Dynamic Web TWAIN

Developer's Guide

August 2017

Dynamsoft™

10+ 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. It provides an overview of most of the things you can achieve with the SDK.

Audience

This guide is meant for all developers interested in the Dynamic Web TWAIN 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.

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.

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 device. 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, etc.
            - 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
        - Ubuntu 10+, Debian 8 and Fedora 19+
            - Chrome/Firefox 27+: HTML5
    2. Scanner /Other Devices (must be TWAIN compliant)

**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 several editions: ActiveX, HTML5 for Windows, HTML5 for Mac and HTML5 for Linux. 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;
- **HTML5 for Mac**: supports Chrome/Firefox 27+, Safari 7+ on Mac OS.
- **HTML5 for Linux:** supports Chrome/Firefox 27+ on Ubuntu 10+, Debian 8 and Fedora 19+.

Building the "Hello World" Scan Page

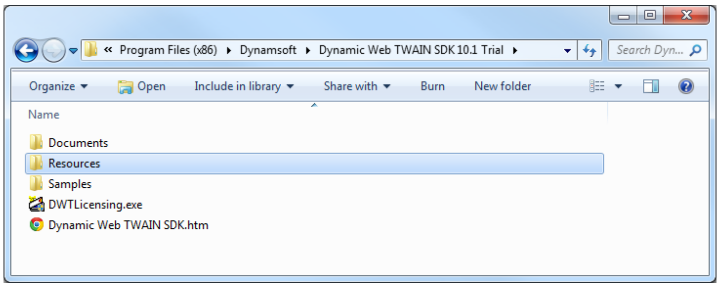
NOTE: Before you start, please make sure you've downloaded and installed the latest version of Dynamic Web TWAIN. If you haven't done so, you can get the 30-day free trial here.

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

Step 1: Start a Web Application

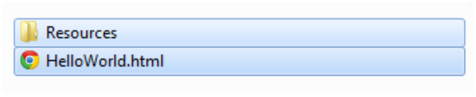
1.1 Copy the Dynamsoft Resources folder to your project

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

Put an 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

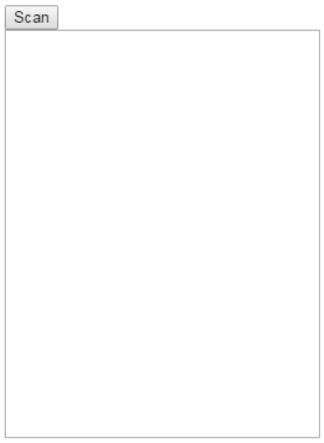
```
  <input type="button" value="Scan" onclick="AcquireImage();" />
  <script type="text/javascript">
         var DWObject;
         function Dynamsoft_OnReady(){
           DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
         }
         function AcquireImage(){
          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

3.2 Review the completed code

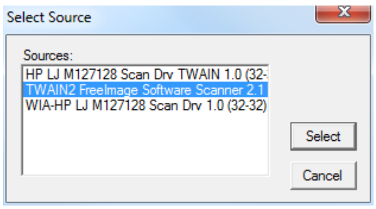
```
<html>
<head><title>Hello World</title>
  <script type="text/javascript" src="Resources/dynamsoft.webtwain.initiate.js"> </script>
  <script type="text/javascript" src="Resources/dynamsoft.webtwain.config.js"> </script>
</head>
<body>
  <input type="button" value="Scan" onclick="AcquireImage();" />
  <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 browser, it should look like this:



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

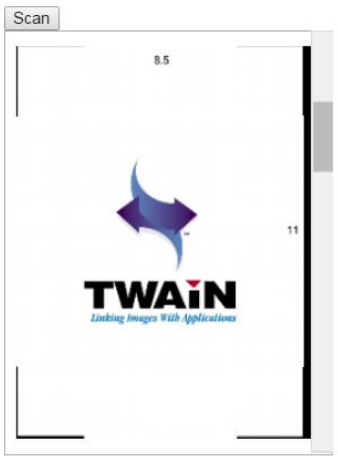

Select Source
Sources:
HP LJ M127128 Scan Drv TWAIN 1.0 (32-
TWAIN2 FreeImage Software Scanner 2.1
WIA-HP LJ M127128 Scan Drv 1.0 (32-32)
Select
Cancel

NOTE:

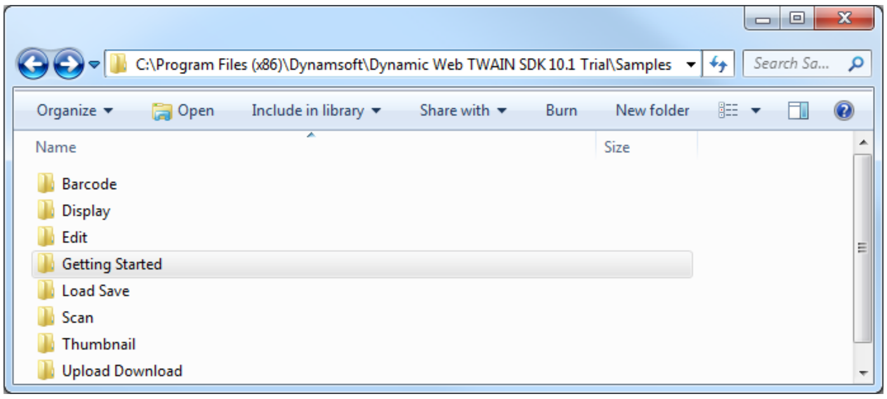
1) *Only TWAIN compatible devices are listed in the Select Source dialog. If your connected scanner is not shown in the list, please follow this article to troubleshoot.*

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 samples (source code provided) there. You can try them to see the different features of Dynamic Web TWAIN.

Using JavaScript IntelliSense

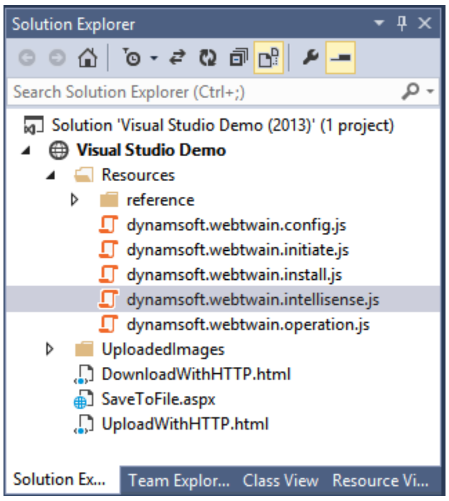
IntelliSense is designed to make the development of your application much easier. It is a powerful feature that helps you write code faster and with fewer errors by providing information while you code.

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*.

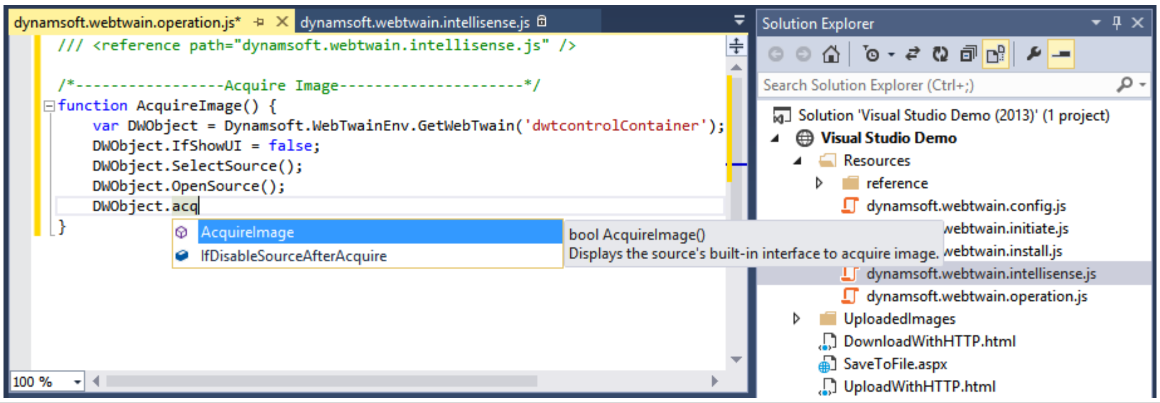


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Visual Studio Demo (2013)' (1 project)
Visual Studio Demo
Resources
reference
dynamsoft.webtwain.config.js
dynamsoft.webtwain.initiate.js
dynamsoft.webtwain.install.js
dynamsoft.webtwain.intellisense.js
dynamsoft.webtwain.operation.js
UploadedImages
DownloadWithHTTP.html
SaveToFile.aspx
UploadWithHTTP.html
Solution Ex...
Team Explor...
Class View
Resource Vi...

Next, drag *dynamsoft.webtwain.intellisense.js* from the solution explorer into the editor window of the JavaScript file where you want to use 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.



dynamsoft.webtwain.operation.js*
dynamsoft.webtwain.intellisense.js
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Visual Studio Demo (2013)' (1 project)
Visual Studio Demo
Resources
reference
dynamsoft.webtwain.config.js
webtwain.initiate.js
webtwain.install.js
dynamsoft.webtwain.intellisense.js
dynamsoft.webtwain.operation.js
UploadedImages
DownloadWithHTTP.html
SaveToFile.aspx
UploadWithHTTP.html
100 %

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

/*----------------Acquire Image--------------------*/
function AcquireImage() {
    var DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
    DWObject.IfShowUI = false;
    DWObject.SelectSource();
    DWObject.OpenSource();
    DWObject.acq
}
```

AcquireImage
IfDisableSourceAfterAcquire

bool AcquireImage()
Displays the source's built-in interface to acquire image.

Note:

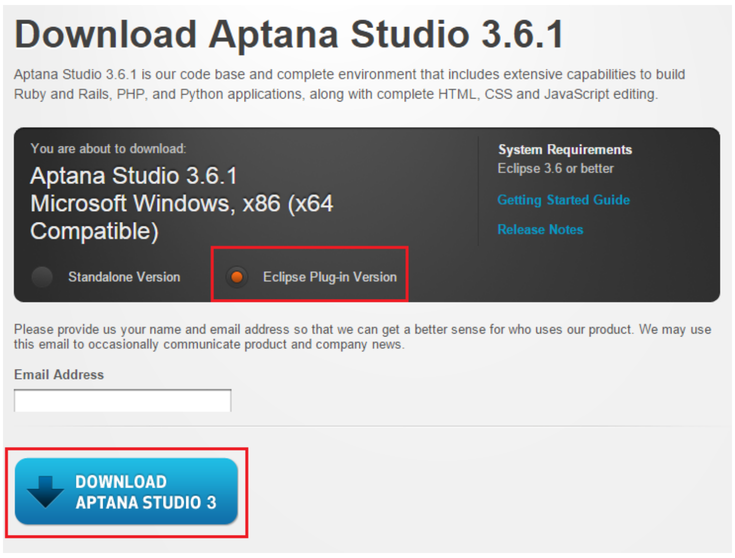
1. *The reference directive must be declared earlier than any script in the JavaScript file, in other words, you should put it at the very top.*
2. *The path (path = "dynamsoft.webtwain.intellisense.js") is relative to the current JavaScript file.*
3. *IntelliSense only supports pure JavaScript files (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 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.



Download Aptana Studio 3.6.1

Aptana Studio 3.6.1 is our code base and complete environment that includes extensive capabilities to build Ruby and Rails, PHP, and Python applications, along with complete HTML, CSS and JavaScript editing.

You are about to download:
Aptana Studio 3.6.1
Microsoft Windows, x86 (x64 Compatible)

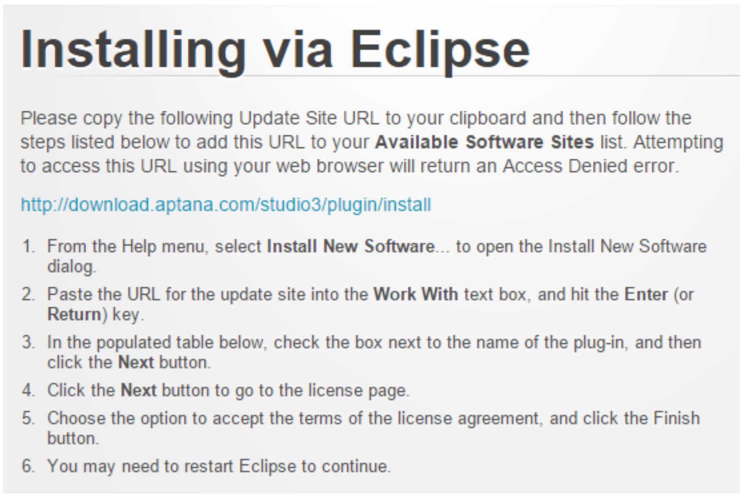
System Requirements
Eclipse 3.6 or better

Please provide us your name and email address so that we can get a better sense for who uses our product. We may use this email to occasionally communicate product and company news.

Email Address

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

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



Installing via Eclipse

Please copy the following Update Site URL to your clipboard and then follow the steps listed below to add this URL to your **Available Software Sites** list. Attempting to access this URL using your web browser will return an Access Denied error.

http://download.aptana.com/studio3/plugin/install

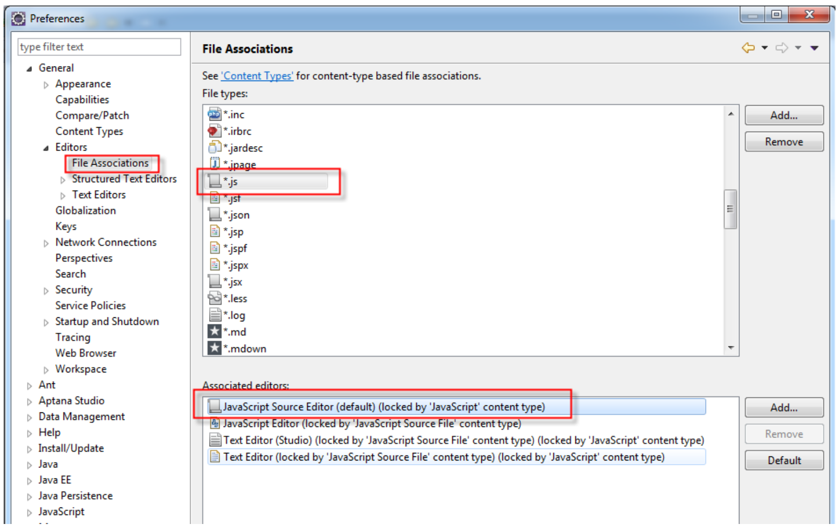
1. From the Help menu, select **Install New Software**... to open the Install New Software dialog.
2. Paste the URL for the update site into the **Work With** text box, and hit the **Enter** (or **Return**) key.
3. In the populated table below, check the box next to the name of the plug-in, and then click the **Next** button.
4. Click the **Next** button to go to the license page.
5. Choose the option to accept the terms of the license agreement, and click the Finish button.
6. You may need to restart Eclipse to continue.

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

Step two: Configuring Eclipse

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

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



File Associations

See 'Content Types' for content-type based file associations.

File types:

*.inc
*.irbrc
*.jardesc
*.jpage
*.js
*.jsf
*.json
*.jsp
*.jspf
*.jspx
*.jsx
*.less
*.log
*.md
*.mdown

Associated editors:

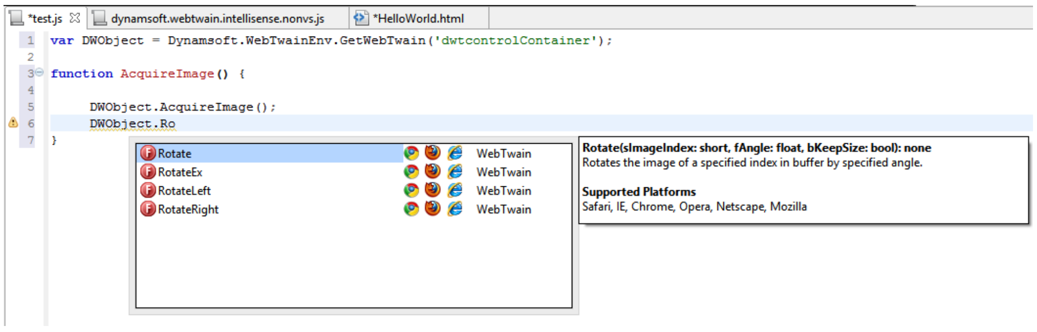
JavaScript Source Editor (default) (locked by 'JavaScript' content type)
JavaScript Editor (locked by 'JavaScript Source File' content type)
Text Editor (Studio) (locked by 'JavaScript Source File' content type) (locked by 'JavaScript' content type)
Text Editor (locked by 'JavaScript Source File' content type) (locked by 'JavaScript' content type)

Step three: Using the IntelliSense

1. Make sure that the *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, 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 '.'



```
var DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');

function AcquireImage() {

    DWObject.AcquireImage();
    DWObject.Ro
}
```

Rotate	WebTwain
RotateEx	WebTwain
RotateLeft	WebTwain
RotateRight	WebTwain

Rotate(sImageIndex: short, fAngle: float, bKeepSize: bool): none
Rotates the image of a specified index in buffer by specified angle.

Supported Platforms
Safari, IE, Chrome, Opera, Netscape, Mozilla

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

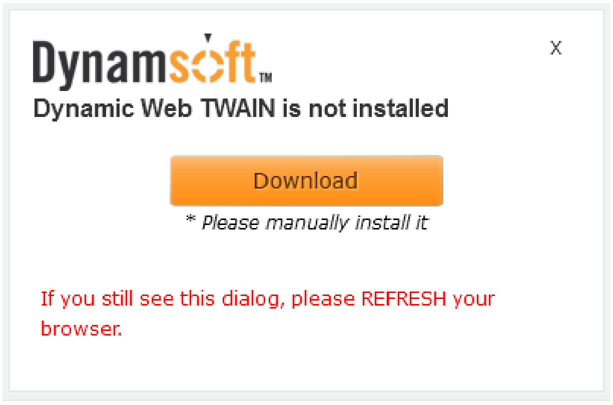
Changing the Size of the Viewer

You can change the size of the container (the built-in viewer) in *dynamsoft.webtwain.config.js.* You can use either 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


Dynamic Web TWAIN is not installed

If you still see this dialog, please REFRESH your browser.

You may want to change the 'Dynamsoft Branding' here. 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.

Dynamsoft

- 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 bit tricky to use. We'll talk about it a little more here.

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 the OnMouseClick event for 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'

For all the events, please refer to the online API Documentation. Of these 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 may 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>
```

Dynamic Web TWAIN Service Configuration

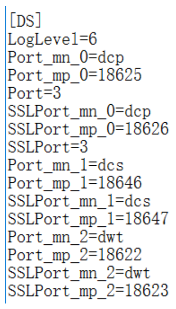
You can configure Dynamic Web TWAIN HTML5 edition in the configuration INI file located in

64bit OS:

C: \Windows\SysWOW64\Dynamsoft\DynamsoftService

32bit OS:

C: \Windows\System32\Dynamsoft\DynamsoftService



```
[DS]
LogLevel=6
Port_mn_0=dcp
Port_mp_0=18625
Port=3
SSLPort_mn_0=dcp
SSLPort_mp_0=18626
SSLPort=3
Port_mn_1=dcs
Port_mp_1=18646
SSLPort_mn_1=dcs
SSLPort_mp_1=18647
Port_mn_2=dwt
Port_mp_2=18622
SSLPort_mn_2=dwt
SSLPort_mp_2=18623
```

General Settings

LogLevel

The level on which Dynamic Web TWAIN writes its log. The default value is 6 which enables minimum simple logging. You can set it to 14 to log more detailed information for debugging purposes.

Port/SSLPort/Port_mn_*/SSLPort_mn_*/

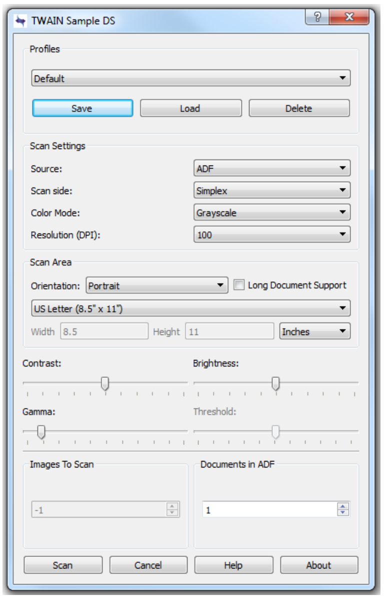
The port numbers used by the Dynamsoft Service or different modules for http communication. *_mn_* means the name of the module(s), *_mp_* means the port(s). Port_* and SSLPort_* mean the port is for HTTP or HTTPS respectively.

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 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 the settings in the scanner's built-in User Interface. Take the Virtual scanner for example:


TWAIN Sample DS

Profiles: Default

Scan Settings
Source: ADF
Scan side: Simplex
Color Mode: Grayscale
Resolution (DPI): 100

Scan Area
Orientation: Portrait
Long Document Support
US Letter (8.5" x 11")
Width 8.5 Height 11 Inches

Images To Scan: -1
Documents in ADF: 1

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
```

Manipulating the image(s)

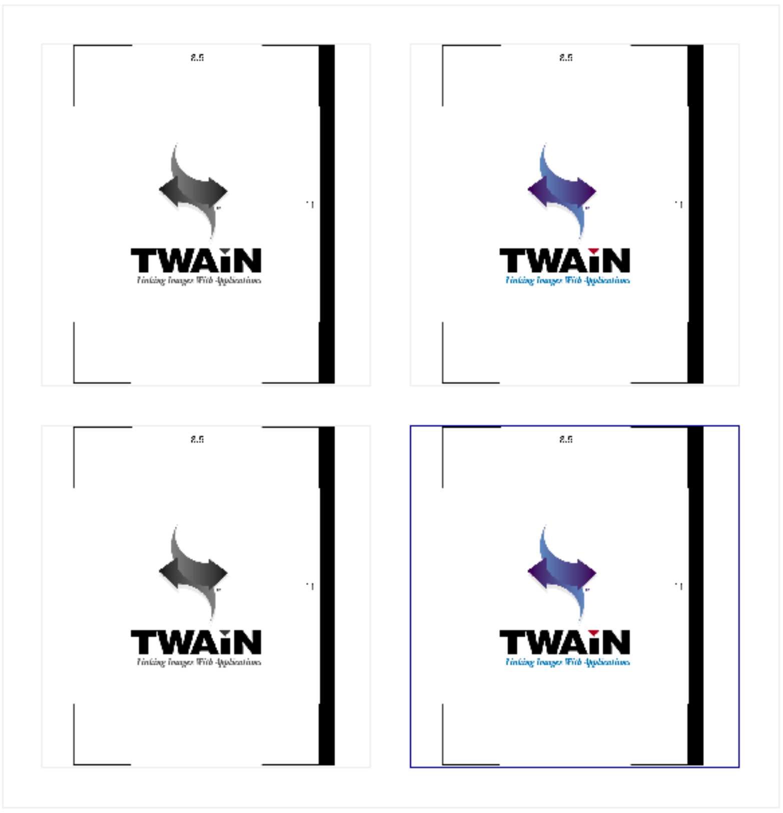
When you have scanned or loaded images in Dynamic Web TWAIN, you can start manipulating the images. You can:

1. Go through each image by changing the property CurrentImageIndexInBuffer

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

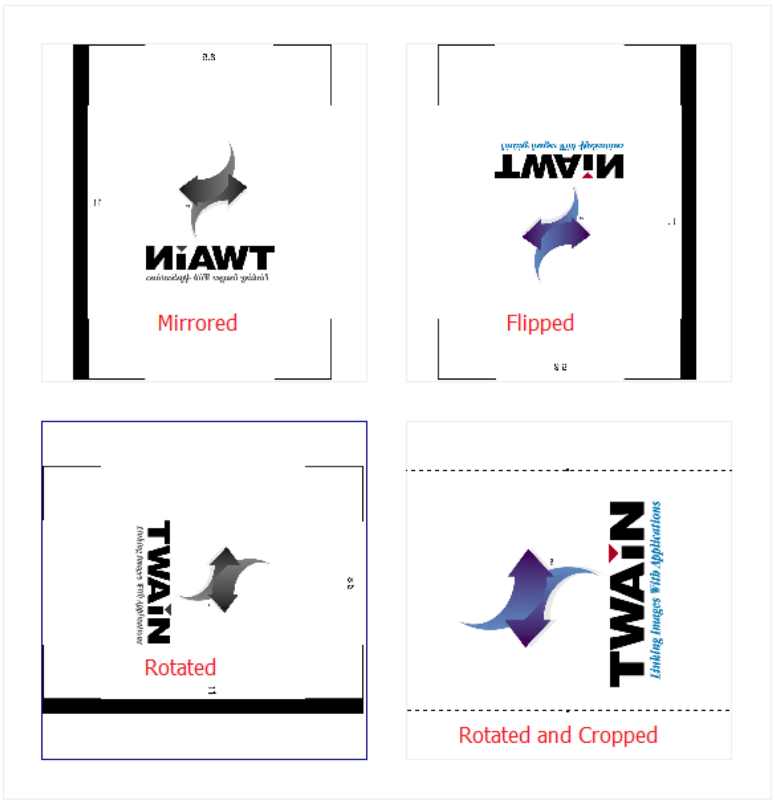
2. Show multiple images by changing the view mode (other than 1*1 or -1*-1) using SetViewMode()

```
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**

Using Thumbnails

Currently Dynamic Web TWAIN does not have a built-in thumbnails view. 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>
```

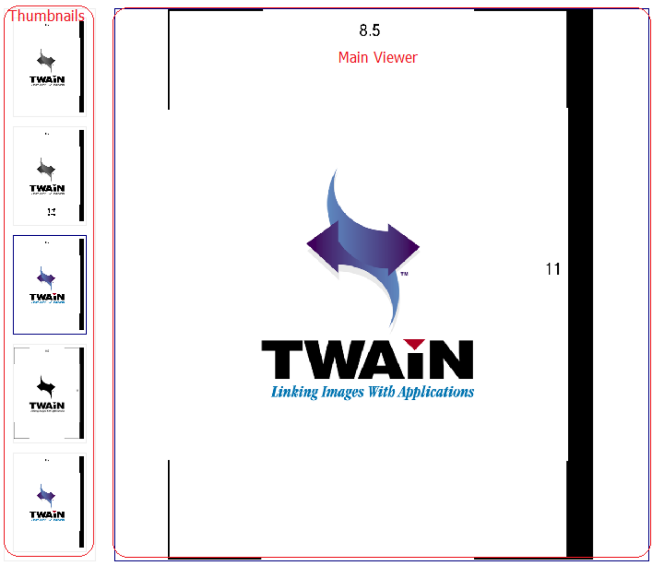
In the dynamsoft.webtwain.config.js

```
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 view). 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
}
```

Adding/Removing Dynamic Web TWAIN object individually

We have the methods *Dynamsoft.WebTwainEnv.CreateDWTObject(id, ip, port, portssl, OnSuccessCallback, OnFailureCallback)* and *Dynamsoft.WebTwainEnv.DeleteDWTObject(id)* which can be used to add or remove a specific Dynamic Web TWAIN object dynamically. The first method is self-explanatory—it's used to add a new Dynamic Web TWAIN object by passing in the id of the DWT placeholder. The second method allows us to remove a Dynamic Web TWAIN object from the web page.

Add a Dynamic Web TWAIN object at runtime

To add a Dynamic Web TWAIN object on the web page, you need to first put a div element as the placeholder for this object:

HTML code:

```
    <div id="dwtcontrolContainer2"></div>
```

The next step is to use the method *Dynamsoft.WebTwainEnv.CreateDWTObject(id, ip, port, portssl, OnSuccessCallback, OnFailureCallback)* to create and initialize the Dynamic Web TWAIN object that will be embedded in the div with id 'dwtcontrolContainer2'. Here's what it looks like in code:

JavaScript code:

```
var DWObject2;
Dynamsoft.WebTwainEnv.CreateDWTObject(
   "dwtcontrolContainer2",
   "127.0.0.1",
   "18618",
   "18619",
   function(newDWObject){ DWObject2 = newDWObject;},
   function(errorString){ alert(errorString);}
);
```

NOTE:

If the div element with id "dwtcontrolContainer2" already exists in Dynamsoft.WebTwainEnv.Containers, you will get the following error:

"Duplicate ID detected for creating Dynamic Web TWAIN objects, please check and modify."

When this happens, please check if you've set "dwtcontrolContainer2" in dynamsoft.WebTwainEnv.Containers in the dynamsoft.webtwain.config.js file like this:

In the dynamsoft.webtwain.config.js

```
Dynamsoft.WebTwainEnv.Containers = [{ContainerId:'dwtcontrolContainer', Width: 120, Height: 350}, {ContainerId:'dwtcontrolContainer2', Width: 270, Height: 350},];
```

If so, please modify the id "dwtcontrolContainer2" in HTML and JavaScript and try again.

Okay, that's it. You can now add some style to the new Dynamic Web TWAIN object and manipulate it. For example:

```
DWObject2.Width = 580;
```

```
DWObject2.Height = 600;
DWObject2.SelectSource();
DWObject2.AcquireImage();
```

Remove a Dynamic Web TWAIN object at runtime

To remove a Dynamic Web TWAIN object from the web page, it is quite easy. You just need to call the method Dynamsoft.WebTwainEnv.DeleteDWTObject(id) specifying the container id.

JavaScript code:

```
Dynamsoft.WebTwainEnv.DeleteDWTObject("dwtcontrolContainer2");
```

NOTE:

> ***Dynamsoft.WebTwainEnv.Unload()*** *can't release the Dynamic Web TWAIN objects generated by the method* ***Dynamsoft.WebTwainEnv.CreateDWTObject.*** *You can only use the method* ***Dynamsoft.WebTwainEnv.DeleteDWTObject(id)*** *to release that object.*

Scanning a large number of Documents - Disk Caching

When you scan hundreds or even thousands of documents, 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 (typically around 1,500M), 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, set **IfAllowLocalCache** to true. With this mode on, the default value of **BufferMemoryLimit** is 800M.

If the image data < **BufferMemoryLimit** (default value: 800M), the images will be stored in compressed mode in physical memory. When the memory limit is reached, the extra image data will be encrypted and stored on the local hard disk. You can set **BufferMemoryLimit** to adjust the maximum memory setting according to your needs.

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\DynamsoftService\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\DynamsoftService\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 the 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

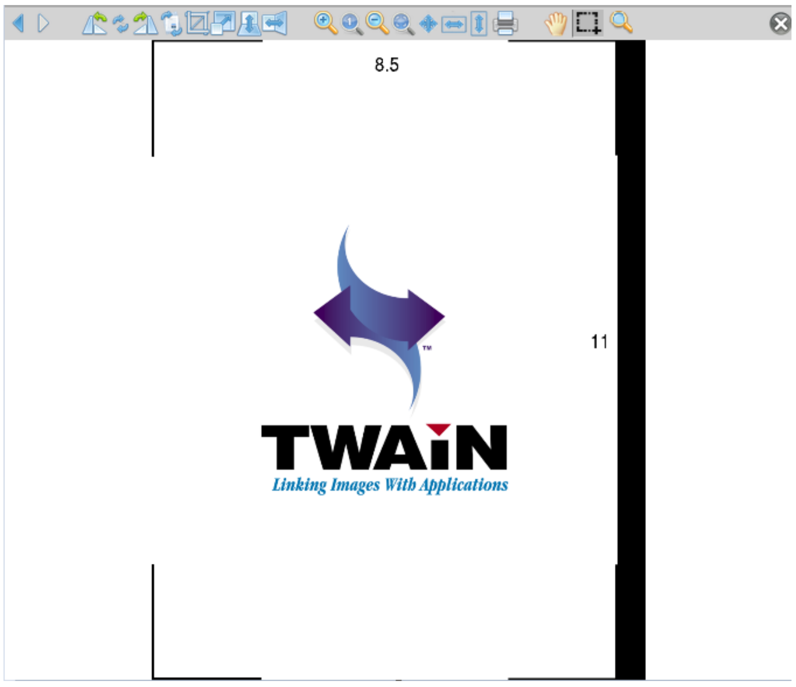
2. Edit an image in the following ways: rotate, mirror, crop, flip, change size, change bit depth, change DPI, convert an image to gray scale, erase and add text.

3. Print the images, etc.

Opening the 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);
```

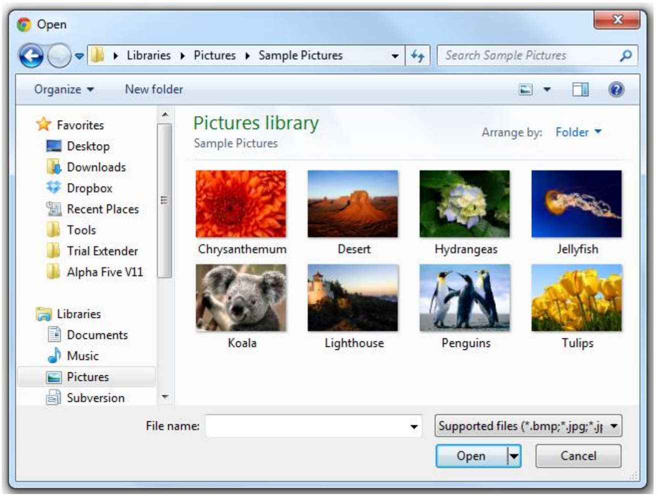
```
DWObject.LoadImageEx("C:\\WebTWAIN\\Images\\ImageData.jpg", EnumDWT_ImageType.IT_JPG,
optionalAsyncSuccessFunc, optionalAsyncFailureFunc); // ImageType: JPG

//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()

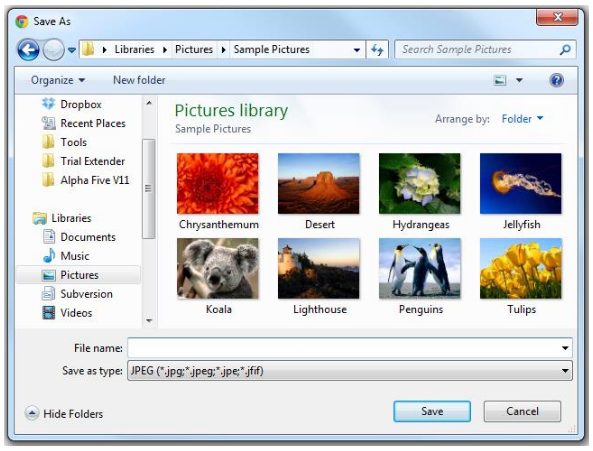
Code snippet:

```
//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:

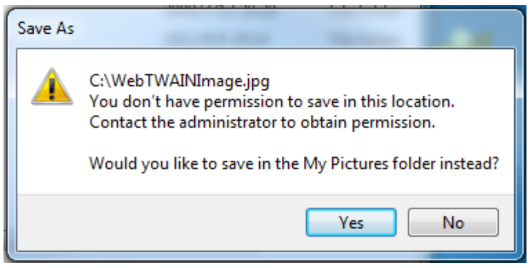
```
DWObject.IfShowFileDialog = true;
DWObject.SaveAsJPEG("",0);
```

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.


Save As

C:\WebTWAINImage.jpg
You don't have permission to save in this location.
Contact the administrator to obtain permission.

Would you like to save in the My Pictures folder instead?

Uploading image(s) to the web server

Preparation

Before we upload the image(s), we need to set the server IP/name, set the port number, 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 8 methods:

Format	Method
All files	**HTTPUploadThroughPostDirectly**()

All supported image formats	HTTPUpload()
	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 http 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()

Code Snippet

```
DWObject.FTPUserName = 'test';
DWObject.FTPPort = 21;
DWObject.FTPPassword = 'test';
DWObject.FTPUploadAllAsPDF(
       '192.168.8.222',
       'test.pdf',
       OnFtpUploadSuccess, OnFtpUploadFailure
);
```

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 http 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:

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:

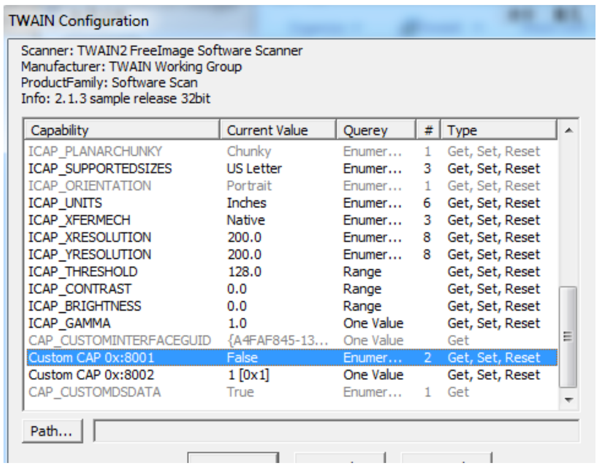
```
<?xml version="1.0" encoding="utf-8"?>
  <configuration>
    <system.webServer>
      <httpProtocol>
        <customHeaders>
          <add name="Access-Control-Allow-Origin" value="*" />
          <add name="Access-Control-Allow-Methods" value="OPTIONS,POST,GET,PUT"/>
          <add name="Access-Control-Allow-Headers"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



Scanner: TWAIN2 FreeImage Software Scanner
Manufacturer: TWAIN Working Group
ProductFamily: Software Scan
Info: 2.1.3 sample release 32bit

Capability	Current Value	Querey	#	Type
ICAP_PLANARCHUNKY	Chunky	Enumer...	1	Get, Set, Reset
ICAP_SUPPORTEDSIZES	US Letter	Enumer...	3	Get, Set, Reset
ICAP_ORIENTATION	Portrait	Enumer...	1	Get, Set, Reset
ICAP_UNITS	Inches	Enumer...	6	Get, Set, Reset
ICAP_XFERMECH	Native	Enumer...	3	Get, Set, Reset
ICAP_XRESOLUTION	200.0	Enumer...	8	Get, Set, Reset
ICAP_YRESOLUTION	200.0	Enumer...	8	Get, Set, Reset
ICAP_THRESHOLD	128.0	Range		Get, Set, Reset
ICAP_CONTRAST	0.0	Range		Get, Set, Reset
ICAP_BRIGHTNESS	0.0	Range		Get, Set, Reset
ICAP_GAMMA	1.0	One Value		Get, Set, Reset
CAP_CUSTOMINTERFACEGUID	{A4FAF845-13...	One Value		Get
Custom CAP 0x:8001	False	Enumer...	2	Get, Set, Reset
Custom CAP 0x:8002	1 [0x1]	One Value		Get, Set, Reset
CAP_CUSTOMDSDATA	True	Enumer...	1	Get

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 How to perform Capability Negotiation for more details.

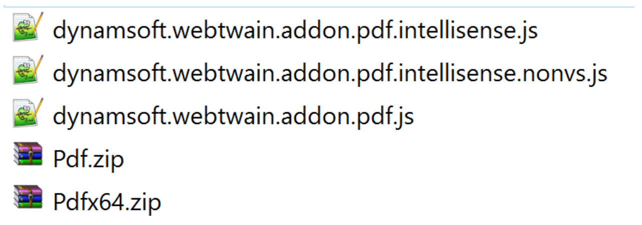
Using Add-ons

How to use PDF Rasterizer

The PDF Rasterizer was designed to support non-image PDF files. There are 5 APIs

Download, SetResolution, SetPassword, IsTextBasedPDF, SetConvertMode

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



dynamsoft.webtwain.addon.pdf.intellisense.js
dynamsoft.webtwain.addon.pdf.intellisense.nonvs.js
dynamsoft.webtwain.addon.pdf.js
Pdf.zip
Pdfx64.zip

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 () {
               DWObject.Addon.PDF.SetResolution(200);
               DWObject.Addon.PDF.SetConvertMode(EnumDWT_ConverMode.CM_RENDERALL);
             },
             function (errorCode, errorString) {alert(errorString); }
        );
  }
}
```

How it works

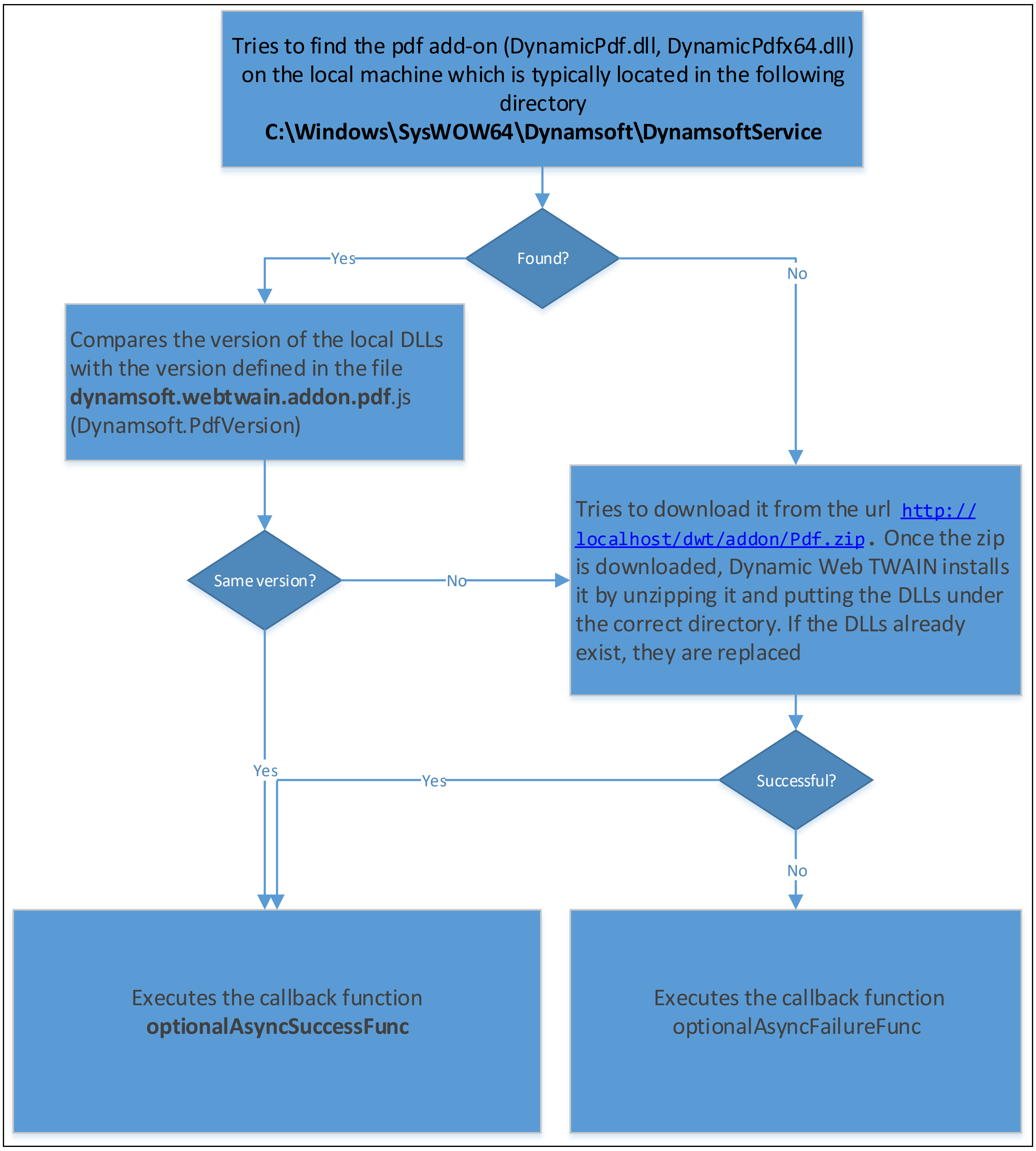
Dynamsoft has optimized this PDF Rasterizer internally so that when you have done the 2 steps above, the PDF rasterizer will be ready to work in your application. Basically, when you try to **load** or **download** any PDF file, the rasterizer will be called automatically to convert the PDF into images 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


Tries to find the pdf add-on (DynamicPdf.dll, DynamicPdfx64.dll) on the local machine which is typically located in the following directory **C:\Windows\SysWOW64\Dynamsoft\DynamsoftService**
Found?
Yes
No
Compares the version of the local DLLs with the version defined in the file **dynamsoft.webtwain.addon.pdf**.js (Dynamsoft.PdfVersion)
Tries to download it from the url http://localhost/dwt/addon/Pdf.zip. Once the zip is downloaded, Dynamic Web TWAIN installs it by unzipping it and putting the DLLs under the correct directory. If the DLLs already exist, they are replaced
Same version?
No
Yes
Yes
Successful?
No
Executes the callback function **optionalAsyncSuccessFunc**
Executes the callback function optionalAsyncFailureFunc

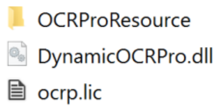
How to use the Professional OCR Add-on

Relatively speaking, the OCR add-on is the most difficult one out of all the add-ons for the Dynamic Web TWAIN SDK. In this guide, we'll talk about how to integrate the **Professional OCR** add-on into your web application.

Dynamic Web TWAIN has been a popular SDK in the market for more than a decade, over the years, we've got many requests for an OCR solution to integrate with it. After carefully evaluating the options, Dynamsoft chose to work with the industry leader Nuance® to provide a good OCR solution which is simply called the **Dynamsoft OCR Professional** Module.

Powered by the OCR engine from Nuance®, this new solution generates industry-leading OCR results. Dynamsoft carefully designed the APIs so that you can do powerful things with just a few lines of code. All APIs can be found in the Dynamsoft Developer Center.

The following files are directly related to this add-on. If you don't have them, please contact Dynamsoft Support (support@dynamsoft.com).



OCRProResource
DynamicOCRPro.dll
ocrp.lic

Figure 1 OCR Professional Resources

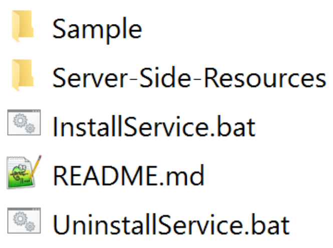
How does it work

The professional OCR add-on is designed to work on the server-side as a HTTP service. Essentially it forms a local Client/Server structure on the server. Once the original document is uploaded on the server, all you need to do is send a HTTP POST request to the OCR service with information like where the original document is and where you'd like the result to be saved. Then you wait for the response from the service before notifying the user who initially started the OCR.

To implement this solution, please follow the steps below

1. Download and **unzip** the all-in-one sample from Dynamsoft Code Gallery. You'll see the structure as shown in Figure 2.

2. **[Installation]**
   If you have installed Dynamsoft Service on the server already, you need to **manually** copy the following highlighted files shown in Figure 3 and paste them in the directory **C:\Windows\SysWOW64\Dynamsoft\DynamsoftService\**.


Sample
Server-Side-Resources
InstallService.bat
README.md
UninstallService.bat

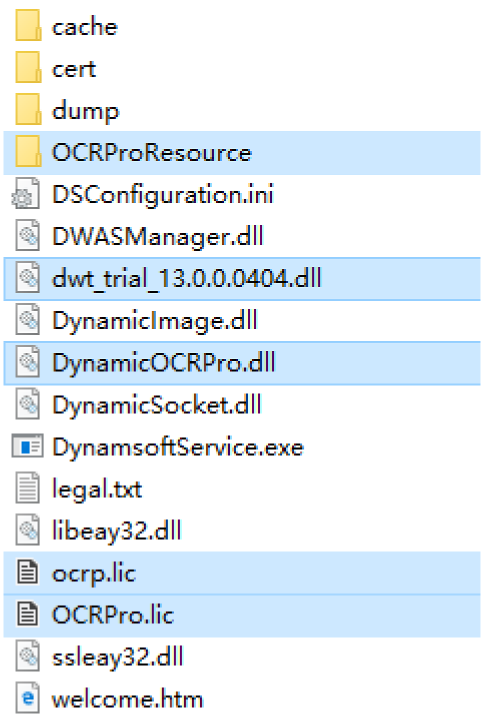

cache
cert
dump
OCRProResource
DSConfiguration.ini
DWASManager.dll
dwt_trial_13.0.0.0404.dll
DynamicImage.dll
DynamicOCRPro.dll
DynamicSocket.dll
DynamsoftService.exe
legal.txt
libeay32.dll
ocrp.lic
OCRPro.lic
ssleay32.dll
welcome.htm

Figure 2 Demo structure

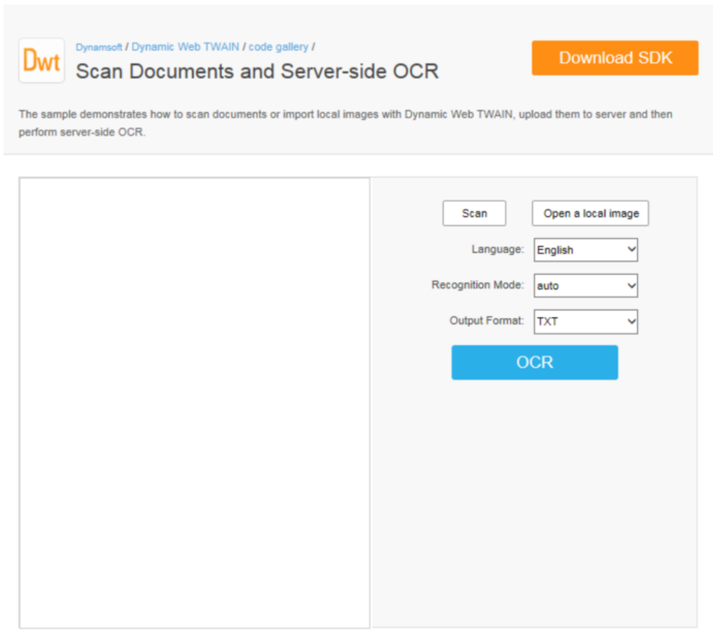
Figure 3 Resources for OCR Professional

3. If you have never installed Dynamsoft Service on the server

   a. If you will be testing the demo app with Visual Studio, you can start by installing the OCR pro add-on by running **InstallService.bat** as Administrator

   b. If you will be testing the demo app in a C#-enabled IIS or other web servers, you need to copy all the unzipped files in step 1 to a content directory configured in the web server and then run the file **InstallService.bat** as Administrator to install the OCR-pro add-on (service)

4. **[Testing the Demo App]**
   If you have Visual Studio installed, you can open the Demo application by the according solution file (***.sln***). If you have already configured a C#-enabled IIS or other web server, you can browse to the directory where you had previous deployed the demo files. The following image shows what you will see as the UI of the demo and where you can start testing the OCR functionalities.



The sample demonstrates how to scan documents or import local images with Dynamic Web TWAIN, upload them to server and then perform server-side OCR.

Figure 4 The UI for DWT OCR Professional Demo

5. **[How it works]**
   Here is how the demo works:

   a. First of all, you load a local image/images or scan in an image or multiple images

   b. You configure the OCR by specifying the language and resulting format etc.

   c. When you hit the button "OCR", the image(s) will be uploaded to the server as a multi-page PDF. The configuration for the OCR is also included as part of the HTTP request in the fields **OutputFormat** and **RequestBody**

   d. The server-side code written in C# in the file **SaveToOCR.aspx** does the following things

      i. It receives the uploaded PDF and save it on the server's disk

      ii. It updates the configuration for OCR with the correct file path and result format

      iii. It then sends a HTTP request (a JSON string) containing the OCR configuration to the OCR service we have previously installed. More info on the request>>>

      iv. Once the OCR service finishes the task, it sends back a HTTP response which contains information like the path of the result file. More info on the response>>>

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 can also be bound to a specific domain (since version 11)*

Generating a Product Key

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

http://developer.dynamsoft.com/dwt/kb/2040

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******’,**

Upgrading from Previous Versions

Please Refer to the Knowledge Base article below:

http://developer.dynamsoft.com/dwt/kb/2575

If you are upgrading from v9 or earlier to the new versions, 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 to obtain a TWAIN driver.

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

Please check out http://developer.dynamsoft.com/dwt/kb/2541.

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://developer.dynamsoft.com/dwt/kb

Useful Resources

Online Demo Site

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

Knowledge Base

http://developer.dynamsoft.com/dwt/kb

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://developer.dynamsoft.com/dwt/api-reference

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

Appendix A: How to Perform Capability Negotiation

This article describes how to negotiate capabilities with different capability container types using Dynamic Web TWAIN. For an advanced sample, please check out this demo page. You can also download the source code here.

Description

Capabilities represent the features that a specified TWAIN source (e.g. a scanner) provides. In order to make full use of such a source/device, TWAIN applications needs to perform the operation called capability negotiation. With the negotiation, TWAIN applications can understand the source and then guide it to provide the images they would like to receive from it.

Capability Containers

Capabilities exist in many varieties but all have a Default Value, Current Value, and may have other values available that can be supported if selected. To help categorize the supported values into clear structures, TWAIN defines four types of containers for capabilities which are

Name of the Data Structure for the Container	Type of Contents
TW_ONEVALUE	A single value whose current and default values are coincident. The range of available values for this type of capability is simply this single value. For example, a capability that indicates the presence of a document feeder could be of this type.
TW_ARRAY	An array of values that describes the current logical item. The available values may be a larger array of values. For example, a list of the names, such as the supported capabilities list returned by the CAP_SUPPORTEDCAPS capability, would use this type of container.
TW_RANGE	Many capabilities allow users to select their current value from a range of regularly spaced values. The capability can specify the minimum and maximum acceptable values and the incremental step size between values. For example, resolution might be supported from 100 to 600 in steps of 50 (100, 150, 200, ..., 550, 600).
TW_ENUMERATION	This is the most general type because it defines a list of values from which the Current Value can be chosen. The values do not progress uniformly through a range and there is not a consistent step size between the values. For example, if a Source's

	resolution options did not occur in even step sizes then an enumeration would be used (for example, 150, 400, and 600).

What is involved

To perform capability negotiation, you basically do two things

- Get a Capability. This is used to ask the source about a specified capability and get its type, value, etc.
- Set a Capability. This is generally used to request the source to set/change the value of a capability.

Now we'll talk about how to perform capability negotiation

Before doing any capability negotiation, please keep in mind that it can only be done when the source is open (**DataSourceStatus** is 1). Basically you need to use the methods **SelectSource**() and **OpenSource**() to select a TWAIN source and get it ready for the negotiation.

Get

To get information about a capability, you can simply use the following code snippet

```
DWObject.OpenSource();
DWObject.Capability = EnumDWT_Cap.***;
DWObject.CapGet();
var tempValue = '';
DWObject.CapValueType > 8?
/*STR*/tempValue = DWObject.CapValueString
:
/*NUM*/tempValue = DWObject.CapValue;
/*
* Special for BOOL
*/
if(DWObject.CapValueType == EnumDWT_CapValueType.TWTY_BOOL) {
  DynamsoftCapabilityNegotiation.CurrentCapabilityHasBoolValue = true;
  tempValue == 0 ? tempValue='FALSE' : tempValue='TRUE';
}
alert('The type of the capability is ' + DWObject.CapType); /*More info*/
alert('The value of the capability is ' + tempValue);
```

Set

Please NOTE that the container type and available values for a capability is generally dictated by the TWAIN source vendor. When you try to set a capability, you are basically just trying to change it so that it uses a different but available value. Therefore, we recommend that you try to 'get' the capability before you set it.

TW_ONEVALUE

To set a capability with TW_ONEVALUE container, the following steps are needed:

- Set the **Capability** property to the capability to be negotiated;
- Set the **CapType** property to TW_ONEVALUE (`EnumDWT_CapType.TWON_ONEVALUE(5)`) (if you did CapGet() first, this step can be skipped);
- Input the value that you'd like to set to this capability. Please NOTE that you can only set certain values (for example, only 'true' and 'false' are allowed for a capability with the CapValueType of `TWTY_BOOL);`
- If the data type (**CapValueType**) of the capability is ***string*** (`EnumDWT_CapValueType.TWTY_STR32/64/128/255`), set the value using the **CapValueString** property. Otherwise, set the value using the **CapValue** property;
- Call **CapSet**() to actually set the value.

```
DWObject.OpenSource();
DWObject.Capability = EnumDWT_Cap.***; /*Make sure this Capability is TW_ONEVALUE*/
DWObject.CapGet(); /*Recommended*/
DWObject.CapType = EnumDWT_TWAINCONTAINERTYPE.TWON_ONEVALUE;
DWObject.CapValueType > 8 ?
/*STR*/DWObject.CapValue = someStringValue;
:
/*NUM*/DWObject.CapValue = someNonStringValue;
DWObject.CapSet();
```

TW_ARRAY

TW_ARRAY capability container type can be used to set or return a group of associated individual values.

To set a capability with TW_ARRAY container, the following steps are needed:

- Set the **Capability** property to the capability to be negotiated;
- Set the **CapType** property to TWON_ARRAY (`EnumDWT_CapType.TWON_ARRAY (3)`) (if you did CapGet() first, this step can be skipped);
- Set the **CapNumItems** property to indicate how many items are in the array;
- If the data type (**CapValueType**) of the capability is ***string*** (`EnumDWT_CapValueType.TWTY_STR32/64/128/255`), set the value using the **CapValueString** property. Otherwise, set the value using the **CapValue** property;

- Call **CapSet**() to actually set the value.

```
DWObject.OpenSource();
DWObject.Capability = EnumDWT_Cap.***; /*Make sure this Capability is TWON_ARRAY*/
DWObject.CapGet(); /*Recommended*/
DWObject.CapType = EnumDWT_TWAINCONTAINERTYPE.TWON_ARRAY;
DWObject.CapNumItems = *;
if(DWObject.CapValueType > 8 ){
/*STR*/
  DWObject. SetCapItemsString (0, someStringValue);
  DWObject. SetCapItemsString (1, someStringValue);
  …
} else {
/*NUM*/
  DWObject. SetCapItems(0, someNonStringValue);
  DWObject. SetCapItems(1, someNonStringValue);
  …
}
DWObject.CapSet();
```

TW_RANGE

TW_RANGE capability container type can be used to set or return a range of individual values describing a capability. The values are uniformly distributed between a minimum and a maximum value. The step size between every two values is constant.

To set a capability with TW_RANGE container, the following steps are needed:

- Set the **Capability** property to the capability to be negotiated;
- Set the **CapType** property to TWON_RANGE (`EnumDWT_CapType.TWON_RANGE (6)`) (if you did CapGet() first, this step can be skipped);
- Set the **CapMinValue**, **CapMaxValue**, **CapStepSize** and **CapCurrentValue** properties;
- Call **CapSet**() to actually set the value.

```
DWObject.OpenSource();
DWObject.Capability = EnumDWT_Cap.***; /*Make sure this Capability is TWON_RANGE*/
DWObject.CapGet(); /*Recommended*/
DWObject.CapType = EnumDWT_TWAINCONTAINERTYPE.TWON_RANGE;
DWObject.CapMinValue = 80;
DWObject.CapMaxValue = 200;
DWObject.CapStepSize = 20;
DWObject.CapCurrentValue = 100;
DWObject.CapSet();
```

TW_ENUMERATION

TW_ENUMERATION capability container type can be used to set or return a group of associated individual values describing a capability. The values are ordered from the lowest to highest values, but the step size between every two values is probably not uniform.

To set a capability with TW_ENUMERATION container, the following steps are needed:

- Set the **Capability** property to the capability to be negotiated;
- Set the **CapType** property to TWON_ENUMERATION (`EnumDWT_CapType.TWON_ENUMERATION (4)`) (if you did **CapGet**() first, this step can be skipped);
- Set the **CapNumItems** property to indicate how many items are in the array;
- If the data type (**CapValueType**) of the capability is ***string*** (`EnumDWT_CapValueType.TWTY_STR32/64/128/255`), set the value using the **CapValueString** property. Otherwise, set the value using the **CapValue** property;
- Set the **CapCurrentIndex** to indicate the index of the current value;
- Call **CapSet**() to actually set the value.

```
DWObject.OpenSource();
DWObject.Capability = EnumDWT_Cap.***; /*Make sure this Capability is TWON_ENUMERATION*/
DWObject.CapGet(); /*Recommended*/
DWObject.CapType = EnumDWT_TWAINCONTAINERTYPE.TWON_ENUMERATION;
DWObject.CapNumItems = *;
if(DWObject.CapValueType > 8 ){
/*STR*/
  DWObject. SetCapItemsString (0, someStringValue);
  DWObject. SetCapItemsString (1, someStringValue);
  …
} else {
/*NUM*/
  DWObject. SetCapItems(0, someNonStringValue);
  DWObject. SetCapItems(1, someNonStringValue);
  …
}
DWObject.CapCurrentIndex = 1;
DWObject.CapSet();
```

Using a custom capability that the TWAIN Specification doesn’t have

Some TWAIN sources come with their own custom capabilities that are only supported by their corresponding devices. In such cases, you can refer to the following KB article on how to make use of these capabilities.

http://developer.dynamsoft.com/dwt/kb/2724