode != null)
      {
             Alert.show(event.barcode.getString(), "Found Barcode!");
      }
}
```

Using this code, when the application is loaded, the `ScannerPanel` component is created and starts scanning for barcodes immediately. Once the `ScannerPanel` component detects a barcode, it calls the `displayBarcode` handler function, which displays the detected barcode to the user.

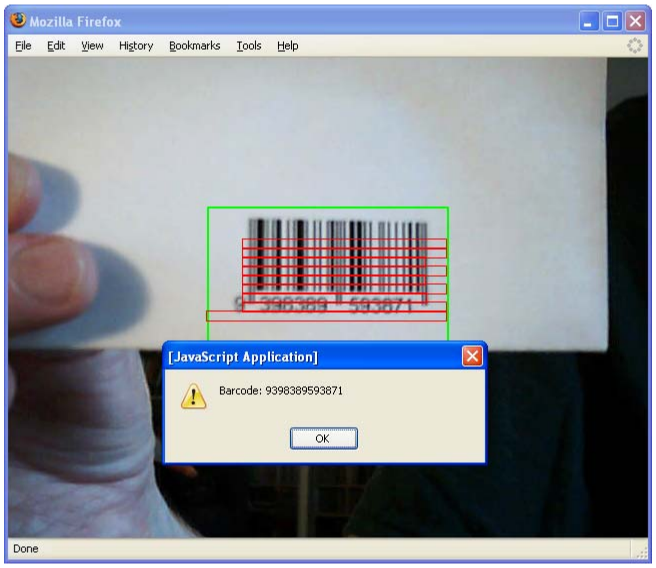
The JavascriptScan Application

The JavascriptScan application is a more complex variant of the SimpleScan application that displays detected barcodes using a Javascript function that gets registered by the HTML page container. This sample shows how to easily integrate barcode recognition with an existing web application using the Scannerfly SDK.

Run the JavascriptScan sample application

1 Navigate to the `javascriptscan\bin` directory.
2 Double-click `JavascriptScan.html` to launch it in your web browser.

Once the application has started, hold a barcode in front of your computer's webcam. When a barcode is detected, the JavascriptScan application displays the detected barcode's value in a Javascript dialog box:



Barcode: 9398389593871

Components of the Application

There are three components of the JavascriptScan sample application:

- **JavascriptScan.mxml:** This is the source code file that defines the application's user interface and functionality.
- **JavascriptScan.js:** This file in the `html-template` directory defines all of the Javascript code required by the HTML container to enable the page to properly interact with the Flash application.
- **index.template.html:** This file in the `html-template` directory defines a template that is used to generate the application's HTML container. This file has been altered from the default template generated by Flex Builder 2 to include the JavascriptScan.js source file, and add an `onLoad` event handler.

All other files in the project are automatically generated by the project and have not been altered from their defaults.

How It Works

Unlike the SimpleScan sample application, JavascriptScan has to ensure that the HTML container page and associated Javascript is completely loaded before beginning to scan for barcodes. This is necessary because JavascriptScan relies on a Javascript handler in the HTML container to report detected barcodes to the user, and these Javascript functions cannot be accessed until the container has been loaded completely.

The process for ensuring that the container is loaded and registering the Javascript function that will handle detected barcodes is as follows:

1. Once the JavascriptScan application is loaded, it polls the HTML container by calling the `isReady` Javascript function to see if the container has finished loading.
2. When the HTML container has completed loading, it calls the `pageInit` Javascript function; the next time JavascriptScan calls `isReady`, it will be notified that the HTML container has finishing loading.
3. Once JavascriptScan is notified that the container has finished loading, it registers a function called `addListener` with the container that will allow the container to register one or more functions to handle barcode detection events. It also starts `ScannerPanel` scanning for barcodes.
4. JavascriptScan calls the `setSWFIsReady` Javascript function to signal to the container that it can now register barcode detection handler functions.
5. The container registers one or more handler functions using JavascriptScan's `addListener` function.
6. When a barcode is detected, the JavascriptScan application calls each of the Javascript functions previously registered using `addListener`, and passes a string representation of the detected barcode.

Using JavascriptScan in your Web Application

The JavascriptScan sample is especially useful for developers that simply wish to add barcode recognition to an existing web application to replace manual dat entry. As all of the code responsible for interacting with the HTML container is contained in `JavascriptScan.js`, you can easily tailor this sample to your needs without even needing to alter the Flash application itself.

To change how JavascriptScan handles a detected barcode, you need to understand the `setSWFIsReady` Javascript function:

```
function setSWFIsReady()
{
    swfReady = true;

    getSWF("JavascriptScan").addListener("processBarcode");
}
```

This function registers a Javascript handler function called `processBarcode`, which displays the detected barcode to the user, as shown below:

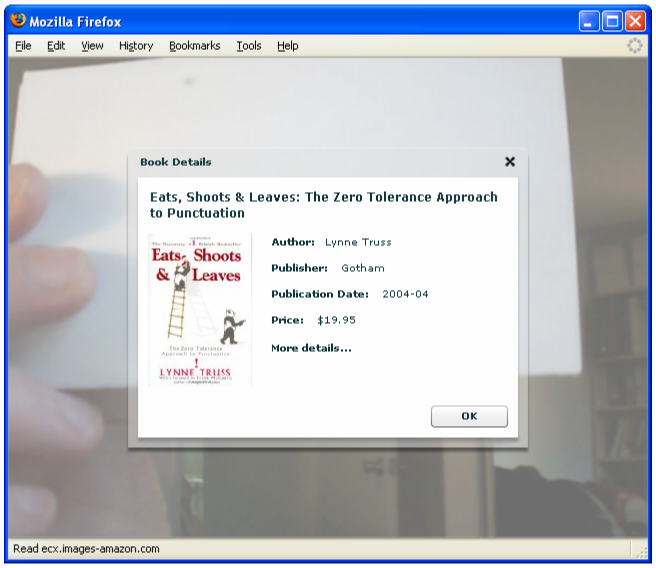
```
function processBarcode(barcodes)
{
     if (barcodes != null)
     {
            alert("Barcode: " + barcodes);
     }
}
```

If you want to integrate barcode recognition into your web application, you can either change `setSWFIsReady` to register your own handler function, or alter the contents of the `processBarcode` function.

Caution: If you choose to change the `name` attribute used to embed the JavascriptScan application in an HTML page, you must also update `setSWFIsReady`‘s call to `getSWF` to use the new name. Otherwise, `setSWFIsReady` will not be able to find the Flash application in the HTML container. When this occurs, the barcode detection handler function will never get registered with the Flash application, and your Javascript handler function will never be called when JavascriptScan detects a barcode.

The AmazonScan Application

The AmazonScan application provides a more complete Flash application that uses the `ScannerPanel` to scan book barcodes, and Amazon.com’s web service to retrieve and display details about the book:



Book Details

Eats, Shoots & Leaves: The Zero Tolerance Approach to Punctuation

Author: Lynne Truss

Publisher: Gotham

Publication Date: 2004-04

Price: $19.95

More details...

OK

Read ecx.images-amazon.com

Warning: Unlike the other sample applications, AmazonScan does not contain a `bin` directory until you build the application. You need to build the sample application using Flex Builder 2 before you can run the sample application. This is necessary because the sample application requires you to add your Amazon Web Services developer token to the code to allow the application to function properly.

See the *Building the AmazonScan Application* section for instructions on building the AmazonScan sample application.

Components of the Application

There are three components of the AmazonScan sample application:

- **AmazonScan.mxml:** This is the source code file that defines the main application user interface and functionality. This interface primarily consists of a `ScannerPanel` that is used to scan for barcodes.
- **BookDetail.mxml:** This is the source code for the dialog box used by `AmazonScan.mxml` to display the details of a book that the application retrieves from Amazon.com.

All other files in the project are automatically generated by the project and have not been altered from their defaults.

How It Works

As with the SimpleScan sample application, `AmazonScan.mxml` adds a `ScannerPanel` instance to the user interface and begins scanning for barcodes immediately. When a barcode is detected, the application uses Flash's `WebService` class to query Amazon.com's web service to retrieve information about the book that carries that barcode. Information that is retrieved is used to populate the user interface defined by `BookDetail.mxml`, which is then displayed to the user.

Caution: AmazonScan is designed to scan only book barcodes. Any other barcodes that are scanned by the application will not be found in Amazon.com's book database, and hence will not result in any information being presented to the user.

Building the AmazonScan Application

AmazonScan relies on Amazon.com web services to provide information about books that are scanned by the application. To access the Amazon.com web service, AmazonScan requires you to update the code to use your Amazon Web Services (AWS) Access Key ID when performing queries.

Note: You can obtain an AWS Access Key ID for free from Amazon.com by registering at:

http://aws.amazon.com/

You will need to obtain an AWS Access Key ID before you can proceed with the remainder of this section.

The AWS Access Key ID used by AmazonScan is defined by the following variable:

```
[Bindable]
private var amazonDeveloperId:String = "####################";
```

Build the AmazonScan sample application

1. Open the AmazonScan project in Flex Builder 2 (see the *Using the Sample Applications* section for instruction on how to do this).
2. Open `AmazonScan.mxml` in the MXML Editor.
3. Locate the `amazonDeveloperId` variable declaration described above.
4. Edit the value of the `amazonDeveloperId` variable to match your AWS Access Key Id.
5. Save the file.

If you have Build Automatically selected, then Flex Builder 2 will rebuild the project for you automatically; otherwise, you'll have to select Build Project from the Project menu to rebuild the project. Once the project has been rebuilt without errors, you will be able to run the application successfully.

Run the AmazonScan sample application

1. Navigate to the `amazonscan\bin` directory.
2. Double-click `AmazonScan.html` to launch it in your web browser.

Once the application has started, hold a book's barcode in front of your computer's webcam. When a barcode is detected, the AmazonScan application will query Amazon.com and display the book's details using the user interface defined by `BookDetail.mxml`.