Dynamic Web TWAIN 9.0

Developer's Guide

March 26, 2013

Dynamsoft™

The leading provider of version control solutions and TWAIN SDK

Contents

Preface

Description

This guide provides an instruction to the Dynamsoft's Dynamic Web TWAIN SDK. It provides a general overview of the types of things you can do with the SDK and the APIs that are available to you through the SDK.

Audience

This guide is meant for both experienced Dynamic Web TWAIN SDK developers and those who are still new to our SDK.

For developers who are new to the SDK, this guide provides a general description and basic technologies you can use to develop a scanning page in your web application.

For those who have used earlier versions of the SDK, this guide provides information on advanced APIs to help polish your scanning page and new features you can take advantage in this version of Dynamic Web TWAIN SDK.

Getting Started

What Is TWAIN

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

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

We, Dynamsoft Corporation, are also a member of TWAIN Working Group.

What Is Dynamic Web TWAIN

Dynamic Web TWAIN is a TWAIN scanning SDK specifically optimized for **web apps**. This TWAIN interface allows you to write code of a couple of lines which supports **scanning** documents from **TWAIN compatible** scanners or **acquiring** images from digital cameras, webcams or capture cards. Then users can **edit** the images and **save** them to a variety of formats or a remote database.

The TWAIN control also supports loading and processing local images.

Basic Requirements

- **Server Side:**
  1. Operation System: Windows, Linux, UNIX, Mac, etc.
  2. Web Server: IIS, Apache, Tomcat, ColdFusion, etc.
  3. Programming Languages: ASP,VB.NET,C#,PHP,JSP,CFM, etc.

- **Client Side:**
  1. Web Browser:
     ActiveX Edition: IE (32-bit or 64-bit),
     Plug-in Edition: Chrome, Firefox, Safari, or Opera on Windows
     Mac Edition: Safari, Opera, Chrome and Firefox on Mac OS X 10.5 or above.
  2. Scanner/Camera: TWAIN compatible.

3. Please verify the following security settings of IE are set to "Prompt" or "Enabled" on the client machine:
   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 three editions: ActiveX Edition, Plug-in Edition and Mac Edition.

- **ActiveX Edition** is for use in IE (both 32bit and 64-bit) on Windows;
- **Plug-in Edition** is for Firefox, Chrome, Safari and Opera on Windows;
- **Mac Edition** is for Safari, Firefox, Chrome and Opera on Mac.

According to what browsers and operating systems your application is going to support, you can decide which edition(s) of Dynamic Web TWAIN to use.

Build the "Hello World" page

There are several easy steps involved to deploy and integrate Dynamic Web TWAIN into your client web page. Here we are going to guide you through these steps by adding basic Scan functionality to a simple web page, so as to give you the flavor about the necessary steps to make Dynamic Web TWAIN work in your web page.

For advanced features

To suit your web page to more OS and Web Browser types, and to know more about the Dynamic Web TWAIN license details, , you need to find the relevant information in section Advanced Techniques.

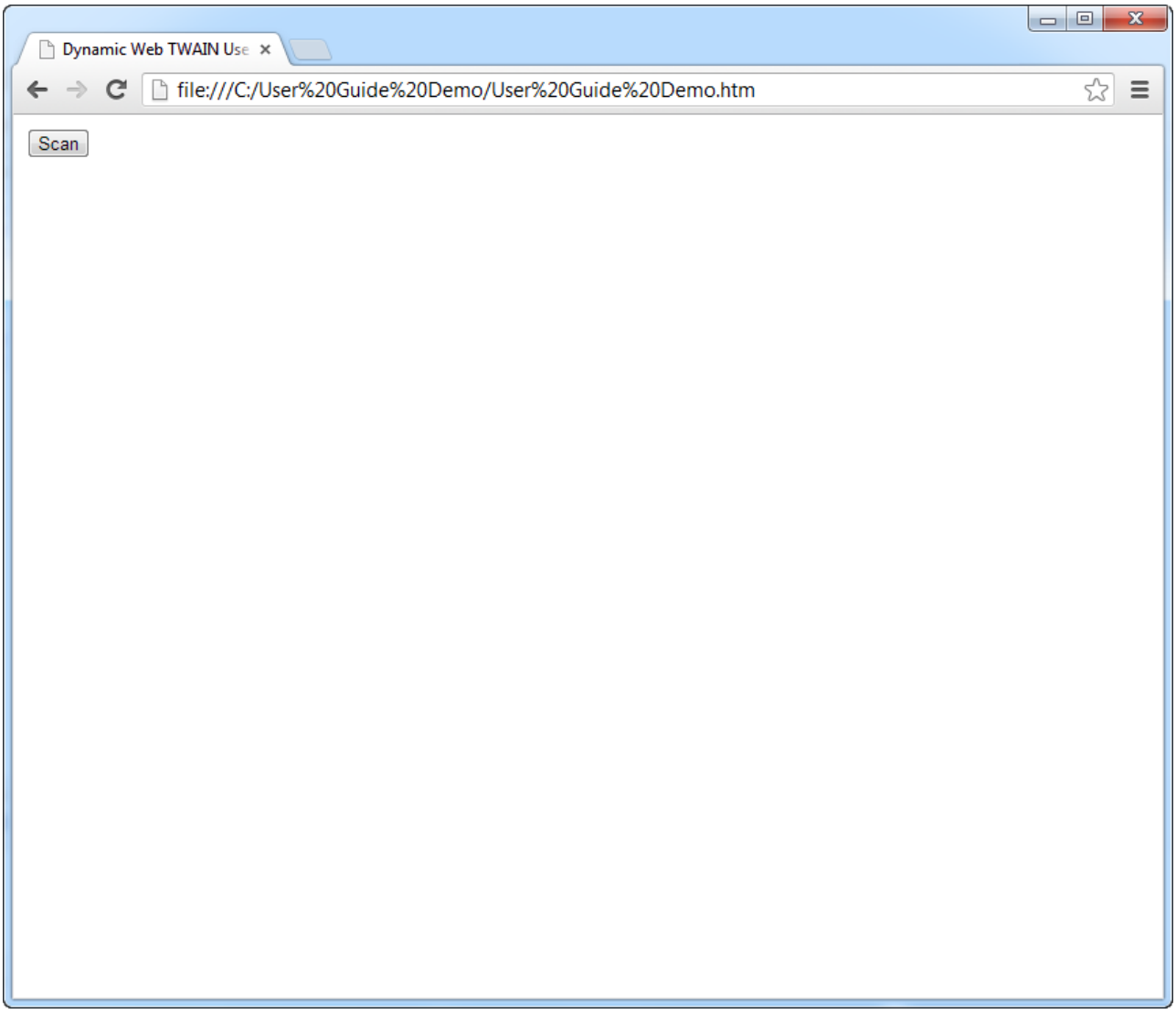
To integrated more functionality, such as Image Viewer, load/save image(s) from/to local file system, download/upload image(s) from/to the web server etc., into your web pages, you need to find the relevant information in section Explore the Features.

Starting From an Dummy Web Page

First let's create an html file, named User Guide Demo.htm. The code of User Guide Demo.htm is as following,

```
<html>
<head>
    <title>Dynamic Web TWAIN User Guide Demo</title>
</head>
<body>
    <input type="button" value="Scan" />
</body>
</html>
```

The web page User Guide Demo.htm only has a dummy “Scan” button, and should look like below,



Define Place Holder for Dynamic Web TWAIN object

We named Dynamic Web TWAIN object **DWObject** throughout the code implementation. In order to use Dynamic Web TWAIN object **DWObject**, you need to define a place holder for Dynamic Web TWAIN in your web page (HTML).

```
<html>
<head>
    <title>Dynamic Web TWAIN User Guide Demo</title>
</head>
<body>
    <input type="button" value="Scan" />
    <div id="dwtcontrolContainer">
```

```
        <div id="DWTContainerID" >
                <!-- Place holder for the ActiveX 32bit -->
        </div>
    </div>
</body>
</html>
```

Initialize DWObject by Including JavaScript File Provided

Second, include the JavaScript file DWT_BasicPageInitiate.js provided by Dynamsoft Corporation, to fully initialize the Dynamic Web TWAIN object **DWObject**.

```
<html>
<head>
    <title>Dynamic Web TWAIN User Guide Demo</title>
</head>
<body>
    <input type="button" value="Scan" />
    <div id="dwtcontrolContainer">
        <div id="DWTContainerID" >
                <!-- Place holder for the ActiveX 32bit -->
        </div>
    </div>
    <input id="IfInitInfo" style="visibility:hidden ;" value = 0 /> <!-- Disable function InitInfo()-->
<script type="text/javascript" language="javascript" src="Scripts/DWT_BasicPageInitiate.js">
</script>
</body>
</html>
```

DWObject initialization is actually done in the JavaScript file DWT_BasicPageInitiate.js provided. With DWT_BasicPageInitiate.js, you can initialize **DWObject** easily and make use of it later.

DWT_BasicPageInitiate.js file can either be downloaded from the link directly: http://www.dynamsoft.com/download/Support/DWT/DWT_BasicPageInitiate.js, or be found from any download package file of the samples page: http://www.dynamsoft.com/Downloads/WebTWAIN-Sample-Download.aspx .

And of course, you can always check and modify DWT_BasicPageInitiate.js to suit best your business scenarios. The relevant information can be found in section Advanced Techniques.

Control Dynamic Web TWAIN object with JavaScript

Once Dynamic Web TWAIN object **DWObject** is fully initialized, you can control it just like other JavaScript object.

Basically, there are 3 ways to manage Dynamic Web TWAIN object **DWObject** directly:

Via Properties

Properties are used to retrieve or set a certain value in Dynamic Web TWAIN object. For example, Resolution, Duplex, IfShowUI, etc…

Via Methods

Methods are used to call the build-in functions of Dynamic Web TWAIN object. For example, AcquireImage, SaveAsJPEG, Rotate, etc… The syntax is fairly simple:

```
//Property
DWObject.Resolution = 200; //Scan pages in 200 DPI
//Method
DWObject.Rotate(Int16 sImageIndex, float fAngle, Bool bKeepSize);
```

Via Events

Events are triggered when certain trigger points are reached. For example, we have **OnMouseClick** event for mouse clicking, **OnPostTransfer** event for end of transferring one image, etc. For more information, please refer to Handling Events in section Explore the Features for more details.

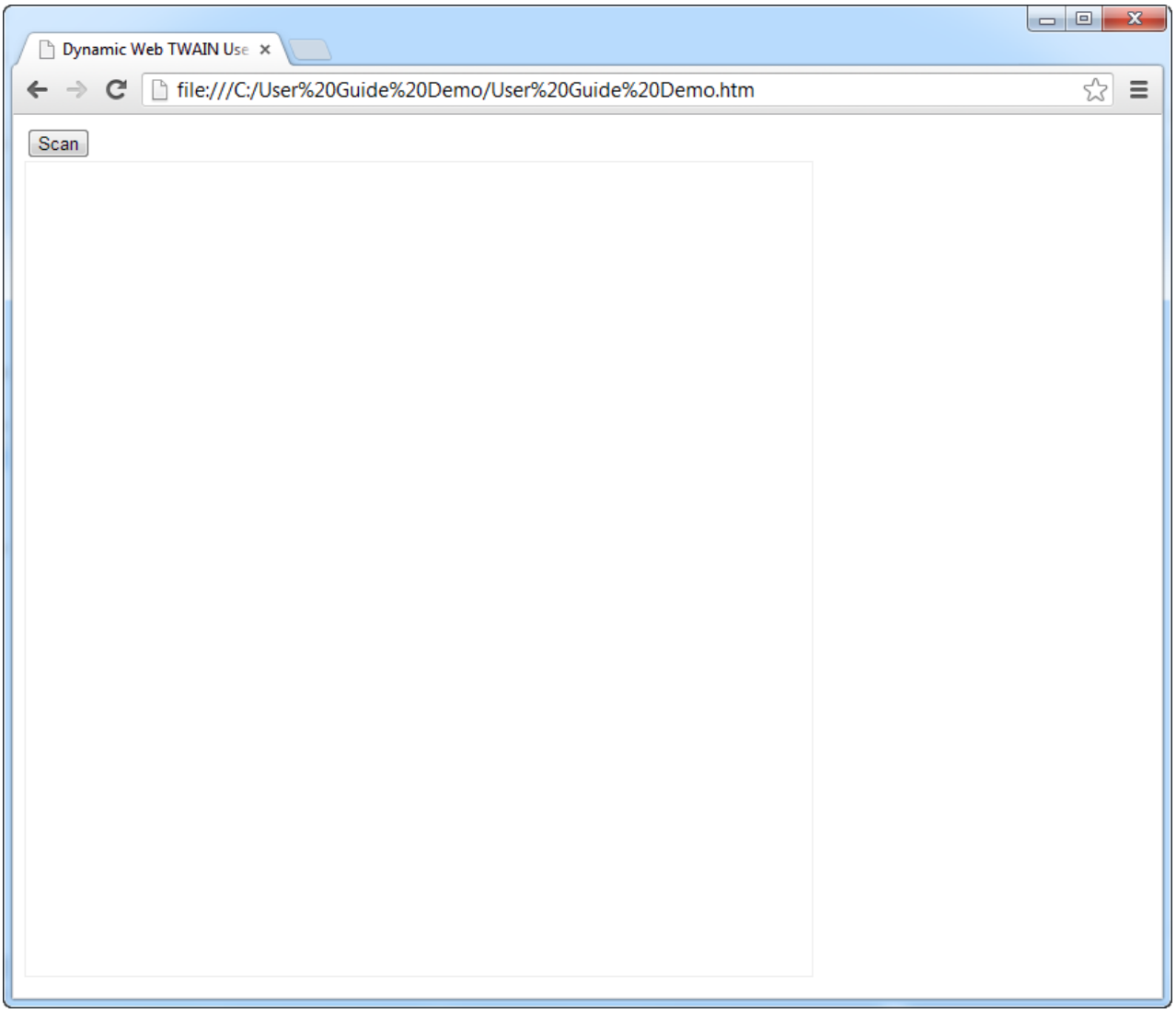
For our “scan” example in User Guide Demo.htm, we are going to add a JavaScript function Simple_AcquireImage() and add it to the “Scan” button.

The function Simple_AcquireImage() managed Dynamic Web TWAIN object **DWObject** via both ways, by Properties, like IfShowUI and Methods like SelectSource(). The code is as shown below,

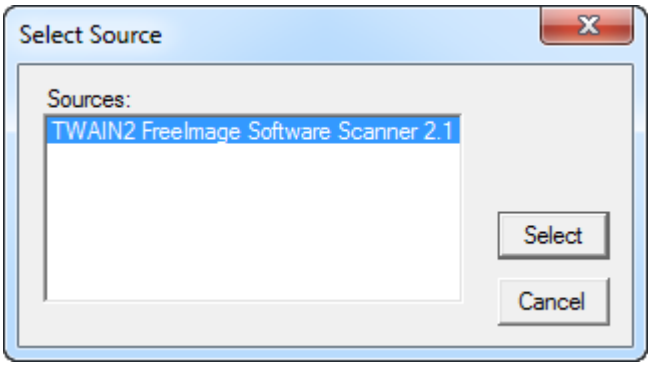
```
<html>
<head>
    <title>Dynamic Web TWAIN User Guide Demo</title>
</head>
<body>
    <input type="button" value="Scan" onclick = "Simple_AcquireImage();" />
    <div id="dwtcontrolContainer">
        <div id="DWTContainerID" >
                <!-- Place holder for the ActiveX 32bit -->
        </div>
```

```
    </div>
    <input id="IfInitInfo" style="visibility:hidden ;" value = 0 /> <!-- Disable function InitInfo()-->
<script type="text/javascript" language="javascript" src="Scripts/DWT_BasicPageInitiate.js">
</script>

<script type="text/javascript" language="javascript">
function Simple_AcquireImage() {

    DWObject.SelectSource();
    DWObject.CloseSource();
    DWObject.OpenSource();

    DWObject.IfShowUI = false;

    DWObject.AcquireImage();
}
</script>

</body>
</html>
```

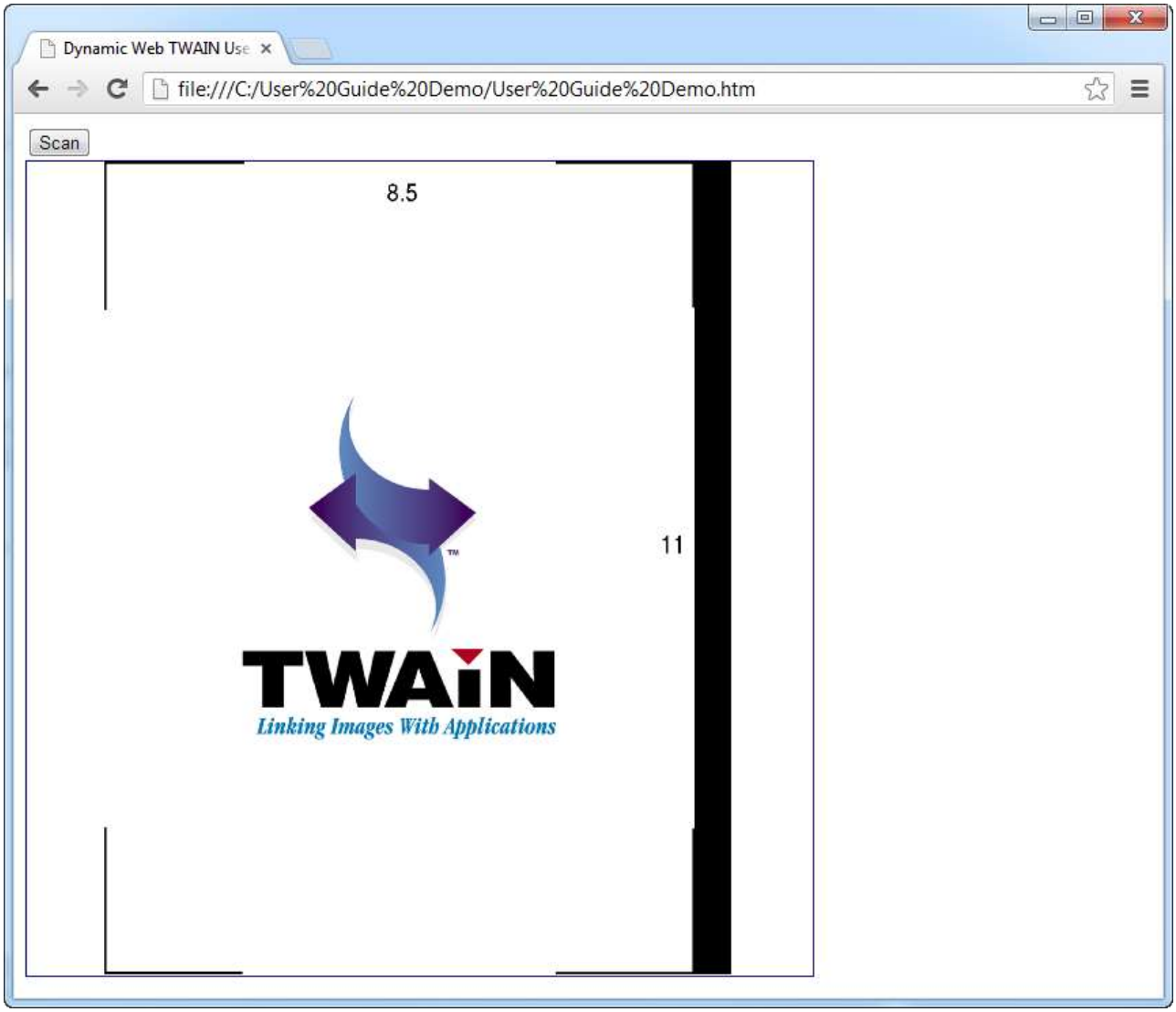
Refresh the web page once the code change is done. Your User Guide Demo.htm should appear like below,



Now, click “Scan” button. A window as below will prompt out asking which scanner you would like to use. If no scanner is available, please check the following link http://kb.dynamsoft.com/questions/541/Why+is+my+scanner+not+shown+or+not+responding+in+the+browser%3F for more information.



In our demo, we simply select “TWAIN2 FreeImage Software Scanner 2.1” for testing purpose. Once the “Select” button is clicked, the web page will scan a TWAIN sample page instantly, as shown below,



Bravo! Scan as you like. Your web page now has scan functionality integrated, after following three steps listed, Define Place Holder for Dynamic Web TWAIN object, Initialize DWObject by Including JavaScript File Provided and Control Dynamic Web TWAIN object with JavaScript.

Advanced Techniques

As shown from the above example, Dynamic Web TWAIN is easy to be integrated into your web page. But in order to suit your web page to more OS and Web Browser types, and also to get to know more about the Dynamic Web TWAIN license details, you need to look into some details of the JavaScript file included, namely DWT_BasicPageInitiate.js.

OS and Web Browser Type Detection

In JavaScript file DWT_BasicPageInitiate.js, you need to detect the OS and browser type. Because in different browsers, we use different ways to create the Web TWAIN object.

```
function DW_PageonloadInner() { //Detect Environment
    // Get User Agent Value
    ua = (navigator.userAgent.toLowerCase());

    // Set the Explorer Type
    if (ua.indexOf("msie") != -1)
        DW_InIE = true;
    else
        DW_InIE = false;

    // Set the Operating System Type
    if (ua.indexOf("macintosh") != -1)
        DW_InWindows = false;
    else
        DW_InWindows = true;

    // Set the x86 and x64 type
    if (ua.indexOf("win64") != -1 && ua.indexOf("x64") != -1)
        DW_InWindowsX86 = false;
    else
        DW_InWindowsX86 = true;
}

function DW_CreateControl() {
    var objString = "";
    var DWTContainer;

    // For IE, render the ActiveX Object
    if (DW_InIE) {
        /*Only useful in version 8.* or earlier
        ////////////////////////////////////////
        objString = "<object classid='clsid:" + DW_PROCLASSID + "' style='display:none;'><param
name='LPKPath' value='" + DW_LPKPath + "'/></object>";
        ////////////////////////////////////////
        */
        objString += "<object id='" + DW_ObjectName + "' style='width:" + DW_Width + "px;height:" +
DW_Height + "px'";
```

```
        if (DW_InWindowsX86)
            objString += "codebase='" + DW_CABX86Path + "#version=" + DW_VersionCode + "' ";
        else
            objString += "codebase='" + DW_CABX64Path + "#version=" + DW_VersionCode + "' ";

        var temp = DW_IsTrial ? DW_TRAILCLASSID : DW_FULLCLASSID;
        objString += " classid='clsid:" + temp + "' viewastext>";
        objString += " <param name='Manufacturer' value='DynamSoft Corporation' />";
        objString += " <param name='ProductFamily' value='" + DW_ProductName + "' />";
        objString += " <param name='ProductName' value='" + DW_ProductName + "' />";
        //objString += " <param name='wmode' value='transparent'/>  ";
        objString += " </object>";
    }
    // For non-IE, render the embed object
    else {
        objString = " <embed id='" + DW_ObjectName + "'style='display: inline; width:" + DW_Width +
"px;height:" + DW_Height + "px' id='" + DW_ObjectName + "' type='" + DW_MIMETYPE + "'";
        objString += " OnPostTransfer='Dynamsoft_OnPostTransfer'
OnPostAllTransfers='Dynamsoft_OnPostAllTransfers'";
        objString += " OnMouseClick='Dynamsoft_OnMouseClick'
OnPostLoad='Dynamsoft_OnPostLoadfunction'";
        objString += " OnImageAreaSelected = 'Dynamsoft_OnImageAreaSelected'";
        objString += " OnImageAreaDeSelected = 'Dynamsoft_OnImageAreaDeselected'";
        objString += " OnMouseDoubleClick = 'Dynamsoft_OnMouseDoubleClick'";
        objString += " OnMouseRightClick = 'Dynamsoft_OnMouseRightClick'";
        objString += " OnTopImageInTheViewChanged = 'Dynamsoft_OnTopImageInTheViewChanged'";
        objString += " OnGetFilePath='Dynamsoft_OnGetFilePath'";
        if (DW_InWindows)
            objString += " pluginspage='" + DW_MSIPath + "'></embed>";
        else
            objString += " pluginspage='" + DW_PKGPath + "'></embed>";
    }

    DWTContainer = document.getElementById(DW_DWTContainerID);
    DWTContainer.innerHTML = objString;
    DWObject = document.getElementById(DW_ObjectName);
}

function DW_Pageonload() {
    DW_PageonloadInner();  //Detect environment
    if (document.getElementById("IfInitInfo").value != 0) {
        InitInfo();           //Add guide info
```

```
    }
    DW_CreateControl(); //Create an instance of the component in the DIV assigned by DW_DWTContainerID

    vShowNoControl = false; //By default, we assume the control is not loaded
    //Set interval to check if the control is fully loaded.
    DW_Seed = setInterval(DW_ControlDetect, 500);
}

// Check if the control is fully loaded.

function DW_ControlDetect() {
    // If the ErrorCode is 0, it means everything is fine for the control. It is fully loaded.
    if (DWObject.ErrorCode == 0) {
        /*Only useful in version 9.0 or later*/
        ////////////////////////////////////// Please put your product key below
        DWObject.ProductKey = "";
        //////////////////////////////////////

        DW_Pause();
        // For IE, attach events
        if (DW_InIE) {
            DWObject.attachEvent('OnPostTransfer', Dynamsoft_OnPostTransfer);
            DWObject.attachEvent('OnPostAllTransfers', Dynamsoft_OnPostAllTransfers);
            DWObject.attachEvent('OnMouseClick', Dynamsoft_OnMouseClick);
            DWObject.attachEvent('OnPostLoad', Dynamsoft_OnPostLoadfunction);
            DWObject.attachEvent('OnImageAreaSelected', Dynamsoft_OnImageAreaSelected);
            DWObject.attachEvent('OnMouseDoubleClick', Dynamsoft_OnMouseDoubleClick);
            DWObject.attachEvent('OnMouseRightClick', Dynamsoft_OnMouseRightClick);
            DWObject.attachEvent('OnTopImageInTheViewChanged',
Dynamsoft_OnTopImageInTheViewChanged);
            DWObject.attachEvent('OnImageAreaDeSelected', Dynamsoft_OnImageAreaDeselected);
            DWObject.attachEvent('OnGetFilePath', Dynamsoft_OnGetFilePath);
        }
    }
    else {
        if (vShowNoControl == false) {
            DW_NoControl();
            vShowNoControl = true;
        }
    }
    DW_Timeout = setTimeout(function () { }, 10);
}
```

1. Events for IE can be attached later using the method **attachEvent** in the function **DW_ControlDetect().**

2. As you can see in the code, we check the ErrorCode property of the TWAIN object every 0.5 second. This is to make sure that Dynamic Web TWAIN object is fully initialized before you use it. Dynamic Web TWAIN object may take a little time to initialize and this is the recommended way to handle it.

Verify License

Basic info

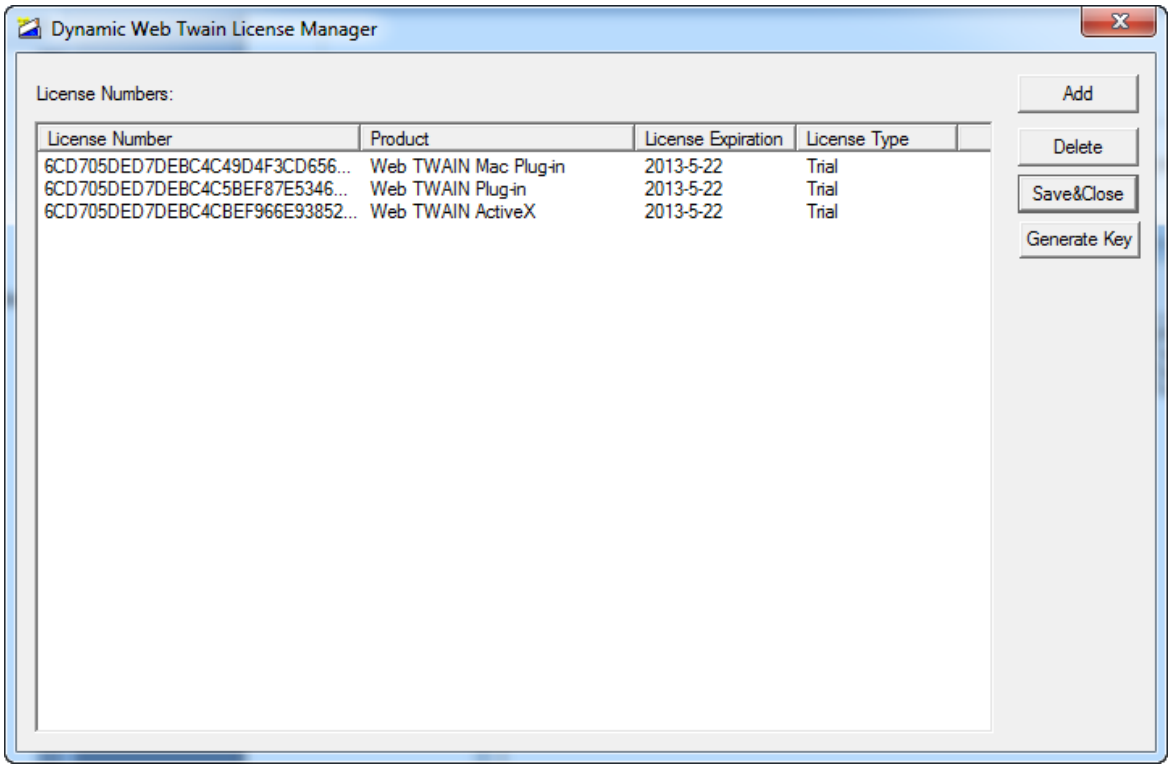
Since version 9.0, Dynamic Web TWAIN use ProductKey property to set a series of alphanumeric code for license verification at runtime. All editions of the authentication mechanism are the same.

For developers, they only need to generate a product key with Licensing Tool for each of the web servers which they want to deploy their application on.

For end users, it would be much easier for them to activate the control.

How to activate Dynamic Web TWAIN with license?

1) Open the Licensing Tool which is normally located at “{Disk}:\Program Files (x86)\Dynamsoft\Dynamic Web TWAIN {version}\”.
2) In the Licensing Tool, click "Generate Key" button to generate a product key.
3) At runtime, assign the product key to the property ProductKey before you use any of the other methods, properties of Dynamic Web TWAIN.



How to use ProductKey?

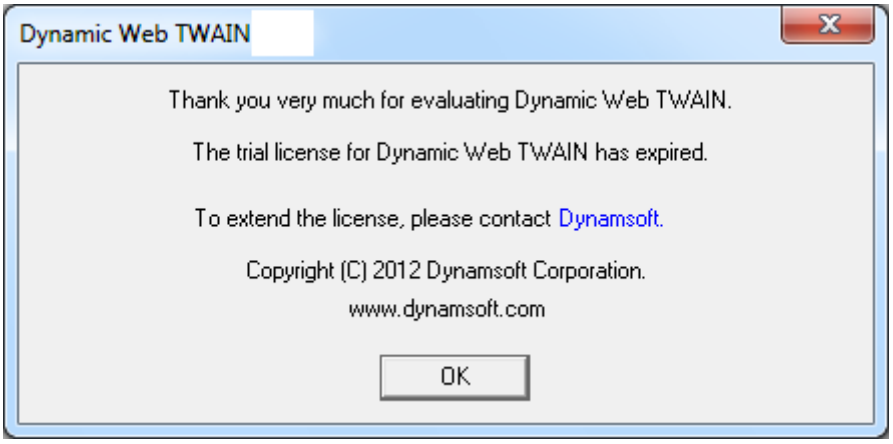
Set ProductKey in the JavaScript file DWT_BasicPageInitiate.js:

```
/*Only useful in version 9.0 or later*/
////////////////////////////////////// Please put your product key below
DWObject.ProductKey = "";
//////////////////////////////////////
```

For more details about how to use the license, please check the page:
http://www.dynamsoft.com/help/TWAIN/WebTwain/Programmer%20Guide/Guide%20How%20to%20use%20the%20license.htm

Common license errors

As stated in the error message, the license is missing or the ProductKey is invalid. Please make sure you have assigned the valid key to ProductKey.



As stated in the error message, the trial license has expired thus the ProductKey has also expired. In this case, you can send a mail to our support team (twainsupport@dynamsoft.com) and ask for an extended license.

Explore the Features

Use Image Viewer

What is the image viewer

Image Viewer is a built-in feature of Dynamic Web TWAIN. You can use the image viewer to manage images you have acquired.

What can you achieve with image viewer

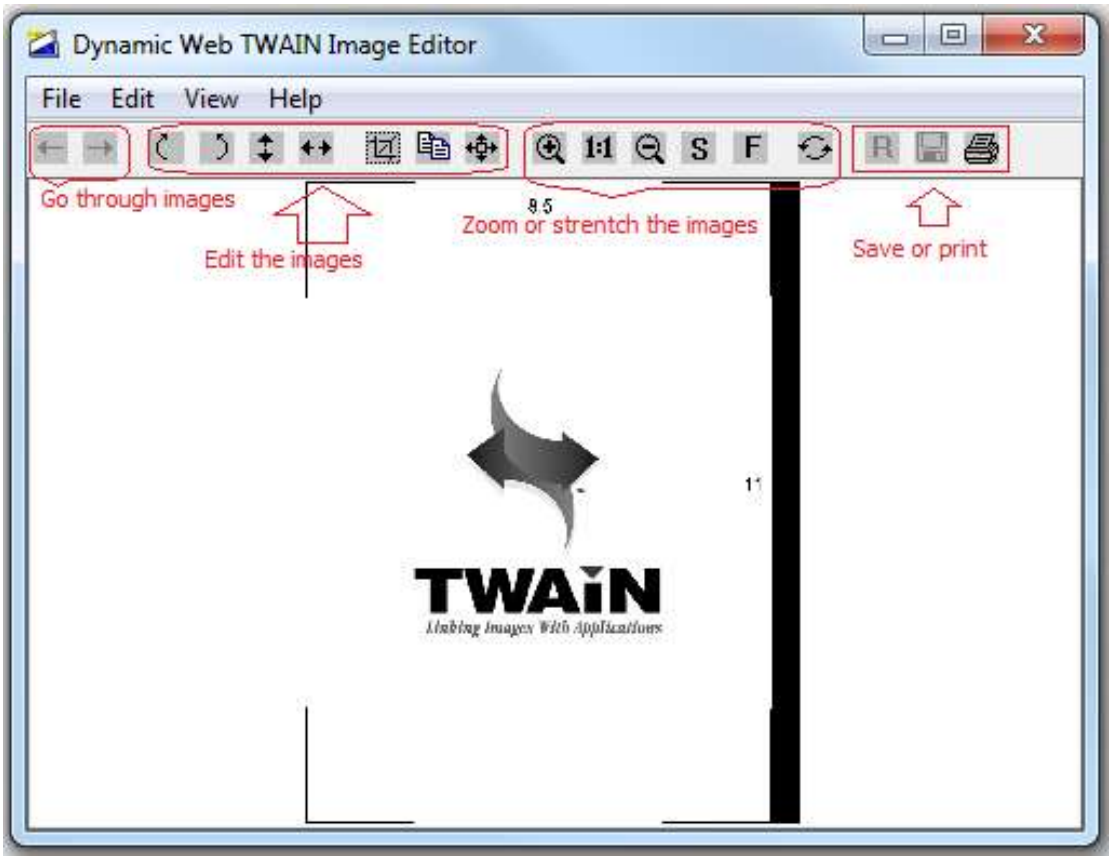
In Image Viewer, you can:

1. Go through all the images currently scanned or loaded, you can zoom in or out the images.
2. Edit an image in the following ways: Rotate, Mirror, Crop, Change size and etc…
3. Save changes or print the image directly

How to use the image viewer

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

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

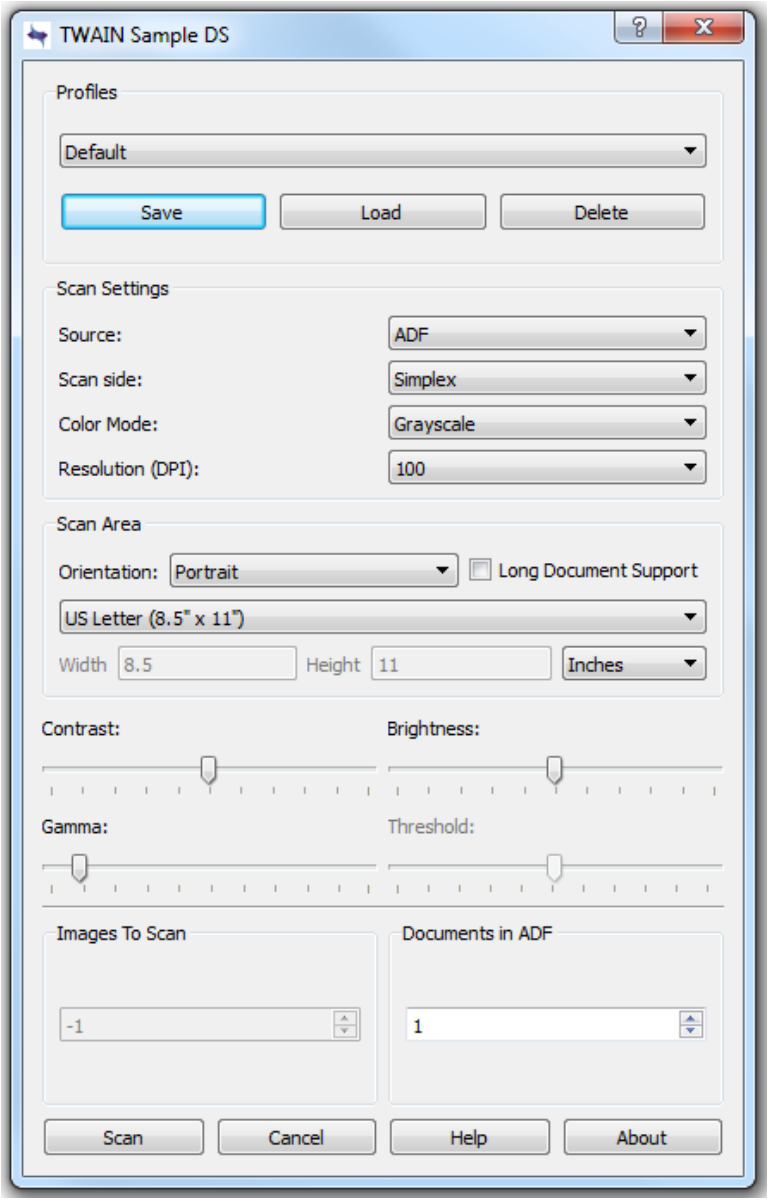


Customize the editor

Currently you can change the title of the editor window, make it a modal dialog, and make it read-only. Also, you can choose whether or not to allow the users to go through images. The related properties are: ImageEditorIfEnableEnumerator, ImageEditorIfReadonly, ImageEditorIfModal, and ImageEditorWindowTitle.

Customize your scan settings

Before you start an actual scan session, you can choose how you'd want your documents to be scanned. Normally, you can do all of the settings in the User's Interface of a scanner as you can see below:



All these settings seem overwhelming for end users, especially for these 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 = 1; //Scan pages in Grey
DWObject.Resolution = 200; //Scan pages in 200 DPI
DWObject.AcquireImage(); //Start scanning
```

Related settings

BitDepth	Brightness	Contrast	ImageBitsPerPixel	IfAutoDiscardBlankpages
IfAutoFeed	IfAutoScan	IfDuplexEnabled	IfFeederEnabled	IfAutomaticBorderDetection
Duplex	IfShowUI	IfAutoBright	ImageLayoutPageNumber	IfAutomaticDeskew
JPEGQuality	PageSize	PendingXfers	IfPaperDetectable	ImageLayoutDocumentNumber
PixelFlavor	PixelType	Resolution	ImageLayoutFrameRight	ImageLayoutFrameBottom
TransferMode	XferCount	IfFeederLoaded	ImageLayoutFrameTop	ImageLayoutFrameLeft

Manipulate the image(s)

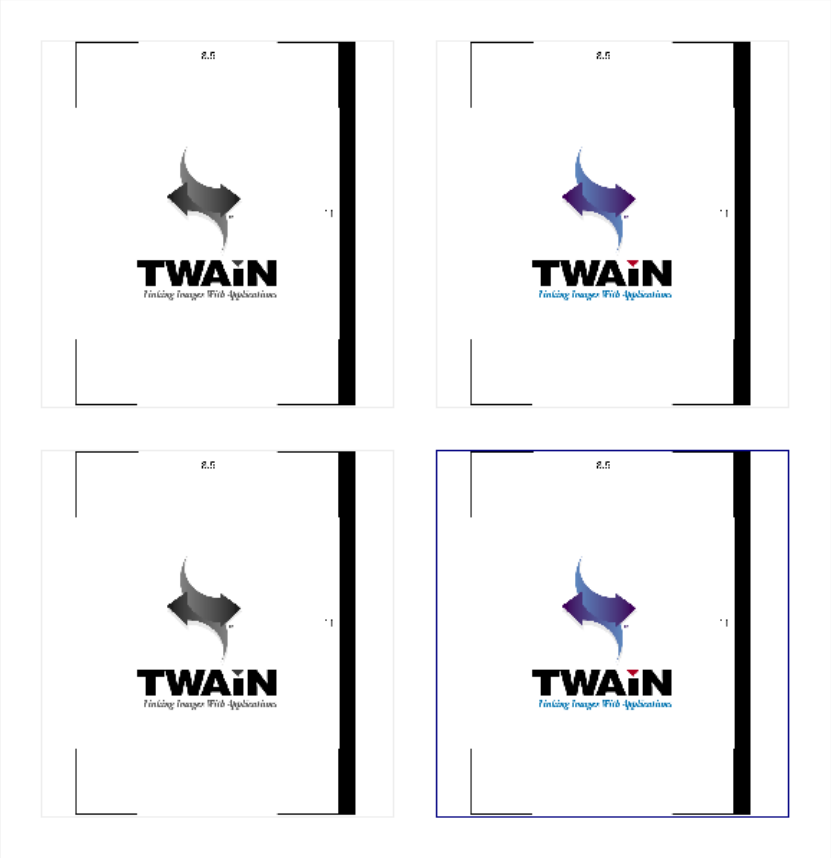
When you have scanned or loaded an image in Dynamic Web TWAIN, you can start manipulating the image. You can

1. Go through each image by changing the currently displayed image or the property CurrentImageIndexInBuffer

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

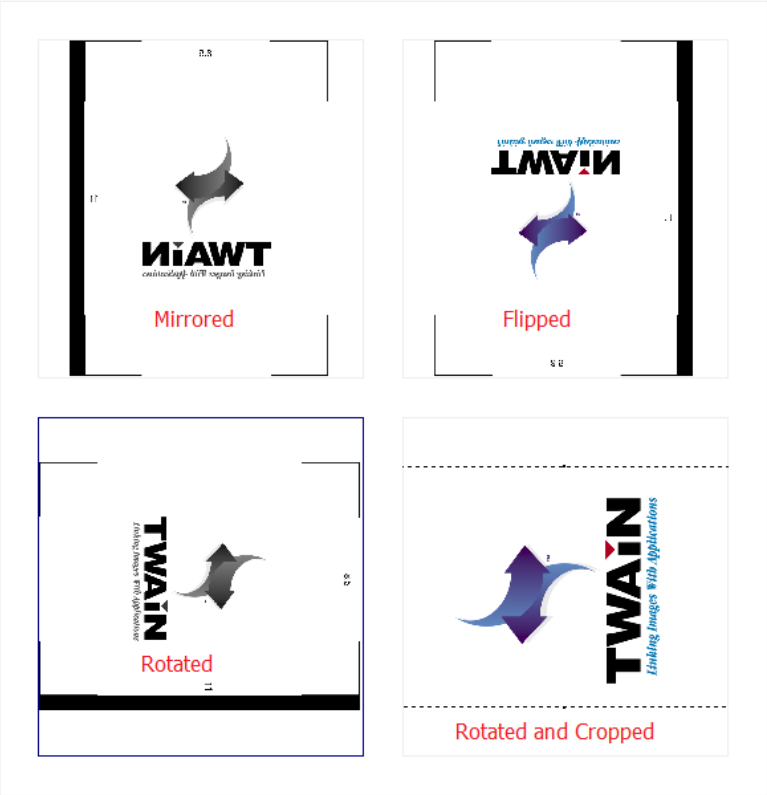
2. Show multiple images by changing the view mode to larger than 1*1. SetViewMode()

```
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);
```



Related methods

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

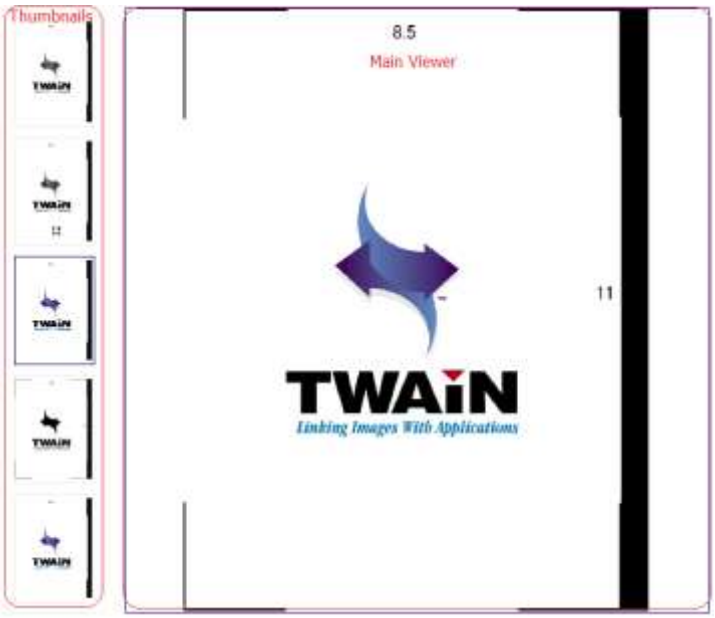
Use Thumbnail

Description

There isn't a built-in thumbnail for Dynamic Web TWAIN. To show the thumbnails, you can put 2 Dynamic Web TWAIN controls on the page. Below is how it would be look like:

*Please NOTE that 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 plus it's set to hold only 1 image at a time.*

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



With the above implementation, you only need to use the thumbnails of the components for scanning/editing/uploading, etc. And you update the current image to the main viewer when it's changed. For example, when you click on one of the images in the thumbnails, you copy it to the main viewer:

```
function DynamicDWObjectLeft_OnMouseClick(index) {
  DWObjectLeft.CopyToClipboard(index); //Copy the image you just clicked on
  DWObjectRight.LoadDibFromClipboard(); //Load the same image
}
```

Load 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 the below article: http://kb.dynamsoft.com/questions/612/

Calling the method

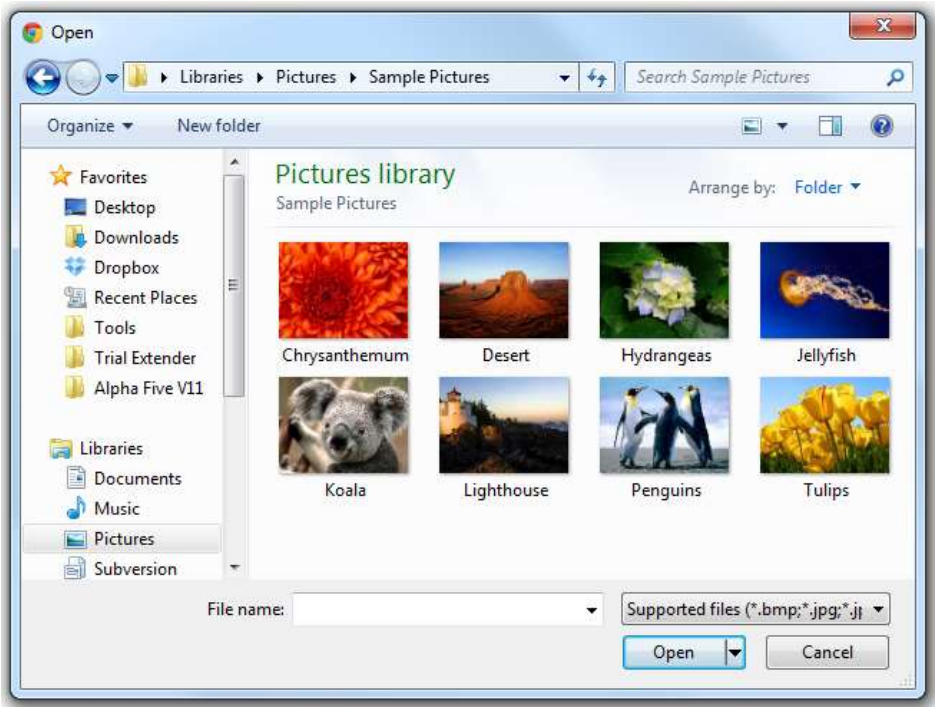
With Dynamic Web TWAIN, you can load the local images with the methods LoadImage() or LoadImageEx(). Below is a simple code snippet for you:

```
DWObject.LoadImage("G:/wwwroot/WebTWAIN/Images/ImageData.jpg");
DWObject.LoadImageEx("G:/wwwroot/WebTWAIN/Images/ImageData.jpg",1);
```

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("",5);
```

Please NOTE that the second parameter "ImageType" in the method LoadImageEx() would determine the format filter in the "Select File…" dialog.



Save the image(s) to local system

Preparation

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

Calling the method

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

Format	Methods
All supported formats	SaveAsBMP() SaveAsJPEG() SaveAsPDF() SaveAsPNG() SaveAsTIFF()
Multi-page PDF	SaveSelectedImagesAsMultiPagePDF() SaveAllAsPDF()
Multi-page TIFF	SaveAllAsMultiPageTIFF() SaveSelectedImagesAsMultiPageTIFF()

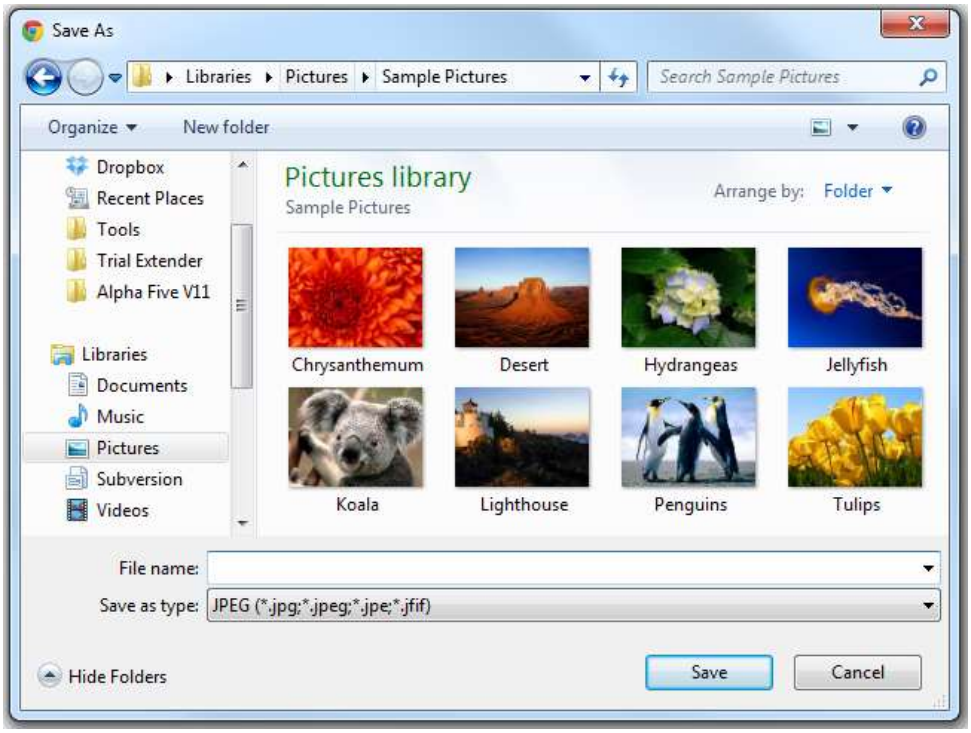
Code snippet:

```
DWObject.SaveAsJPEG("G:/wwwroot/WebTWAIN/Images/ImageData.jpg",0);
DWObject.SaveAllAsPDF("G:/wwwroot/WebTWAIN/Images/ImageData.pdf");
```

From the above code, you can see that you need to provide the complete file path to save an image locally. Like loading an image, Dynamic Web TWAIN can 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:

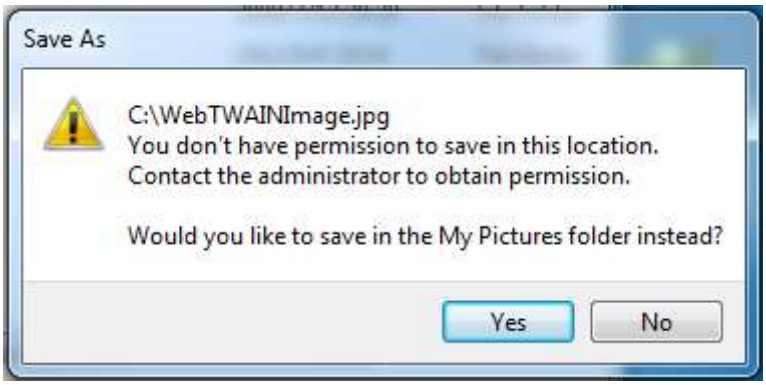
```
DWObject.IfShowFileDialog = true;
DWObject.SaveAsJPEG("",0);
```

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



Possible issue:

On Windows Vista or above, the strict security means you can only save images to certain places that you have write permission. If you try to save to other places, you will get the below error:



Upload the image(s) to web server

Preparation

Before we upload the image(s), we need to set the server address, port and 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 server disk or database or email the data.

Here is an example:

```
var strHTTPServer = location.hostname;
DWObject.HTTPPort = location.port == "" ? 80 : location.port;
var CurrentPathName =
    unescape(location.pathname); // get current PathName in plain ASCII
var CurrentPath =
    CurrentPathName.substring(0, CurrentPathName.lastIndexOf("/") + 1);
var strActionPage = CurrentPath + "actionPage.aspx"; //the ActionPage's file path
var uploadfilename = "TestImage.pdf";
```

The **strHTTPServer** variable is used to store the server address. The server address specifies which server you want to upload the image(s) to, it can be either the hostname or the ip of the server. If you want to upload the image(s) to the same server as the current page, we suggest you use the JavaScript object: location to get the hostname in runtime as shown above.

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

The **CurrentPathName** and **CurrentPath** variable is used to build the relative path of the action page.

The **strActionPage** variable stores the relative path of the action page.

The **uploadfilename** variable stores the file name for the uploaded images. You should change the extension of the name accordingly.

Calling the method

Ok, now we can call the HTTP upload method to upload the image(s). We have 14 methods to upload the image data:

Format\method	HTTP Post	HTTP Put
All formats	HTTPUploadThroughPostDirectly()	HTTPUploadThroughPutDirectly()
All supported formats	HTTPUploadThroughPost() HTTPUploadThroughPostEx()	HTTPUploadThroughPut() HTTPUploadThroughPutEx()
Multi-page PDF	HTTPUploadAllThroughPostAsPDF() HTTPUploadThroughPostAsMultiPagePDF()	HTTPUploadAllThroughPutAsPDF() HTTPUploadThroughPutAsMultiPagePDF()
Multi-page TIFF	HTTPUploadAllThroughPostAsMultiPageTIFF() HTTPUploadThroughPostAsMultiPageTIFF()	HTTPUploadAllThroughPutAsMultiPageTIFF() HTTPUploadThroughPutAsMultiPageTIFF()

As you can see in the table above, there are 7 different methods to post image(s) with HTTP Post (recommended). And there are 7 corresponding methods for HTTP Put.

HTTPUploadThroughPostDirectly() and **HTTPUploadThroughPutDirectly()** are special. With these methods, you can upload a local file (any type of file) directly to the web server without loading it in the component.

Since we have prepared the server address, action page path and file name, the calling of the HTTP upload method is quite simple. Let's use HTTPUploadAllThroughPostAsPDF() as an example:

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

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

If you want to upload one image only, you can use **HTTPUploadThroughPost (Ex)()** or **HTTPUploadThroughPut (Ex)()**.

If you want to upload selected images, you can use **HTTPUploadThroughPostAsMultiPagePDF()**, **HTTPUploadThroughPostAsMultiPageTIFF()** , **HTTPUploadThroughPutAsMultiPagePDF()** or **HTTPUploadThroughPutAsMultiPageTIFF()**.

Action Page

After the HTTP upload method, the HTTP request will reach your action page. Your action page can be written in most of the server-side languages (C#, VB, PHP, Java, etc...)

Here is the example in C#:

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

Note: RemoteFile is the default value for the image data field. If you like, you can change it with the property HttpFieldNameOfUploadedImage.

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

Here is the example in PHP:

```
$fileTempName = $_FILES['RemoteFile']['tmp_name'];
$fileSize = $_FILES['RemoteFile']['size'];
$fileName = $_FILES['RemoteFile']['name'];

if (file_exists($fileName))
 $fWriteHandle = fopen($fileName, 'w');
else
 $fWriteHandle = fopen($fileName, 'w');

$fReadHandle = fopen($fileTempName, 'rb');
$fileContent = fread($fReadHandle, $fileSize);
fwrite($fWriteHandle, $fileContent);
fclose($fWriteHandle);
```

PHP action page use the $_FILES object to retrieve the image data and save it to the current directory.

Other information

Dynamic Web TWAIN Plugin edition sends two requests to the action page. The first one is blank while the second is the one that contains the image data. The first one is used to verify the authentication. If your action page would throw exception on receiving blank request, you can modify your action page to ignore the blank request, or make Dynamic Web TWAIN skip the first request with AllowPluginAuthentication property set to false.

Beside the HTTP upload methods, you can also use the FTP Upload methods to update image(s) to your web server. Of course you would need to have a FTP site on your web server to receive the upload request and store the file.

You can search keyword ‘FTPUpload’ in the help document for more details.

Upload the image(s) to Database

Actually there is no way to directly upload the image(s) to your database. What we can do is upload the image data to the action page first and then use the server-side code 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 topic “Upload the image(s) to 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 the example in C# and 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 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”.

Upload 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 “Upload the image(s) to 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 HTTP upload method, we need to use a method called “SetHTTPFormField”.

SetHTTPFormField(String FieldName, String FieldValue)

To use this method, you just need to pass two parameters:

- String FieldName: specifies the name of a text field in web form.
- String FieldValue: specifies the value of a text field in 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. Just like other standard HTTP Form field in classic HTML web application.

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

Download and load image from the web

You can use **HTTPDownload()** or **HTTPDownloadEx()** method to download an image from the web and load it in Dynamic Web TWAIN.

```
DWObject.HTTPDownload("localhost", "/WebTWAIN/Images/ImageData.jpg");
```

With this line of code, we can download ImageData.jpg from http://localhost/WebTWAIN/Images/ImageData.jpg and load it in Dynamic Web TWAIN.

This is especially useful when you want to load back an image which is created by Dynamic Web TWAIN. Even when the image data is stored in the database, you can write a page to download the image data and use HTTPDownloadEx() method to download the image data from that page.

Besides the HTTP download methods, you can also use the FTP download methods to download and load the image(s) in Dynamic Web TWAIN.

You can search keyword 'FTPDownload' in the help document for more details.

Web page authentication

Basic Authentication

You need to set **HTTPUserName** property and **HTTPPassword** property accordingly. The user name and password should be the same as the user ID and password in the authentication system.

Windows Authentication

Windows authentication is supported by default. You don't need to modify your code.

Forms Authentication

In Dynamic Web TWAIN, we can use **HTTPUploadThroughPostAs***()** methods to upload image(s) to the server. But if you use Forms-Based Authentication in Your ASP.NET Application, you might not be able to upload the image successfully because the action page verifies the login information of the HTTP post request before accepting the data and by default the **HTTPUploadThroughPostAs***()** methods don't pass the login information to the action page.

The simplest way is to get the cookie string from the server and pass it to the Dynamic Web TWAIN object. For example in your asp.net page:

```
DWObject.SetCookie("<%=Request.Headers["Cookie"] %>");
```

You can put this code anywhere before the HTTP Post method.

Alternatively, you can follow below steps to fix this issue:

1. Go to the login page, and make sure you have stored the login information in the HTTP cookie. At a later stage, we will pass the HTTP cookie to the action page for authentication.
   Example:

```
private void cmdLogin_ServerClick(object sender, System.EventArgs e)
{
   if (ValidateUser(txtUserName.Value,txtUserPass.Value) )
   {
      FormsAuthenticationTicket tkt;
      string cookiestr;
      HttpCookie ck;
   a  tkt = new FormsAuthenticationTicket(1, txtUserName.Value, DateTime.Now,
DateTime.Now.AddMinutes(30), chkPersistCookie.Checked, "your custom data");
   b  cookiestr = FormsAuthentication.Encrypt(tkt);
   c  ck = new HttpCookie(FormsAuthentication.FormsCookieName, cookiestr);
      if (chkPersistCookie.Checked)
      ck.Expires=tkt.Expiration;
                  ck.Path = FormsAuthentication.FormsCookiePath;
   d  Response.Cookies.Add(ck);

      string strRedirect;
      strRedirect = Request["ReturnUrl"];
      if (strRedirect==null)
            strRedirect = "default.aspx";
         Response.Redirect(strRedirect, true);
   }
   else
      Response.Redirect("logon.aspx", true);
}
```

   a. Create a **FormsAuthenticationTicket**.
   b. Get the encrypt cookie string.
   c. Create a HTTP cookie.
   d. Add the cookie to the response cookie, so JavaScript can access this cookie.

2. Go to the scan page, and add below code before you call **HTTPUploadThroughPost()** method.

```
DWObject.SetCookie(document.cookie);
```

Thus, the cookie will be attached to the HTTP post request and the action page will use the cookie to verify the login information.

Please NOTE that in the Web.Config file, the settings for Forms Authentication should follow the below code:

```
<authentication mode="Forms">
  <forms name="testform" loginUrl="login.aspx" domain="localhost" defaultUrl="default.aspx" path="/"
timeout="2">
  </forms>
```

```
</authentication>
```

* The "**Domain**" is necessary here.

Handling Events

Subscribe to an event

For ActiveX Edition, we suggest you use the attachEvent() function to subscribe to an event. Please refer to the sample code below:

```
DWObject.attachEvent('OnPostTransfer', DynamicWebTwain_OnPostTransfer);
```

This line of code attaches the **OnPostTransfer** event to JavaScript function: DynamicWebTwain_OnPostTransfer(). We should attach the event before any method is called. So put this code in the initialization function is a good choice.

For Plugin Edition and Mac Edition, we can attach the event in the object, just like the other standard HTML elements. Here is an example:

```
<embed id='mainDynamicWebTWAIN' type='Application/DynamicWebTwain-Plugin'
    OnPostTransfer='DynamicWebTwain_OnPostTransfer'
    pluginspage='DynamicWebTWAIN/DynamicWebTWAINPlugInTrial.msi'>
</embed>
```

In the above code, we attach the **OnPostTransfer** event to JavaScript function DynamicWebTwain_OnPostTransfer().

Event with parameter

Some of the events have parameter(s). For example, OnMouseClick Event:

OnMouseClick(Short sImageIndex)

So when you create the corresponding JavaScript function, you can include the parameter and retrieve the value from it:

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

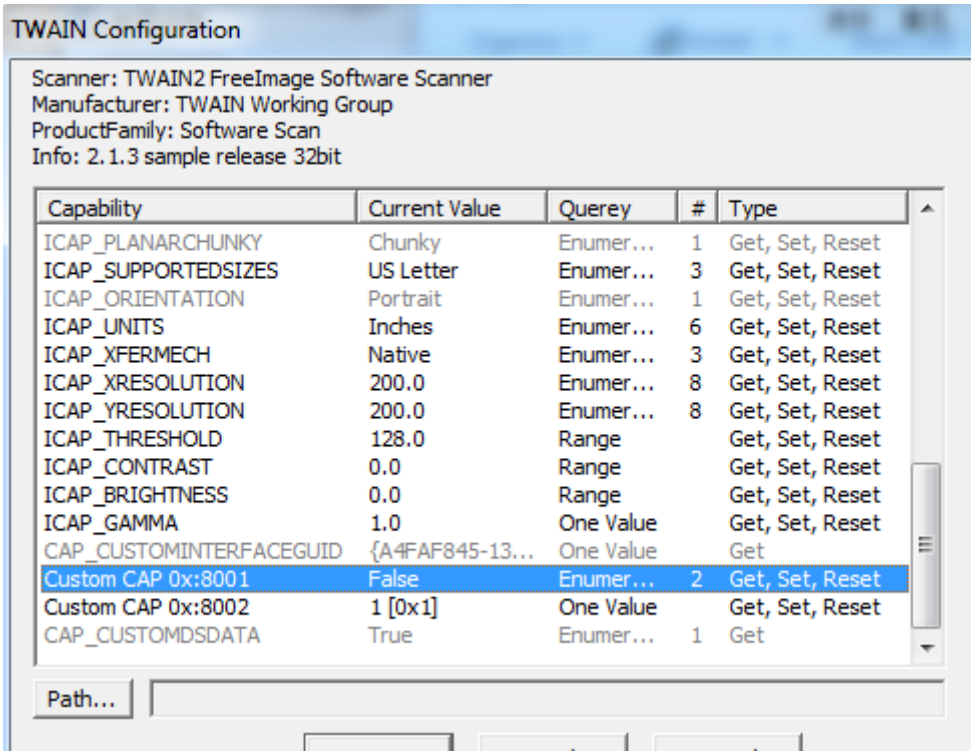
```
}
```

From this event, you will know which image is clicked.

Custom capabilities

To use a custom capability, you need to know what code stands for this capability. 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 then check what the hexadecimal value of the capability is.



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

```
DWObject.Capability = 0x8001;
DWObject.CapType = 5; //TWON_ONEVALUE
DWObject.CapValue = 1;
```

```
if (DWObject.CapSet())
    alert("Successful");
else
    alert("Source doesn't support this capability");
```

We have 4 types of capability. The one we used in the sample is called "**TWON_ONEVALUE**" which represents a single value being Boolean, string or number. Besides it, we have **TWON_ARRAY** which represents an array of associated individual values and **TWON_ENUMERATION** which represents a group of associated individual values. The values in the group are ordered from low to high and the step size between every two values is probably not the same. Lastly, we have **TWON_RANGE** which represents a range of individual values.

Please refer to the page below on How to Negotiate Capability with Different Capability Container Types:

http://www.dynamsoft.com/help/TWAIN/WebTwain/Programmer%20Guide/Capability%20Negotiation/Guide_cap%20negotiate%20with%20different%20cap%20container%20type.htm

Deploy Dynamic Web TWAIN on Web Server

ActiveX Edition

The followings are the steps needed to deploy Dynamic Web TWAIN on web server:

Embed the ActiveX Control

1) Copy DynamicWebTWAIN.cab to your web server. The CAB file can be found in the installation folder of Dynamic Web TWAIN.

   There are 32-bit and 64-bit CAB files:

   DynamicWebTWAIN.cab is for 32-bit TWAIN drivers and 32-bit IE;
   DynamicWebTWAINx64.cab is for 64-bit TWAIN drivers and 64-bit IE (Currently few device vendors provide 64-bit TWAIN drivers) .

2) Initialize the ActiveX control.

   Insert an <object> tag for Dynamic Web TWAIN in the scan page. Please pay more attention to the value of the **CodeBase** parameter if you are using a case-sensitive server such as Tomcat.

   The trial and full versions of Dynamic Web TWAIN use different classids.

   For TRIAL version of Dynamic Web TWAIN ActiveX:

```
<object classid="clsid:FFC6F181-A5CF-4ec4-A441-093D7134FBF2" id="DynamicWebTwain1" width="143"
height="156" CodeBase = "DynamicWebTWAIN.cab#version=9,0">
</object>
```

   For FULL version of Dynamic Web TWAIN ActiveX:

```
<object classid="clsid:E7DA7F8D-27AB-4EE9-8FC0-3FEC9ECFE758" id="DynamicWebTwain1" width="143"
height="156" CodeBase = "DynamicWebTWAIN.cab#version=9,0">
</object>
```

- **CodeBase**: The value of CodeBase is the relative path of the CAB file to the scan page. The above sample code indicates the CAB file is in the same folder with the scan page.

- If you are using a Friendly URL, please place the CAB file at the root directory on your web server and then modify the value of CodeBase to CodeBase = "/DynamicWebTWAIN.cab#version=9,0".

- 9.0 is the version number of Dynamic Web TWAIN ActiveX.

3) Activate Dynamic Web TWAIN with your license. For evaluation key, you can deploy Dynamic Web TWAIN on test web server for evaluation purpose only. For more info, please refer to Verify License section.

Plug-in Edition

You can use the HTML EMBED as the following sample to invoke Dynamic Web TWAIN Plug-in.

```
<embed    id="DynamicWebTWAIN"
          TYPE="Application/DynamicWebTwain-Plugin"
          PLUGINSPAGE="DynamicWebTWAINPlugIn.msi"
          height="528" width="100%">
</embed>
```

Notes:

1. The **PLUGINSPAGE** is the relative URL of DynamicWebTWAINPlugIn.msi for download. For Chrome, Opera, Safari and etc., you need to show a download link on your page for the users to download DynamicWebTWAINPlugIn.msi
2. You *must* include the TYPE attribute in an EMBED tag. If you do not, then there is no way of determining the media type, and no plug-in will load. Use **TYPE** to specify MIME type necessary to display the plug-in.
3. After DynamicWebTWAINPlugIn.msi is downloaded, you need to run and install the plug-in.

Mac Edition

The followings are the steps needed to deploy Dynamic Web TWAIN Mac Edition on web server:

1. Embed the plugin to your webpage

```
<embed id="DynamicWebTWAIN" type="application/dynamicwebtwain-plugin" height="528" width="100%">
</embed>
```

This is the object of Dynamic Web TWAIN Mac Edition. You can modify the ID, height and width if needed, but please *do not* change the type of the object.

2. Install Dynamic Web TWAIN Mac Edition. Download Page

Go to the installation folder of Dynamic Web TWAIN Mac Edition and you can find DynamicWebTWAINMacEdition.pkg.zip. Please unzip it and copy DynamicWebTWAINMacEdition.pkg to the folder where the web pages reside.

Deploy All Editions

To deploy all three editions to support all mainstream browsers on Windows and Mac, please follow the below steps:

1) **License verification.**

   Please generate a product key with Licensing Tool for web server. And assign the product key to the property ProductKey before you use any of the other methods, properties of Dynamic Web TWAIN.

2) **Load the corresponding ActiveX control/plugin according to the browser and OS type.**

   You can try out Dynamic Web TWAIN online demo to see how it works. Build the "Hello World" page section will explain more details for this part.

3) **Deploy Resource folder to your web server.** The folder can be found in the installation folder of Dynamic Web TWAIN.

   ActiveX Edition – DynamicWebTWAIN.cab and DynamicWebTWAINx64.cab
   Plug-in Edition – DynamicWebTWAINPlugIn.msi (recommended) and DynamicWebTwain.xpi (optional, Firefox only)
   Mac Edition – DynamicWebTWAINMacEditionTrial.pkg

Troubleshooting

Install Virtual Scanner for testing

If you don't have an actual scanner available for testing, you can download and install a virtual scanner, which is developed by 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 compatible?

If you are not sure whether your scanner comes with a TWAIN driver, you can use TWACKER, which is a tool developed by 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 don't 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 failed, you may search online or contact the device vendor for a TWAIN driver.

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

Symptom:
Sometimes, you will encounter an issue where your scanner is not shown, not responding or shown as busy in the browser.

Causes:

1. The scanner is not connected to the machine or the scanner driver is not installed.

 * Especially when you are using **64-bit Internet Explorer** but you don't have 64-bit scanner (TWAIN) driver installed on the machine.

2. The Browser doesn't have enough permission to access the scanner driver.

 *This mostly happens on **Windows 7/Vista/2008 OS** where high security level is applied to Internet Explorer. It would also happen in Firefox 4+ and Safari.

Resolution:

1. Make sure the driver of the scanner is installed and the scanner is running and properly connected to the machine. You should be able to see the scanner in Windows' Scanners and Cameras Wizard. And you can use Twacker to check if the driver is TWAIN compatible.

In addition, if you are using 64-bit Internet Explorer, please make sure you have a 64-bit TWAIN driver of the scanner installed.

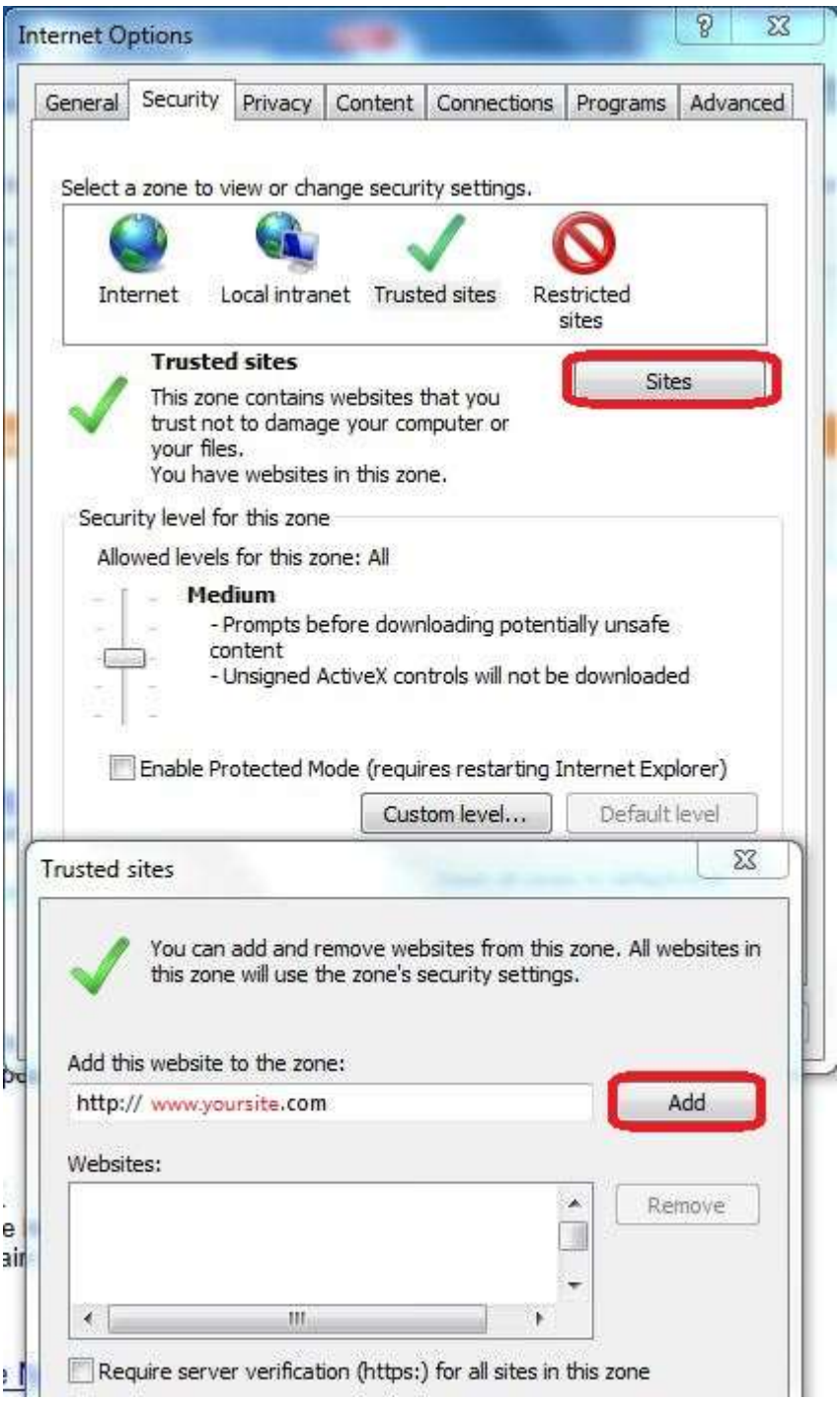
2.

For Firefox, Safari, etc., please go to C:\Windows\twain_32 and you should be able to find that your scanner driver has a sub-folder there. In order to show the scanner as an available source, you need to copy the dlls in the sub-folder and paste them to the folder C:\Windows\System32 (or C:\Windows\SysWOW64 on a 64-bit OS).

For Internet Explorer, you can try one of the following tricks:

#1:

Add the website to the zone of trusted sites. **IE | Tools | Internet Options | Security | Trusted Sites.**



#2

Turn **Protected Mode** off.

To turn off the protected mode of IE, you can go to **Tools | Internet Options | Security** and uncheck **"Enable Protected Mode (requires restarting Internet Explorer)"** option.

FYI, Protected Mode helps protect users from attack by running an IE process with greatly restricted

privileges. It uses the Windows Vista integrity mechanism to run the Internet Explorer process at low integrity. But with Protected Mode turned on, you may encounter problems when running Dynamic Web TWAIN applications.

#3
To temporarily elevate permissions of IE: click Start, point to All Programs, right-click IE, and then select Run as Administrator (with Administrator permissions).

If you need any further support, please email twainsupport@dynamsoft.com.

Red X on the image viewer

Symptom:
If you access a web page with Dynamic Web TWAIN in IE and see a Red "X", it means the ActiveX control is not downloaded and installed on your machine successfully.

Resolution:

First verify if it's a server-side or client-side problem.

You can test with the online demo of Dynamic Web TWAIN and see if you can download the ActiveX successfully.

If yes, there might be some problem with the ActiveX deployment on the server side. Make sure you follow the correct steps for deploying Dynamic Web TWAIN ActiveX.

If it failed with the online demo too, please check the following things:

- Check the **security settings** of IE (Tools->Internet Options->Security tab) on the client machine and verify the following security settings of IE to "Prompt" or "Enabled":

  1) Download signed ActiveX controls
  2) Run ActiveX Controls and plug-ins
  3) Script ActiveX controls marked safe for scripting

- Try adding the site as Trusted Site (Tools->Internet Options->Security tab) in IE and test again.

- Check if the ActiveX control was already installed on the machine. Go to IE menu **Tools**->**ManageAdd-ons**, choose **All add-ons in the Show drop-down list and see if there is any DynamicWebTwain class.**

If the control is already there, please clear them and try accessing the scan page again.

- The **DynamicWebTwain.cab** file on your server may be **invalid**.

You can copy the cab from the installation folder of Dynamic Web TWAIN and update it on server to test again.

If you are not sure if your CAB file is invalid, you can send it to **support@dynamsoft.com** and we will check it for you.

Useful Resources

Online Demo Site

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

Knowledage 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

All APIs (Property, Method, Capability and Event)

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

Contact Us

Email: twainsupport@dynamsoft.com
Online Chat: http://www.dynamsoft.com/Support/LiveHelp.aspx
Telephone: 1-877-605-5491 (Toll Free); 1-604-605-5491
FAX: 1-866-410-8856

Outsource development to Dynamsoft

Description

In relative terms, TWAIN is a special area which needs special expertise and knowledge. To truly relieve you of any efforts in TWAIN related learning, designing, coding and debugging, we proudly announce that we offer **Virtual Developer Service**.

With this service, our developers participate in your web scanning module as an active team member from requirement analysis, design, coding, testing and deployment to maintenance.

Virtual Developer Service for Dynamic Web TWAIN

- Top priority of request response.
- We take care of all the TWAIN related development for your project, including requirement analysis, design, coding, testing, deployment and maintenance during the contact period.
- 100% Satisfaction guaranteed. One-month unconditional money back guarantee.
- The available techniques for desktop application are: VB, VC, VB.NET, C#, Delphi, C++ Builder, Access, SQL Server. For web application, the available techniques are: ASP/ASP.NET + SQL Server, PHP + MySQL, Java, VBScript, JavaScript. Please contact us at twainsupport@dynamsoft.com if your desired techniques are not listed here.

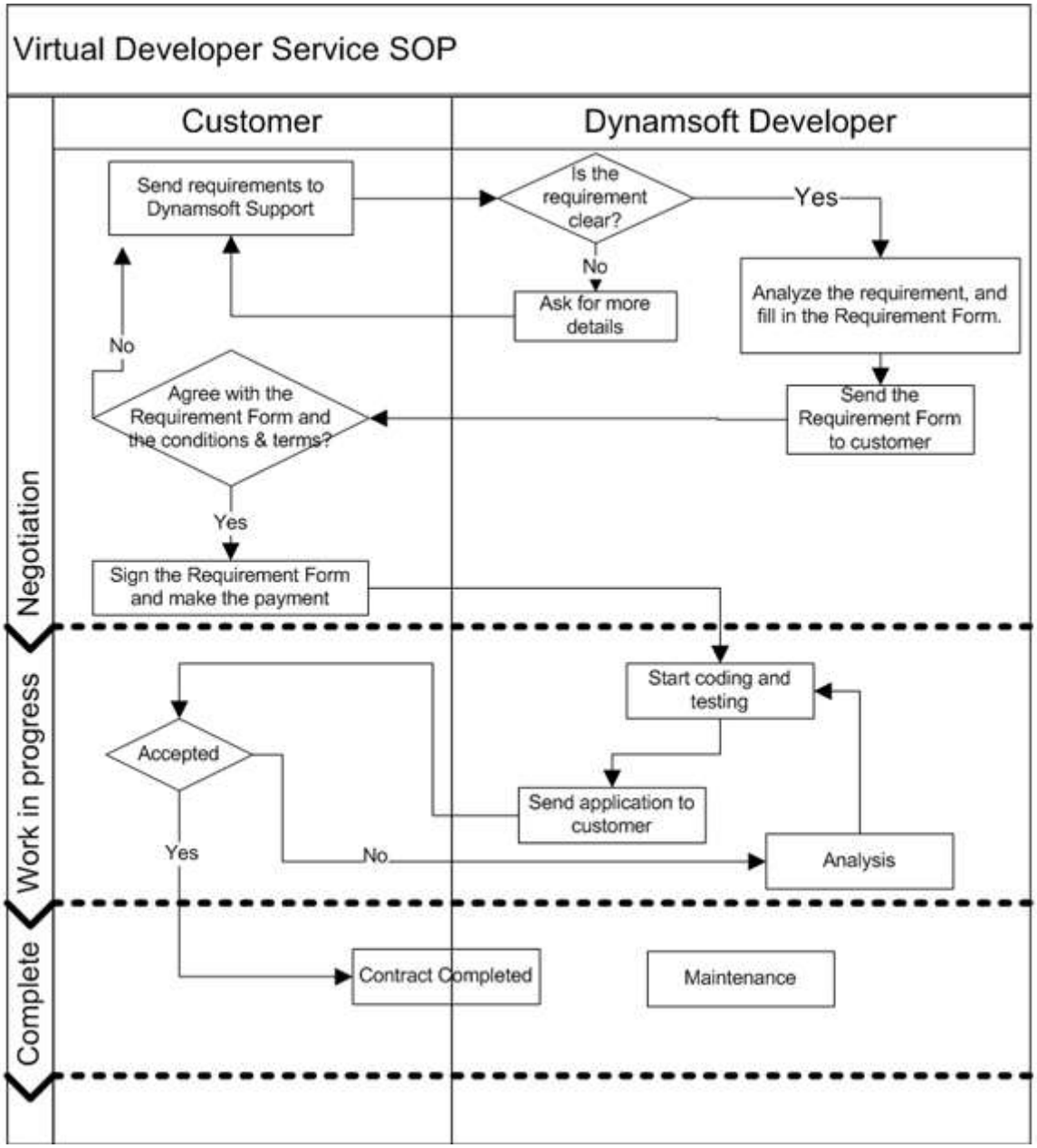
Requirement Form

Virtual Developer Service Requirements (Form)
Virtual Developer Service Requirements (Sample)

Standard Operating Procedure (SOP)

1. Complete the Requirements Form and send it to the support team at twainsupport@dynamsoft.com.
2. Our developers will analyze your requirements, and send the form with comments along with a quotation for the Virtual Developer Service back to you.
3. If you agree with the requirement analysis, please sign the form and send us the License you own and we will start coding and testing.
   Purchase Dynamsoft Products
4. Check and accept the application.

5. During the contact period, you can bring up any new requirements and we will modify the application with the steps above.



Conditions and Terms

The following conditions and terms apply to our Virtual Developer Service. By using the service, you agree to be bound by them.

- Although we try our best to provide accurate service, we are not responsible for errors that may occur. Under no circumstances, Dynamsoft should be responsible for any damages caused by our service. You agree to use our service at your own risk.

- With the Virtual Developer Service, all the requirements must be directly TWAIN related and can be implemented with Dynamic Web TWAIN.
- Only code is provided. No documentation will be provided.
- Dynamsoft agrees to keep all the communication between Dynamsoft and you confidential.
- You agree to keep all the communication between Dynamsoft and you confidential.
- The contract begin date should be the date you sign the requirement analysis form.
- If integration to your existing system is required, you need to provide the access of the related source code, database and etc...

Dynamsoft reserves the right to not carry out the Virtual Developer Service at Dynamsoft's discretion. If we decide not to carry out the Virtual Developer Service, we will refund you the full service fee.