Dynamic Web TWAIN

Developer's Guide

Based on Version 11.2

November, 2015

Dynamsoft™

11 Years of Experience in TWAIN SDKs and Version Control Solutions

Contents

Preface

Description

This guide provides instructions on how to use Dynamsoft's Dynamic Web TWAIN SDK. You can get an overview of most of the things you can achieve with the SDK.

Audience

This guide is meant for developers new to Dynamic Web TWAIN SDK as well as those who are experienced in the SDK.

For new developers, there is a step-by-step guide to help you develop a scanning page in your web application from scratch.

For those who have used the SDK before, you can find information on advanced APIs that you can use to polish your scanning page. You will also find new features introduced in this version of the SDK.

Getting Started

What is TWAIN

TWAIN is a standard software protocol and application programming interface (API) that regulates communication between software applications and imaging devices such as scanners and digital cameras.

The TWAIN standard, including the specification, data source manager and sample code, is maintained by the not-for-profit organization TWAIN Working Group.

Dynamsoft Corporation is a member of the TWAIN Working Group.

What is Dynamic Web TWAIN

Dynamic Web TWAIN is a TWAIN scanning SDK specifically optimized for **web applications**. This TWAIN interface enables you to write code, in just a few lines, to **scan** documents from a **TWAIN Compliant** scanner or **acquire** images from any other TWAIN Compliant devices. Users can then **edit** the images, **save** them locally, or **upload** them to a remote server in a variety of formats.

With the SDK, you can also import local images or download files from the web.

Basic Requirements

- **Server Side:**
    1. Operating System: Windows, Linux, UNIX, Mac, etc.
    2. Web Server: IIS, Apache, Tomcat, ColdFusion, etc.
    3. Programming Languages:
        - Front-end: HTML, JavaScript, CSS
        - Back-end: ASP.NET(C# and VB), PHP, JSP, ASP, etc.

- **Client Side:**
    1. Browser/OS Support
        - Windows XP, Vista, 7/8/10, Windows Server 2003/2008/2012
            - IE 6 ~ 9: ActiveX
            - IE 10 ~ 11: HTML5/ActiveX
            - Edge: HTML5
            - Chrome/Firefox 26-: NPAPI Plug-in (obsolete)
            - Chrome/Firefox 27+: HTML5
        - Mac OS X 10.6 and later
            - Chrome/Firefox 26-, Safari 6-: NPAPI Plug-in (obsolete)
            - Chrome/Firefox 27+, Safari 7+: HTML5
    2. Scanner/Camera/Other Devices (must be TWAIN compliant), Webcams (must be UVC compliant, in other words, it must support DirectShow).

    **For Internet Explorer 6 ~9, please verify that the following security settings are set to "Prompt" or "Enabled" (typically these are the default settings):*

    a) Download signed ActiveX controls

    b) Run ActiveX Controls and plug-ins

    c) Script ActiveX controls marked safe for scripting

Deciding which Dynamic Web TWAIN Edition to use

Dynamic Web TWAIN has four editions: ActiveX, Plugin, HTML5 for Windows and HTML5 for Mac. Based on the browser(s) your end users use, you can decide which edition(s) to use.

- **ActiveX**: supports IE 6 ~ 9 on Windows by default, it can be configured to support IE 10, 11 as well;
- **HTML5 for Windows**: supports Firefox/Chrome 27+, IE 10/11 and Edge on Windows;
- **Plug-in Edition**: supports Firefox/Chrome 26-, Safari 6- and Opera on Windows and Mac;
- **HTML5 for Mac**: supports Chrome/Firefox 27+, Safari 7+ on Mac OS.

Building the "Hello World" Scan Page

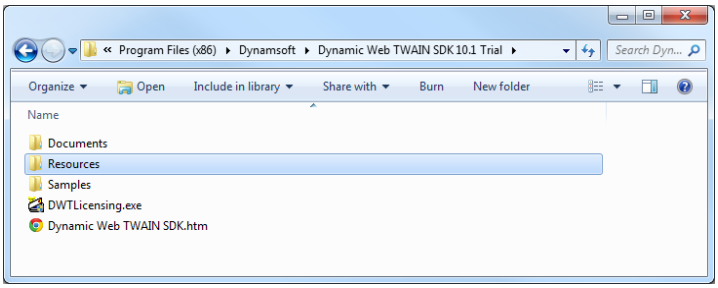
NOTE: Before you start, please make sure you've downloaded and installed the latest version of Dynamic Web TWAIN. If you haven't done so, you can get the 30-day free trial here.

The following 3 steps will show you how to create your first web-based scanning application in just 5 minutes!

Step 1: Start a Web Application

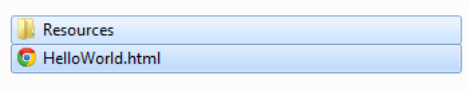
1.1 Copy the Dynamsoft's Resources folder to your project

The Resources folder can normally be copied from C:\Program Files (x86)\Dynamsoft\Dynamic Web TWAIN SDK {Version Number} {Trial}\



1.2 Create an empty HTML page

Please put the empty html page together with the Resources folder, as shown below:



Step 2: Add Dynamic Web TWAIN to the HTML Page

2.1 Include the two Dynamsoft JS files in the <head> tag

```
<script type="text/javascript" src="Resources/dynamsoft.webtwain.initiate.js"> </script>
<script type="text/javascript" src="Resources/dynamsoft.webtwain.config.js"> </script>
```

2.2 Add Dynamic Web TWAIN container to the <body> tag

```
<div id="dwtcontrolContainer"> </div>
```

***Note**: "dwtcontrolContainer" is the default id for the div. You can change it in the file dynamsoft.webtwain.config.js if necessary.*

Step 3: Use Dynamic Web TWAIN

3.1 Add a Scan button and the minimum scripts to scan

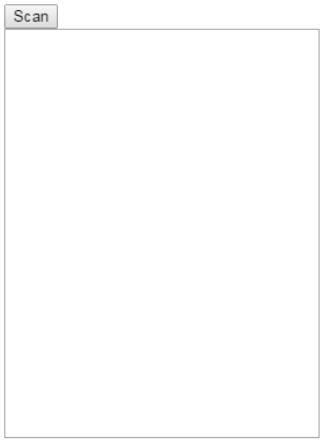
```
  <input type="button" value="Scan" onclick="AcquireImage();" />
  <script type="text/javascript">
         var DWObject;
         function Dynamsoft_OnReady(){
           DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
         }
         function AcquireImage(){
           if(DWObject) {
               DWObject.IfDisableSourceAfterAcquire = true;
               DWObject.SelectSource();
               DWObject.OpenSource();
               DWObject.AcquireImage();
           }
         }
  </script>
```

3.2 Review the completed code

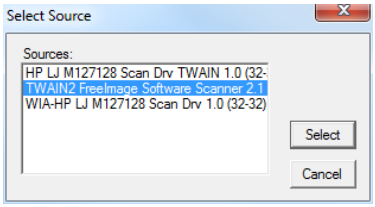
```
<html><head><title>Hello World</title>
  <script type="text/javascript" src="Resources/dynamsoft.webtwain.initiate.js"> </script>
  <script type="text/javascript" src="Resources/dynamsoft.webtwain.config.js"> </script>
</head>
<body>
  <input type="button" value="Scan" onclick="AcquireImage();" />
  <div id="dwtcontrolContainer"> </div>
  <script type="text/javascript">
         var DWObject;
         function Dynamsoft_OnReady(){
           DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
         }
         function AcquireImage(){
           if(DWObject) {
               DWObject.IfDisableSourceAfterAcquire = true;
               DWObject.SelectSource();
               DWObject.OpenSource();
               DWObject.AcquireImage();
           }
         }
  </script>
</body></html>
```

3.3 See the scan page in action

If you open the Hello World page in your project, it should look like this:



Now, you can click on the **Scan** button to select a device, as shown below:



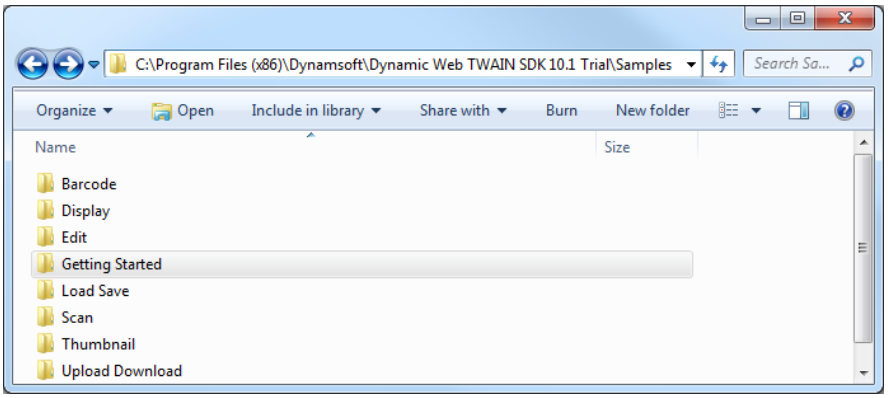
NOTE:

1) *Only TWAIN compatible devices are listed in the Select Source dialog. If your connected scanner is not shown in the list, please follow this article to troubleshoot.*

2) *If you don't have a real scanner at hand, you can install the Virtual Scanner – a scanner simulator which is developed by the TWAIN Working Group – for testing purposes.*

Once scanning is done, image(s) will show up in the built-in Dynamic Web TWAIN viewer:

If you have installed the [30-day trial version of Dynamic Web TWAIN], you can normally find the complete Hello World application at C:\Program Files (x86)\Dynamsoft\Dynamic Web TWAIN SDK {Version Number} {Trial}\Samples\Getting Started\.



As you can see, there are many other sample applications (source code provided) available for you to explore more features of Dynamic Web TWAIN. Enjoy! ☺

Using JavaScript IntelliSense

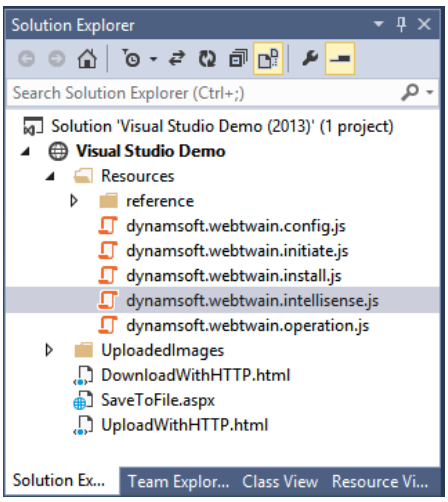
IntelliSense is designed to make the development of your application much easier. It is a powerful feature that helps you write code faster and with fewer errors by providing information while you code.

NOTE: this feature is available from version 10.1.

In Visual Studio

Dynamic Web TWAIN IntelliSense supports Visual Studio 2010 and later.

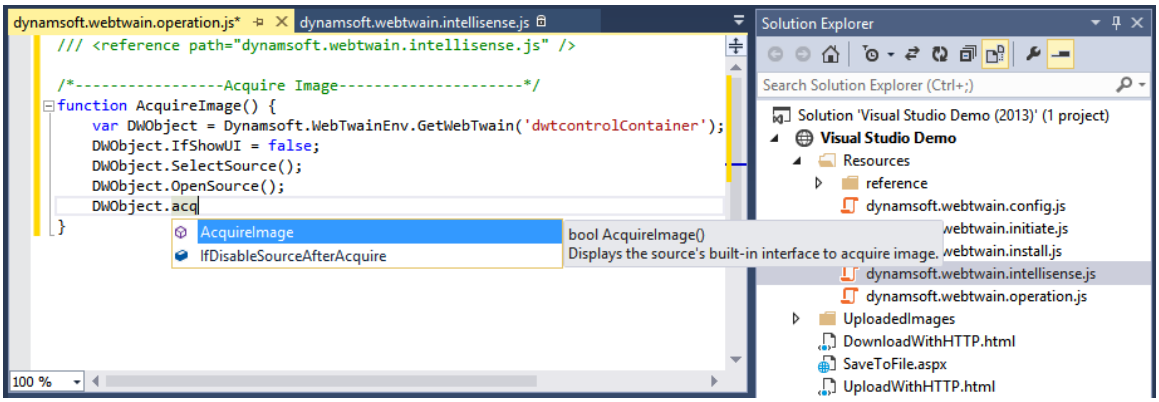
Firstly, you need to make sure *dynamsoft.webtwain.intellisense.js* is in your web project. If not, please add it to your project manually. Typically, it is under *C:\Program Files (x86)\Dynamsoft\Dynamic Web TWAIN* {Version Number} {Trial}*Resources*.



Next, drag *dynamsoft.webtwain.intellisense.js* from the solution explorer into the editor window of the JavaScript file where you want to use the intellisense. This will create the reference directive in the current JavaScript file which will look like this

```
/// <reference path="dynamsoft.webtwain.intellisense.js" />
```

Alternatively, you can add the above line manually to reference *Dynamsoft.webtwain.intellisense.js* and enable the IntelliSense.



***Note**:*

1. *The reference directive must be declared earlier than any script in the JavaScript file, in other words, you should put it at the very top.*

2. *The path (path = "dynamsoft.webtwain.intellisense.js") is relative to the current JavaScript file.*

3. *IntelliSense only supports pure JavaScript files (with the extension **.js**).*

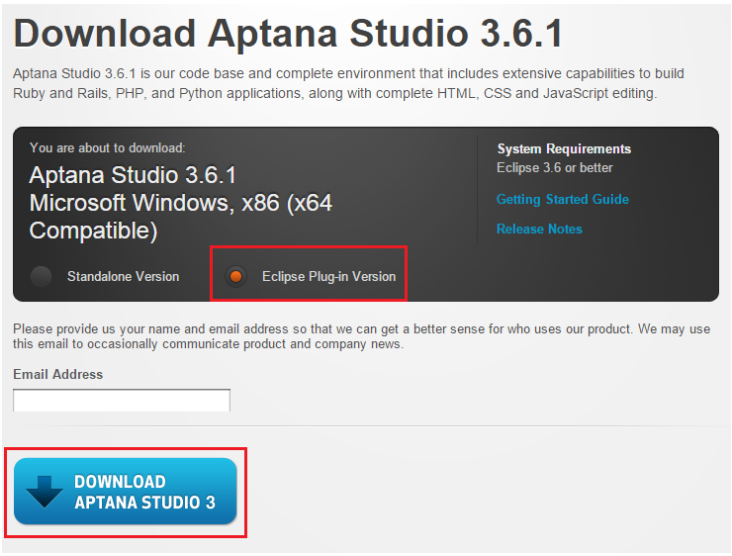
4. *This will only provide IntelliSense for the current JavaScript file.*

In Eclipse

Dynamic Web TWAIN IntelliSense supports Eclipse V4.2 and later. In order to use the IntelliSense in Eclipse, you need to first install **Aptana Studio**.

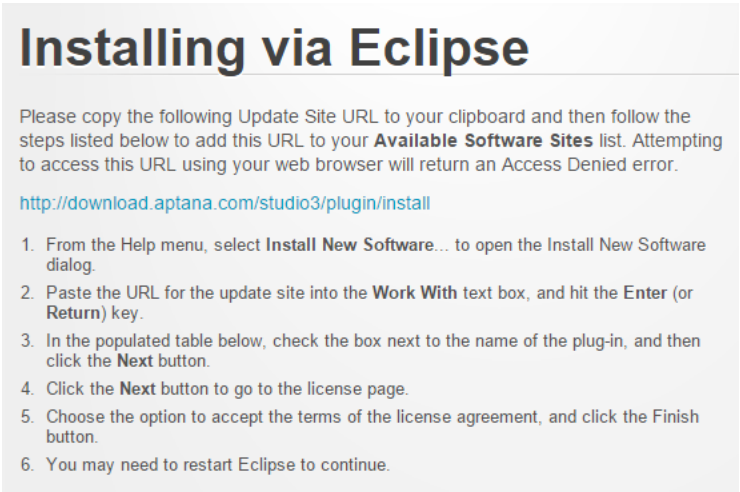
Step one. Installing Aptana Studio as an Eclipse plug-in

1. Download Aptana Studio Eclipse Plug-in Version 3.6 or above from [Aptana download page](#).



2. Choose **Eclipse Plug-in Version** and click on **Download Aptana Studio 3**.

3. Then you will be redirected to **Installing via Eclipse** page as shown below.

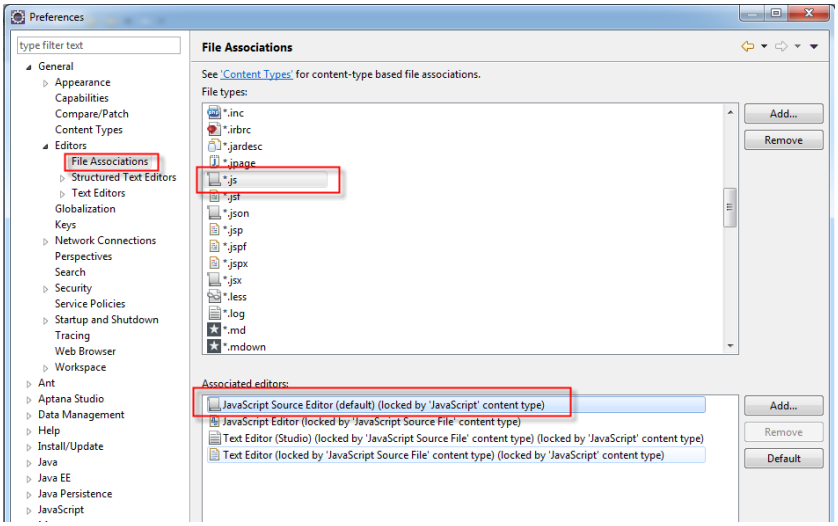


4. Please follow the six steps to complete the installation.

Step two. Configuring Eclipse

1. Open Eclipse IDE and navigate to the menu ***Window*** –> ***Preferences***.

2. In the ***Preferences*** dialog, expand ***General*** –> ***Editors*** –> ***File Associations*** -> Select ****.js*** from the ***File types*** list -> choose ***JavaScript Source Editor*** at ***Associated editors*** and then click on ***Default*** button.

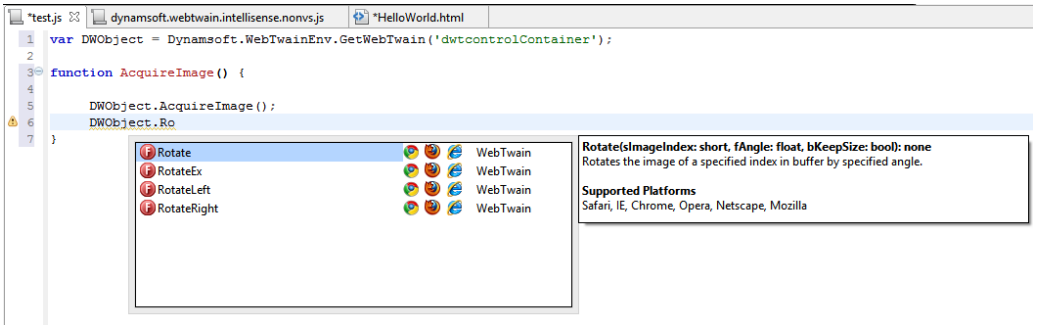


Step three. Using the IntelliSense

1. Make sure that *dynamsoft.webtwain.intellisense.nonvs.js* file is in your web project. If not, please add it to your project manually. Typically, it is under *C:\Program Files (x86)\Dynamsoft\Dynamic Web TWAIN* {Version Number} {Trial}*Resources*.

2. Open *dynamsoft.webtwain.intellisense.nonvs.js* using the default JavaScript Source Editor.

3. That's it! To verify it, you can create or open any existing JavaScript file and type 'DWObject' and you should get functions and comments in the popup list after pressing '.'



Customizing the Dynamic Web TWAIN Object

Naming the Dynamic Web TWAIN object

By default, the (first) Dynamic Web TWAIN object is named "**DWObject**". You should set it before using any other Dynamic Web TWAIN's properties or methods. A good place to do this is the built-in function ***Dynamsoft_OnReady***. For example, in our Hello World sample:

```
<html>
<head><title>Hello World</title>
   <script type="text/javascript" src="Resources/dynamsoft.webtwain.initiate.js"> </script>
   <script type="text/javascript" src="Resources/dynamsoft.webtwain.config.js"> </script>
</head>
<body>
   <input type="button" value="Scan" onclick="AcquireImage();" />
   <div id="dwtcontrolContainer"> </div>
   <script type="text/javascript">
          var DWObject;
          function Dynamsoft_OnReady(){
            DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
          }
          function AcquireImage(){
            if(DWObject) {
                DWObject.IfDisableSourceAfterAcquire = true;
                DWObject.SelectSource();
                DWObject.OpenSource();
                DWObject.AcquireImage();
            }
          }
   </script>
</body></html>
```

The div with id 'dwtcontrolContainer' is the place holder for Dynamic Web TWAIN. Its name and size are defined in the file *dynamsoft.webtwain.config.js* as shown below. You can change it if necessary.

```
    Dynamsoft.WebTwainEnv.Containers = [{ContainerId:'dwtcontrolContainer',Width:270,Height:350}];
```

Changing the Size of the Viewer

You can simply change the size of the container (the built-in viewer) in *dynamsoft.webtwain.config.js.* You can both use a number or a percentage here. For example

```
Dynamsoft.WebTwainEnv.Containers = [{ContainerId:'dwtcontrolContainer',Width: '50%',Height:350}];
```

Changing the Look of the Installation Prompt

If Dynamic Web TWAIN is not installed, you will see this built-in interface that prompts the user to install the SDK

In most cases, you might want to change the 'Dynamsoft Branding'. The following is how you can achieve this

- The 'branding' is defined in the file **\Resources\reference*hint.css***

```
.DYNLogo
{
    background:url(logo.gif) left top no-repeat;
    width:159px;
    height:39px;
}
```

As you can see, the following image (**\Resources\reference*logo.gif***) is the file you need to change. The simplest way is to replace it with your own logo but keep the same name and size.

- If you'd like to change further the style of this prompt, please feel free to change the css mentioned above or change the code in the file **\Resources*dynamsoft.webtwain.install.js***

```
function OnWebTwainNotFoundOnWindowsCallback(ProductName, InstallerUrl, bHTML5, bIE, bSafari, bSSL, strIEVersion) { }
//This callback is triggered when Dynamic Web TWAIN is not installed on a PC running Windows
function OnWebTwainNotFoundOnMacCallback(ProductName, InstallerUrl, bHTML5, bIE, bSafari, bSSL, strIEVersion) { }
//This callback is triggered when Dynamic Web TWAIN is not installed on a MAC
function OnWebTwainOldPluginNotAllowedCallback(ProductName) { }
//This callback is triggered when Dynamic Web TWAIN is disabled by a non-IE browser
function OnWebTwainNeedUpgradeCallback(ProductName, InstallerUrl, bHTML5, bMac, bIE, bSafari, bSSL, strIEVersion) { }
//This callback is triggered when Dynamic Web TWAIN installed on the machine is older than the
//one on the server and upgrade is needed
```

Using Dynamic Web TWAIN

Dynamic Web TWAIN is automatically initialized in the accompanying JS files. Once the Dynamic Web TWAIN object is initialized, you can control it like a normal JS object. You can refer to our online API Documentation to check all built-in properties, methods and events of Dynamic Web TWAIN.

Basically, there are 3 ways to use Dynamic Web TWAIN:

Properties

Properties are used to get or set a certain value in the Dynamic Web TWAIN object at runtime such as Resolution, Duplex, IfShowUI, etc.

```
// Property
DWObject.Resolution = 200; // Scan pages in 200 DPI
```

Methods

Methods are used to call the built-in functions of the Dynamic Web TWAIN object such as AcquireImage, SaveAsJPEG, Rotate, etc. The syntax is fairly simple:

```
// Method

/// <summary>
/// Rotates the image of a specified index in buffer by a specified angle.
/// </summary>
/// <param name="sImageIndex" type="short"> specifies the index of image in buffer. The index is 0-based.</param>
/// <param name="fAngle" type="float"> specifies the angle.</param>
/// <param name="bKeepSize" type="bool"> specifies whether to keep the original size</param>
/// <returns type="bool"></returns>
DWObject.Rotate(0, 45, false); // rotate the 1st image in the buffer by 45 degrees
```

Events

Events are triggered when certain trigger points are reached. For example, we have an OnMouseClick event for mouse clicking, an OnPostTransfer event for the end of transferring one image, etc. Compared with Properties and Methods, Events are a little bit tricky to use. Here we'll talk about it a little more.

Handling Events

Add an event listener

To add an event listener, you can use the built-in method RegisterEvent. Please refer to the sample code below:

```
<script type="text/javascript">
Dynamsoft.WebTwainEnv.RegisterEvent('OnWebTwainReady', Dynamsoft_OnReady);
var DWObject;
/* OnWebTwainReady event fires as soon as Dynamic Web TWAIN is initialized and ready to be used. It is the best place to add event listeners */
function Dynamsoft_OnReady() {
   DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
   DWObject.RegisterEvent("OnPostTransfer", Dynamsoft_OnPostTransfer);
}
function Dynamsoft_OnPostTransfer() {
   /* This event OnPostTransfer will be triggered after a transfer ends. */
   /* your code goes here*/
}
</script>
```

In the above code, we added the JavaScript function Dynamsoft_OnPostTransfer() as an event listener for the event **OnPostTransfer**.

Alternatively, you can write code like this:

```
<script type="text/javascript">
Dynamsoft.WebTwainEnv.RegisterEvent('OnWebTwainReady', Dynamsoft_OnReady);
var DWObject;
function Dynamsoft_OnReady() {
   DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
   DWObject.RegisterEvent("OnPostTransfer", function() {
      /* your code goes here*/
   };
}
</script>
```

Event with parameter(s)

Some of the events have parameter(s). Take OnMouseClick Event as an example:

OnMouseClick(short sImageIndex) `/* sImageIndex is the index of the image you clicked on*/`

When you create the corresponding JavaScript function (A.K.A., the event listener), you can include the parameter(s) and retrieve the value at runtime.

```
function DynamicWebTwain_OnMouseClick(index) {
    CurrentImage.value = index + 1;
}
```

Or

```
DWObject.RegisterEvent("OnPostTransfer", function(index) {
    CurrentImage.value = index + 1;
};
```

Special Event - 'OnWebTwainReady'

In version 10.1, there are 17 events in total. Please refer to the online API Documentation. Of the 17 events, there is one called ***'OnWebTwainReady'*** which is special. Basically this event fires as soon as the Dynamic Web TWAIN object is initialized and ready to be used. As you have seen earlier in the document, the recommended way to use it is:

```
    <script type="text/javascript">
    Dynamsoft.WebTwainEnv.RegisterEvent('OnWebTwainReady', Dynamsoft_OnReady);
    var DWObject;
    function Dynamsoft_OnReady() {
       DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
    }
    </script>
```

Or

```
    <script type="text/javascript">
    Dynamsoft.WebTwainEnv.RegisterEvent('OnWebTwainReady', function() {
       DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
    };
    </script>
```

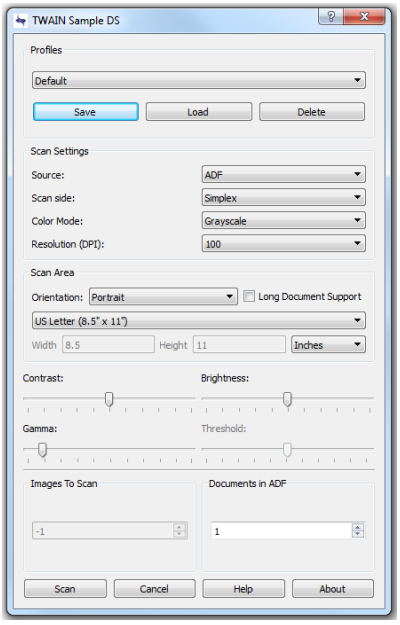
Exploring the Features

As we introduced in the previous section Three ways to use Dynamic Web TWAIN, You can control Dynamic Web TWAIN objects in three ways: Properties, Methods and Events. The complete list of all built-in properties, methods and events of Dynamic Web TWAIN is available in our online API Documentation for your reference.

Here we will introduce Dynamic Web TWAIN's functionality in more details:

Customizing your scan settings

Before you start an actual scan session, you can choose how you want your documents to be scanned. Normally, you can change all of the settings in the scanner's built-in User Interface. Take the Virtual scanner for example:



All these settings might be overwhelming for end users, especially for those without a technical background. With Dynamic Web TWAIN, you can customize all these settings in your JavaScript code. For example:

```
DWObject.SelectSource();
DWObject.OpenSource();                  // You should customize the settings after opening a source
DWObject.IfShowUI = false;              // Hide the User Interface of the scanner
DWObject.IfFeederEnabled = true;        // Use the document feeder to scan in batches
DWObject.IfDuplexEnabled = false;       // Scan in Simplex mode (only 1 side of the page)
DWObject.PixelType = EnumDWT_PixelType.TWPT_GRAY; // Scan pages in GRAY
DWObject.Resolution = 200;              // Scan pages in 200 DPI
DWObject.AcquireImage();                // Start scanning
```

Related settings

BitDepth	Brightness	Contrast	PageSize	Resolution
PixelType	PixelFlavor	IfDuplexEnabled	IfFeederEnabled	IfShowUI
XferCount	IfAutoDiscardBlankpages	IfAutomaticDeskew	IfAutomaticBorderDetection	

Manipulating the image(s)

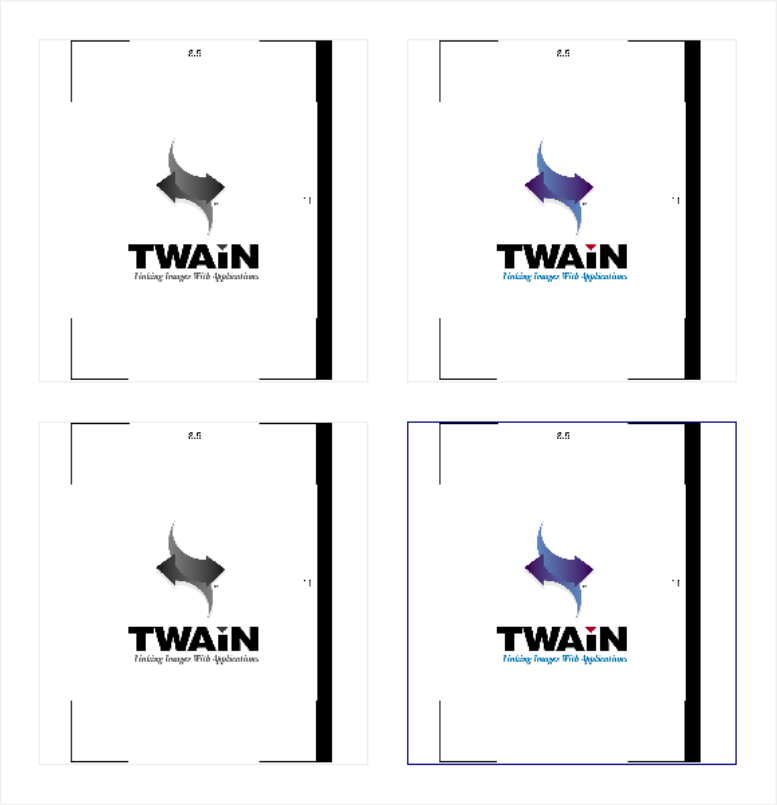
When you have scanned or loaded images in Dynamic Web TWAIN, you can start manipulating the images. You can:

1. Go through each image by changing the property CurrentImageIndexInBuffer

```
DWObject.CurrentImageIndexInBuffer = 2; // Show the 3rd image in the buffer
```

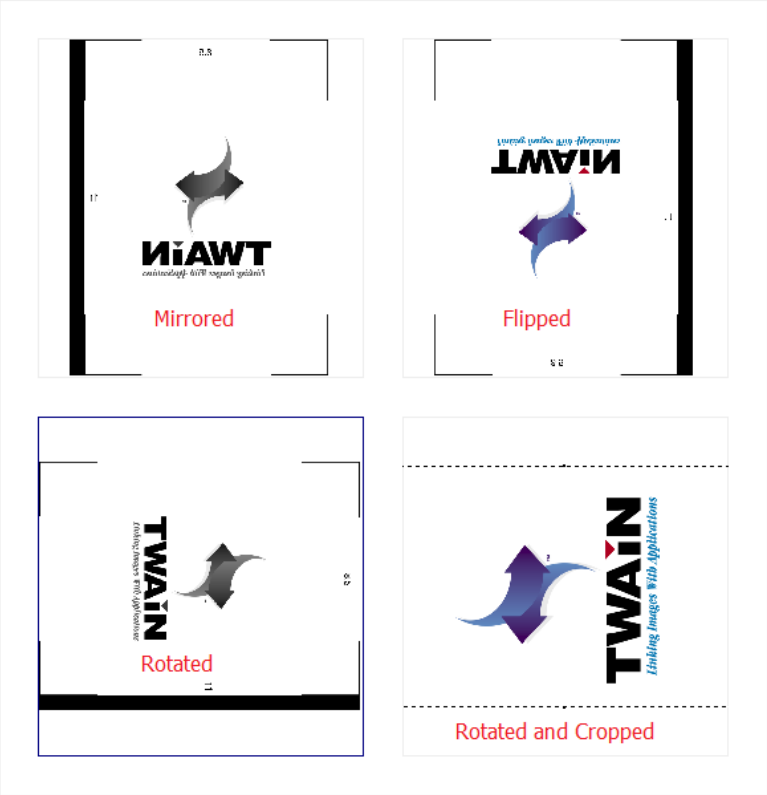
2. Show multiple images by changing the view mode (other than 1*1 or -1*-1) using SetViewMode()

```
DWObject.SetViewMode(2, 2); // Show images in buffer with 2 * 2 view
```



3. Rotate, flip, mirror or crop an image, etc.

```
DWObject.Mirror(0);
DWObject.Flip(1);
DWObject.RotateRight(2);
DWObject.Crop(3,101,243,680,831);
DWObject.RotateLeft(3);
```



Also, you can remove/delete an image by its index or remove/delete selected or all images at once. The methods are RemoveImage, RemoveAllSelectedImages, RemoveAllImages

Related methods

ChangeImageSize	CutToClipboard	LoadDibFromClipboard	RemoveAllImages	RotateLeft
CopyToClipboard	CutFrameToClipboard	Mirror	RemoveImage	RotateRight
Crop	Erase	MoveImage	Rotate	SetDPI
CropToClipboard	Flip	RemoveAllSelectedImages	RotateEx	SwitchImage

Using Thumbnails

Currently Dynamic Web TWAIN does not have a built-in thumbnail. To show thumbnails, you can put 2 Dynamic Web TWAIN controls on the same page.

How to define two controls within one page

In the HtmlPage.html

```
<html>
<head>
    <title> Thumbnails </title>
</head>
<body>
    <div id="dwtcontrolContainer" style="float:left; width: 120px;"></div>
    <div id="dwtcontrolContainerLargeViewer" style="float:left;"></div>
    <script type="text/javascript" src="Resources/dynamsoft.webtwain.initiate.js"></script>
    <script type="text/javascript" src="Resources/dynamsoft.webtwain.config.js"></script>
    <script type="text/javascript">
      var DWObject, DWObjectLargeViewer;
      Dynamsoft.WebTwainEnv.RegisterEvent('OnWebTwainReady', Dynamsoft_OnReady);
      function Dynamsoft_OnReady() {
        DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer'); // create a thumbnail
        DWObjectLargeViewer = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainerLargeViewer');
        // create a large viewer
      }
    </script>
</body>
</html>
```

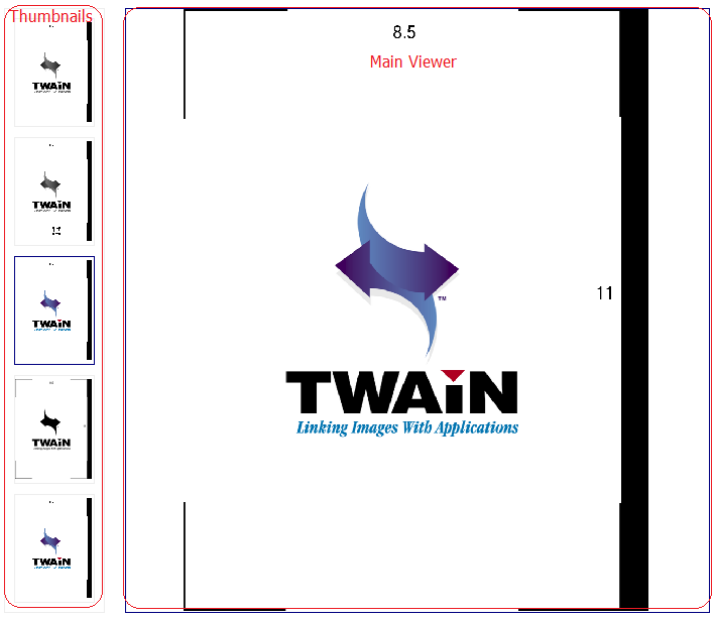
In the dynamsoft.webtwain.config.js

```
Dynamsoft.WebTwainEnv.Containers = [{ContainerId:'dwtcontrolContainer', Width: 120, Height: 350},
{ContainerId:'dwtcontrolContainerLargeViewer', Width: 270, Height: 350},];
///
Dynamsoft.WebTwainEnv.ProductKey = '******';
///
Dynamsoft.WebTwainEnv.Trial = true;
///
Dynamsoft.WebTwainEnv.Debug = false;
```

Below is what it looks like:

*NOTE: the control on the left has a view mode of 1*5 and the one on the right has a view mode of 1*1 and it is set to hold only 1 image at a time.*

```
DWObject.SetViewMode(1, 5);
DWObjectLargeViewer.SetViewMode(1, 1);     // This is actually the default setting
DWObjectLargeViewer.MaxImagesInBuffer = 1; // Set it to hold one image only
```



With the above implementation, you only need to use the object on the left (the thumbnails for scanning/editing/uploading, etc. The main viewer shows only the current image. When the current image changes, you need to update the main viewer. For example, when you click on one of the images in the thumbnails, you need to copy it to the main viewer:

```
DWObject.RegisterEvent("OnMouseClick", Dynamsoft_OnMouseClick);
function Dynamsoft_OnMouseClick(index) {
  DWObject.CopyToClipboard(index);  // Copy the image you just clicked on to the clipboard
  DWObjectLargeViewer.LoadDibFromClipboard(); // Load the same image from clipboard into the large viewer
}
```

Scanning a large number of Documents - Disk Caching

When you scan documents in hundreds or even thousands, the disk caching feature will come in handy. The related properties are IfAllowLocalCache and BufferMemoryLimit.

By default, the value of **IfAllowLocalCache** is false. All the image data is stored in DIB format as part of the memory occupied by the browser. However, there is a limit of up to 2,048M of physical memory for all 32-bit browsers. The more images you scan, the more memory the browser will consume. Once the maximum

memory (2,048M) is reached (mostly only around 1,300M~1,500M because our SDK isn't the only one running in the browser), the newly scanned images will not be loaded into Dynamic Web TWAIN as there is not enough memory and the browser might appear to be hanging.

To turn on disk caching, you only need to set **IfAllowLocalCache** to true. With this mode on, the default value of **BufferMemoryLimit** is 800M. You can adjust the maximum memory setting according to your needs.

If **BufferMemoryLimit** < 800M, all images will be stored in compressed mode in physical memory. When the memory limit is reached, the image data will be stored on the local hard disk.

If **BufferMemoryLimit** >= 800M, the images will be stored in non-compressed mode in physical memory. When the memory limit is reached, the image data will be stored on the local hard disk.

NOTE:

1. *All cached data will be encrypted and can only be accessed by Dynamic Web TWAIN.*

   On Windows Vista and later

   For ActiveX and Plug-in Edition:

   The cached data is stored at "C:\Users\{User Name}\AppData\LocalLow\Dynamsoft\cache".

   For HTML5 Edition:

   It is stored at "C:\Windows\SysWOW64 {or system32}\Dynamsoft\DynamicWebTwain\ForChrome\cache"

   On Windows XP

   For ActiveX and Plug-in Edition:

   The cached data is stored at "C:\Documents and Settings\ {User Name}\Application Data\Dynamsoft\cache".

   For HTML5 Edition:

   It is stored at "C:\WINDOWS\SysWOW64 {or system32}\Dynamsoft\DynamicWebTwain\ForChrome\cache"

2. *When the scanning page is closed, the cached data will be destroyed and removed from the hard disk automatically.*

3. *Although you can scan and load as many images as you want into Dynamic Web TWAIN control, you might not be able to upload all these images to the web server as there is a memory limit (2048M) for 32-bit browsers. When that limit is reached, your browser will probably freeze and/or crash.*

The following example shows how to enable disk caching:

```
var DWObject;
Dynamsoft.WebTwainEnv.RegisterEvent('OnWebTwainReady', function() {
  DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
  DWObject.IfAllowLocalCache = true;
};
function AcquireImage(){
  DWObject.SelectSource();
  DWObject.OpenSource();
  DWObject.AcquireImage();
}
```

Using the Image Editor

What is Image Editor

The Image Editor is a built-in feature of Dynamic Web TWAIN. You can use the Image Editor to manage images.

What can you do with the Image Editor

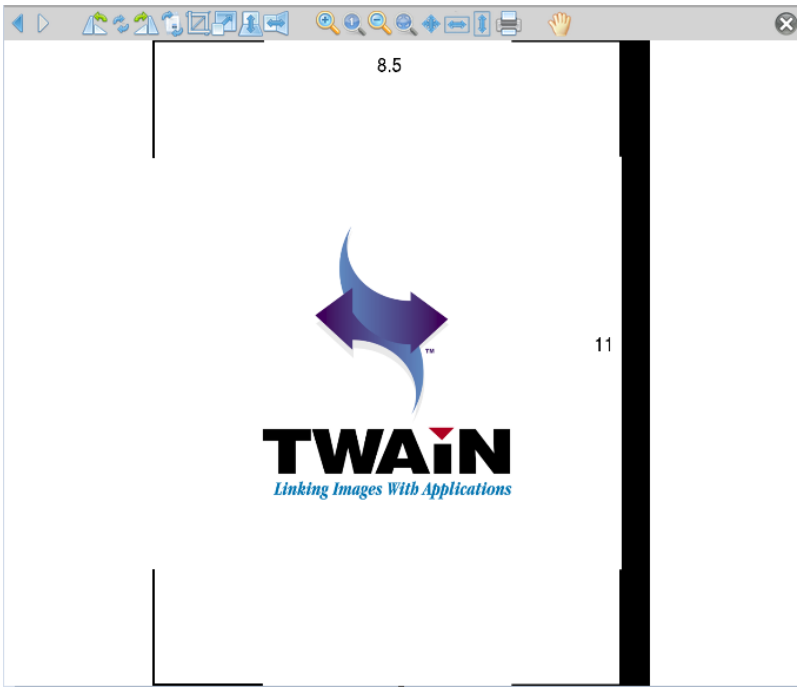
In Image Viewer, you can:

1. Go through all the images currently scanned or loaded
2. Edit an image in the following ways: rotate, mirror, crop, change size, etc.
3. Print the images, etc.

Opening Image Editor

When there is at least one image in the buffer, you can use the method ShowImageEditor() to show the editor window:

```
DWObject.ShowImageEditor();
```



Loading local image(s) into Dynamic Web TWAIN

Preparation

First of all, bear in mind that as a lightweight component running in web browsers, Dynamic Web TWAIN is only designed to deal with the most basic images in the following formats: BMP, JPEG, PNG, TIFF and PDF. We only guarantee that images generated by Dynamic Web TWAIN can be successfully loaded. If you are trying to load an image that was not generated by Dynamic Web TWAIN, you can check out this article.

Calling the methods

With Dynamic Web TWAIN, you can load local images with the methods LoadImage() or LoadImageEx(). Below is a simple code snippet:

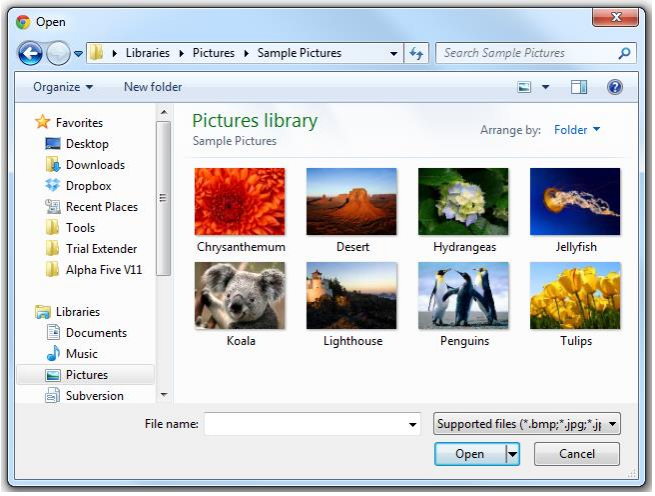
```
DWObject.LoadImage("C:\\WebTWAIN\\Images\\ImageData.jpg", optionalAsyncSuccessFunc,
optionalAsyncFailureFunc);
DWObject.LoadImageEx("C:\\WebTWAIN\\Images\\ImageData.jpg", EnumDWT_ImageType.IT_JPG,
optionalAsyncSuccessFunc, optionalAsyncFailureFunc); // ImageType: JPG

//Callback functions for async APIs
function optionalAsyncSuccessFunc(){
    console.log('successful');
}
function optionalAsyncFailureFunc(errorCode, errorString){
    alert(errorString);
}
```

Please note that the last two parameters **optionalAsyncSuccessFunc** and **optionalAsyncFailureFunc** are optional callback functions.

As you can see, you need to provide the complete file path in order to load an image. This is somewhat clumsy especially when you need to load more than one image. But no worries, Dynamic Web TWAIN can open a “Select File…” dialog for you to locate the image(s) you want to load. And like other properties and methods, it’s very easy to use. Below is a code snippet:

```
DWObject.IfShowFileDialog = true;
DWObject.LoadImageEx("", EnumDWT_ImageType.IT_ALL); // ImageType: ALL (BMP, JPG, PNG, PDF, TIFF)
```

Please note that the second parameter “ImageType” in the method **LoadImageEx**() would determine the file filter in the “Select File…” dialog.



Saving image(s) locally

Preparation

Dynamic Web TWAIN can save all scanned or loaded images locally in the following formats: BMP, JPEG, PNG, TIFF (single-page or multi-page) and PDF (single-page or multi-page).

Calling the methods

With Dynamic Web TWAIN, you can choose one of the following methods to save an image or images:

Format	Methods
One-Page File	SaveAsBMP() SaveAsJPEG() SaveAsPDF() SaveAsPNG() SaveAsTIFF()
Multi-page PDF	SaveSelectedImagesAsMultiPagePDF() SaveAllAsPDF()
Multi-page TIFF	SaveAllAsMultiPageTIFF() SaveSelectedImagesAsMultiPageTIFF()

Code snippet:

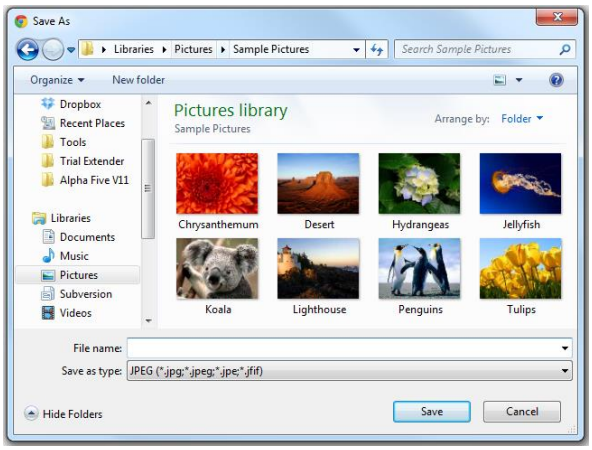
```
//Use it synchronously
DWObject.SaveAsJPEG("C:\\WebTWAIN\\Images\\ImageData.jpg", 0);

//Use it asynchronously
```

```
DWObject.SaveAllAsPDF("C:\\WebTWAIN\\Images\\ImageData.pdf", optionalAsyncSuccessFunc,
optionalAsyncFailureFunc);

//Callback functions for Async APIs
function optionalAsyncSuccessFunc(){
    console.log('successful');
}
function optionalAsyncFailureFunc(errorCode, errorString){
    alert(errorString);
}
```

From the above code, you can see that you need to provide the complete file path to save an image locally which is sometimes inconvenient. But no worries, just like loading an image, Dynamic Web TWAIN can also open a "Save As…" dialog for you to locate the path that you want to save the image(s) to. Below is a code snippet:

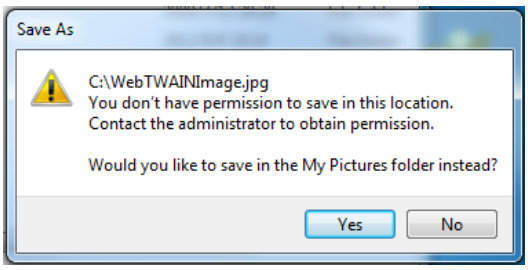
```
DWObject.IfShowFileDialog = true;
DWObject.SaveAsJPEG("",0);
```

This will bring up this dialog box with the "Save as type" specified by the method you use:



NOTE:

On Windows Vista and above, Microsoft has strengthened security which means you can only save images to certain places where you have write permission. If you try to save to other places, you will get the below error message. You can then save to a different directory or first obtain permission for that directory.



Uploading image(s) to the web server

Preparation

Before we upload the image(s), we need to set the server IP/name, port as well as define the path for the action page. An action page is used to receive the image data and handle all the server-side operation like saving the data on the hard disk or database, etc.

Here is an example:

```
var strHTTPServer = location.hostname;
DWObject.HTTPPort = location.port == "" ? 80 : location.port;
var CurrentPathName = unescape(location.pathname);
var CurrentPath =  CurrentPathName.substring(0, CurrentPathName.lastIndexOf("/") + 1);
var strActionPage = CurrentPath + "actionPage.aspx";
var uploadfilename = "TestImage.pdf";
```

strHTTPServer is used to store the server name which specifies which server the image(s) will be uploaded to. You can also use the server's IP for the same purpose. If you want to upload image(s) to the same server as the current page, we suggest you use "**location.hostname**" to get the hostname at runtime.

The property HTTPPort specifies the HTTP port to be used for the upload. Normally, port 80 is for HTTP, port 443 is for HTTPS, etc. If you are not sure about the port number, it's recommended that you use "location.port == "" ? 80 : location.port" to get the current port number at runtime.

CurrentPathName and **CurrentPath** are used to build the relative path of the action page.

strActionPage stores the relative path of the action page.

uploadfilename stores the file name for the uploaded image(s). You should change the extension of the name accordingly.

NOTE:

In version 10.0 and above, we are using the browser as the upload agent. Due to browser security restrictions, client-side scripts (e.g., JavaScript) are not allowed to make requests to another domain. Therefore, when you try to upload an image to a server with a different domain, subdomain, port, or protocol, you need to configure your server to allow such requests by adding one HTTP Response Header, namely:

```
Access-Control-Allow-Origin: *
// the asterisk wild-card permits scripts hosted on any site to load your resources
```

Take IIS 7 for example, what you need to do is merge the following lines into the web.config file at the root of your application / site:

```
<?xml version="1.0" encoding="utf-8"?>
  <configuration>
    <system.webServer>
      <httpProtocol>
        <customHeaders>
          <add name="Access-Control-Allow-Origin" value="*" />
          <add name="Access-Control-Allow-Methods" value="OPTIONS,POST,GET,PUT"/>
          <add name="Access-Control-Allow-Headers" value="x-requested-with"/>
          <add name="Access-Control-Allow-Credentials" value="true" />
        </customHeaders>
      </httpProtocol>
    </system.webServer>
  </configuration>
```

If you don't have a web.config file already, just create a new file called "web.config" and add the snippet above.

Calling the methods

Now, we can call one of the HTTP upload methods to upload the image(s). We have 7 methods:

Format	Method
All files	HTTPUploadThroughPostDirectly()
All supported image formats	HTTPUploadThroughPost() HTTPUploadThroughPostEx()
Multi-page PDF	HTTPUploadAllThroughPostAsPDF() HTTPUploadThroughPostAsMultiPagePDF()
Multi-page TIFF	HTTPUploadAllThroughPostAsMultiPageTIFF() HTTPUploadThroughPostAsMultiPageTIFF()

Let's take the method **HTTPUploadAllThroughPostAsPDF**() for example:

```
DWObject.HTTPUploadAllThroughPostAsPDF(
      strHTTPServer,
      strActionPage,
      uploadfilename,
     OnHttpUploadSuccess, OnHttpUploadFailure
);
```

With this method, all the images in the Dynamic Web TWAIN control will be sent to the web server as one multi-page PDF file.

In the above code, the parameters **OnHttpUploadSuccess** and **OnHttpUploadFailure** are optional callback functions. If they are present, the method is asynchronous, otherwise, the method is synchronous. It's recommended that you use the methods asynchronously to avoid possible browser hanging.

The following is a simple implementation of these two functions:

```
function OnHttpUploadSuccess(){
    console.log('successful');
}
function OnHttpUploadFailure(errorCode, errorString, sHttpResponse){
    alert(errorString + sHttpResponse);
}
```

If you want to upload one image as a single-page file, you can use HTTPUploadThroughPost or HTTPUploadThroughPostEx.

If you want to upload selected images as a multi-page file, you can use HTTPUploadThroughPostAsMultiPagePDF or HTTPUploadThroughPostAsMultiPageTIFF.

Action Page

The HTTP upload method(s) makes a standard HTTP post request to the action page on the server. The request contains the image data, image name, etc. In the action page, you can process the image data according to your requirements. Technically you can write the action page in any server-side language (C#, VB, PHP, Java, etc...).

Here is an example in C#:

This action page retrieves the image data from the current request object and saves it as a local file on the server.

```
HttpFileCollection files = HttpContext.Current.Request.Files;
HttpPostedFile uploadfile = files["RemoteFile"];
uploadfile.SaveAs(System.Web.HttpContext.Current.Request.MapPath(".") + "/" + uploadfile.FileName);
```

***Note**: Please note that '**RemoteFile**' is the default name/key for the uploaded image data. If necessary, you can change it using the property HttpFieldNameOfUploadedImage.*

To do the same thing in PHP:

```
$fileTempName = $_FILES['RemoteFile']['tmp_name'];
$fileSize = $_FILES['RemoteFile']['size'];
$fileName = $_FILES['RemoteFile']['name'];
move_uploaded_file($fileTempName, $fileName) ;
```

Uploading to FTP

Besides the HTTP upload methods, you can also use the FTP Upload methods to update image(s) to your FTP web server. The available APIs are:

Format	Method
All files	FTPUploadDirectly ()
All supported formats	FTPUpload() FTPUploadEx()
Multi-page PDF	FTPUploadAllAsPDF() FTPUploadAsMultiPagePDF()
Multi-page TIFF	FTPUploadAllAsMultiPageTIFF() FTPUploadAsMultiPageTIFF()

Uploading image(s) to a Database

Dynamic Web TWAIN doesn't save/upload the image(s) to your database directly. Instead, we can upload the image data to the action page first and then use the action page to store the image data in the database.

If you are not sure how to upload the image data to the server, please refer to the previous section Uploading image(s) to the web server.

Different database systems have different data types for image data. We normally use ***BLOB*** or ***varbinary*** in SQL Server, ***Long raw*** or ***BLOB*** in Oracle, ***BLOB*** in MySQL, etc.

Here is an example in C# with SQL Server:

```
int iFileLength;
HttpFileCollection files = HttpContext.Current.Request.Files;
HttpPostedFile uploadfile = files["RemoteFile"];
String strImageName = uploadfile.FileName;

iFileLength = uploadfile.ContentLength;
Byte[] inputBuffer = new Byte[iFileLength];
System.IO.Stream inputStream;
inputStream = uploadfile.InputStream;
inputStream.Read(inputBuffer,0,iFileLength);

// add code to connect to database
String SqlCmdText = "INSERT INTO tblImage (strImageName,imgImageData) VALUES (@ImageName,@Image)";
System.Data.SqlClient.SqlCommand sqlCmdObj = new System.Data.SqlClient.SqlCommand(SqlCmdText,
sqlConnection);

sqlCmdObj.Parameters.Add("@Image",System.Data.SqlDbType.Binary,iFileLength).Value = inputBuffer;
```

```
sqlCmdObj.Parameters.Add("@ImageName",System.Data.SqlDbType.VarChar,255).Value = strImageName;

sqlConnection.Open();
sqlCmdObj.ExecuteNonQuery();
sqlConnection.Close();
```

We get the file object from the current request and write the image data to the byte array. In the SQL statement, we pass the byte array to the database as *System.Data.SqlDbType.Binary* and store the data in a BL field called "imgImageData".

Uploading image(s) with extra data

If you are not sure how to upload the image data to the server, please refer to the previous topic "Uploading image(s) to the web server".

Sometimes we need to pass more information to the server. For example, document type, employee ID, document description, etc. Since we don't have any options to pass extra data in the HTTP upload method, we need to use a method called SetHTTPFormField.

SetHTTPFormField(string sFieldName, string sFieldValue)

- string sFieldName: specifies the name of a text field in the web form.
- string sFieldValue: specifies the value of a text field in the web form.

We need to use this method before the HTTP Upload method. Here is an example:

```
DWObject.ClearAllHTTPFormField(); // Clear all fields first
DWObject.SetHTTPFormField("EmployeeID", "2012000054");
DWObject.SetHTTPFormField("DocumentType", "Invoice");
DWObject.SetHTTPFormField("DocumentDesc", "This is an invoice from ...");
```

In the action page, you can retrieve the data from the request object by the field names. For example:

```
String EmployeeID = HttpContext.Current.Request.Form["EmployeeID"];
```

Downloading image(s) from the web

You can use the method HTTPDownload() or HTTPDownloadEx() to download an image from the web server into Dynamic Web TWAIN.

```
DWObject.HTTPDownload("www.dynamsoft.com","/images/dwt-logo.png",
optionalAsyncSuccessFunc, optionalAsyncFailureFunc);
```

```
//Callback functions for async APIs
function optionalAsyncSuccessFunc (){
    console.log('successful');
}

function optionalAsyncFailureFunc (errorCode, errorString){
    alert(errorString);
}
```

This is especially useful when you want to review an image created and uploaded by Dynamic Web TWAIN. Even when the image data is stored in the database, you can write an action page to pull the data from the database and get it downloaded (in this case, you need to use the method HTTPDownloadEx because the image format needs to be specified explicitly). Besides the HTTP download methods, you can also use the FTP download methods to download image(s) from a FTP server. Available methods are FTPDownload, FTPDownloadEx, etc.

NOTE:

Same as mentioned earlier in the section Uploading image(s) to the web server, special configuration has to be made on the server to overcome browser security restrictions. When you try to download an image from a server with a different domain, subdomain, port, or protocol, you need to configure your server to allow such requests by adding one HTTP Response Header, namely:

```
Access-Control-Allow-Origin: *
```

Take IIS 7 for example, what you need to do is merge the following lines into the web.config file at the root of your application / site:

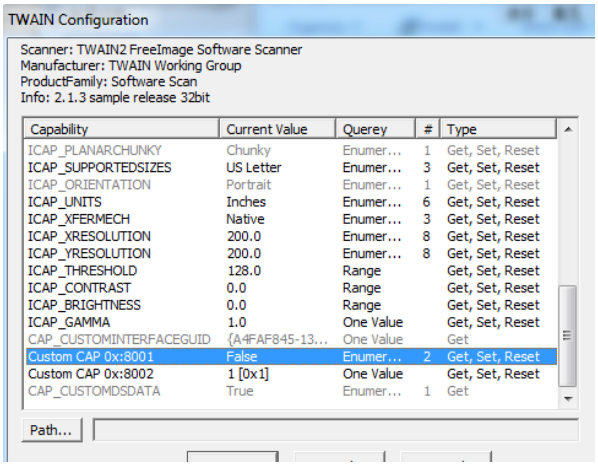
```
<?xml version="1.0" encoding="utf-8"?>
  <configuration>
    <system.webServer>
      <httpProtocol>
        <customHeaders>
          <add name="Access-Control-Allow-Origin" value="*" />
          <add name="Access-Control-Allow-Methods" value="OPTIONS,POST,GET,PUT"/>
          <add name="Access-Control-Allow-Headers" value="x-requested-with"/>
          <add name="Access-Control-Allow-Credentials" value="true" />
        </customHeaders>
      </httpProtocol>
    </system.webServer>
  </configuration>
```

If you don't have a web.config file already, just create a new file called "web.config" and add the snippet above.

Using Custom capabilities

To use a custom capability, you need to know what code represents this capability. To do this, you can follow the steps below:

1. Install the TWAIN sample application
    - 32-bit: http://www.dynamsoft.com/download/support/twainapp.win32.installer.msi
    - 64-bit: http://www.dynamsoft.com/download/support/twainapp.win64.installer.msi
2. Use the TWAIN Sample App to open the source and check what the code of the capability is.



3. In the above example, code “0x8001” is the hexadecimal value for the highlighted custom capability. Now, we can use the code to negotiate the capability with the scanner driver:

```
DWObject.Capability = 0x8001;
DWObject.CapType = EnumDWT_CapType.TWON_ONEVALUE; // TWON_ONEVALUE
DWObject.CapValue = 1;
if (DWObject.CapSet())
    alert("Successful");
else
    alert("Source doesn't support this capability");
```

Please refer to [PDF] How to perform Capability Negotiation for more details.

Using Add-ons

Basic Information

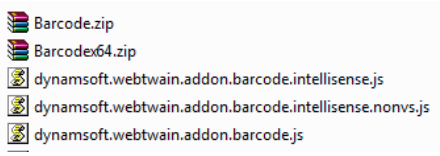
In version 10.1, barcode reader was added to Dynamic Web TWAIN as the first add-on. In version 10.2, webcam add-on was added. Then we added the PDF Rasterizer in version 11.2. Here we'll discuss how to use these add-ons.

How to use the barcode reader

The barcode reader was designed for simplicity. There are merely 3 APIs

Addon.Barcode.Download, Addon.Barcode.Read, Addon.Barcode.ReadRect

At the same time, there are 3 JS files and 2 zip files directly related to this add-on, they are found under /Resources/addon/



In order to use this add-on, you need to first make sure you have referenced the file dynamsoft.webtwain.addon.barcode.js in your code.

```
<script type="text/javascript" src="Resources/dynamsoft.webtwain.initiate.js"> </script>
<script type="text/javascript" src="Resources/dynamsoft.webtwain.config.js"> </script>
<script type="text/javascript" src="Resources/addon/dynamsoft.webtwain.addon.barcode.js"> </script>
```

Let's start with a simple code snippet

```
function ReadBarcode() {
    if (DWObject) {
        DWObject.Addon.Barcode.Download(
            "http://localhost/dwt/addon/Barcode.zip", function () {
                DWObject.Addon.Barcode.Read(DWObject.CurrentImageIndexInBuffer,
                    EnumDWT_BarcodeFormat.CODE_39, GetBarcodeInfo, GetErrorInfo);
            }, function (errorCode, errorString) {alert(errorString); });
    }
}
```

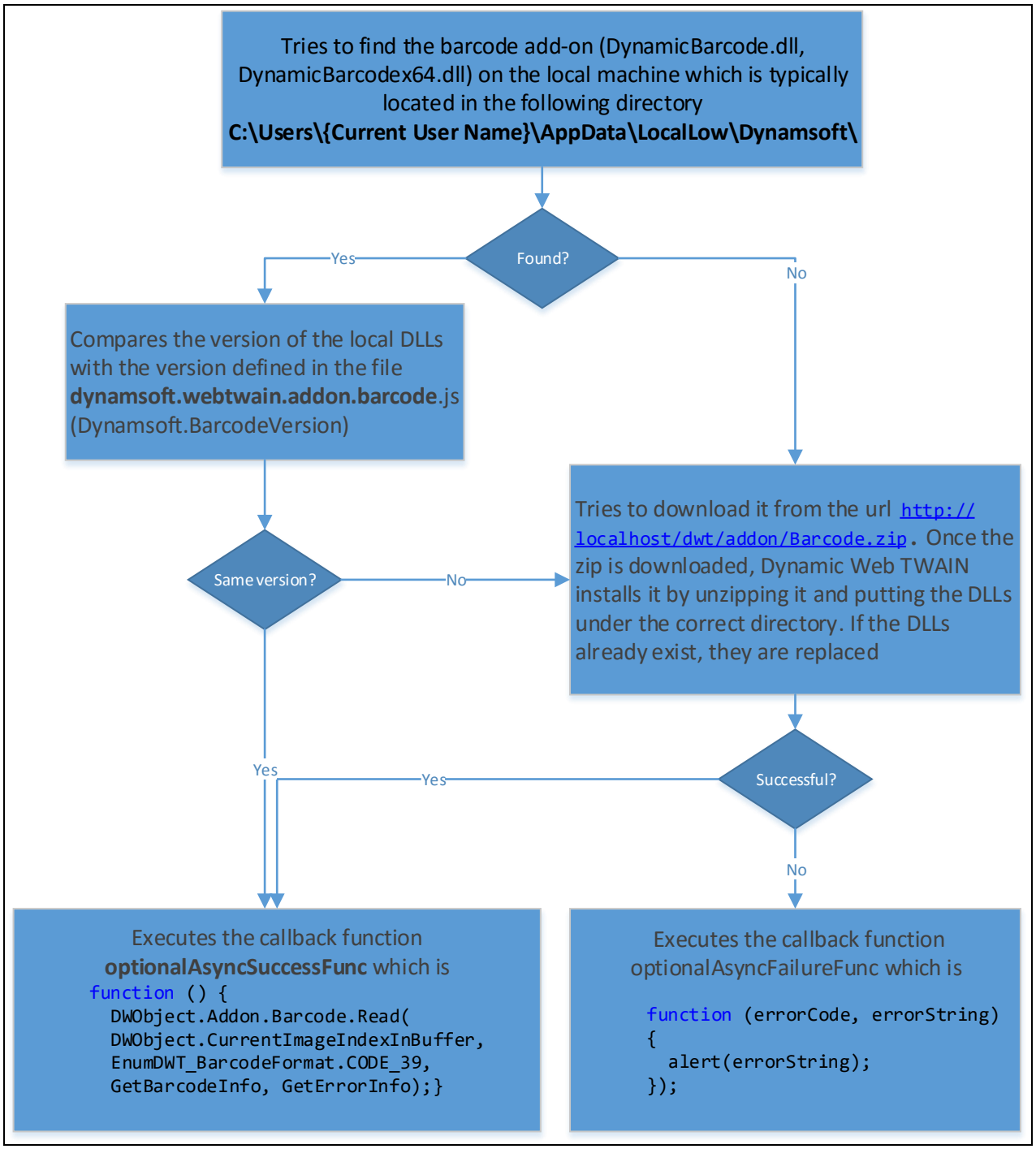
The first thing to note here is the JS-style '**asynchronous** syntax' of both '**Download'** and '**Read'** methods.

Here is how the 'download' method works

Syntax

```
.Addon.Barcode.Download(remoteFile, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc])
```

Workflow

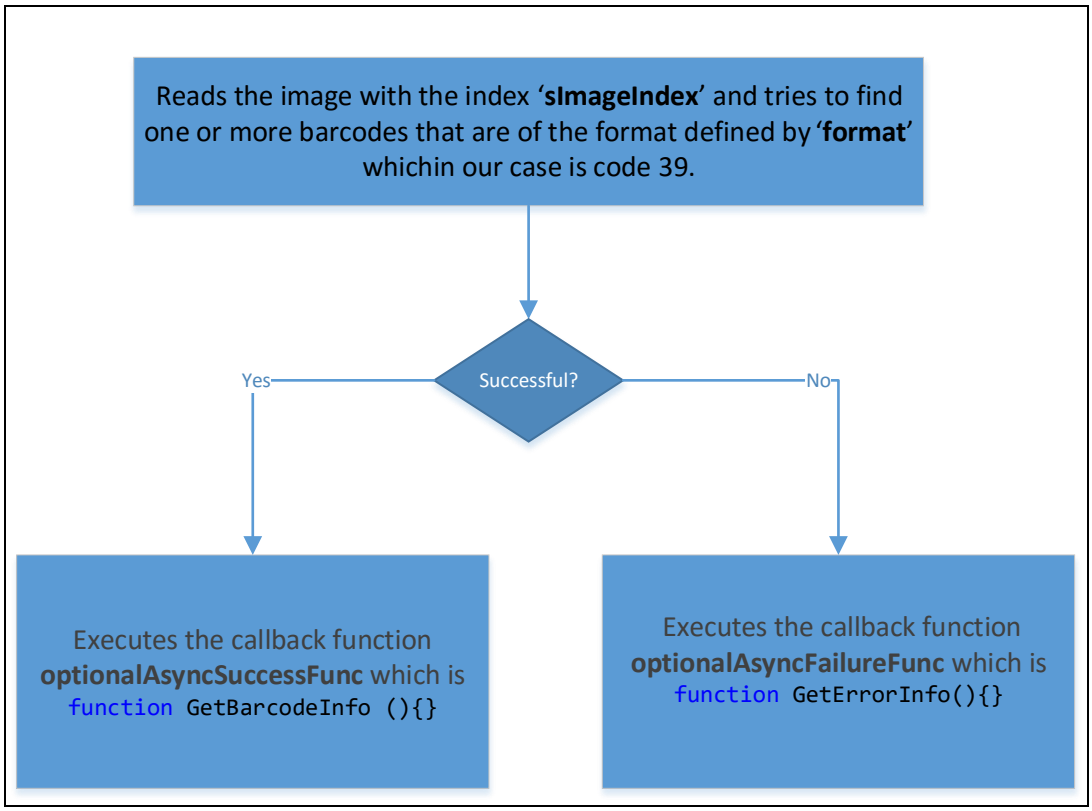


When the barcode reader (DLLs) is in place, barcode reading begins. Here is how the 'read' method works

Syntax

```
.Addon.Barcode.Read(sImageIndex, format, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
```

Workflow



Here is how we get the actual barcode information in the `function GetBarcodeInfo`

```
function GetBarcodeInfo(sImageIndex, result) {
    var count = result.GetCount();
    if (count == 0) { alert("The barcode for the selected format is not found."); return;
    } else {
        for (i = 0; i < count; i++) {
            var text = result.GetContent(i); var format = result.GetFormat(i);
            var x = result.GetX1(i); var y = result.GetY1(i);
            var barcodeText = ("barcode[" + (i + 1) + "]: " + text + "\n");
            barcodeText += ("format:" + (format==4?"Code 39":"Code 128") + "\n");
            barcodeText += ("x: " + x + " y:" + y + "\n");
            alert(barcodeText);
        }
    }
}
```

`result` is the key object that contains all the info about the barcode(s). More info can be found on the API document **Addon.Barcode.Read**.

Compared with the main API 'Read', the only difference of the last API 'ReadRect' is that it only reads the selected area of an image instead of the entire image. Because of this, it works much faster and is especially useful when all documents being scanned have a barcode in a fixed position. The following is a simple code snippet on how to use it.

```
function Dynamsoft_OnReady() {
    DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
    if (DWObject) {
        DWObject.RegisterEvent('OnImageAreaSelected',
            function (sImageIndex, left, top, right, bottom){
                iLeft = left; iTop = top; iRight = right; iBottom = bottom;
        });
    }
}
function ReadBarcode() {
   if (DWObject) {
       DWObject.Addon.Barcode.Download(
           "http://localhost/dwt/addon/Barcode.zip",
           function () {
               DWObject.Addon.Barcode.ReadRect(DWObject.CurrentImageIndexInBuffer,
                   iLeft, iTop, iRight, iBottom,
                   EnumDWT_BarcodeFormat.CODE_39, GetBarcodeInfo, GetErrorInfo);
           },
           function (errorCode, errorString) {
               alert(errorString);
           });
   }
}
```

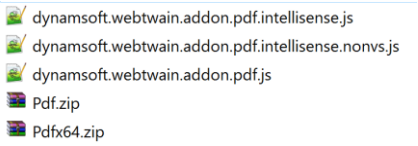
More info can be found on this page Addon.Barcode.ReadRect.

How to use the PDF Rasterizer

The PDF Rasterizer was designed to support non-image PDF files. There are merely 3 APIs

Addon.PDF.Download, Addon.PDF.SetResolution, Addon.PDF.SetConvertMode

There are 3 JS files and 2 zip files directly related to this add-on, they are found under /Resources/addon/



In order to use this add-on, the steps are

1. Reference the file dynamsoft.webtwain.addon.pdf.js in your code.

```
<script type="text/javascript" src="Resources/dynamsoft.webtwain.initiate.js"> </script>
<script type="text/javascript" src="Resources/dynamsoft.webtwain.config.js"> </script>
<script type="text/javascript" src="Resources/addon/dynamsoft.webtwain.addon.pdf.js"> </script>
```

2. Download the necessary DLL file and Set up the add-on

```
function Dynamsoft_OnReady() {
    DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
    if (DWObject) {
       DWObject.Addon.PDF.Download(
           "http://localhost/dwt/addon/Pdf.zip",
            function () { console.log('Successfully Downloaded PDF add-on');},
            function (errorCode, errorString) {alert(errorString); }
        );
       DWObject.Addon.PDF.SetResolution(200);
       DWObject.Addon.PDF.SetConvertMode(EnumDWT_ConverMode.CM_RENDERALL);
    }
}
```

How it works

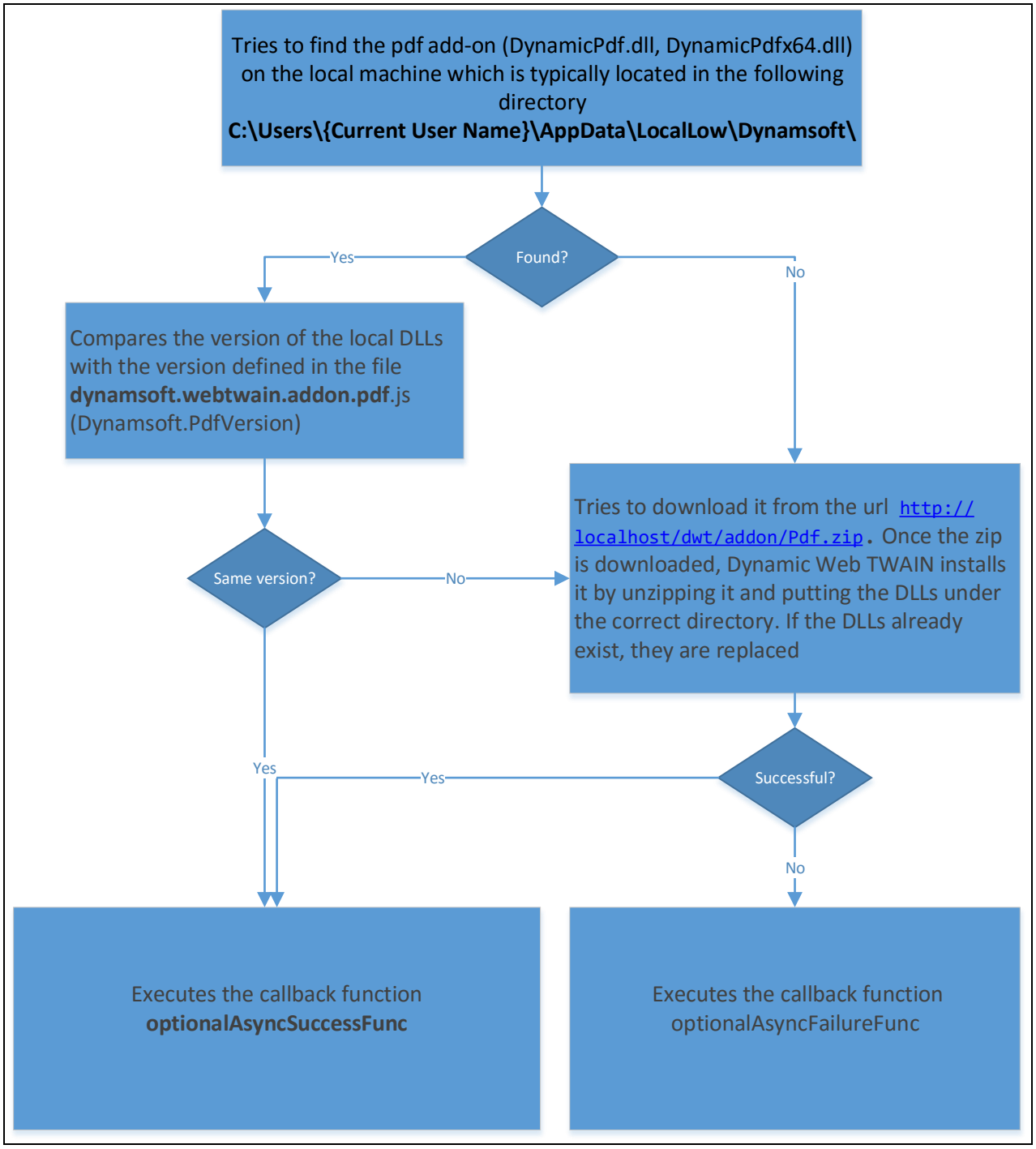
Dynamsoft has optimized this PDF Rasterizer internally so that when you have done the 2 steps above, the PDF rasterizer will be ready to work in your application. Basically, when you try to load or download any PDF file, the rasterizer will be called automatically to convert the PDF into images before loading them. In other words, the feature is turned on and you don't need to write any extra code to use it.

Here is how the 'download' method works

Syntax

```
.Addon.PDF.Download(remoteFile, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc])
```

Workflow



How to use the webcam add-on

The webcam add-on is based on the Microsoft DirectShow API. As an additional image capture engine, it has quite a few APIs.

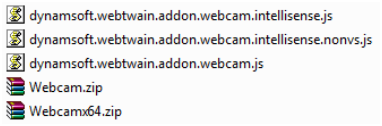
Basic APIs

Addon.Webcam.Download, Addon.Webcam.GetSourceList, Addon.Webcam.SelectSource
Addon.Webcam.CaptureImage, Addon.Webcam.CloseSource, Addon.Webcam.GetMediaType
Addon.Webcam.SetMediaType, Addon.Webcam.GetResolution, Addon.Webcam.SetResolution
Addon.Webcam.GetFrameRate, Addon.Webcam.SetFrameRate

Advanced APIs

Addon.Webcam.GetCameraControlPropertySetting Addon.Webcam.GetCameraControlPropertyMoreSetting
Addon.Webcam.SetCameraControlPropertySetting Addon.Webcam.GetVideoPropertySetting
Addon.Webcam.GetVideoPropertyMoreSetting Addon.Webcam.SetVideoPropertySetting

At the same time, there are 3 JS files and 2 zip files directly related to this add-on, they are found under /Resources/addon/



Make sure you reference the file dynamsoft.webtwain.addon.barcode.js when using this add-on.

```
<script type="text/javascript" src="Resources/dynamsoft.webtwain.initiate.js"> </script>
<script type="text/javascript" src="Resources/dynamsoft.webtwain.config.js"> </script>
<script type="text/javascript" src="Resources/addon/dynamsoft.webtwain.addon.webcam.js"> </script>
```

Let's start with a simple code snippet

```
function CaptureImage() {
    if (DWObject) {
        DWObject.Addon.Webcam.Download(
            "http://localhost/dwt/addon/Webcam.zip", doCapture,
            function (errorCode, errorString) { alert(errorString); });
    }
}
function doCapture() {
        DWObject.Addon.Webcam.SelectSource('Integrated Camera');
        var showUI = false; var OnCaptureStart = function () { };
        var OnCaptureError = function (error, errorstr) { alert(errorstr); }
        var OnCaptureSuccess = function () { }; var OnCaptureEnd = function () { };
        DWObject.Addon.Webcam.CaptureImage(showUI, OnCaptureStart, OnCaptureSuccess,
                                           OnCaptureError, OnCaptureEnd);
}
```

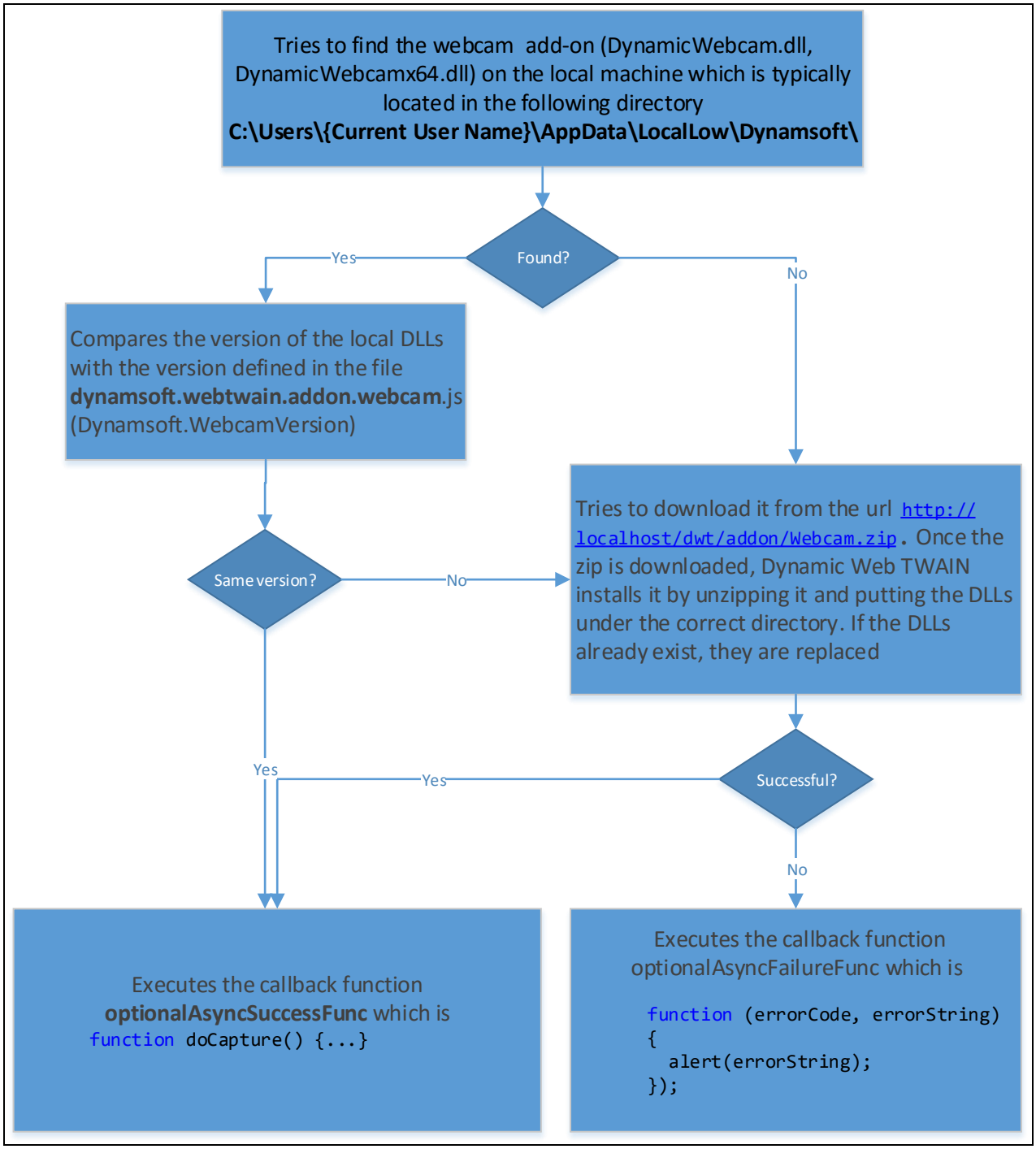
The first thing to note here is also the JS-style '**asynchronous** syntax' of the method 'Download'.

Here is how the 'download' method works

Syntax

```
.Addon.Webcam.Download(remoteFile, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc])
```

Workflow



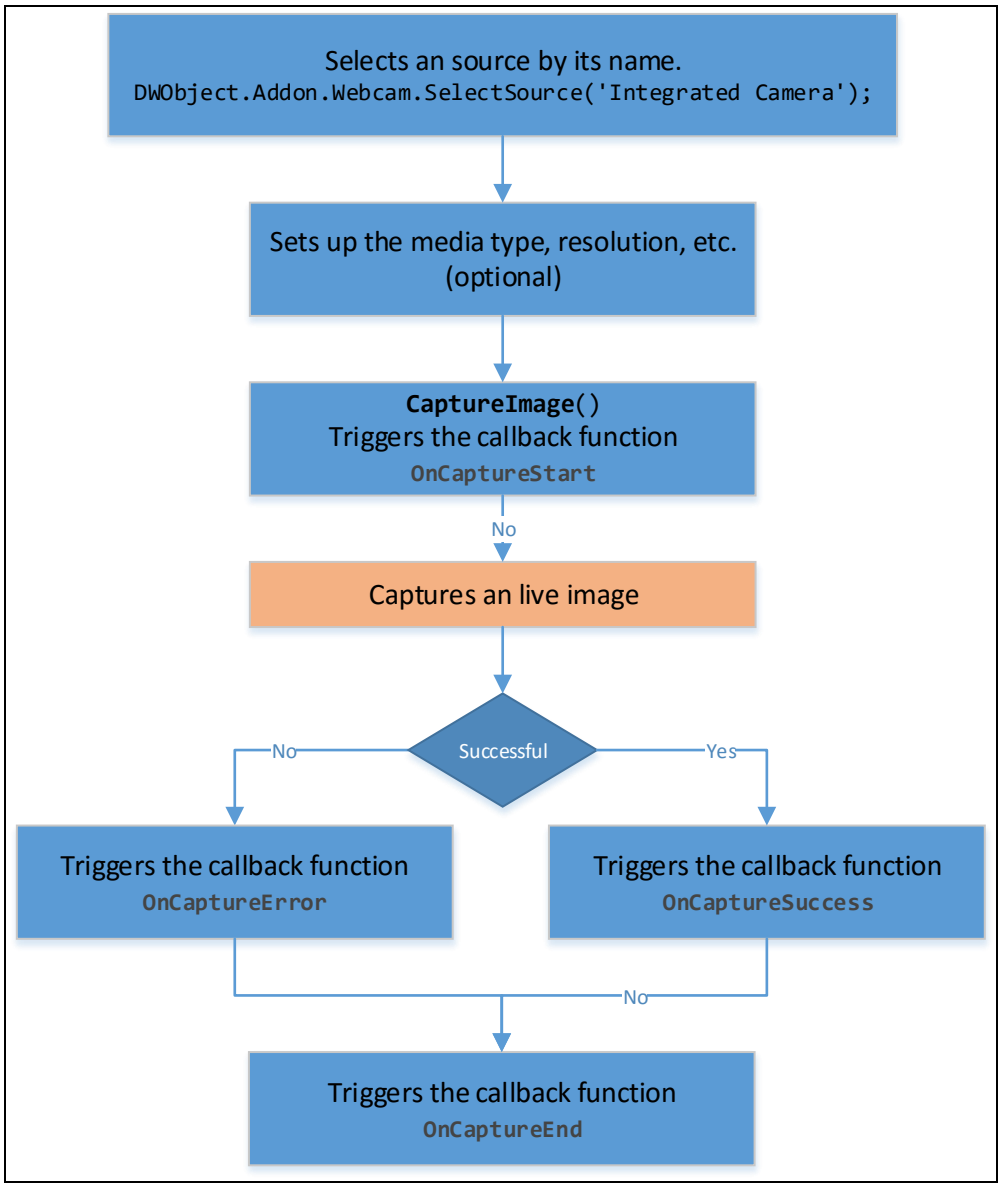
When the webcam add-on (DLLs) is in place, you can start capturing from webcams. Let's check how the method '**CaptureImage**' works.

Syntax

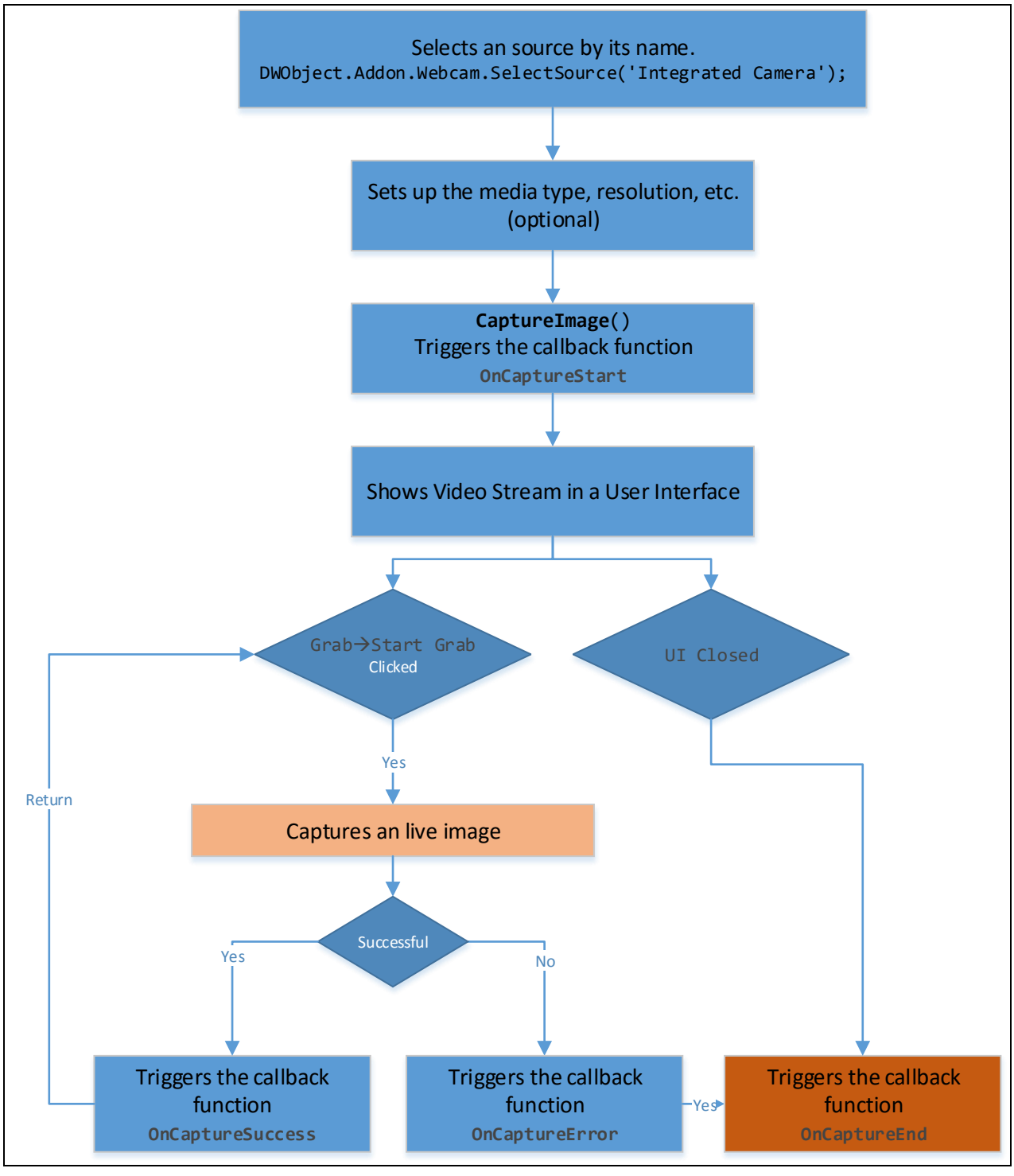
```
.Addon.Webcam.CaptureImage(bool bIfShowUI, OnCaptureStart, OnCaptureSuccess, OnCaptureError, OnCaptureEnd);
```

Workflow

Condition: ***bIfShowUI == true***



Condition: ***bIfShowUI == false***



Setting up the Webcam

Finally, let's check out how to set up the webcam for capturing. We'll take 'resolution' as an example

```
function doCapture() {
    DWObject.Addon.Webcam.SelectSource('Integrated Camera');
    DWObject.Addon.Webcam.SetResolution("1280 x 720");
    DWObject.Addon.Webcam.CaptureImage(……
}
```

3 things to note here

1. To select a webcam, you need to use the name string of the camera to select it. If you wish to use an index (like the method SelectSourceByIndex), you can do something like this

   ```
   DWObject.Addon.Webcam.SelectSource(DWObject.Addon.Webcam.GetSourceList()[0]);//Selects the 1st
   ```

2. To set up a property like 'Resolution', you also need to pass a string to the method **SetResolution**. However the string must be one of the supported ones which means you need to first know about all the strings that you can use. To do this, you can use the method **GetResolution**. The following APIs work similarly in pairs

   Addon.Webcam.GetMediaType & Addon.Webcam.SetMediaType
   Addon.Webcam.GetResolution & Addon.Webcam.SetResolution
   Addon.Webcam.GetFrameRate & Addon.Webcam.SetFrameRate

The following code **lists** all available media types, resolutions and frame rates of the selected camera

HTML

```
<select size="1" id="source" style="position: relative; width: 220px;" onchange="updatePage();"></select>
<select size="1" id="mediaType" style="position: relative; width: 220px;"></select>
<select size="1" id="Resolution" style="position: relative; width: 220px;"></select>
<select size="1" id="frameRate" style="position: relative; width: 220px;"></select>
```

JavaScript

```
function updatePage() {
    DWObject.Addon.Webcam.SelectSource(document.getElementById("source").options[
        document.getElementById("source").selectedIndex].text);
    var mediaType = DWObject.Addon.Webcam.GetMediaType();
    document.getElementById("mediaType").options.length = 0;
    for (var i = 0; i < mediaType.GetCount(); i++)
        document.getElementById("mediaType").options.add(new Option(mediaType.Get(i)));
    var Resolution = DWObject.Addon.Webcam.GetResolution();
    document.getElementById("Resolution").options.length = 0;
    for (var i = 0; i < Resolution.GetCount() ; i++)
```

```
        document.getElementById("Resolution").options.add(new Option(Resolution.Get(i)));
    var frameRate = DWObject.Addon.Webcam.GetFrameRate();
    document.getElementById("frameRate").options.length = 0;
    for (var i = 0; i < frameRate.GetCount() ; i++)
        document.getElementById("frameRate").options.add(new Option(frameRate.Get(i)));
    DWObject.Addon.Webcam.CloseSource();
}
```

The following code sets up the properties with supported values

```
DWObject.Addon.Webcam.SelectSource(document.getElementById("source").options[
    document.getElementById("source").selectedIndex].text);
DWObject.Addon.Webcam.SetMediaType(document.getElementById("mediaType").options[
    document.getElementById("mediaType").selectedIndex].text);
DWObject.Addon.Webcam.SetResolution(document.getElementById("Resolution").options[
    document.getElementById("Resolution").selectedIndex].text);
DWObject.Addon.Webcam.SetFrameRate(document.getElementById("frameRate").options[
    document.getElementById("frameRate").selectedIndex].text);
DWObject.Addon.Webcam.CaptureImage(false, OnCaptureStart, OnCaptureSuccess, OnCaptureError,
    OnCaptureEnd);
```

3. For more advanced settings, the add-on provides the following methods

   Addon.Webcam.GetCameraControlPropertySetting
   Addon.Webcam.GetCameraControlPropertyMoreSetting
   Addon.Webcam.SetCameraControlPropertySetting
   Addon.Webcam.GetVideoPropertySetting
   Addon.Webcam.GetVideoPropertyMoreSetting
   Addon.Webcam.SetVideoPropertySetting

   The following is a simple code snippets on getting and setting the property **Zoom**

```
//Get
var zoom =
DWObject.Addon.Webcam.GetCameraControlPropertySetting(EnumDWT_CameraControlProperty.CCP_ZOOM);
alert(zoom.GetValue());
alert(zoom.GetIfAuto());
//Set
DWObject.Addon.Webcam.SetCameraControlPropertySetting(EnumDWT_CameraControlProperty.CCP_ZOOM, 10,
true);
```

Please make sure the values you set to the properties are supported by the current webcam.

License Verification

Basic Information

Since version 9.0, Dynamic Web TWAIN has been using the property ProductKey for license verification at runtime. The property accepts a series of alphanumeric code as the product key which is generated based on the ***license***(s) you own. All editions share the same authentication mechanism.

Note:

1. *One product key can be generated from one or many license(s)*
2. *The Product Key represents the encrypted license(s). Every product key is unique*
3. *The Product Key is also bound to a certain domain since version 11*

Generating a Product Key

In version 11, we have applied a new security featureby binding licenses to the purchasers' domain. To generate your product key after you get the full license, please refer to

http://kb.dynamsoft.com/questions/40/

Using the Product Key

Set it during initialization (recommended)

Set ProductKey in the file *Dynamsoft.webtwain.config.js*

```
/// <reference path="dynamsoft.webtwain.initiate.js" />

///
Dynamsoft.WebTwainEnv.Containers = [{ContainerId:'dwtcontrolContainer',Width:270,Height:350}];
///
// please replace ****** with your product key
Dynamsoft.WebTwainEnv.ProductKey = '5AKDNCIE2832***';
///
Dynamsoft.WebTwainEnv.Trial = true;
///
Dynamsoft.WebTwainEnv.Debug = false; // only for debugger output
///
```

Set it when necessary (not recommended)

Set ProductKey in your own code before calling AcquireImage() method

```
function AcquireImage() {
	var DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
	DWObject.SelectSource();
	DWObject.OpenSource();
	DWObject.IfShowUI = false;
	DWObject.ProductKey = '5AKDNCIE2832***';
	DWObject.AcquireImage();
}
```

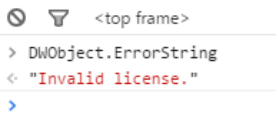
Multiple Product Keys

If you have multiple product keys generated from multiple serial numbers, you can combine all of them and assign them to the ProductKey property. You will need to separate the keys by semi-colons (;). For example

Dynamsoft.WebTwainEnv.ProductKey = '3A0761****;6685BA******',**

Common Licensing Errors

Invalid license



As stated in the error message, the product key doesn't exist or is invalid. When this happens, you are still able to scan documents but there will be a watermark on every image. This error message only appears in the purchased version. Please make sure

1) You have generated the product key with the correct purchased license(s)
2) You have bound the license to the correct domain in the product key

The current product key does not match the domain

As stated in the error message, the license(s) is not binding to the correct domain. Please make sure you have done the binding correctly. Please refer to

http://kb.dynamsoft.com/questions/40/

The trial license for Dynamic Web TWAIN has expired

As stated in the error message, the trial license(s) used to generate the product key has expired and you will find a watermark on the image scanned.

In this case, you can email our support team (support@dynamsoft.com) and ask for a trial license extension.

Upgrading from Previous Versions

Check out the Knowledge Base article below:

http://kb.dynamsoft.com/questions/575/How+to+deploy+or+upgrade+to+the+latest+version

Or you can check out Dynamic Web TWAIN upgrade guide [PDF] at:

http://www.dynamsoft.com/download/Support/Dynamic%20Web%20TWAIN%20Upgrade%20Guide.pdf

Troubleshooting

Installing the Virtual Scanner for testing

If you don't have an actual scanner available for testing, you can download and install a virtual scanner (a scanner simulator), which was developed by the TWAIN Working Group, to test the basic scanning features of Dynamic Web TWAIN.

Download 32 bit virtual scanner
Download 64 bit virtual scanner

Is my scanner driver TWAIN compliant

If you are not sure whether your scanner is TWAIN compliant, you can use TWACKER, a tool developed by the TWAIN Working Group, to verify it.

TWACKER can be downloaded here: TWACKER 32 bit TWACKER 64 bit

After installation:

- Launch the program.
- Click menu File->Select Source and select your device in the 'Select Source' list.
  - If your device is not listed, please check if the driver is installed. Or, you can try running TWACKER as "Admin" since you may not have permission to access the data source.
- Click menu File->Acquire to do scanning.

If the scanning is successful without any errors, then your device should be TWAIN compliant.

If it fails, you may have to search online or contact the device vendor for a TWAIN driver.

Why is my scanner not shown or not responding in the browser

Please check out http://kb.dynamsoft.com/questions/541/

Why does Dynamic Web TWAIN fail to upload my documents

When you use forms authentication in your web application, you might fail to upload documents to the server using **Dynamic Web TWAIN in IE 6~9.** This happens because Dynamic Web TWAIN still works as an ActiveX in IE 6~9 and it performs the upload as an independent user-agent, one that might not have permission to get the authentication cookie during the upload.

The root cause of the issue is the missing cookie for the authentication. So the solution is to locate the cookie and bind it to the upload request. To do this, Dynamsoft has provided an API called SetCookie. The following are the steps, we use ASP.NET (C#) in the example:

1. Print the cookie out explicitly in the page and assign it to a JavaScript variable:
   var cookie = "<%=Request.Headers["Cookie"] %>";
2. Set the cookie at runtime before you call any HTTP Upload method:
   DWObject.SetCookie(cookie);
   DWObject.HTTPUpload…

That's it.

NOTE:

The above solution is very straightforward but it exposes the authentication cookie which might be seen as a security breach. If this is not an acceptable solution, we also offer a workaround using AJAX. For more info, please contact us at live help:
http://www.dynamsoft.com/Support/LiveHelp.aspx

More Troubleshooting Topics

For more troubleshooting topics, please check out:
http://kb.dynamsoft.com/categories/Dynamic+Web+TWAIN/

Useful Resources

Online Demo Site

http://www.dynamsoft.com/demo/DWT/online_demo_scan.aspx

Knowledge Base

http://kb.dynamsoft.com/categories/Dynamic+Web+TWAIN/

Forum

http://forums.dynamictwain.com/support-and-discussion-dynamic-web-twain-f9.html

FAQ

http://www.dynamsoft.com/Products/WebTWAIN_FAQ.aspx

API Documentation

http://www.dynamsoft.com/help/TWAIN/WebTwain/index.htm

Contact Us

Email: support@dynamsoft.com

Online Chat: http://www.dynamsoft.com/Support/LiveHelp.aspx

Telephone: +**1 604.605.5491** | +**1 877.605.5491** (Toll-Free)

FAX: 1-866-410-8856