Table of Contents

Developers Guide

API Documentation

Knowledge Base

Dynamic Web TWAIN

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 or new APIs that you can use to polish your scanning page.

Get Started

What is TWAIN | ICA | SANE

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

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

Dynamsoft Corporation is a member of the TWAIN Working Group.

The TWAIN protocol works very well on Windows but not as good on macOS. Thus, a substitute is also used by Dynamsoft's SDK on macOS which is called Image Capture Architecture or ICA for short.

On Linux, TWAIN isn't available; therefore SANE is used. As described on the official introduction page, SANE stands for "Scanner Access Now Easy" and is an application programming interface (API) that provides standardized access to any raster image scanner hardware such as flatbed scanners, hand-held scanners, video and still cameras, frame-grabbers.

What is Dynamic Web TWAIN

Dynamic Web TWAIN is a scanning SDK specifically optimized for **web applications**. It was initially designed for Windows at the beginning, and only TWAIN protocol was supported at the time which is why **TWAIN** is in the name of the SDK. However, it has been dramatically improved and extended over the years, and now it supports TWAIN on Windows & macOS, ICA on macOS and SANE on Linux. The SDK enables you to write code, in just a few lines, to **scan** documents from a `TWAIN|ICA|SANE` compliant device which typically is a scanner. 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 files in the formats `BMP|JPG|PNG|TIF|PDF` from a local disk or the web via HTTP(s) or FTP.

Basic Requirements

Server Side

- Operating System: Windows, macOS, Linux, etc.
- Web Server: IIS, Apache, Tomcat, NGINX, WebSphere, ColdFusion, etc.
- Programming Languages:
  - Front-end: HTML, JavaScript, TypeScript, CSS, etc.
  - Back-end: ASP.NET (C# and VB), PHP, JSP (JAVA), ASP, Python, NodeJS, etc.

Client Side

- Browser/OS Support
  - Windows XP/7/8/2008/2012/2016 and 10; 32-bit and 64-bit
    - IE 6-9: ActiveX
    - IE 10-11: HTML5/ActiveX
    - Edge: HTML5
    - Chrome/Firefox 27+: HTML5
  - Mac OS X 10.6.8 and later
    - Chrome/Firefox 27+, Safari 7+: HTML5
  - Ubuntu 12.0.4+, Debian 8+, Fedora 24+, mint 18.3; 64-bit
    - Chrome/Firefox 27+: HTML5
  - IOS
    - Safari v11+: MBC
  - Android
    - Chrome v58+: MBC

Choose which Dynamic Web TWAIN Edition to use

Dynamic Web TWAIN has five editions: ActiveX, HTML5 for Windows, HTML5 for Mac, HTML5 for Linux and Mobile Browser Capture for IOS & Android. Based on the browser(s) your end users use, you can decide which edition(s) you need.

- `ActiveX` : supports IE 6-9 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
- `HTML5 for Mac` : supports Chrome/Firefox 27+, Safari 7+

- `HTML5 for Linux` : supports Chrome/Firefox 27+
- `MBC for IOS & Android` : supports Safari v11+ on IOS and Chrome v58+ on Android

Build the "Hello World" Scan Page

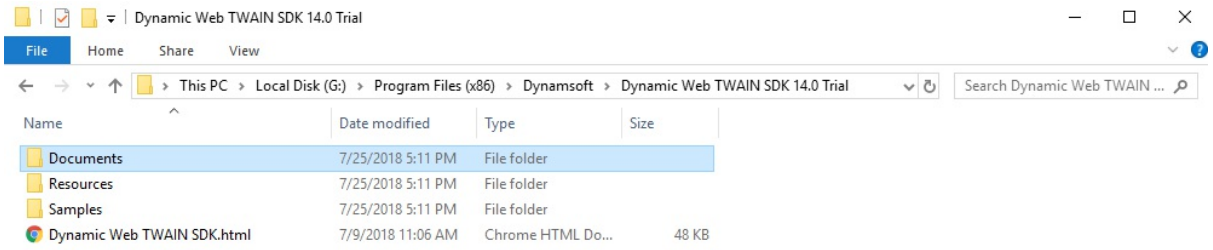
> *NOTE: Before you start, please make sure you've downloaded and installed the latest version of Dynamic Web TWAIN. If you haven't done so, you can get the 30-day free trial here.*

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

Step 1: Start a Web Application

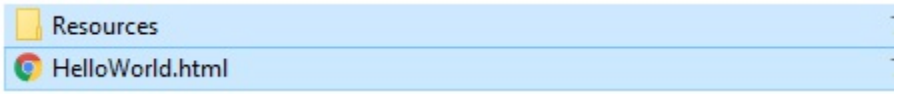
Copy the Dynamsoft Resources folder to your project

You can typically find the Resources folder in `C:\Program Files (x86)\Dynamsoft\Dynamic Web TWAIN SDK {Version Number}{Trial}\`



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

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

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

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

Add a Scan button and the minimum code to scan

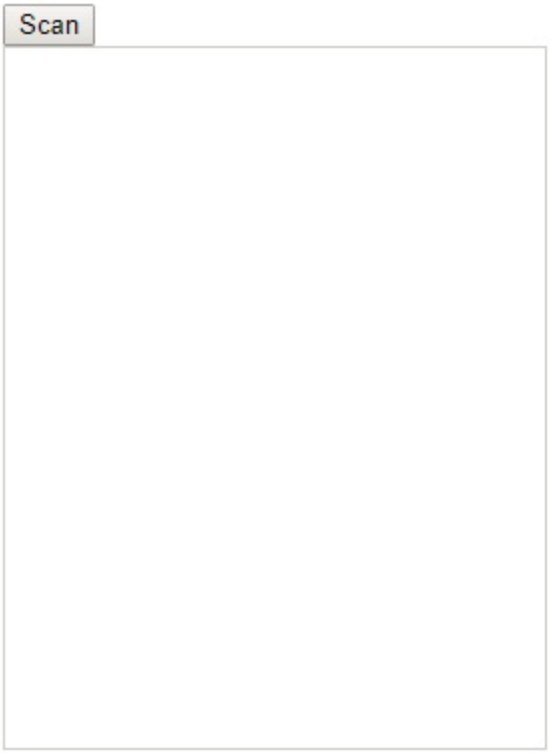
```
<input type="button" value="Scan" onclick="AcquireImage();" />
<script type="text/javascript">
    var DWObject;
    function Dynamsoft_OnReady() {
        DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
    }
    function AcquireImage() {
        if (DWObject) {
           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

Review the completed code

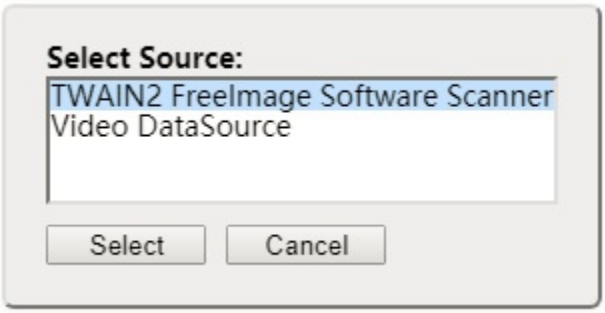
```
<html>
<head>
    <title>Hello World</title>
    <script src="Resources/dynamsoft.webtwain.initiate.js"> </script>
    <script src="Resources/dynamsoft.webtwain.config.js"> </script>
</head>
<body>
    <input type="button" value="Scan" onclick="AcquireImage();" />
    <div id="dwtcontrolContainer">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

See the scan page in action

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

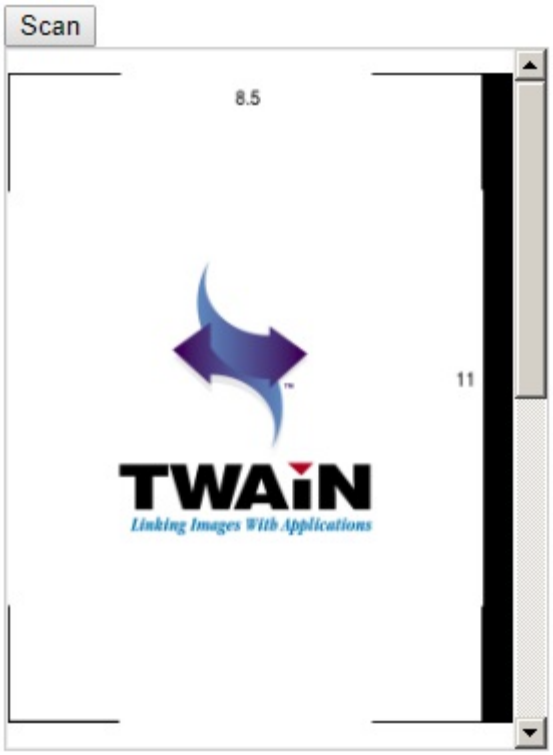


Now, you can click on the Scan button to select a device, as shown below:

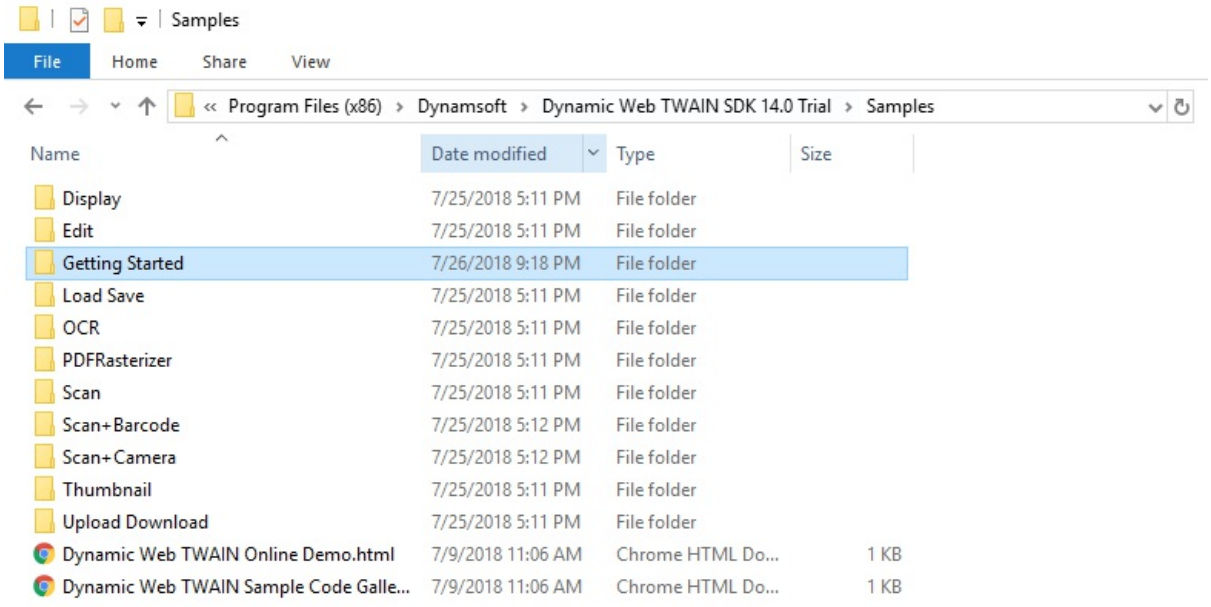


> NOTE:
>
> - Only `TWAIN|ICA|SANE`-compliant devices are listed in the Select Source dialog. If your connected scanner doesn't show up in the list, please make sure the proper driver is installed.
> - If you are using Windows and don’t have a real scanner at hand, you can install the Virtual Scanner – a scanner simulator which is developed by the TWAIN Working Group for testing purposes.

Once the 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 lots of samples there (source code provided) for you to try out the many features of Dynamic Web TWAIN. You can also find advanced samples on the online sample gallery.

Customize the Dynamic Web TWAIN Object

Change the name of the object

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

```
function Dynamsoft_OnReady() {
    DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
}
```

The div with id `dwtcontrolContainer` is the placeholder for Dynamic Web TWAIN. Its initial 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}];
```

Change the Size of the Viewer

You can change the initial 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: '513px' }];
```

> NOTE
>
> You can also change the size of the container at runtime using the properties Width and Height.

Use Dynamic Web TWAIN

By default, Dynamic Web TWAIN automatically initializes after the page finishes loading. Once the Dynamic Web TWAIN object finishes initializing, you can start to call its methods, set its properties, etc. You can refer to our API Documentation to check all properties, methods and events of Dynamic Web TWAIN.

Properties

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

```
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 like this:

```
DWObject.Rotate(0, 45, false); // Rotate the 1st image in the buffer by 45 degrees
```

Events

Events are triggered when the program reaches certain trigger points. For example, `OnMouseClick` is triggered when you click the mouse, `OnPostTransfer` is triggered when one image is transferred, 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:

```
Dynamsoft.WebTwainEnv.RegisterEvent('OnWebTwainReady', Dynamsoft_OnReady);
var DWObject;
/* OnWebTwainReady event fires as soon as Dynamic Web TWAIN is initialized. It is the best place to add event listene
rs */
function Dynamsoft_OnReady() {
    DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
    DWObject.RegisterEvent("OnPostTransfer", Dynamsoft_OnPostTransfer);
}
function Dynamsoft_OnPostTransfer() {
    /* This event handler will be called after a transfer ends. */
    /* Your code goes here*/
}
```

In the code above, we added the JavaScript function `Dynamsoft_OnPostTransfer()` as an event listener for the event `OnPostTransfer`. Alternatively, you can also write code as shown below:

```
Dynamsoft.WebTwainEnv.RegisterEvent('OnWebTwainReady', Dynamsoft_OnReady);
var DWObject;
function Dynamsoft_OnReady() {
    DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
    DWObject.RegisterEvent("OnPostTransfer", function () {
        /* Your code goes here*/
    };
}
```

Event with argument(s)

Some of the events have argument(s). Take the `OnMouseClick` event for an example:

```
OnMouseClick(Number nImageIndex) /* nImageIndex refers to the image you clicked on*/
```

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

```
function DynamicWebTwain_OnMouseClick(index) {
    console.log(index);
}
```

or

```
DWObject.RegisterEvent("OnMouseClick", function (index) {
    console.log(index);
};
```

Special Event - `OnWebTwainReady`

To check all the events, please refer to the API Documentation. Of all these events, there is one called `OnWebTwainReady` that is special. This event fires as soon as the Dynamic Web TWAIN object finishes initializing. As you may have seen earlier in the document, the recommended way to use it is:

```
Dynamsoft.WebTwainEnv.RegisterEvent('OnWebTwainReady', Dynamsoft_OnReady);
var DWObject;
function Dynamsoft_OnReady() {
    DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
}
```

or

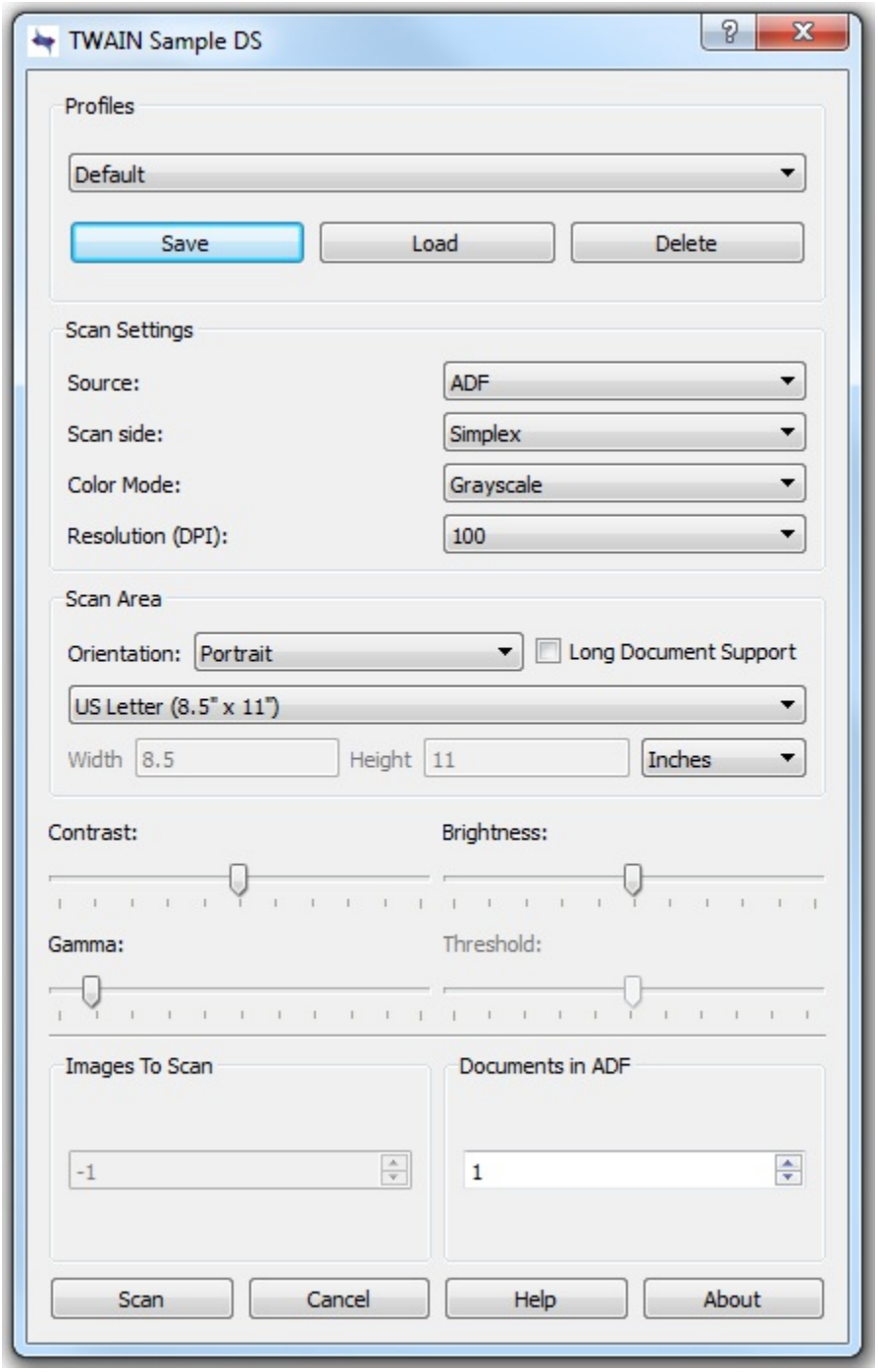
```
Dynamsoft.WebTwainEnv.RegisterEvent('OnWebTwainReady', function () {
    DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
};
```

Explore the Features

- Customize your scan settings
- Manipulate the images
- Use the ImageEditor
- Create a thumbnails view
- Scan large amounts of documents
- Load local images
- Save images locally
- Upload images to the web server
- Download images from the web
- Use Capability Negotiation
- Add/Remove the object individually

Customize scan settings

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



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

```
DWObject.SelectSource();
DWObject.OpenSource();// You should customize the settings after opening a source
DWObject.IfShowUI = false;// Hide the User Interface of the scanner
```

```
DWObject.IfFeederEnabled = true;// Use the document feeder to scan in batches
DWObject.IfDuplexEnabled = false;// Scan in Simplex mode (only 1 side of the page)
DWObject.PixelType = EnumDWT_PixelType.TWPT_GRAY; // Scan pages in GRAY
DWObject.Resolution = 200; // Scan pages in 200 DPI
DWObject.AcquireImage();// Start scanning
```

Manipulate the image(s)

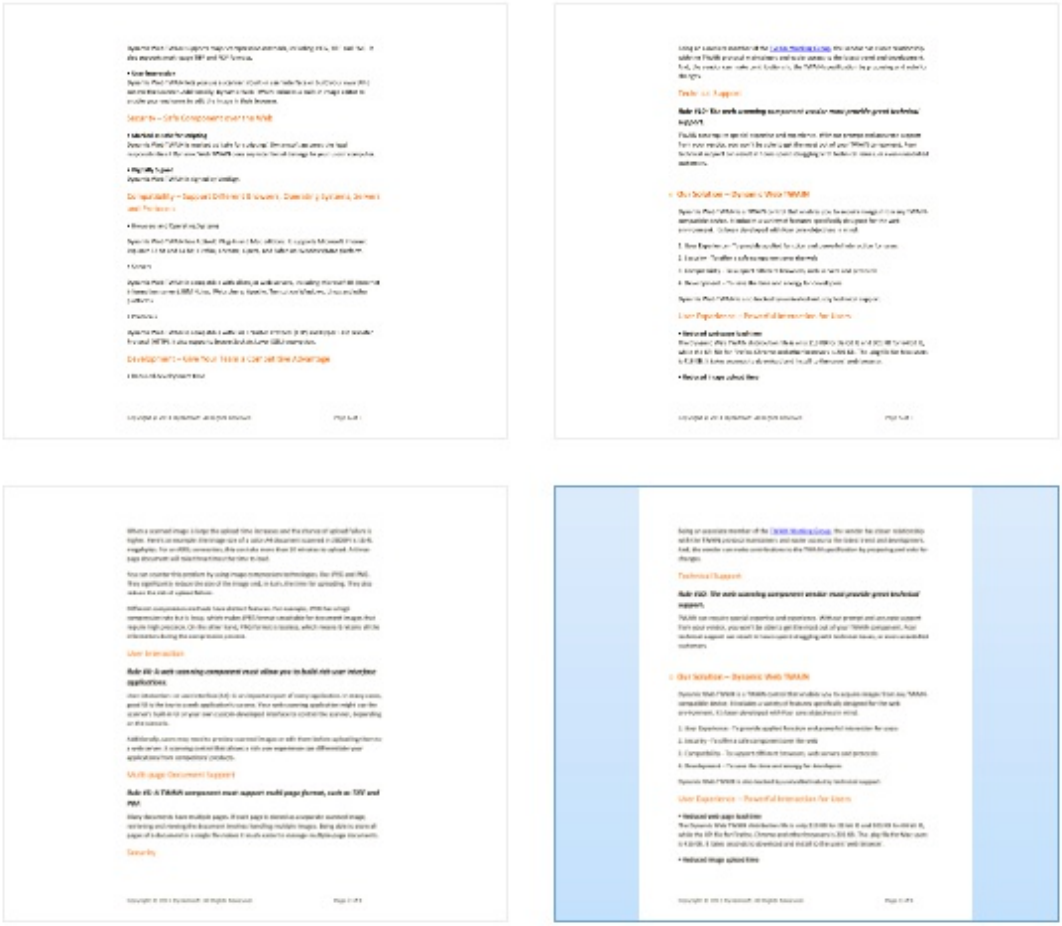
When you have scanned or loaded images in Dynamic Web TWAIN, you can start manipulating the images. You can:

1. Go through each image by changing the property `CurrentImageIndexInBuffer`

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

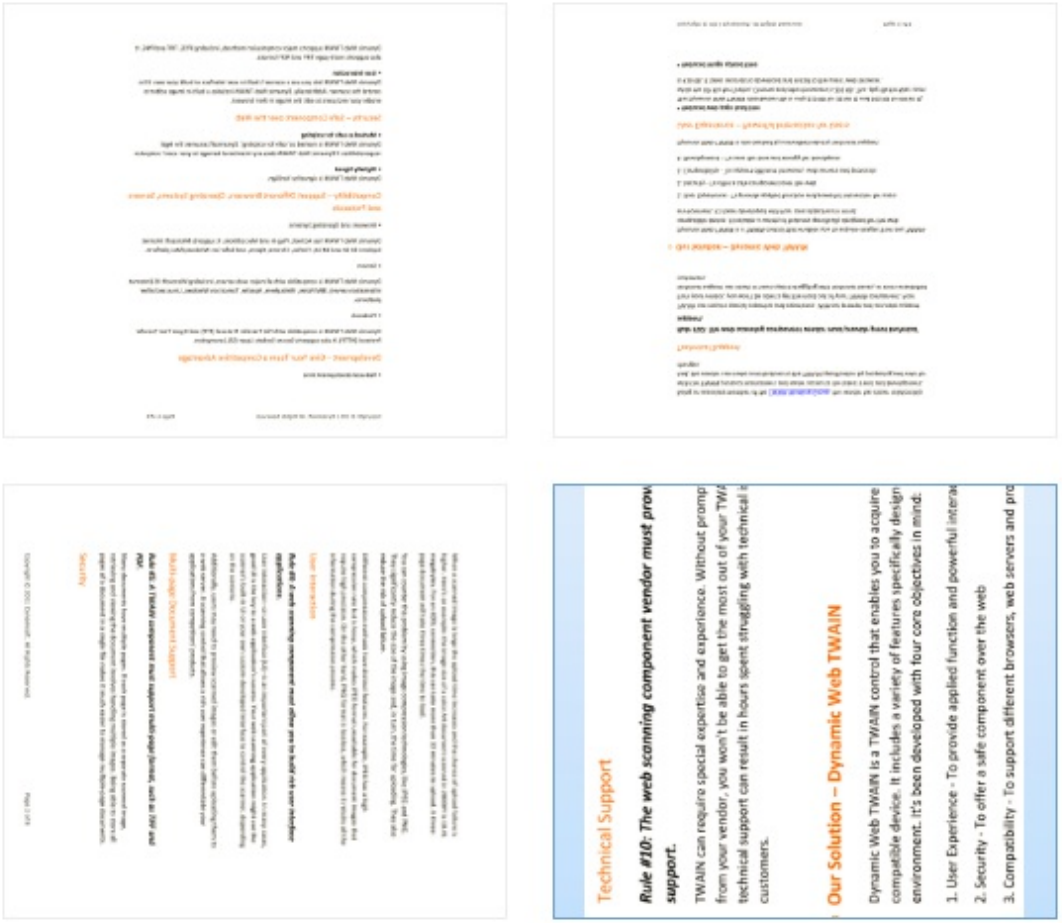
2. Show multiple images by changing the view mode (other than 1 by 1 or -1 by -1) using `SetViewMode()`

```
/* Show 4 images in the viewer 2 by 2*/
DWObject.SetViewMode(2, 2);
```



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 an image by its index or remove selected or all images at once. The methods are `RemoveImage()`, `RemoveAllSelectedImages()`, `RemoveAllImages()`.

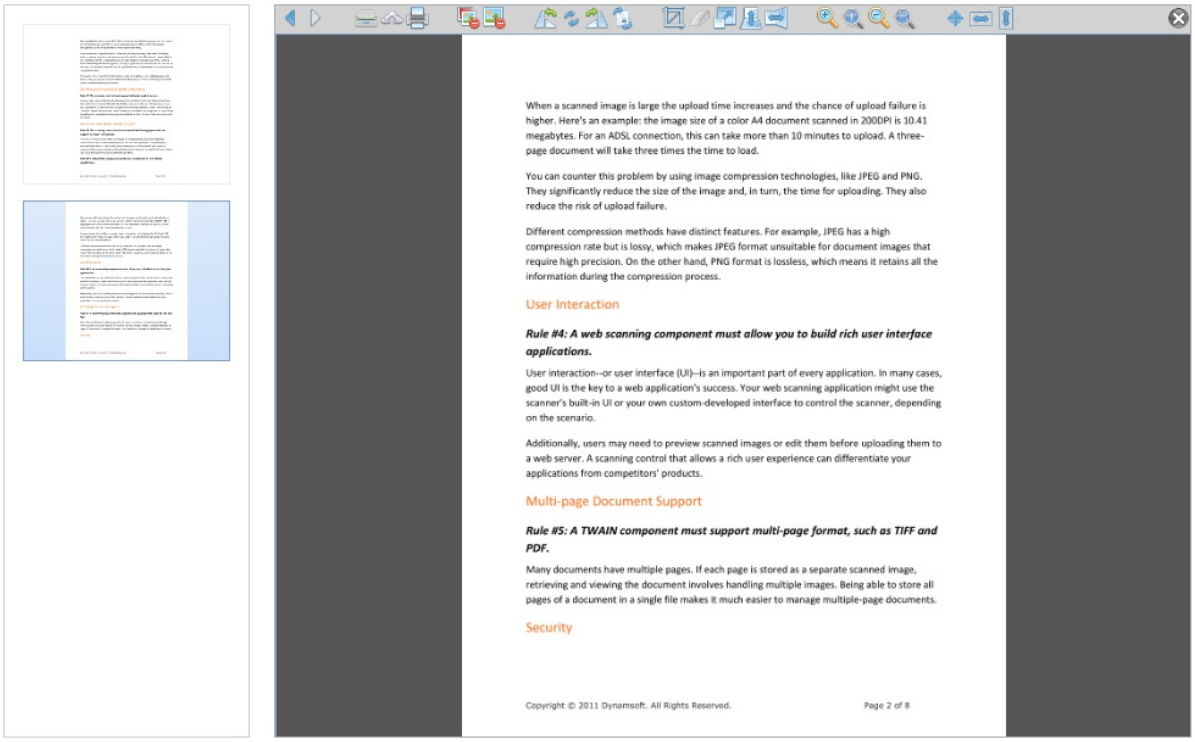
Create A Thumbnails View

In the HTML5 edition of version 14.0+, you can use the built-in editor of Dynamic Web TWAIN together with the main image viewer to create a thumbnails view. The following code shows how to do it.

> Note:
>
> For the ActiveX, you will still need to use two controls to simulate the thumbnails.

```
<!DOCTYPE html>
<html>
<head>
    <title>Thumbnails View</title>
    <script src="Resources/dynamsoft.webtwain.config.js"></script>
    <script src="Resources/dynamsoft.webtwain.initiate.js"></script>
</head>
<body>
    <div id="dwtcontrolContainer" style="float: left; margin-right:20px;"></div>
    <div id="dwtcontrolContainerLargeViewer" style="float: left;"></div>
    <script type="text/javascript">
        Dynamsoft.WebTwainEnv.RegisterEvent('OnWebTwainReady', Dynamsoft_OnReady);
        var DWObject;
        function Dynamsoft_OnReady() {
            DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
            if (DWObject) {
                DWObject.Width = 200;
                DWObject.Height = 600;
                DWObject.ShowImageEditor("dwtcontrolContainerLargeViewer", 750, 600);
                DWObject.SetViewMode(1, 4);
            }
        }
    </script>
</body>
</html>
```

The following shows what it looks like



As shown above, the built-in editor comes with many features itself. By clicking the buttons, you can scan, load, remove images and then edit them or zoom in/out to view them better.

> Note:
>
> If the DIV holding the editor isn't wide enough, the buttons are hidden group by group automatically.

All buttons are displayed by default. If you like, you can choose to hide/show one or more buttons in the file `dynamsoft.webtwain.config.js` .

In the dynamsoft.webtwain.config.js

The following are the default configurations, and you can edit the visibility to show/hide some buttons.

```
bShowAllButtons: true,
visibility: {
//only valid when bShowAllButtons is true, otherwise changing visibility does nothing
    'scan': true, 'load': true, 'print': true,
    'removeall': true, 'removeselected': true,
    'rotateleft': true, 'rotate': true, 'rotateright': true, 'deskew': true,
    'crop': true, 'erase': true, 'changeimagesize': true, 'flip': true, 'mirror': true,
    'zoomin': true, 'originalsize': true, 'zoomout': true, 'stretch': true,
    'fit': true, 'fitw': true, 'fith': true,
    'hand': true, 'rectselect': true, 'zoom': true
}
```

Scan Lots of Documents at a Time - Disk Caching

Sometimes, you may need to scan hundreds or even thousands of documents at a time. In this case, the disk caching feature may come in handy. The related properties are `IfAllowLocalCache` and `BufferMemoryLimit`.

Although Dynamic Web TWAIN can run both in 32bit and 64bit, it's 32bit by default which means it can utilize no more than 2 GB of physical memory. However, the data the SDK deals with are images which take up much space. For example, one A4 paper scanned in 300 DPI takes around 24MB in memory (DIB) and even if you can use 2GB to store images, you can store no more than 85 of them. Therefore Dynamsoft added the disk-caching feature to the SDK which, when enabled, caches most images temporarily on the disk while keeping a few active ones in the memory to maintain high performance.

The disk caching feature is enabled by default and can be disabled by setting `IfAllowLocalCache` to `false`.

We can also set how much memory we want the SDK to use before images start to be cached. By default, `800MB` is used. You can change it using the property `BufferMemoryLimit`.

> NOTE:
>
> - All cached data is encrypted and can only be accessed by Dynamic Web TWAIN
>   - For ActiveX Edition: the cached data is stored in `C:\Users\{User Name}\AppData\LocalLow\Dynamsoft\cache`
>   - For HTML5 Edition: it is stored in `C:\Windows\SysWOW64 {or system32}\Dynamsoft\DynamsoftService\cache`
> - When the SDK is unloaded (like when the browser tab is closed), the cached data is destroyed and removed from the disk automatically.
> - Although you can scan and load as many images as you like, you need to handle them in a smaller volume instead of processing them all at once. For example, you should not upload too many images as one file because it may exceed the memory limit.

Load local image(s) into Dynamic Web TWAIN

> **NOTE**
>
> Before you try to load any images, 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 load successfully. If you are trying to load an image that was not generated by Dynamic Web TWAIN, it may or may not work.

Methods

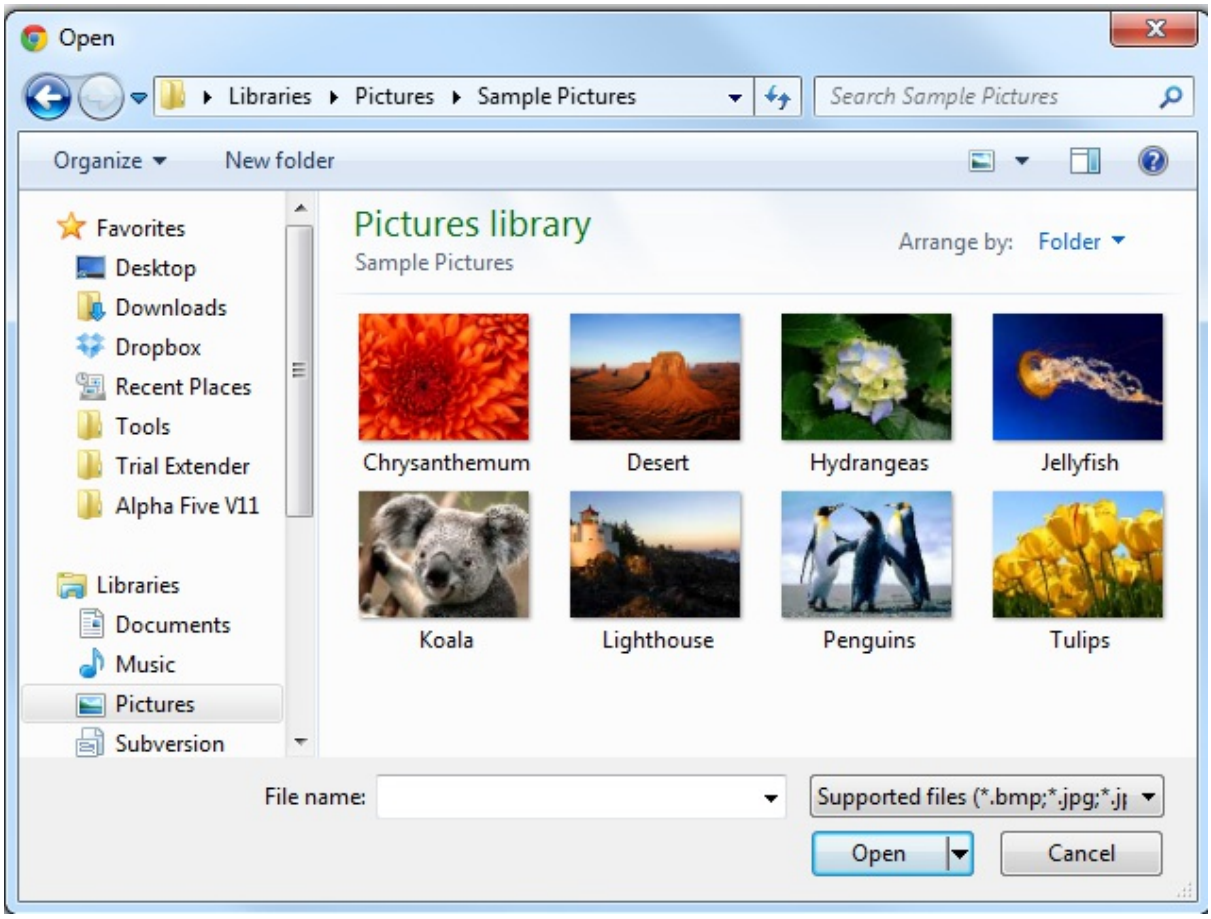
With Dynamic Web TWAIN, you can load local images with the methods `LoadImage()` or `LoadImageEx()`. Below is a simple code snippet:

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

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

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

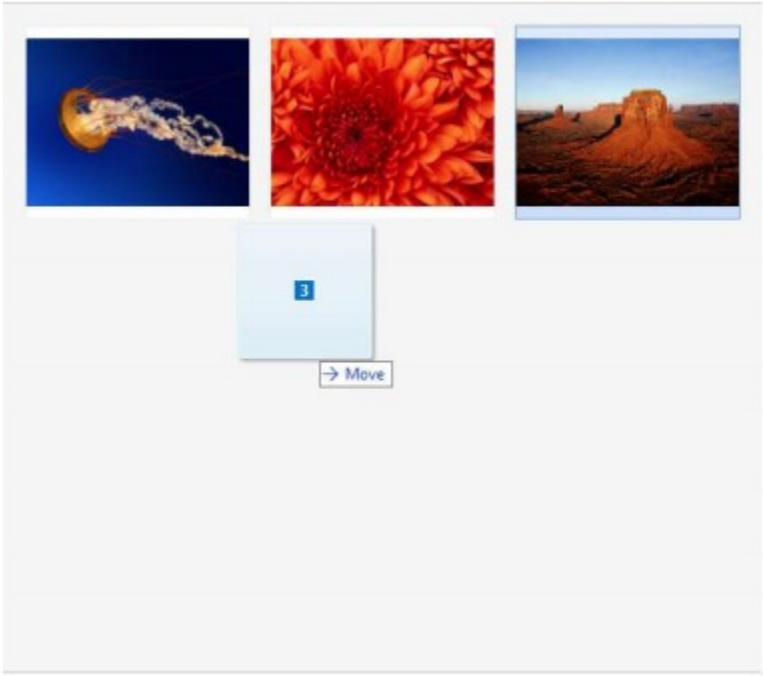
As you can see, you need to provide the complete file path to load an image. It is somewhat clumsy especially when you need to load more than one image. But no worries, Dynamic Web TWAIN can open a "Select File…" dialog for you to locate the image(s) you want to load. Moreover, like other properties and methods, it's effortless to use. Below is a code snippet:

```
DWObject.IfShowFileDialog = true;
DWObject.LoadImageEx("", EnumDWT_ImageType.IT_ALL); //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.



Starting from v14.0, you can also drag and drop images onto the Dynamic Web TWAIN viewer to load them.



Save image(s) locally

> **NOTE**
>
> 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).

Methods

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

Format	Method
Single-Page	`SaveAsBMP()` `SaveAsJPEG()` `SaveAsPDF()` `SaveAsPNG()` `SaveAsTIFF()`
Multi-Page PDF	`SaveSelectedImagesAsMultiPagePDF()` `SaveAllAsPDF()`
Multi-Page TIFF	`SaveAllAsMultiPageTIFF()` `SaveSelectedImagesAsMultiPageTIFF()`

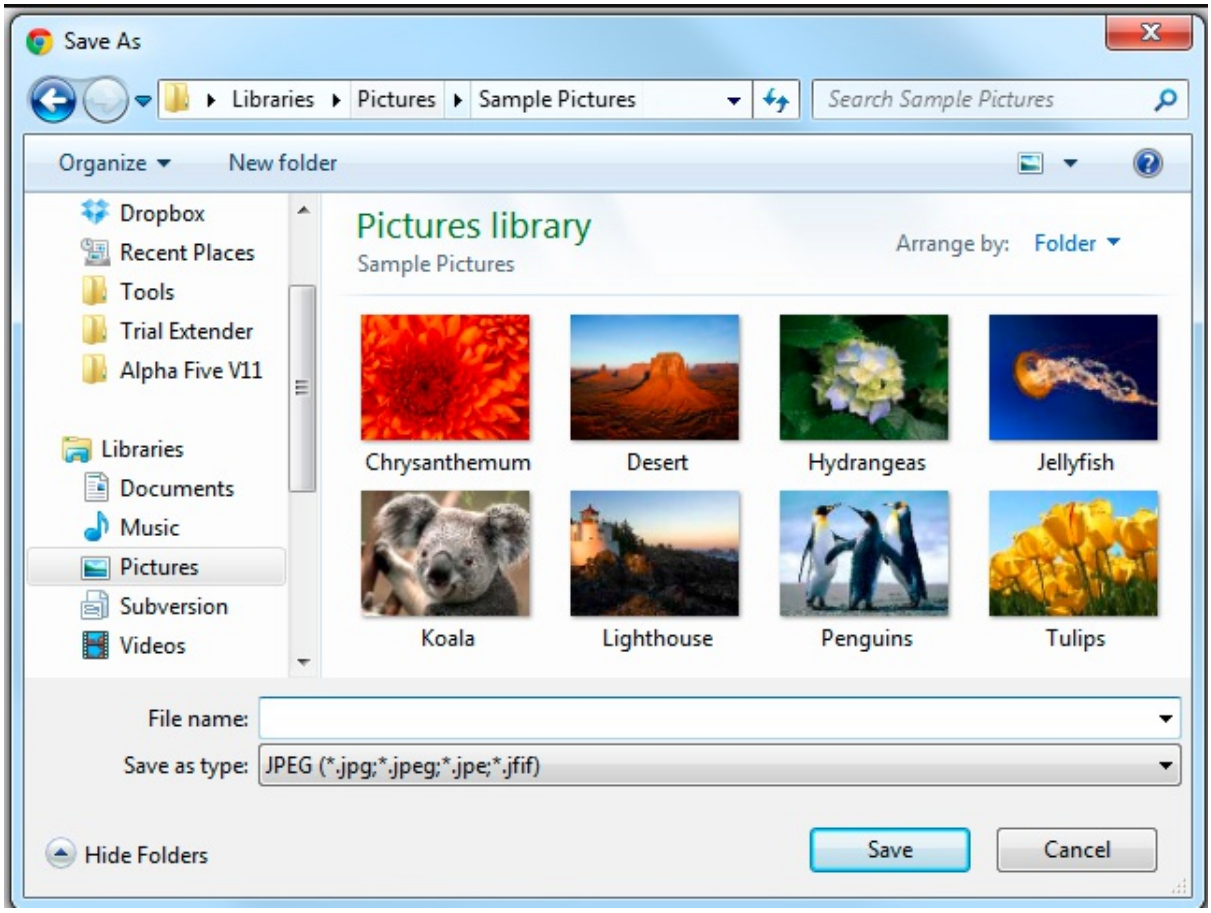
Code snippet:

```
//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:

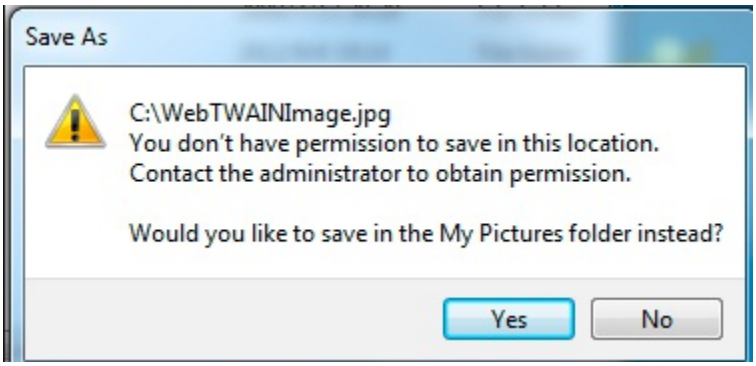
```
DWObject.IfShowFileDialog = true;
DWObject.SaveAsJPEG("",0);
```

It brings up this dialog box with the "Save as type" specified by the method you use:



NOTE

On Windows 7 and above, Microsoft has strengthened security which means you can only save images to certain places where you have the 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.



Upload image(s) to the web server

> **NOTE**
>
> 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. The action page refers to the target script that receives the HTTP Post request containing the image data and handles all the server-side operation like saving the data on the hard disk or database, etc. Here is an example:

Upload and Save on the Server Disk

```
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";
```

In the code snippet

`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`. If you are not sure about the port number, you can 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 an 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.

Methods

Now, we can call one of the HTTP upload methods to upload the image(s). We have 8 methods:

Format	Method
Any Type	`HTTPUploadThroughPostDirectly()`
Supported Images	`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. You can 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 (such as `C#`, `VB`, `PHP`, `Java`).

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) ;
```

Upload to FTP

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

Format	Method
Any Type	`FTPUploadDirectly()`
Supported Images	`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
);
```

Upload image(s) to a Database

Dynamic Web TWAIN doesn't save/upload the image(s) to your database directly. Instead, the image data hits the action page first, and then the code in the action page decides where to store it.

If you are not sure how to upload the image data to the server, please refer to the previous section Upload and Save on the Server Disk.

Different database systems may have different data types for image data. We generally use `BLOB` or `varbinary` in `MSSQL` Server, `Long raw` or `BLOB` in `Oracle`, `BLOB` in `MySQL`.

Here is an example in `C#` with `MSSQL` 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();
```

In the code snippet, we get the file object from the current HTTP request and write the image data to a 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`.

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 section Upload and Save on the Server Disk.

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 Upload. 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"];
```

Download 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. It is especially useful when you want to review an image created and uploaded by 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);
}
```

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 an FTP server. Available methods are `FTPDownload()`, `FTPDownloadEx()`, etc.

> **NOTE**
>
> 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.

Use Capability Negotiation

Introduction

Capabilities represent the features that a specified TWAIN source (e.g., a scanner) provides. To make full use of such a source/device, TWAIN applications need 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

Container Data Structure	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 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, the 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 do two things

- **Get a Capability**. It is used to ask the source about a specified capability and get its type, value, etc.
- **Set a Capability**. It 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 take place when the source is open (`DataSourceStatus` is 1). You need to use the methods `SelectSource()` and `OpenSource()` to select a TWAIN source and get it ready for the negotiation. The related APIs are

- `Capability`

- `CapGet`
- `CapSet`
- `CapReset`
- `CapType`
- `CapValue`
- `CapValueType`
- `CapValueString`
- `CapGetCurrent`
- `CapGetDefault`
- `CapCurrentIndex`
- `CapCurrentValue`
- `CapDefaultIndex`
- `CapDefaultValue`
- `CapMaxValue`
- `CapMinValue`
- `CapNumItems`
- `CapStepSize`
- `GetCapItemsString`
- `GetCapItems`

Get

To get information about a capability, you can use the following code snippet by specifying the capability.

```
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) {
    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 TWAIN source vendor generally dictates the container type and available values for a capability. When you try to set a capability, you are just trying to change it so that it uses a different but available value. Therefore, you should try to **get** the capability before you set it.

The code to set capabilities is different if they have different container types.

TW_ONEVALUE

```
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

```
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

```
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

```
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();
```

Add/Remove Additional Dynamic Web TWAIN object(s)

> NOTE: What is discussed here works only for the HTML5 editions of Dynamic Web TWAIN.
>
> To add/remove an additional Dynamic Web TWAIN object, you can use the following methods.

```
Dynamsoft.WebTwainEnv.CreateDWTObject(id, OnSuccessCallback, OnFailureCallback)
Dynamsoft.WebTwainEnv.DeleteDWTObject(id).
```

Add a Dynamic Web TWAIN object at runtime

1. Create a new DIV element as the placeholder for this object.

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

2. Use the method `Dynamsoft.WebTwainEnv.CreateDWTObject(id, OnSuccessCallback, OnFailureCallback)` to create and initialize the Dynamic Web TWAIN object that will be embedded in the div with id `dwtcontrolContainer2` .

   ```
   var DWObject2;
   Dynamsoft.WebTwainEnv.CreateDWTObject(
       "dwtcontrolContainer2",
       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 `dynamsoft.webtwain.config.js`

3. You can now use the new Dynamic Web TWAIN object.

   ```
   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. Just call the method `Dynamsoft.WebTwainEnv.DeleteDWTObject(id)` with `id` specifying the container id. For example

```
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.

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 licenses, this is done by Dynamsoft
> 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)

Use the Product Key

Set it during initialization (recommended)

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

```
Dynamsoft.WebTwainEnv.ProductKey = 't0068MgAA…';
```

Set it when necessary (not recommended)

Set ProductKey in your code before you call the method `AcquireImage()`

```
function AcquireImage() {
    DWObject.SelectSource();
    DWObject.OpenSource();
    DWObject.IfShowUI = false;
    DWObject.ProductKey = 't0068Mg…';
    DWObject.AcquireImage();
}
```

Use Multiple ProductKeys

If you have multiple product keys generated from multiple serial numbers, you can combine all of them using the semi-colon `;` and assign them to the `ProductKey` property.

```
Dynamsoft.WebTwainEnv.ProductKey = 't0068MgAA…;t006…;t00…';
```

Use Add-ons

In summary, the functionality of the Dynamic Web TWAIN SDK itself is mainly

- Get images from a scanner or other source (such as local images, server-side images)
- Display the captured image in the image viewer
- Edit images in ways like rotating, cropping, deleting, etc.
- Encode the image data into various image formats (PDF, TIFF, JPEG, PNG, BMP)
- Save encoded images on the local disk or onto a remote server/database.

In many document management systems, in addition to the above functions, you may also need features like barcode recognition, text extraction, etc. In the light of this, Dynamsoft also offers the following extensions so that you can design more efficient and automated business processes:

PDF Rasterizer

Convert an existing text-based PDF file (that is, the text in the file is searchable) into an image. Because only then can this PDF file be displayed in the image viewer of the SDK.

Barcode Reader

Identify the barcode data directly from the image acquired from the scanner or local disk. The use of barcodes on documents can significantly improve the efficiency of document categorization or content extraction.

Dynamsoft Camera Capture

Display the video stream of a Webcam attached to the machine directly on the web page and capture still images from the stream.

Mobile Browser Capture

Perform document capture using the cameras from mobile devices.

OCR

Extract the text information from acquired images or convert the images to text or searchable PDF files.

Quickly convert PDF to images with the PDF Rasterizer

> ## Introduction
>
> PDF is one of the most popular formats on the market. In most cases, PDF files are readable, meaning that they contain text content internally. One way to convert this to an image is to take a screenshot of the page. However, this is very inefficient. By using the PDF Rasterizer, you can get it done quickly.

Environment

node

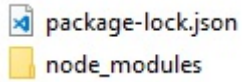
> NOTE:
>
> The PDF Rasterizer itself doesn't rely on `Node.js`, and it's needed in this article just because it's faster for us to use its package manager (npm) to get required files.

Steps

Step 1 Create a new directory, open the command line tool inside (shortcut is `Ctrl+Shift+right click`). Download the core control used in this article through npm

```
npm install dwt@14.2.0
```

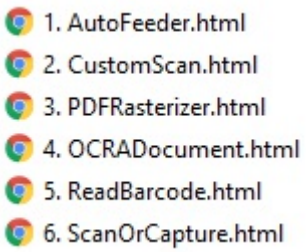
Then you can see the following in this directory



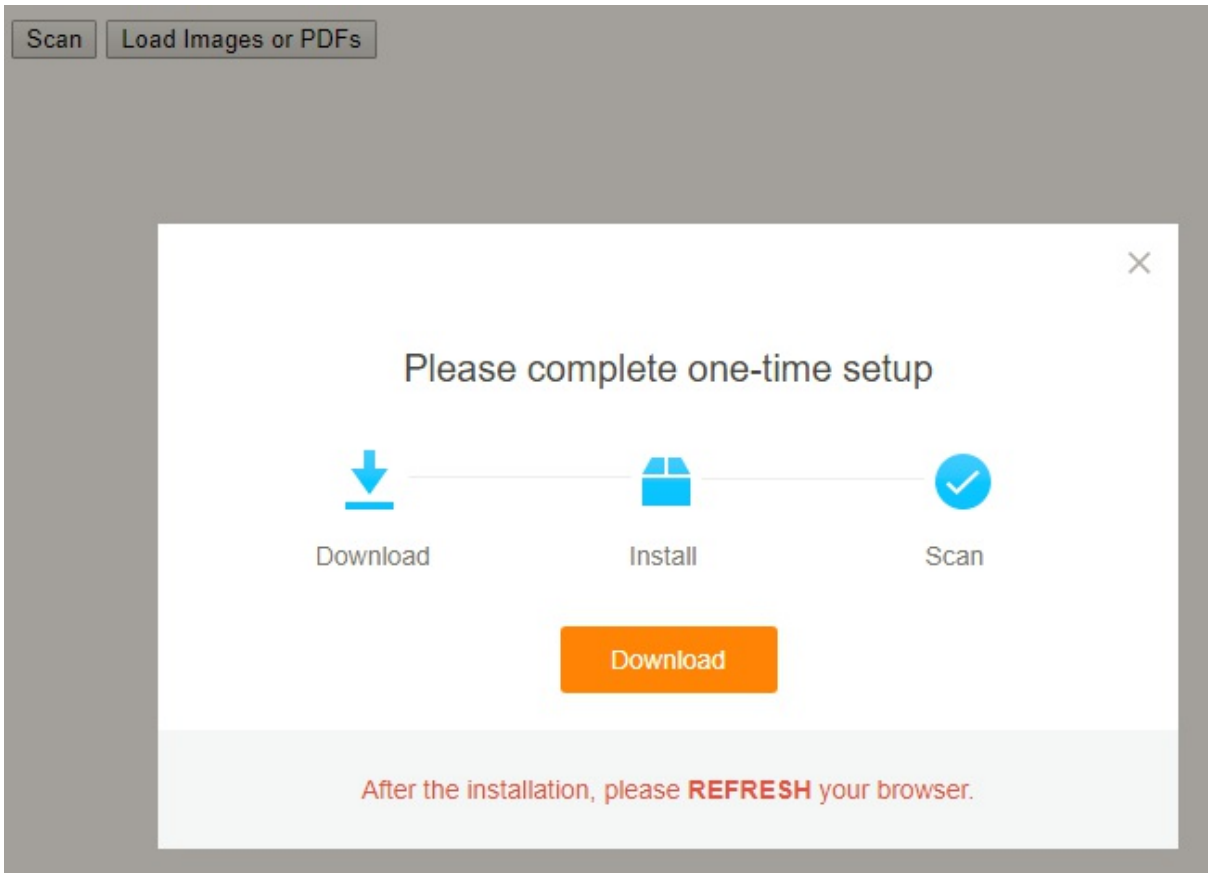
Step 2 Open to the following directory

`node_modules\dwt\sample`

where you can see



Step 3 In this article, we are going to check `PDFRasterizer.html` . Double click it to open. If the related controls are not yet available, follow the prompts to install them



Under normal circumstances, the installed files can be found in the `C:\Windows\SysWOW64\Dynamsoft\DynamsoftService` directory. The core files here are mainly

> `DynamsoftService.exe`
>
> `dwt_trial_14.1.0.0828.dll`
>
> `DynamicPdf_10.3.0.0712.dll`

Step 4 After the installation is complete, refresh the page, click the second button and open a local `PDF` file. Soon this `PDF` file will show up as an image(s) in the image viewer on the page

At this point, you can open the browser's developer interface (F12) and try to execute the following code which allows you to view multiple images at a time.

```
DWObject.SetViewMode(3,3);
```

You can also save these images to your local disk (in formats like `JPEG`, `BMP`, `PNG`, `TIF`, etc.)

```
DWObject.IfShowFileDialog = true;
DWObject.SaveAsJPEG('');
```

How it is done

Open `PDFRasterizer.html` in a text editor

References to the Core JavaScript files

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

Here the files referenced are

> JS library for the core SDK `Dynamic Web TWAIN`
>
> `node_modules\dwt\dist\dynamsoft.webtwain.initiate.js` `node_modules\dwt\dist\dynamsoft.webtwain.config.js`
>
> JS library for the `PDF Rasterizer addon`
>
> `node_modules\dwt\dist\addon\dynamsoft.webtwain.addon.pdf.js`

If you have previously installed the Dynamic Web TWAIN locally, the same files are located in the following directory.

`C:\Program Files (x86)\Dynamsoft\Dynamic Web TWAIN SDK {version number} {Trial}\Resources`

Runtime installation of the necessary files

When you open the page, the JavaScript files execute trying to set up the runtime environment. If the local Dynamsoft Service and the library files including the Dynamic Web TWAIN and PDF Rasterizer are missing, the code will show the prompt (Step 3) so that you can download and install the files.

Use the addon

```
if (DWObject.Addon && DWObject.Addon.PDF) {
    DWObject.Addon.PDF.SetResolution(300);
    DWObject.Addon.PDF.SetConvertMode(EnumDWT_ConvertMode.CM_RENDERALL);
}
DWObject.LoadImageEx('', 5,
    function () {
    },
    function (errorCode, errorString) {
        alert('Load Image:' + errorString);
    }
);
```

The core code is

```
DWObject.Addon.PDF.SetResolution(300); //Set the resolution for the conversion
```

```
DWObject.Addon.PDF.SetConvertMode(EnumDWT_ConvertMode.CM_RENDERALL);//Set the conversion mode, generally just set it to CM_RENDERALL
```

After the above settings, when you import a `PDF` file by calling one of the methods `LoadImage`, `LoadImageEx`, `HTTPDownload`, etc., `PDF Rasterizer` is called automatically to raster the file into an image.

Automatic classification in document digitalization by using Barcode recognition

> ## Introduction
>
> Nowadays, the idea of a paperless office is becoming more and more popular, and the digitalization of paper documents has become a trend. At the same time, many industries, such as hospitals, banks, etc., still need to print documents and then digitalize them after that. As a result, due to the time difference between different operations, it is very likely that a large number of paper documents will pile up and wait for scanning. Then it'll require a one-time scanning of multiple documents during which the documents need to be classified. It is true that these tasks can be done manually, but automation can save a lot of time and effort. In this article, we share how to implement the automation through barcode in a web application.

Environment

- Windows
- Local web server

Steps

Step 1 Create a new directory `DocumentsSeparation` create a new page in it `index.html`

```
<!DOCTYPE html>
  <html>
    <head>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <title>Documents Separation</title>
    </head>
  <body>
  </body>
</html>
```

Step 2 Reference core JavaScript library

```
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Documents Separation</title>
    <script src="https://tst.dynamsoft.com/libs/dwt/14.2/dynamsoft.webtwain.config.js"></script>
    <script src="https://tst.dynamsoft.com/libs/dwt/14.2/dynamsoft.webtwain.initiate.js"></script>
    <script src="https://tst.dynamsoft.com/libs/dbr/6.3/dynamsoft.barcodereader.config.js"> </script>
    <script src="https://tst.dynamsoft.com/libs/dbr/6.3/dynamsoft.barcodereader.initiate.js"></script>
</head>
```

> **Note**

> The online `JavaScript` file is referenced here. In your project, you should instead reference the corresponding file in your project. If you have previously installed the `Dynamic Web TWAIN` product locally, the same files can also be found in the following directory.
>
> `C:\Program Files (x86)\Dynamsoft\Dynamic Web TWAIN SDK {version number} {Trial}\Resource`

Step 3 Add a `DIV` and a few buttons

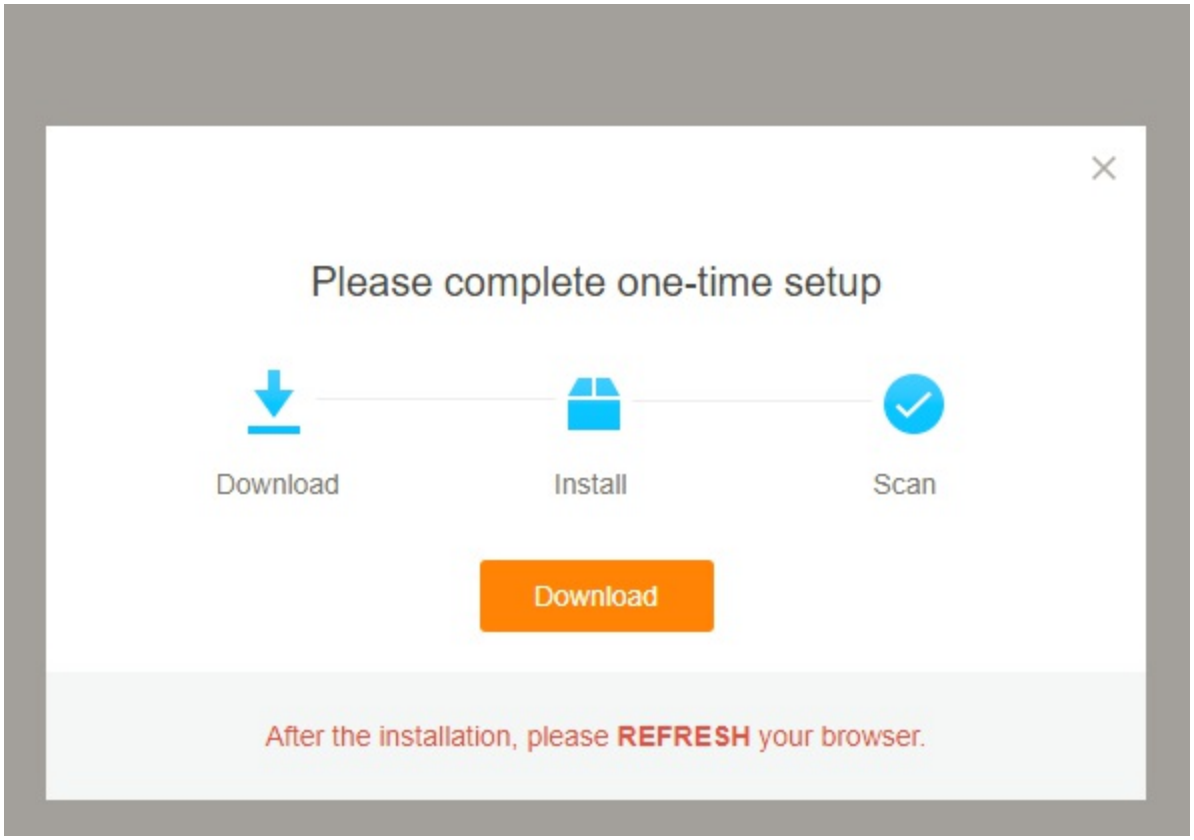
```
<body>
    <input type="button" value="Scan" onclick="AcquireImage();" />
    <input type="button" value="Load" onclick="LoadImages();" />
    <input type="button" value="Delete" onclick="RemoveImages();" />
    <input type="button" value="Separate" onclick="UploadFiles();" />
    <br />
    <br />
    <div id="dwtcontrolContainer" style="float: left"></div>
</body>
```

They'll look like this

Step 4 Add initialization code

```
<script type="text/javascript">
    var dbrObject, DWObject;
    window.onload = function () {
        if (Dynamsoft && (!Dynamsoft.Lib.env.bWin || !Dynamsoft.Lib.product.bChromeEdition)) {
            var ObjString = [];
            ObjString.push('<div class="p15">');
            ObjString.push("Current browser is not supported, please use Chrome, Firefox, Edge or IE 11");
            ObjString.push('</div>');
            Dynamsoft.WebTwainEnv.ShowDialog(400, 180, ObjString.join(''));
            if (document.getElementsByClassName("dynamsoft-dialog-close"))
                document.getElementsByClassName("dynamsoft-dialog-close")[0].style.display = "none";
        } else {
            Dynamsoft.WebTwainEnv.Load();
        }
    };
    Dynamsoft.WebTwainEnv.AutoLoad = false;
    //Dynamsoft.WebTwainEnv.ProductKey = '***';
    Dynamsoft.WebTwainEnv.RegisterEvent('OnWebTwainReady', Dynamsoft_OnReady);

    dynamsoft.dbrEnv.onAutoConnectServiceSuccess = function() {
        dbrObject = new dynamsoft.BarcodeReader();
    }

    dynamsoft.dbrEnv.onAutoConnectServiceError = function(ex) {
        console.log('Initialization failed with error code: ' + (ex.message || ex));
    }
    function Dynamsoft_OnReady() {
        DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
        if (DWObject) {
            DWObject.SetViewMode(3, 3);
        }
    }
</script>
```

Step 5 Open `index.html` in the browser. If it has not been installed before, follow the prompts on the page to install the corresponding scan and Barcode recognition controls. This installation process only needs to be done once on each computer



Typically they are installed to

`C:\Windows\SysWOW64\Dynamsoft\DynamsoftService`

The following are the major files used in the article.

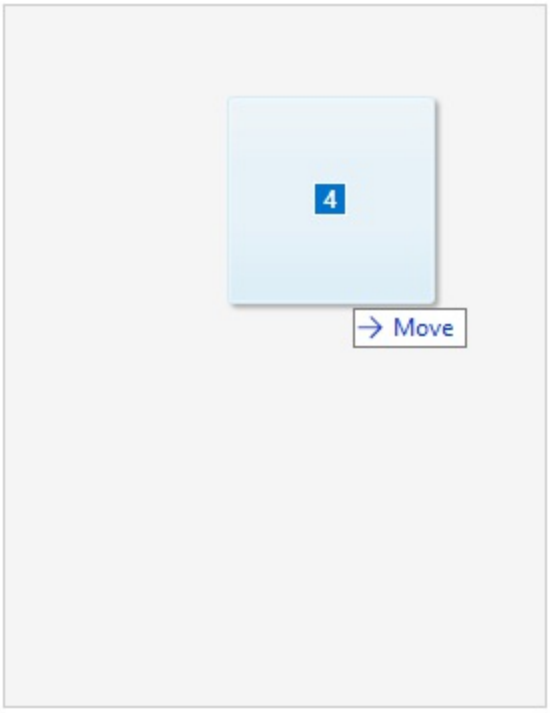
> `DynamsoftService.exe`
>
> `dwt_trial_14.1.0.0828.dll`
>
> `DynamsoftBarcodeReaderx86_6.3.dll`
>
> `dbr_6.3.0.0723.dll`

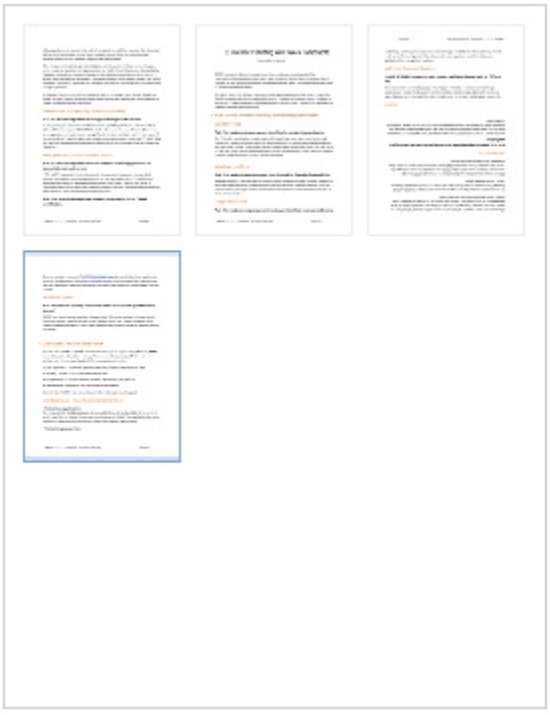
Step 6 Add code for the buttons

```
function AcquireImage() {
    if (DWObject) {
        DWObject.SelectSource(function () {
            var OnAcquireImageSuccess, OnAcquireImageFailure;
            OnAcquireImageSuccess = OnAcquireImageFailure = function () {
                DWObject.CloseSource();
            };
            DWObject.OpenSource();
            DWObject.IfDisableSourceAfterAcquire = true;
            DWObject.AcquireImage(OnAcquireImageSuccess, OnAcquireImageFailure);
        }, function () {
```

```
            console.log('Failed to select a source');
        });
    }
}
function LoadImages() {
    if (DWObject) {
        DWObject.LoadImageEx('', 5,
            function () {
            },
            function (errorCode, errorString) {
                console.log('ailed to load a file with the error code:' + errorString);
            }
        );
    }
}
function RemoveImages() {
    if (DWObject)
        DWObject.RemoveAllSelectedImages();
}
```

Step 7 Refresh the page. At this point, you can call the local scanner or load the local image, or delete the selected image. You can even drag a local image directly to load it directly





Step 8 Add the Barcode recognition code, because there are many types of Barcode, we first add a selection box, the corresponding code is as follows

HTML

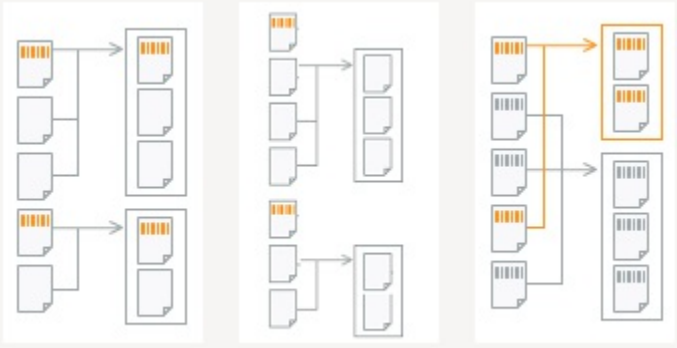
```
<select size="1" id="barcodeformat"></select>
```

JavaScript

```
var BarcodeInfo =
    [
        { desc: "All", val: 503317503 },
        { desc: "1D Barcodes", val: 1023 },
        { desc: "QR Code", val: 67108864 },
        { desc: "PDF417", val: 33554432 },
        { desc: "DATAMATRIX", val: 134217728 },
        { desc: "AZTEC", val: 268435456 },
        { desc: "CODE_39", val: 1 },
        { desc: "CODE_128", val: 2 },
        { desc: "CODE_93", val: 4 },
        { desc: "CODABAR", val: 8 },
        { desc: "ITF", val: 16 },
        { desc: "EAN_13", val: 32 },
        { desc: "EAN_8", val: 64 },
        { desc: "UPC_A", val: 128 },
        { desc: "UPC_E", val: 256 },
        { desc: "INDUSTRIAL_25", val: 512 }
    ];
// Add the following code to the function `Dynamsoft_OnReady` mentioned above
for (var index = 0; index < BarcodeInfo.length; index++)
  document.getElementById("barcodeformat").options.add(new Option(BarcodeInfo[index].desc, index));
document.getElementById("barcodeformat").options.selectedIndex = 0;
```

Step 9 There are three ways to do the classification of documents

- Start each document with an image with barcode
- Use the images with barcode as the separator only without including them in any document
- Each page has a Barode and the image with the same Barcode belongs to the same file

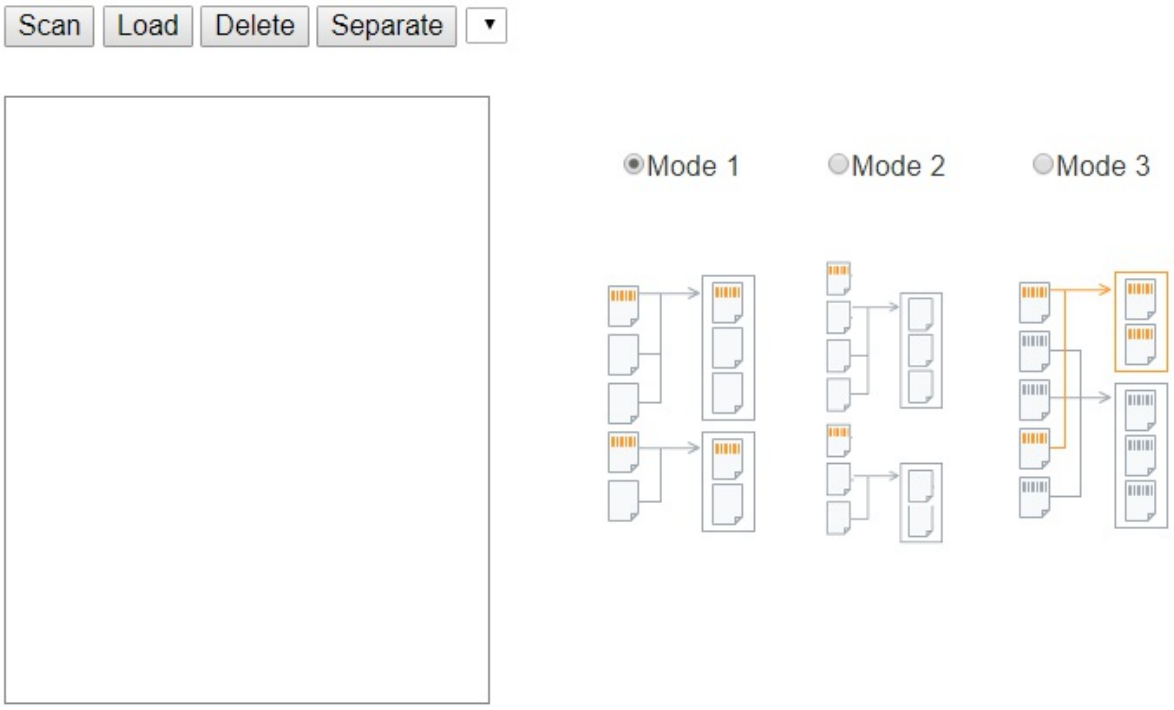


We'll implement all three ways in our code

```
<div style="float: left; margin-left: 20px;">
    <ul style="list-style: none; min-height: 20px;">
        <li style="width:118px; float: left; text-align: center;">
            <label for="mode1">
                <input type="radio" name="UploadModes" checked="checked" value="mode1">Mode 1</label>
        </li>
        <li style="width:118px; float: left; text-align: center;">
            <label for="mode2">
                <input type="radio" name="UploadModes" value="mode2" id="mode2">Mode 2</label>
        </li>
        <li style="width:118px; float: left; text-align: center;">
```

```
            <label for="mode3">
                <input type="radio" name="UploadModes" value="mode3" id="mode3">Mode 3</label>
        </li>
    </ul>
    <br />
    <ul style="list-style: none; min-height: 180px;">
        <li style="width:118px; height:176px; float: left; background: url('https://tst.dynamsoft.com/libs/dbr/modes/
Mode1.png') center no-repeat">
        </li>
        <li style="width:118px; height:176px; float: left; background: url('https://tst.dynamsoft.com/libs/dbr/modes/
Mode2.png') center no-repeat">
        </li>
        <li style="width:118px; height:176px; float: left; background: url('https://tst.dynamsoft.com/libs/dbr/modes/
Mode3.png') center no-repeat">
        </li>
    </ul>
</div>
```

Now the page looks like this



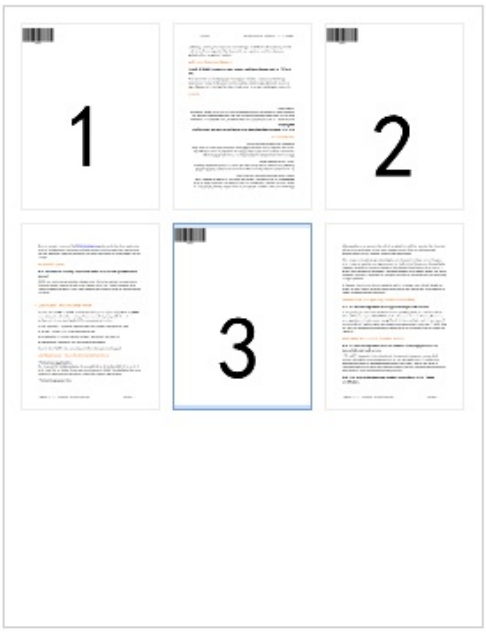
Step 10 Add JavaScript code for the classification

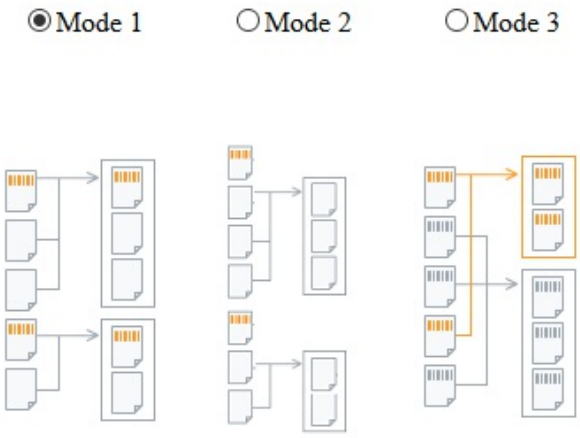
```
function UploadFiles() {
    DWObject.IfShowProgressBar = false;
    ProcssedImagesCount = 0;
    imageArrays = [];
    aryIndicesMode1 = [];
    aryIndicesMode2 = [];
    aryIndicesMode3 = {
        'noBarcode': []
    };
    Dynamsoft.Lib.showMask();
    ReadBarcode(0);
}
```

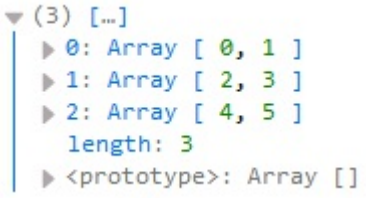
```
function ReadBarcode(i) {
    var j, sImageIndex = i,
        bBarcodeFound = false,
        strSelectedMode = document.getElementsByName('UploadModes');
    for (j = 0; j < strSelectedMode.length; j++) {
        if (strSelectedMode.item(j).checked == true) {
            strSelectedMode = strSelectedMode.item(j).value;
            break;
        }
    }
    if (sImageIndex == DWObject.HowManyImagesInBuffer)
        return;
    if (dbrObject) {
        var settings = dbrObject.getRuntimeSettings();
        settings.mBarcodeFormatIds = BarcodeInfo[document.getElementById("barcodeformat").selectedIndex].val;
        dbrObject.updateRuntimeSettings(settings);
        DWObject.CurrentImageIndexInBuffer = sImageIndex;
        var barcodeImage = DWObject.GetImageURL(sImageIndex, -1, -1);
        dbrObject.decode(barcodeImage).then(function (results) {
            ProcssedImagesCount++;
            if (results.length == 0) {
                console.log("No barcode found on image " + (sImageIndex + 1));
                if (bBarcodeFound == true) {
                    bBarcodeFound = false;
                    aryIndicesMode1[aryIndicesMode1.length - 1].push(sImageIndex);
                    if (aryIndicesMode2.length == 0)
                        aryIndicesMode2.push([sImageIndex]);
                    else
                        aryIndicesMode2[aryIndicesMode2.length - 1].push(sImageIndex);
                } else {
                    if (aryIndicesMode1.length == 0)
                        aryIndicesMode1.push([sImageIndex]);
                    else
                        aryIndicesMode1[aryIndicesMode1.length - 1].push(sImageIndex);
                    if (aryIndicesMode2.length == 0)
                        aryIndicesMode2.push([sImageIndex]);
                    else
                        aryIndicesMode2[aryIndicesMode2.length - 1].push(sImageIndex);
                }
                aryIndicesMode3.noBarcode.push(sImageIndex);
            } else {
                bBarcodeFound = true;
                console.log("Barcode found on image " + (sImageIndex + 1));

                aryIndicesMode1.push([sImageIndex]);
                if (aryIndicesMode2.length == 0)
                    aryIndicesMode2.push([]);
                else if (aryIndicesMode2[aryIndicesMode2.length - 1].length != 0)
                    aryIndicesMode2.push([]);
                var barcodeOnThisImage = [],
                    allKeys = [];
                for (j = 0; j < results.length; j++) {
                    var result = results[j];
                    var barcodeText = result.BarcodeText;
                    if (barcodeOnThisImage.indexOf(barcodeText) == -1)
                        barcodeOnThisImage.push(barcodeText);
                    console.log("The content for barcode number " + (j + 1) + "is: " + barcodeText);
                    var imageArray = {
                        index: sImageIndex,
                        text: barcodeText
                    };
                    imageArrays.push(imageArray);
                }

                Dynamsoft.Lib.each(aryIndicesMode3, function (value, key) {
```

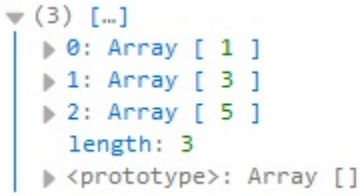
```
                    allKeys.push(key);
                });

                for (j = 0; j < allKeys.length; j++) {
                    var oKey = allKeys[j];
                    if (barcodeOnThisImage.indexOf(oKey) != -1) {
                        barcodeOnThisImage.splice(barcodeOnThisImage.indexOf(oKey), 1);
                        var _value = aryIndicesMode3[oKey];
                        if (_value.indexOf(sImageIndex) == -1) {
                            _value.push(sImageIndex);
                            aryIndicesMode3[oKey] = _value;
                        }
                    }
                }
                for (j = 0; j < barcodeOnThisImage.length; j++) {
                    aryIndicesMode3[barcodeOnThisImage[j]] = [sImageIndex];
                }
            }
            if (ProcssedImagesCount == DWObject.HowManyImagesInBuffer) {
                ProcssedImagesCount = 0;
                var aryTemp = [];
                Dynamsoft.Lib.each(aryIndicesMode3, function (value, key) {
                    aryTemp.push(value);
                });
                aryIndicesMode3 = aryTemp;
                Dynamsoft.Lib.hideMask();
                switch (strSelectedMode) {
                    case 'mode1':
                        console.log(aryIndicesMode1);
                        break;
                    case 'mode2':
                        console.log(aryIndicesMode2);
                        break;
                    case 'mode3':
                        console.log(aryIndicesMode3);
                        break;
                }
            }
            /*
             * Read the next image
             */
            ReadBarcode(sImageIndex + 1);
        }, function (ex) {
            console.log("Error reading barcode: " + ex.message);
            Dynamsoft.Lib.hideMask();
        });
    }
}
```

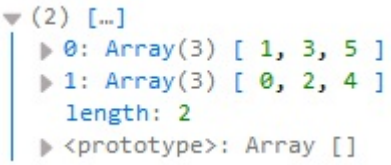
Load a few images and click "`Separate`", then check the browser console (F12)











Step 11 Add upload code and back-end code (in `C#`) to receive the separated files

```
function UploadImagesSeparatedByBarcode(ary) {
    var i, Digital, uploadfilename, CurrentPathName = unescape(location.pathname),
        CurrentPath = CurrentPathName.substring(0, CurrentPathName.lastIndexOf("/") + 1),
        strActionPage = CurrentPath + "SaveToFile.aspx";
    DWObject.IfSSL = Dynamsoft.Lib.detect.ssl;
    var _strPort = location.port == "" ? 80 : location.port;
    if (Dynamsoft.Lib.detect.ssl == true)
        _strPort = location.port == "" ? 443 : location.port;
    DWObject.HTTPPort = _strPort;
    strFullActionPagePath = location.protocol + "//" + location.hostname + ":" + DWObject.HTTPPort + strActionPage;
    for (i = 0; i < ary.length; i++) {
        if (ary[i].length == 0) {
            ary.splice(i, 1);
            i--;
            continue;
        }
        Digital = new Date();
        uploadfilename = 'Doc_' + i + '_' + Digital.getMilliseconds() + '_' + (Math.floor(Math.random() * 1000 + 1)).
toString() + '.pdf';
        DWObject.HTTPUpload(strFullActionPagePath, ary[i], EnumDWT_ImageType.IT_PDF, EnumDWT_UploadDataFormat.Binary,
 uploadfilename, function () { }, function () { });
```

```
    }
}
```

```
HttpFileCollection files = HttpContext.Current.Request.Files;
HttpPostedFile uploadfile = files["RemoteFile"];
String Path = System.Web.HttpContext.Current.Request.MapPath(".") + "/ImageScanned/";
if (!Directory.Exists(Path))
{
    Directory.CreateDirectory(Path);
}
uploadfile.SaveAs(Path + uploadfile.FileName);
```

Step 12 Change code to upload files

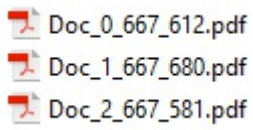
Update the following

```
switch (strSelectedMode) {
    case 'mode1': console.log(aryIndicesMode1); break;
    case 'mode2': console.log(aryIndicesMode2); break;
    case 'mode3': console.log(aryIndicesMode3); break;
}
```

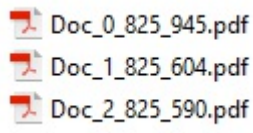
to

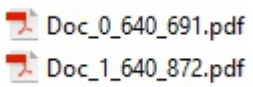
```
switch (strSelectedMode) {
    case 'mode1': UploadImagesSeparatedByBarcode(aryIndicesMode1); break;
    case 'mode2': UploadImagesSeparatedByBarcode(aryIndicesMode2); break;
    case 'mode3': UploadImagesSeparatedByBarcode(aryIndicesMode3); break;
}
```

Step 13 Refresh the page, load the images again and click " `Separate` ". Then you can check the uploaded images on the server



For mode 2, 3





That's it. You can also test the code online.

Quickly integrate mobile device camera into web applications

> ## Introduction
>
> In recent years, with the popularity of smart mobile devices, websites must be designed with a balance of traditional desktop browsers (including the three most common systems `Windows`, `macOS` and `Linux`) and mobile device browsers (generally including `iOS` and `Android`). This article shares how to design a web application that takes into account desktop and mobile browsers.

Preparation

Download the following files.

`Dynamic Web TWAIN v14.2` (.exe)

`Mobile Browser Capture v2.0` (.zip)

Environment

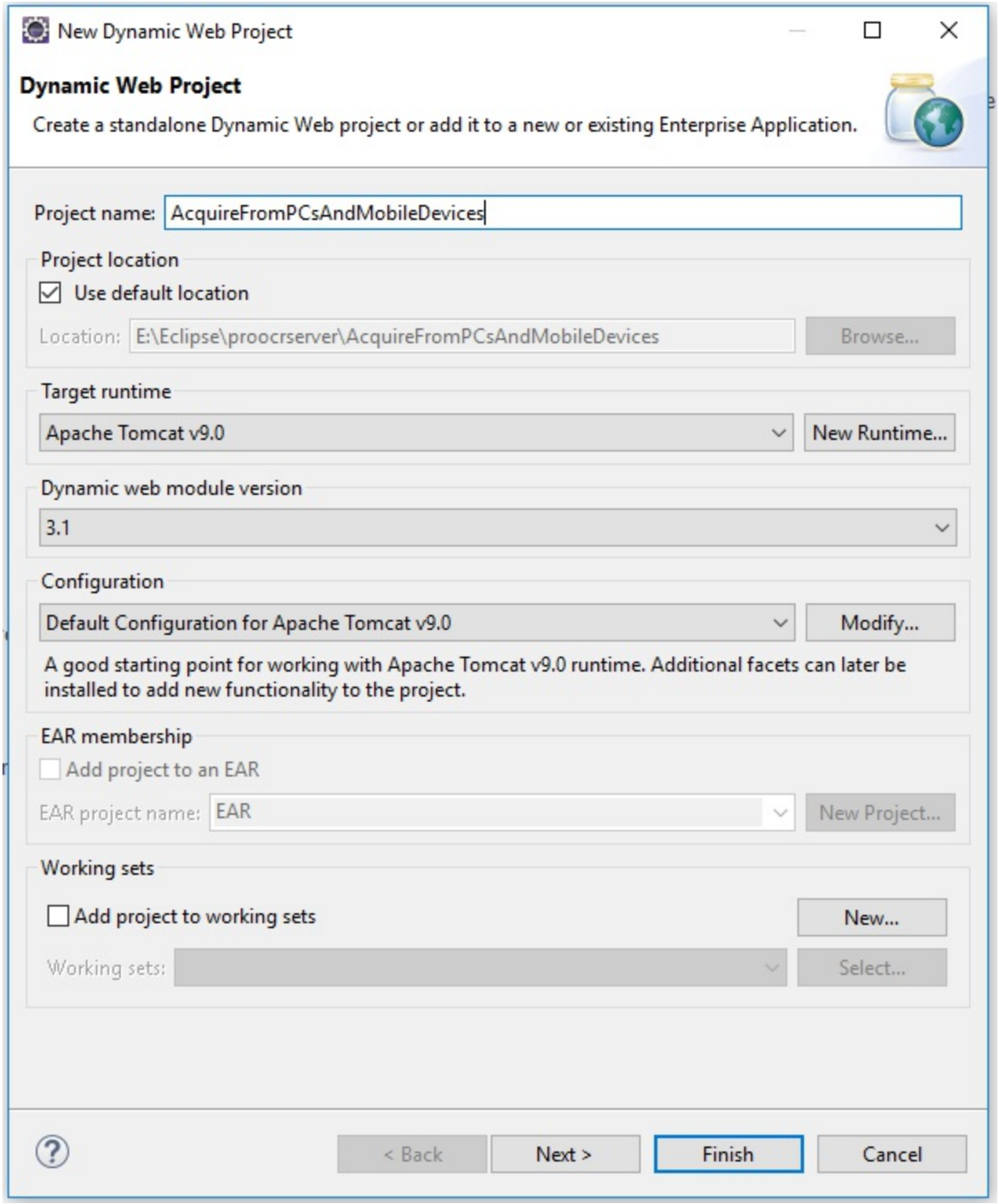
`Dynamic Web TWAIN` can be used across desktop platforms.

`Mobile Browser Capture` currently supports only `.NET` & `Java`. We'll use `Java` in this article.

- `JDK`：`1.8.0_172`
- `Eclipse`：`Oxygen.3a Release (4.7.3a)`
- `64位Tomcat`：`Tomcat v9.0`

Steps

Step 1 Create a `Dynamic Web Application`, name it `AcquireFromPCsAndMobileDevices` and select `Apache Tomcat v9.0` as the runtime environment

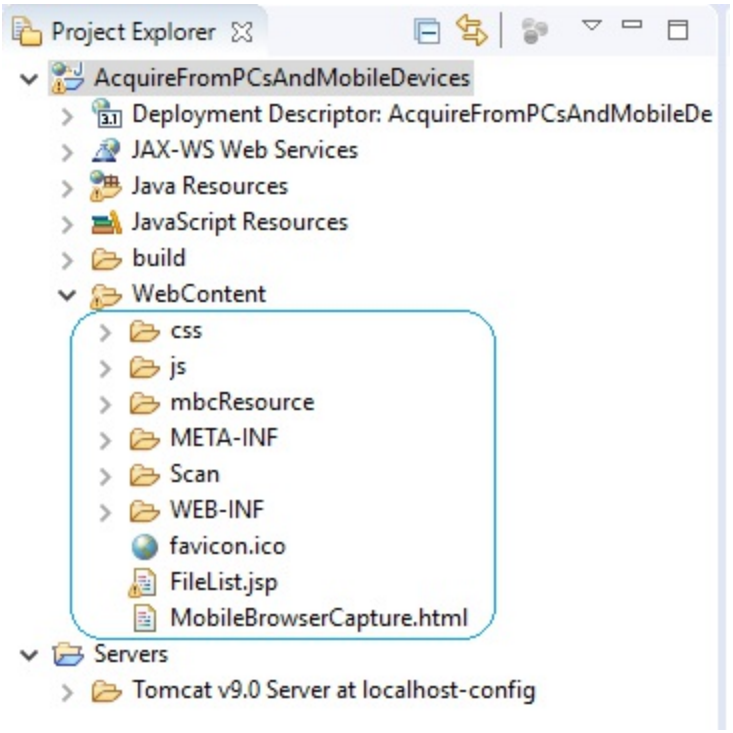


Step 2 Start from a sample

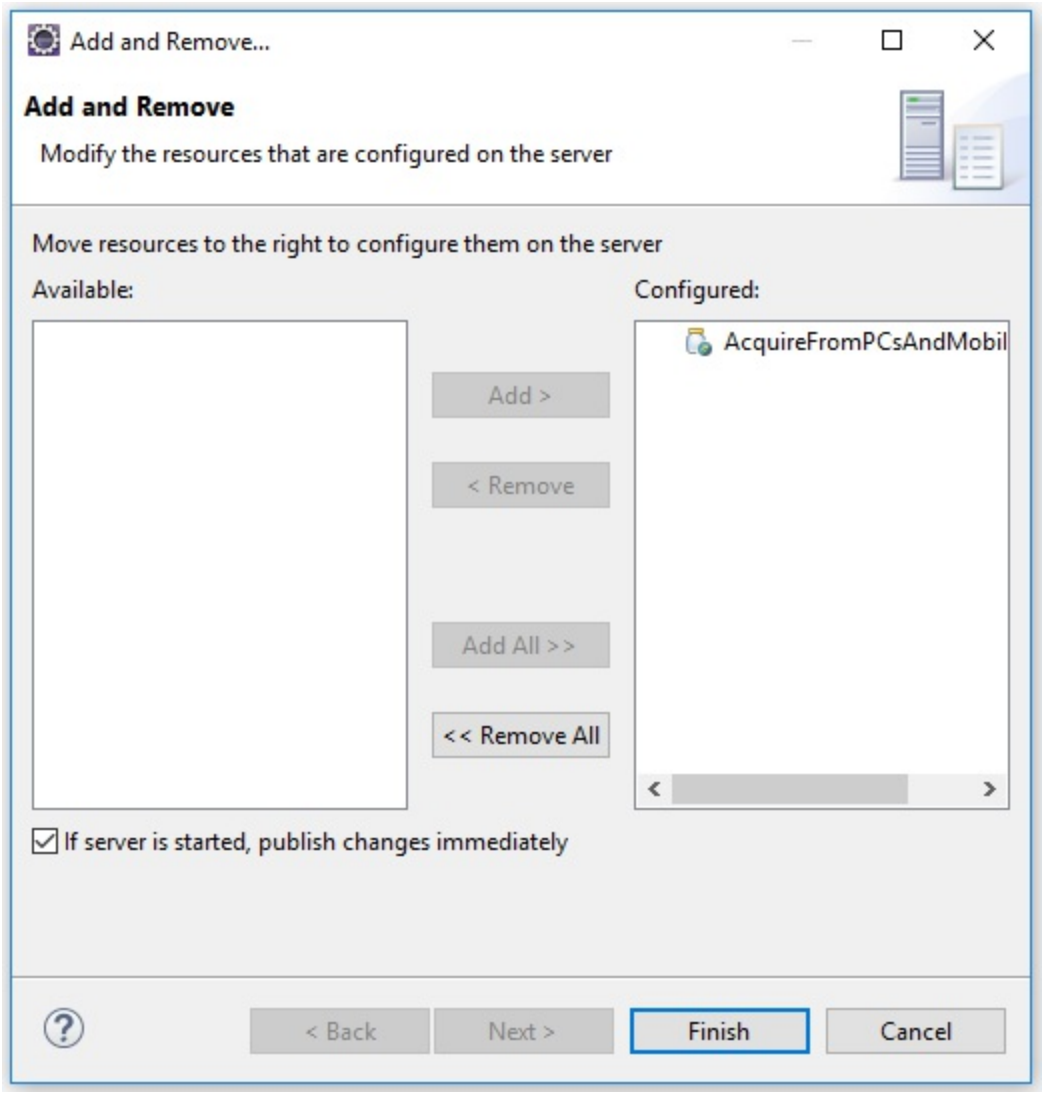
Unzip `Mobile Browser Capture` , copy `src` and `WebContent` from `MobileBrowserCaptureSDK2.0\MobileBrowserCaptureSDK2.0\samples\javaDemo` to application `AcquireFromPCsAndMobileDevices` and replace any conflicts.

Install `Dynamic Web TWAIN` and navigate to `C:\Program Files (x86)\Dynamsoft\Dynamic Web TWAIN SDK 14.1 Trial` . Copy `Samples\Scan` to `AcquireFromPCsAndMobileDevices\WebContent` .

In `Eclipse` , refresh to see the file changes.



Step 3 Add a 'server' in `Eclipse` and add `AcquireFromPCsAndMobileDevices` to it. Then start `Tomcat`



Check the application in the browser.

For mobile browser

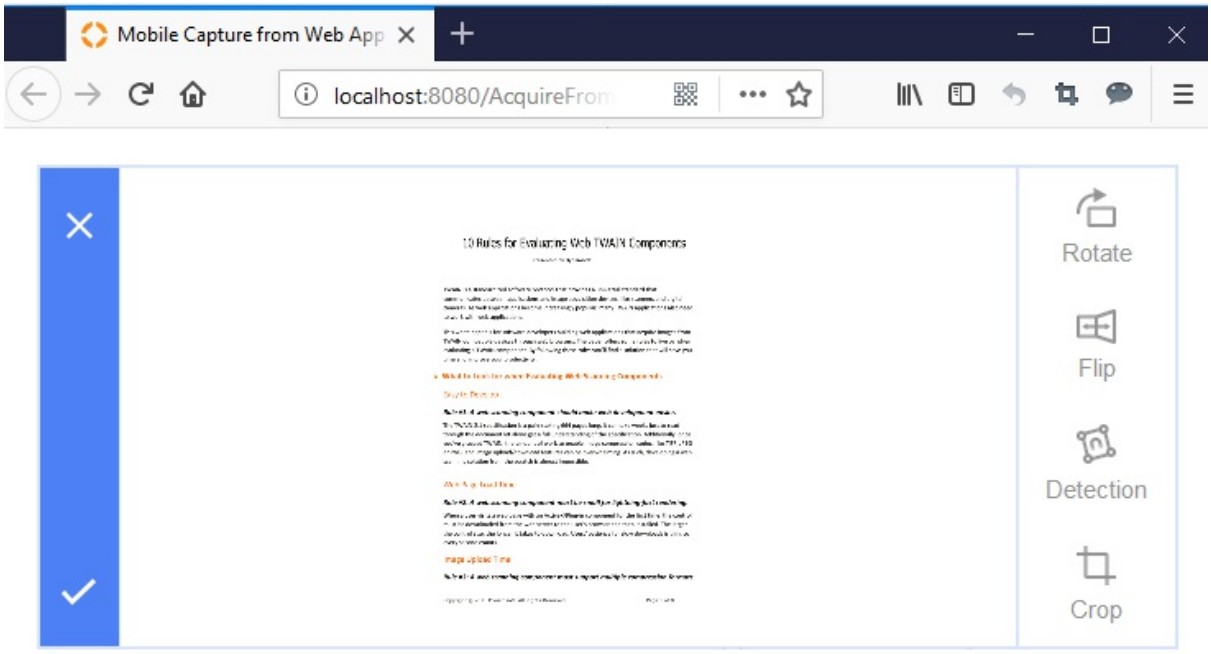
```
http://localhost:8080/AcquireFromPCsAndMobileDevices/MobileBrowserCapture.html
```

For desktop browser

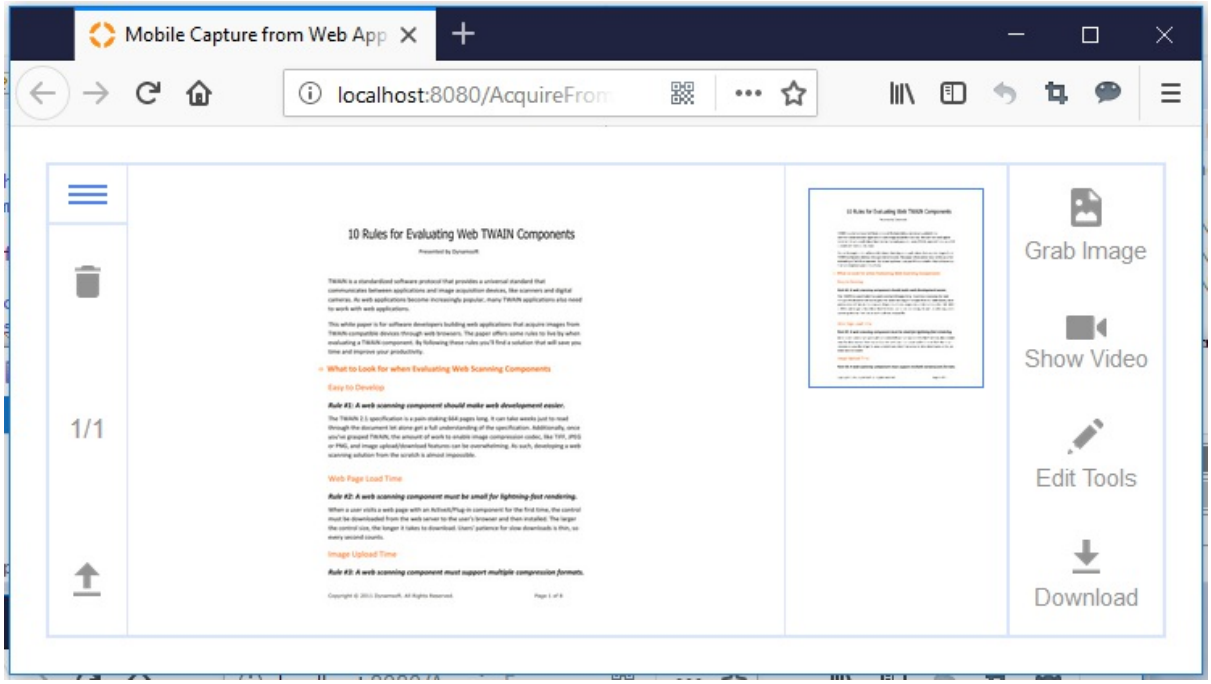
```
http://localhost:8080/AcquireFromPCsAndMobileDevices/Scan/CustomScan.html
```

Step 4 Try the page for mobile browser first

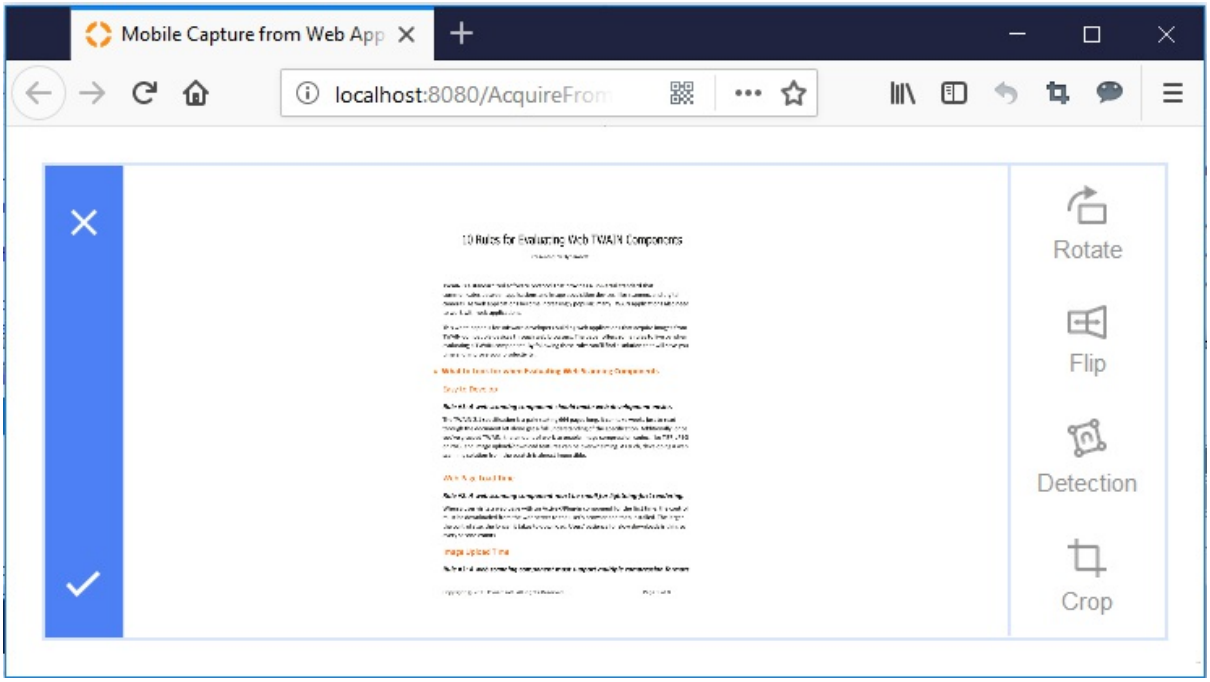
This page works both on desktop browsers and mobile browsers, and in desktop Chrome, it's like this



Here you can click `Show Video` to use the webcam on the PC or click `Grab Image` to load a local file. We'll load a file.



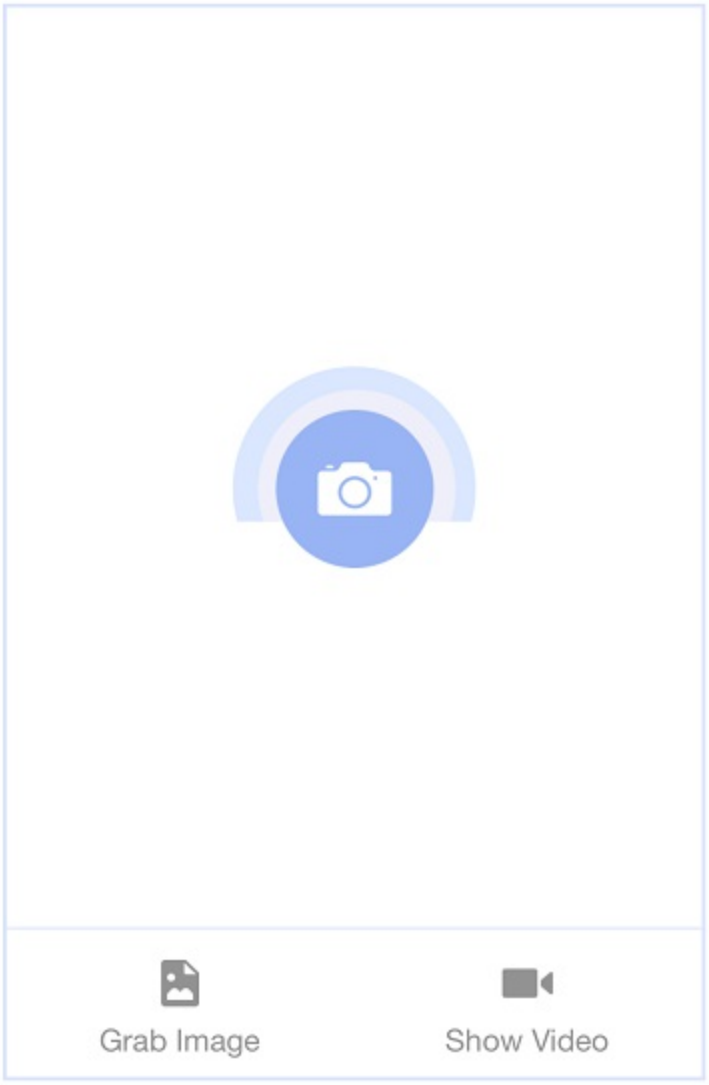
Once the file is loaded, extra features will show up as buttons to delete, upload, edit, download, etc.



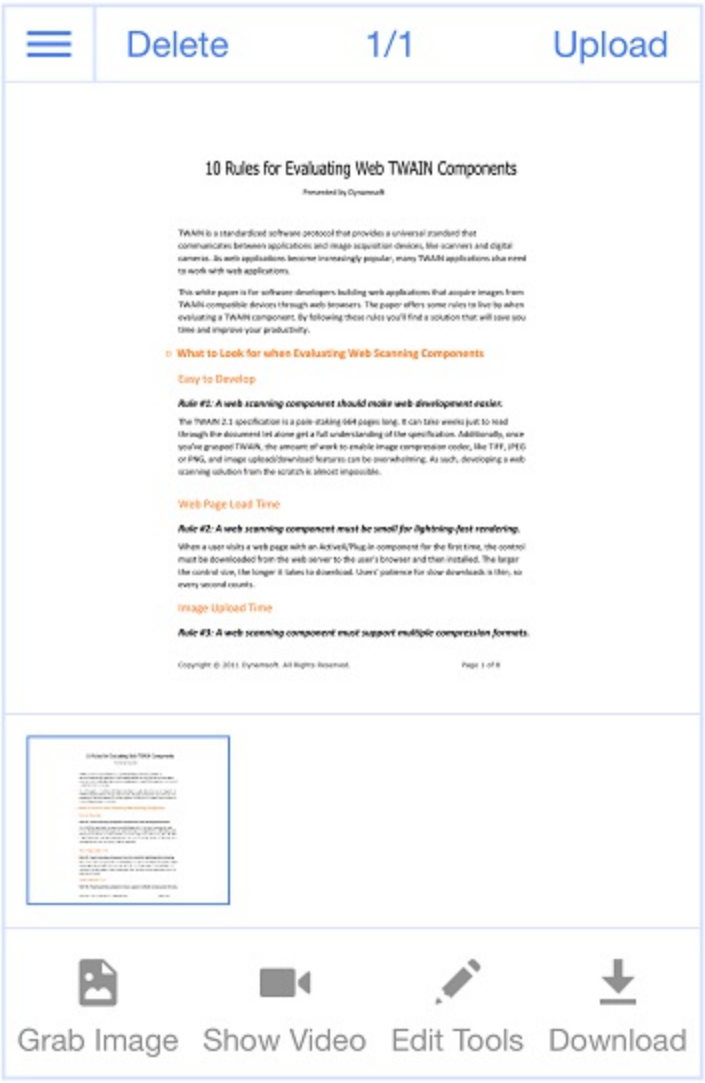
Open it in a mobile browser (assume the `IP` of the dev machine is `192.168.1.100`)

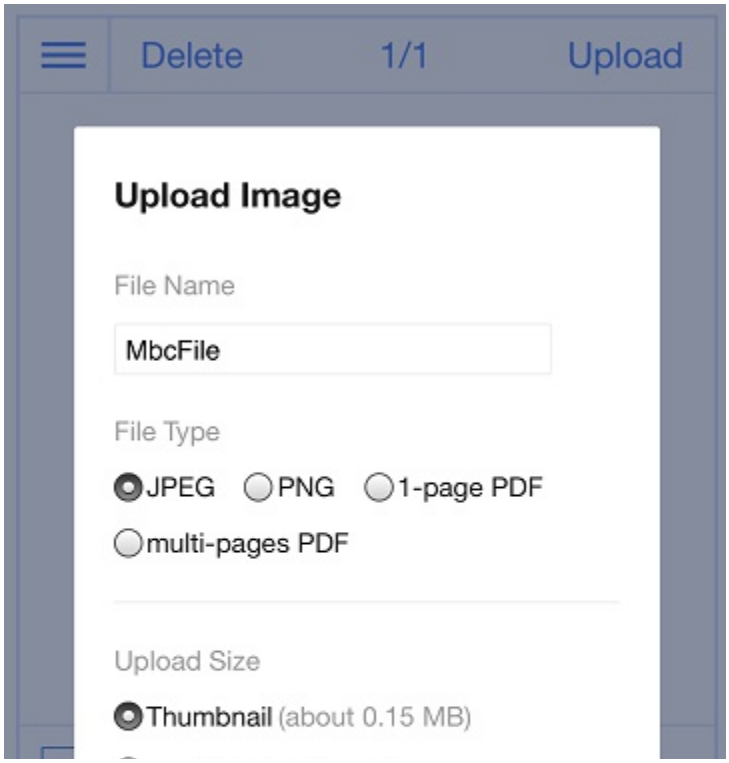
```
http://192.168.1.100:8080/AcquireFromPCsAndMobileDevices/MobileBrowserCapture.html
```

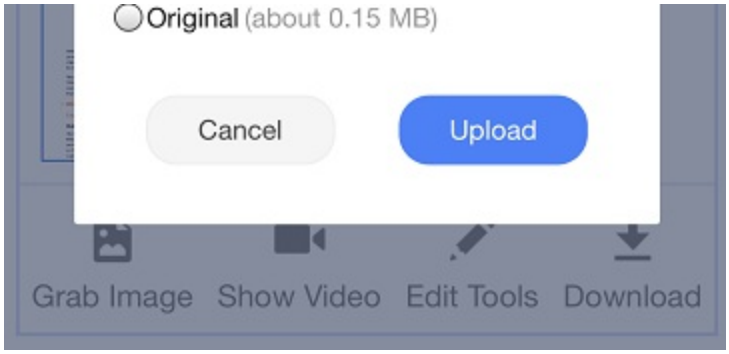
The interface is like the following where you can click 'Grab Image' to take a picture or load a local file



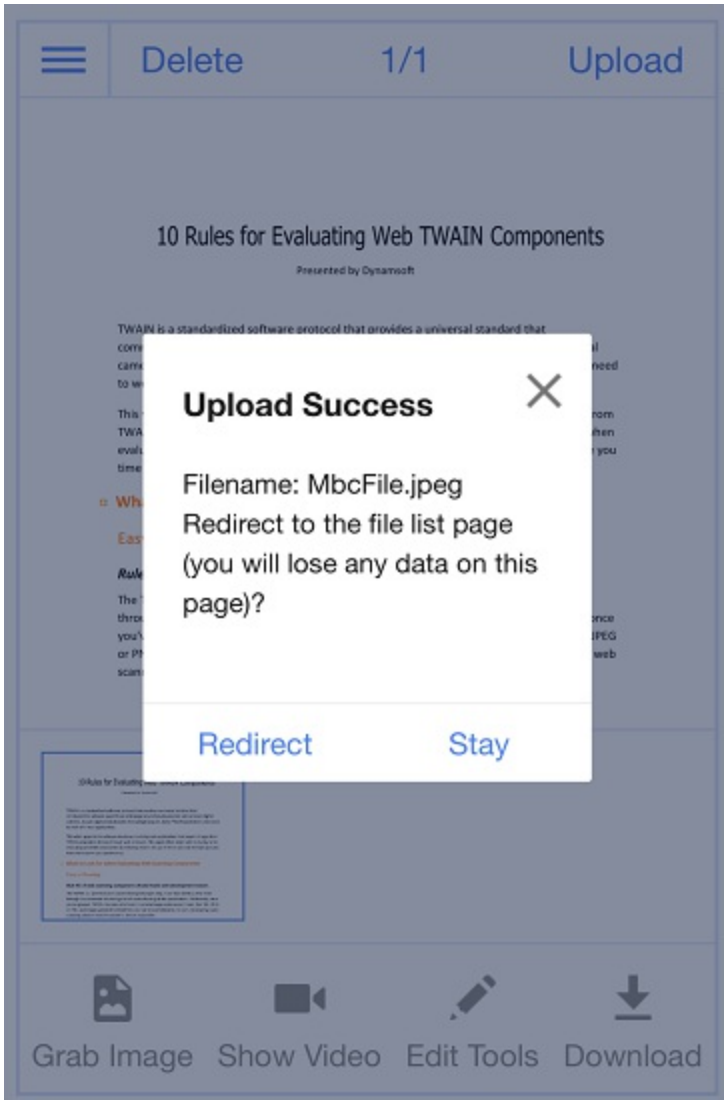
Once an image is loaded/captured

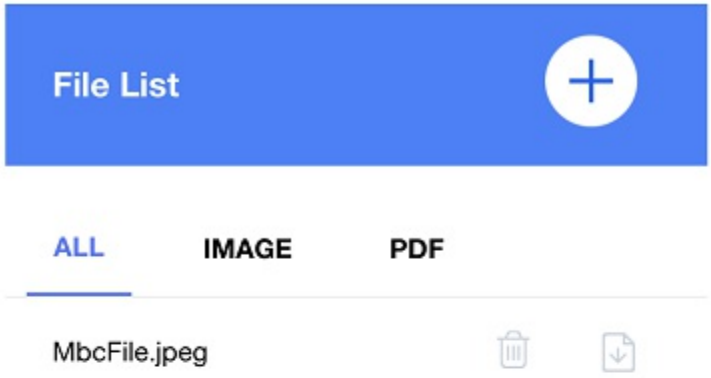

Delete
1/1
Upload
10 Rules for Evaluating Web TWAIN Components
Grab Image
Show Video
Edit Tools
Download


Delete
1/1
Upload
Upload Image
File Name
MbcFile
File Type
JPEG
PNG
1-page PDF
multi-pages PDF
Upload Size
Thumbnail (about 0.15 MB)



Upload and press `Redirect` to see a list of the uploaded files.


Delete
1/1
Upload
10 Rules for Evaluating Web TWAIN Components
Upload Success
Filename: MbcFile.jpeg
Redirect to the file list page (you will lose any data on this page)?
Redirect
Stay
Grab Image
Show Video
Edit Tools
Download


File List
ALL
IMAGE
PDF
MbcFile.jpeg

The uploaded files are saved in a path specified by `System.getProperty("user.dir")` plus a runtime user id. For example

`C:\Program Files\eclipse-jee-oxygen-3a-win32-x86_64\eclipse\Dynamsoft_Upload\391008ba-aa9f-4564-a285-b44a42ec7864`

The path can be changed in the following file.

`AcquireFromPCsAndMobileDevices\WebContent\WEB-INF\web.xml`

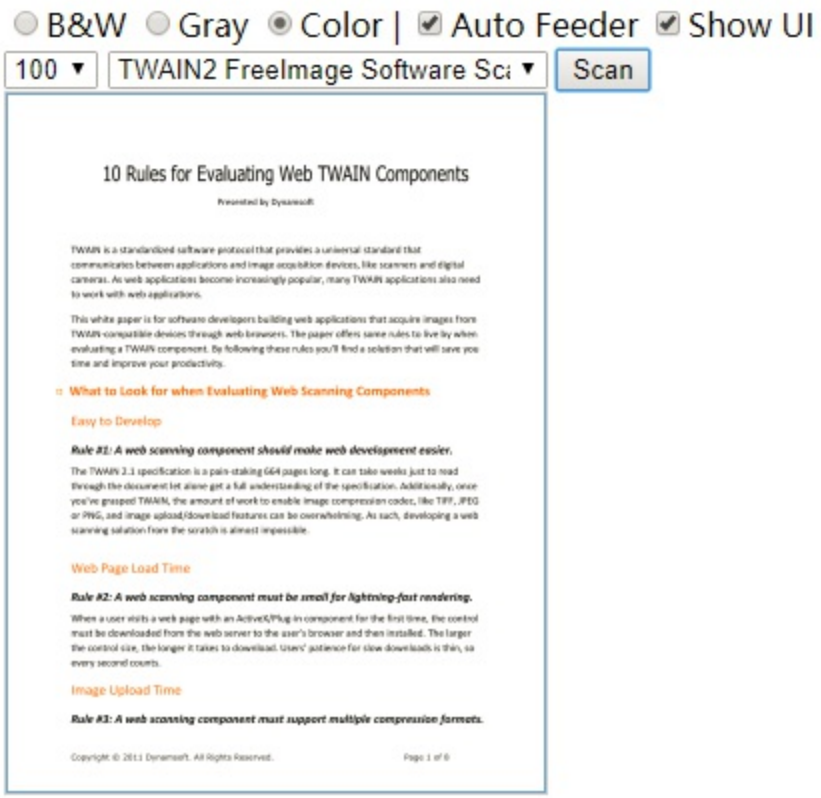
```
<context-param>
    <param-name>dynamsoft_upload</param-name>
    <param-value>D:\\uploadedimages\\</param-value>
</context-param>
```

This will then save the files in

`D:\uploadedimages\391008ba-aa9f-4564-a285-b44a42ec7864`

Step 5 Test the page meant for desktop

Open `http://localhost:8080/AcquireFromPCsAndMobileDevices/Scan/CustomScan.html` >. Acquire an image



Step 6 Add a button to upload

HTML

```
<input type="button" value="Upload" onclick="Upload();" />
```

JavaScript

```
function Upload() {
    if (DWObject) {
        var strFullActionPagePath = location.href.substr(0, location.href.lastIndexOf('/') + 1) + 'upload.jsp';
        DWObject.HTTPUpload(strFullActionPagePath, [DWObject.CurrentImageIndexInBuffer], EnumDWT_ImageType.IT_JPG, En
umDWT_UploadDataFormat.Binary, "test.jpg", function(){}, function(errCode, errString){ console.log(errString);});
    }
}
```

JSP

```
<%@  page language="java" import="java.io.*,java.util.*,org.apache.commons.fileupload.*,org.apache.commons.fileupload
.disk.*,org.apache.commons.fileupload.servlet.*"%><%!
%><%
// Create a factory for disk-based file items
DiskFileItemFactory factory = new DiskFileItemFactory();

// Configure a repository (to ensure a secure temp location is used)
ServletContext servletContext = this.getServletConfig().getServletContext();
File repository = (File) servletContext.getAttribute("javax.servlet.context.tempdir");
factory.setRepository(repository);
```

```
// Set factory constraints
factory.setSizeThreshold(1000000000);// Sets the size threshold beyond which files are written directly to disk.

// Create a new file upload handler
ServletFileUpload upload = new ServletFileUpload(factory);

// Set overall request size constraint
upload.setSizeMax(-1);

// Parse the request
List<FileItem> items = upload.parseRequest(request);

// Process the uploaded items
Iterator<FileItem> iter = items.iterator();
String _fields = "";
String fileName = "";
long sizeInBytes = 0;
String path = application.getRealPath(request.getRequestURI());
String dir = request.getServletContext().getInitParameter("dynamsoft_upload");
dir = dir.replace("\\\\","/");
String _temp_Name = dir + "files-uploaded-in-pc-browsers";
File _fieldsTXT = new File(_temp_Name);
if(!_fieldsTXT.exists())
{
    boolean result = _fieldsTXT.mkdirs();
    System.out.println("File create result:"+result);
}
while (iter.hasNext()) {
    FileItem item = iter.next();
    // Process a regular form field
    if (item.isFormField()) {
    }
    // Process a file upload
    else {
        String fieldName = item.getFieldName();
        fileName = item.getName();
        String contentType = item.getContentType();
        boolean isInMemory = item.isInMemory();
        sizeInBytes = item.getSize();
        if(fileName!=null && sizeInBytes!=0){
            File uploadedFile = new File(_temp_Name + "/" + fileName);
            if(!uploadedFile.exists())
            {
                boolean result = uploadedFile.createNewFile();
                System.out.println("File create result:"+result);
            }
            try {
                item.write(uploadedFile);
            }
            catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
%>
```

Step 7 Add code to detect the environment and redirect accordingly. Name the file `common.js` and place it under `WebContent/js/`

```
var dynamsoft = dynamsoft || {};
(function () {
```

```
var ua = navigator.userAgent.toLowerCase(),
    _platform = navigator.platform.toLowerCase(),
    _bWin = (_platform == 'win32') || (_platform == 'win64') || (_platform == 'windows'),
    _nMSIE = ua.indexOf('msie'),
    _nTrident = ua.indexOf('trident'),
    _nRV = ua.indexOf('rv:'),
    _nEdge = ua.indexOf('edge'),
    _tmp = ua.match(/version\/([\d.]+).*safari/),
    _bSafari = _tmp ? !0 : !1,
    _nSafari = _tmp ? _tmp[1] : 0,
    _nFirefox = ua.indexOf('firefox'),
    _bFirefox = (_nFirefox != -1),
    _bEdge = _bWin && !_bFirefox && (_nEdge != -1),
    _indexOfChrome = ua.indexOf('chrome'),
    _bChrome = !_bEdge && (_indexOfChrome != -1),
    _bIE = _bWin && !_bFirefox && !_bEdge && !_bChrome && (_nMSIE != -1 || _nTrident != -1 || _nRV != -1),
    _strBrowserVersion = '',
    _mainVer = 0;
var _deviceType,
    bIsIpad = ua.match(/ipad/i) == "ipad",
    bIsIphoneOs = ua.match(/iphone os/i) == "iphone os",
    bIsMidp = ua.match(/midp/i) == "midp",
    bIsUc7 = ua.match(/rv:1.2.3.4/i) == "rv:1.2.3.4",
    bIsUc = ua.match(/ucweb/i) == "ucweb",
    bIsAndroid = ua.match(/android/i) == "android",
    bIsCE = ua.match(/windows ce/i) == "windows ce",
    bIsWM = ua.match(/windows mobile/i) == "windows mobile";
if (bIsIpad || bIsIphoneOs || bIsMidp || bIsUc7 || bIsUc || bIsAndroid || bIsCE || bIsWM) {
    _deviceType = 'phone';
} else {
    _deviceType = 'pc';
}
if (_bEdge) {
    _tmp = ua.slice(_nEdge + 5);
    _tmp = _tmp.slice(0, _tmp.indexOf(' '));
    _strBrowserVersion = _tmp;
} else if (_bChrome) {
    _tmp = ua.slice(_indexOfChrome + 7);
    _tmp = _tmp.slice(0, _tmp.indexOf(' '));
    _strBrowserVersion = _tmp;
} else if (_bFirefox) {// FF
    _tmp = ua.slice(_nFirefox + 8);
    _tmp = _tmp.slice(0, _tmp.indexOf(' '));
    _strBrowserVersion = _tmp;
} else if (_bIE) {
    if (_nMSIE != -1) {
        // 'msie'
        _tmp = ua.slice(_nMSIE + 4);
        _tmp = _tmp.slice(0, _tmp.indexOf(';'));
        _strBrowserVersion = _tmp;
    } else if (_nRV != -1) {
        // 'rv:'
        _tmp = ua.slice(_nRV + 3);
        _tmp = _tmp.slice(0, _tmp.indexOf(';'));
        _tmp = _tmp.slice(0, _tmp.indexOf(')'));
        _strBrowserVersion = _tmp;
    } else if (_nTrident != -1) {
        // 'trident'
        _tmp = ua.slice(_nTrident + 7);
        _tmp = _tmp.slice(0, _tmp.indexOf(';'));
        _strBrowserVersion = _tmp;
    }
} else if (_bSafari) {
    if (_tmp) {
        _strBrowserVersion = _tmp[1];
    }
```

```
    }
    if (_strBrowserVersion.indexOf('.') > -1)
        _mainVer = _strBrowserVersion.slice(0, _strBrowserVersion.indexOf('.')) * 1.0;
    dynamsoft.onlineNavInfo = {
        bWin: _bWin,
        bIE: _bIE,
        bEdge: _bEdge,
        bFirefox: _bFirefox,
        bChrome: _bChrome,
        bSafari: _bSafari,
        strVersion: _strBrowserVersion,
        mainVer: _mainVer,
        deviceType: _deviceType
    };
})();
var strHREF = window.location.href;
if (dynamsoft.onlineNavInfo.deviceType == 'pc') {
    if (strHREF.indexOf('CustomScan') == -1)
        window.location.replace(strHREF.substr(0, strHREF.lastIndexOf('AcquireFromPCsAndMobileDevices') + 30) + '/Sca
n/CustomScan.html');
} else {
    if (strHREF.indexOf('MobileBrowserCapture') == -1)
        window.location.replace(strHREF.substr(0, strHREF.lastIndexOf('AcquireFromPCsAndMobileDevices') + 30) + '/Mob
ileBrowserCapture.html');
}
```

Step 8 Create a file `index.jsp` under `WebContent` with the following code

```
<%@ page session="false" pageEncoding="UTF-8" contentType="text/html; charset=UTF-8" %>
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <title>Capture Anywhere</title>
    </head>

    <body>
        <h1>Redirecting</h1>
    </body>
    <script type="text/javascript" src="js/common.js"></script>
</html>
```

Reference `common.js` in both `MobileBrowserCapture.html` and `CustomScan.html`

```
<script type="text/javascript" src="js/common.js"></script>
<script type="text/javascript" src="../js/common.js"></script>
```

Now when you navigate to `http://192.168.1.100:8080/AcquireFromPCsAndMobileDevices/`, you will land on the correct page meant for the browser being used.

Quickly implement text recognition in web applications

> ## Introduction
>
> In the process of document digitization, it is often necessary to extract the required information from the acquired images. `Optical Character Recognition (OCR)` is the technology used for this purpose. In this article, we explore how to quickly scan and recognize text in a browser with Dynamic Web TWAIN and its OCR Add-on.
>
> NOTE
>
> We only discuss the basic OCR engine in this article and we are using it on the client-side. The engine can also be used on the server side. Furthermore, Dynamsoft offers another engine called Professional OCR which is faster and more accurate and can also be used on both client-side and server-side. For more info, please contact us.

Environment

node

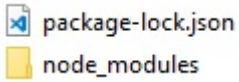
> The OCR module itself doesn't rely on `Node.js`, it's needed in this article just because it's faster to get required files with its package manager (npm).

Steps

Step 1 Create a new directory, open the command line tool inside (shortcut is `Ctrl+Shift+right click`). Download the core control used in this article through npm

```
npm install dwt@14.2.0
```

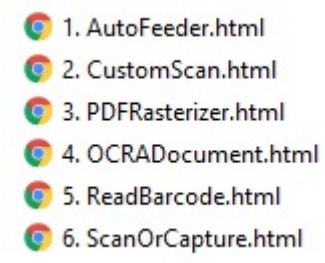
Then you can see the following in this directory



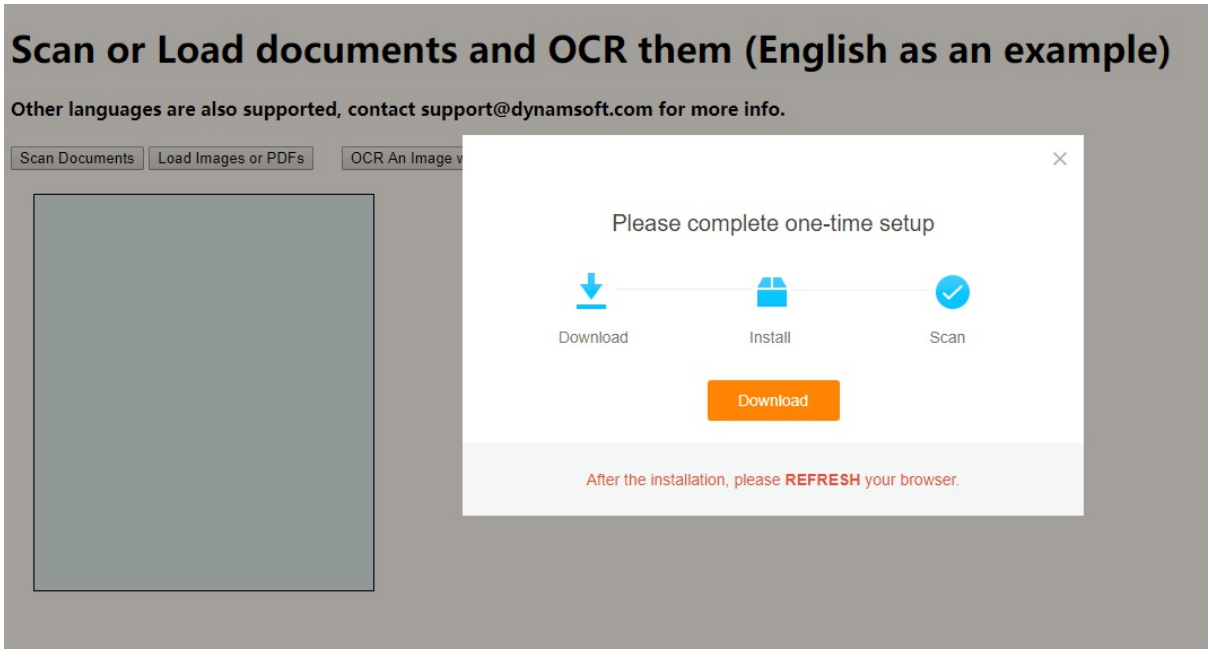
Step 2 Open to the following directory

`node_modules\dwt\sample`

where you can see



Step 3 In this article, we are going to check `OCRADocument.html`. Double click it to open. If the related controls are not yet available, follow the prompts to install them



Under normal circumstances, the installed files can be found in the `C:\Windows\SysWOW64\Dynamsoft\DynamsoftService` directory. The core files here are mainly

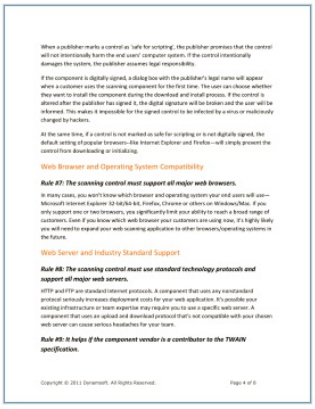
```
DynamsoftService.exe

dwt_trial_14.1.0.0828.dll

DynamicOCR.dll

/DynamicOCR/
```

Step 4 After the installation is complete, refresh the page, click `Scan Documents` (local need scanner) or `Load Images or PDFs` to scan or load local image files with English text. Then click on `OCR An Image with English`. The recognition result of the image will then show up in the result box on the right

Scan or Load documents and OCR them (English as an example)

Other languages are also supported, contact support@dynamsoft.com for more info.

Scan Documents | Load Images or PDFs | OCR An Image with English



When a publisher marks a control as 'safe for scripting' , the publisher promises that the control will not intentionally harm the end users' computer system. If the control intentionally damages the system, the publisher assumes legal responsibility.
If the component is digitally signed, a dialog box with the publisher's legal name will appear when a customer uses the scanning component for the first time. The user can choose whether they want to install the component during the download and install

How it is done

Open `OCRADocument.html` in a text editor

References to the Core JavaScript files

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

Here the files referenced are

> JS library for the core SDK `Dynamic Web TWAIN`
>
> `node_modules\dwt\dis\dynamsoft.webtwain.initiate.js` `node_modules\dwt\dis\dynamsoft.webtwain.config.js`
>
> JS library for the `Dynamsoft OCR Basic`
>
> `node_modules\dwt\dist\addon\dynamsoft.webtwain.addon.ocr.js`
>
> `PDF Rasterizer` is not necessary, check out PDF Rasterizer
>
> `node_modules\dwt\dist\addon\dynamsoft.webtwain.addon.pdf.js`

If you have previously installed the Dynamic Web TWAIN product locally, the same files (except `dynamsoft.webtwain.addon.pdf.js`) can also be found in the following directory.

`C:\Program Files (x86)\Dynamsoft\Dynamic Web TWAIN SDK {version number} {Trial}\Resource`

`Dynamsoft OCR Basic` runtime installation code

```
function downloadOCRBasic(bDownloadDLL) {
    var strOCRPath = Dynamsoft.WebTwainEnv.ResourcesPath + "/OCRResources/OCR.zip",
        strOCRLangPath = Dynamsoft.WebTwainEnv.ResourcesPath + '/OCRResources/OCRBasicLanguages/English.zip';

    if (bDownloadDLL) {
```

```
        DWObject.Addon.OCR.Download(
            strOCRPath,
            function () {/*console.log('OCR dll is installed');*/
                downloadOCRBasic(false);
            },
            function (errorCode, errorString) {
                console.log(errorString);
            }
        );
    } else {
        DWObject.Addon.OCR.DownloadLangData(
            strOCRLangPath,
            function () {
            }, function (errorCode, errorString) {
                console.log(errorString);
            });
    }
}
```

As shown in the above code, the `Dynamsoft OCR Basic` installation takes two steps. The first step is to install the core `DLL` (`DynamicOCR.dll` from `"/OCRResources/OCR.zip"`) with the `DWObject.Addon.OCR.Download` interface, The second step is to install the `OCR` language pack or the recognition dictionary (`'/OCRResources/OCRBasicLanguages/English.zip'`) with the `DWObject.Addon.OCR.DownloadLangData` interface. Only the `English` dictionary is installed here, so the program can only recognize English. If you need to identify other languages (27 main languages in total), you can download a complete example or refer to this online example

Scan Documents and Client-side OCR basic

List of supported languages

> Arabic, Bengali, Chinese_Simplified, Chinese_Traditional, English, French, German, Hindi, Indonesian, Italian, Japanese, Javanese, Korean, Malay, Marathi, Panjabi, Persian, Portuguese, Russian, Spanish, Swahili, Tamil, Telugu, Thai, Turkish, Vietnamese, Urdu.

Use the addon

```
function DoOCR() {
    if (DWObject) {
        if (DWObject.HowManyImagesInBuffer == 0) {
            alert("Please scan or load an image first.");
            return;
        }
        DWObject.Addon.OCR.SetLanguage('eng');
        DWObject.Addon.OCR.SetOutputFormat(EnumDWT_OCROutputFormat.OCROF_TEXT);
        DWObject.Addon.OCR.Recognize(
            DWObject.CurrentImageIndexInBuffer,
            function (sImageIndex, result) {
                if (result == null)
                    return null;
                var _textResult = (Dynamsoft.Lib.base64.decode(result.Get())).split(/\r?\n/g), _resultToShow = [];
                for (var i = 0; i < _textResult.length; i++) {
                    if (i == 0 && _textResult[i].trim() == "")
                        continue;
                    _resultToShow.push(_textResult[i] + '<br />');
                }
                _resultToShow.splice(0, 0, '<p style="padding:5px; margin:0;">');
                _resultToShow.push('</p>');
                document.getElementById('divNoteMessage').innerHTML = _resultToShow.join('');
            },
            function (errorcode, errorstring, result) {
                alert(errorstring);
```

```
            }
        );
    }
}
```

The core code is

```
DWObject.Addon.OCR.SetLanguage('eng'); //Set the language to be recognized
DWObject.Addon.OCR.SetOutputFormat(EnumDWT_OCROutputFormat.OCROF_TEXT); //Set the output format
DWObject.Addon.OCR.Recognize(... //Start Reconizing
```

Check out the supported output formats `EnumDWT_OCROutputFormat`.

Related methods are

`SetLanguage()`, `SetOutputFormat()`

`Recognize()`, `RecognizeFile()`, `RecognizeRect()`, `RecognizeSelectedImages()`

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)`

API Documentation

Basic Scan

Methods		
AcquireImage()	CloseSource()	DisableSource()
EnableSource()	OpenSource()	SelectSource()
SelectSourceByIndex()	SetOpenSourceTimeout()	

Properties		
BitDepth	IfAppendImage	IfDisableSourceAfterAcquire
IfDuplexEnabled	IfFeederEnabled	IfShowUI
ImageCaptureDriverType	PageSize	PixelType
Resolution	SourceCount	

Events	
OnPostAllTransfers	OnPostTransfer

Basic Edit

Methods		
AddText()	ChangeBitDepth()	ChangeImageSize()
ConvertToGrayScale()	CopyToClipboard()	CreateTextFont()
Crop()	CropToClipboard()	CutFrameToClipboard()
CutToClipboard()	Erase()	Flip()
Mirror()	MoveImage()	OverlayRectangle()
RemoveAllImages()	RemoveAllSelectedImages()	RemoveImage()
Rotate()	RotateEx()	RotateLeft()
RotateRight()	SetDPI()	SetImageWidth()
SetSelectedImageArea()	SetSelectedImageIndex()	ShowImageEditor()
SwitchImage()		

Properties
SelectionRectAspectRatio

Events	

OnImageAreaSelected	OnImageAreaDeSelected

UI & Display

Methods
SetViewMode()

Properties		
BackgroundColor	BackgroundFillColor	FitWindowType
Height	IfAutoScroll	IfFitWindow
ImageMargin	MaxImagesInBuffer	MouseShape
SelectionImageBorderColor	Width	Zoom
ShowPageNumber	MouseX	MouseY

Events	
OnTopImageInTheViewChanged	OnMouseClick
OnMouseDoubleClick	OnMouseMove
OnMouseRightClick	

Load & Save

Methods	
FileExists()	LoadDibFromClipboard()
LoadImage()	LoadImageEx()
LoadImageFromBase64Binary()	SaveAllAsMultiPageTIFF()
SaveAllAsPDF()	SaveAsBMP()
SaveAsJPEG()	SaveAsPDF()
SaveAsPNG()	SaveAsTIFF()
SaveSelectedImagesAsMultiPagePDF()	SaveSelectedImagesAsMultiPageTIFF()
SaveSelectedImagesToBase64Binary()	ShowFileDialog()

Properties
IfShowFileDialog

Events	
OnGetFilePath	OnPostLoad

Upload & Download

Methods	
ClearAllHTTPFormField()	FTPDownload()
FTPDownloadDirectly()	FTPDownloadEx()
FTPUpload()	FTPUploadAllAsMultiPageTIFF()
FTPUploadAllAsPDF()	FTPUploadAsMultiPagePDF()
FTPUploadAsMultiPageTIFF()	FTPUploadDirectly()
FTPUploadEx()	HTTPDownload()
HTTPDownloadDirectly()	HTTPDownloadEx()
HTTPUpload()	HTTPUploadAllThroughPostAsMultiPageTIFF()
HTTPUploadAllThroughPostAsPDF()	HTTPUploadThroughPost()
HTTPUploadThroughPostAsMultiPagePDF()	HTTPUploadThroughPostAsMultiPageTIFF()
HTTPUploadThroughPostDirectly()	HTTPUploadThroughPostEx()
SetHTTPFormField()	SetUploadSegment()
SetHTTPHeader()	

Properties	
FTPPassword	FTPPort
FTPUserName	HttpFieldNameOfUploadedImage
HTTPPort	HTTPPostResponseString
IfPASVMode	IfShowCancelDialogWhenImageTransfer
IfSSL	MaxUploadImageSize

Events
OnInternetTransferPercentage

Advanced Scan

Methods		
CancelAllPendingTransfers()	CloseSourceManager()	CloseWorkingProcess()
FeedPage()	GetCustomDSData()	GetCustomDSDataEx()
GetDeviceType()	GetSourceNameItems()	OpenSourceManager()
ResetImageLayout()	RewindPage()	SetCustomDSData()
SetCustomDSDataEx()	SetFileXferInfo()	SetImageLayout()

Properties

Brightness	Contrast	CurrentSourceName
DataSourceStatus	DefaultSourceName	Duplex
IfAutoBright	IfAutoDiscardBlankpages	IfAutoFeed
IfAutomaticBorderDetection	IfAutomaticDeskew	IfAutoScan
IfFeederLoaded	IfPaperDetectable	IfShowIndicator
IfUIControllable	IfUseTwainDSM	PendingXfers
PixelFlavor	TransferMode	Unit
XferCount		

Events		
OnPreAllTransfers	OnPreTransfer	OnSourceUIClose

Capability Negotiation

Methods		
CapGet()	CapGetCurrent()	CapGetDefault()
CapGetFrameBottom()	CapGetFrameLeft()	CapGetFrameRight()
CapGetFrameTop()	CapGetHelp()	CapGetLabel()
CapGetLabels()	CapIfSupported()	CapReset()
CapSet()	CapSetFrame()	GetCapItems()
GetCapItemsString()	SetCapItems()	SetCapItemsString()

Properties		
Capability	CapCurrentIndex	CapCurrentValue
CapDefaultIndex	CapDefaultValue	CapMaxValue
CapMinValue	CapNumItems	CapStepSize
CapType	CapValue	CapValueString
CapValueType		

Encode & Decode

Methods		
ClearTiffCustomTag()	ConvertToBase64()	ConvertToBlob()
SetTiffCustomTag()		

Properties		
IfOpenImageWithGDIPlus	IfTiffMultiPage	JPEGQuality/a>

PDFAuthor	PDFCompressionType	PDFCreationDate
PDFCreator	PDFKeywords	PDFModifiedDate
PDFProducer	PDFSubject	PDFTitle
PDFVersion	TIFFCompressionType	

Runtime Info

Methods		
GetImageBitDepth()	GetImageHeight()	GetImageSize()
GetImageSizeWithSpecifiedType()	GetImageWidth()	GetImageXResolution()
GetImageYResolution()	GetSelectedImageIndex()	GetSelectedImagesSize()
GetSkewAngle()	GetSkewAngleEx()	IsBlankImageExpress()

Properties		
BlankImageCurrentStdDev	BlankImageMaxStdDev	BlankImageThreshold
CurrentImageIndexInBuffer	HowManyImagesInBuffer	ImageLayoutDocumentNumber
ImageLayoutFrameBottom	ImageLayoutFrameLeft	ImageLayoutFrameNumber
ImageLayoutFrameRight	ImageLayoutFrameTop	ImageLayoutPageNumber
ImagePixelType	MagData	MagType
SelectedImagesCount		

General Utilities

Methods		
GetImagePartURL()	GetImageURL()	Print()
RegisterEvent()	UnregisterEvent()	SetLanguage()

Properties		
BufferMemoryLimit	ErrorCode	ErrorString
IfAllowLocalCache	IfShowProgressBar	LogLevel
Manufacturer	ProductFamily	ProductKey
ProductName	VersionInfo	

Events
OnBitmapChanged

PDF Rasterizer

Methods	
Addon.PDF.Download()	Addon.PDF.SetResolution()
Addon.PDF.SetPassword()	Addon.PDF.SetConvertMode()

Webcam Capture

Methods	
Addon.Webcam.CaptureImage()	Addon.Webcam.CloseSource()
Addon.Webcam.Download()	Addon.Webcam.GetCameraControlPropertyMoreSet
Addon.Webcam.GetCameraControlPropertySetting()	Addon.Webcam.GetFrameRate()
Addon.Webcam.GetImagePartUrl()	Addon.Webcam.GetImageUrl()
Addon.Webcam.GetMediaType()	Addon.Webcam.GetResolution()
Addon.Webcam.GetSourceList()	Addon.Webcam.GetVideoPropertyMoreSetting()
Addon.Webcam.GetVideoPropertySetting()	Addon.Webcam.PauseVideo()
Addon.Webcam.PlayVideo()	Addon.Webcam.SelectSource()
Addon.Webcam.SetCameraControlPropertySetting()	Addon.Webcam.SetFrameRate()
Addon.Webcam.SetMediaType()	Addon.Webcam.SetResolution()
Addon.Webcam.SetVideoPropertySetting()	Addon.Webcam.SetVideoRotateMode()
Addon.Webcam.StopVideo()	

File Uploader

Methods		
Init()	CreateJob()	Run()
Cancel()	CancelAllUpload()	GenerateURLForUploadData()

Properties		
ServerUrl	HttpHeader	SourceValue

Events		
OnUploadTransferPercentage	OnRunSuccess	OnRunFailure

Barcode Reader

Constructor
dynamsoft.BarcodeReader()

Methods		
decode()	decodeBase64String()	getRuntimeSettings()
updateRuntimeSettings()	resetRuntimeSettings()	
dynamsoft.BarcodeReader.initServiceConnection()		

Properties	
productKey	bAutoConnectService
resourcesPath	ifCheck64bitServiceFirst

Events	
onAutoConnectServiceSuccess	onAutoConnectServiceError

Enumerations

Errors

OCR Basic

Client-Side

Methods		
Download()	DownloadLangData()	GetIfUseDetectedFont()
GetMinFontSizeforMoreAccurateResult()	GetUnicodeFontName()	IsModuleInstalled()
Recognize()	RecognizeFile()	RecognizeRect()
RecognizeSelectedImages	SetIfUseDetectedFont()	SetLanguage()
SetMinFontSizeforMoreAccurateResult()	SetOutputFormat()	SetPageSetMode()
SetUnicodeFontName()		

OCR Basic Result Object		
Result	PageSetResult	PageResult
LineResult	WordResult	

Server-Side (Java)

OCR Pro

Client-Side

Methods		
Download()	IsModuleInstalled()	Recognize()
RecognizeFile()	RecognizeRect()	RecognizeSelectedImages

Properties
Settings

OCR Pro Result Object	
OCRResult	PageResult
LetterResult	ErrorInfo

Server-Side

OCRPro.ServerSide.Request

OCRPro.ServerSide.Response

Basic Scan

Methods		
AcquireImage()	CloseSource()	DisableSource()
EnableSource()	OpenSource()	SelectSource()
SelectSourceByIndex()	SetOpenSourceTimeout()	

Properties		
BitDepth	IfAppendImage	IfDisableSourceAfterAcquire
IfDuplexEnabled	IfFeederEnabled	IfShowUI
ImageCaptureDriverType	PageSize	PixelType
Resolution	SourceCount	

Events	
OnPostAllTransfers	OnPostTransfer

Code example

The following code example demonstrates how to use the APIs above to perform basic scanning.

```
function BasicScan() {
    if (DWObject) {
        if (Dynamsoft.Lib.env.bMac) {
            DWObject.ImageCaptureDriverType = EnumDWT_Driver.TWAIN_AND_ICA;
        }
        DWObject.RegisterEvent('OnPostTransfer', function () {
            console.log('One page scanned!');
        });
        DWObject.RegisterEvent('OnPostAllTransfers', function () {
            DWObject.CloseSource();
        });
        DWObject.SelectSource(function () {
            DWObject.SetOpenSourceTimeout(3000);
            DWObject.OpenSource();
            DWObject.IfShowUI = false;
            DWObject.IfDisableSourceAfterAcquire = true;
            DWObject.PixelType = EnumDWT_PixelType.TWPT_RGB;
            DWObject.BitDepth = 24;
            DWObject.PageSize = EnumDWT_CapSupportedSizes.TWSS_A4;
            DWObject.Resolution = 300;
            DWObject.IfFeederEnabled = true;
            DWObject.IfDuplexEnabled = true;
            DWObject.IfAppendImage = false;
            DWObject.AcquireImage(function () {
                    /**
                    * Actually not necessary when you have
                    * IfDisableSourceAfterAcquire set to true.
                    */
                    DWObject.DisableSource();
                },
```

```
                function () {
                        DWObject.DisableSource();
                }
            );
        }, function () {
            console.log('Failed to select a source, there are ' +
                DWObject.SourceCount + ' sources in total.');
        });
    }
}
```

Methods

AcquireImage()	
Starts image acquisition.	
Syntax	.AcquireImage([optionalDeviceConfig], [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`optional` `optionalDeviceConfig` : A JS object used to set up the device for image acquisition. `<OnSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when acquiring succeeds. `<OnFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when acquiring fails.
Return value	`boolean`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v1.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes: The parameters are only available in the HTML5 Edition. As the parameters are all optional, there are 4 ways to use this method as demonstrated below.

Example

```
var DeviceConfig = {
    IfShowUI: false,
    PixelType: EnumDWT_PixelType.TWPT_RGB,
    Resolution: 300,
    IfFeederEnabled: true,
    IfDuplexEnabled: false,
    IfDisableSourceAfterAcquire: true
};

function AsyncSuccessFunc() {
    console.log('successful');
}

function AsyncFailureFunc(errorCode, errorString) {
    alert(errorString);
}

function AcquireImage1() {
    DWObject.SelectSource();
    DWObject.OpenSource();
    DWObject.IfShowUI = false;
    DWObject.PixelType = EnumDWT_PixelType.TWPT_RGB;
    DWObject.Resolution = 300;
    DWObject.IfFeederEnabled = true;
    DWObject.IfDuplexEnabled = false;
    DWObject.IfDisableSourceAfterAcquire = true;
    DWObject.AcquireImage();
}

function AcquireImage2() {
    DWObject.SelectSource();
```

```
    DWObject.OpenSource();
    DWObject.AcquireImage(DeviceConfig);
}

function AcquireImage3() {
    DWObject.SelectSource();
    DWObject.OpenSource();
    DWObject.IfShowUI = false;
    DWObject.PixelType = EnumDWT_PixelType.TWPT_RGB;
    DWObject.Resolution = 300;
    DWObject.IfFeederEnabled = true;
    DWObject.IfDuplexEnabled = false;
    DWObject.IfDisableSourceAfterAcquire = true;
    DWObject.AcquireImage(AsyncSuccessFunc, AsyncFailureFunc);
}

function AcquireImage4() {
    DWObject.SelectSource();
    DWObject.OpenSource();
    DWObject.AcquireImage(DeviceConfig, AsyncSuccessFunc, AsyncFailureFunc);
}
```

CloseSource()

Closes the Data Source (a TWAIN/ICA/SANE device which in most cases is a scanner) to free it to be used by other applications.

Syntax	.CloseSource();
Parameters	`None`
Return value	`boolean`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v1.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes: In version 10.0 ~ 11.2, `CloseSource()` is called automatically after all pages have been scanned This method can be called after `SelectSource()` and before `OpenSource()` in order to close any existing connections.

Example

```
DWObject.RegisterEvent('OnPostAllTransfers', function () {
    DWObject.CloseSource();
});
```

DisableSource()

Disables the Data Source (a TWAIN/ICA/SANE device which in most cases is a scanner) to stop the acquiring process. If the source's user interface is displayed, it will be closed.

Syntax	.DisableSource();
Parameters	`None`
Return value	`boolean`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v1.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	After `DisableSource()` is called, the Source is still open and you can continue to acquire images by calling `AcquireImage()` or `EnableSource()` .
Example	`DWObject.RegisterEvent('OnPostAllTransfers', function () {` `  DWObject.DisableSource();` `});`

EnableSource()

Enables the source to start the acquiring process.

Syntax	.EnableSource();
Parameters	`None`
Return value	`boolean`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v1.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	The method is equivalent to `AcquireImage()` without parameters.
Example	`function AcquireImage4() {` `  DWObject.SelectSource();` `  DWObject.OpenSource();` `  DWObject.EnableSource();` `}`

OpenSource()

Loads the currently selected Data Source into memory and initializes it for image acquisition. If no source is specified (neither `SelectSource()` nor `SelectSourceByIndex()` has been called), the default source will be loaded.

Syntax	.OpenSource();
Parameters	`None`
Return value	`boolean`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v1.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	Call OpenSource() before you negotiate any capabilities.
Example	`DWObject.SelectSource();` `DWObject.OpenSource();` `DWObject.IfShowUI = false;` `DWObject.Resolution = 300;` `DWObject.AcquireImage();`

SelectSource()

Brings up the Source Selection User Interface (UI) for the user to choose a Data Source.

Syntax	.SelectSource([optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`<OnSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when selecting succeeds. `<OnFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when selecting fails.
Return value	`boolean` , only when used synchronously (without parameters).
Availability	**ActiveX**: ✓ | v1.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v14.0
Usage notes	The optional parameters which make the method asynchronous were added in v14.0 of the HTML5 edition and in v14.1 of the ActiveX edition.
Example	

```
DWObject.SelectSource();
DWObject.OpenSource();
DWObject.IfShowUI = false;
DWObject.Resolution = 300;
DWObject.AcquireImage();
```

```
DWObject.SelectSource(function () {
    DWObject.OpenSource();
    DWObject.IfShowUI = false;
    DWObject.Resolution = 300;
    DWObject.AcquireImage();
}, function () {
    console.log('Failed to select a source, there are ' +
        DWObject.SourceCount + ' sources in total.');
});
```

SelectSourceByIndex()

Selects a Data Source by its index in the Data Source Manager.

Syntax	.SelectSourceByIndex(Number index);
Parameters	`number` `index` : The index of the targeted Data Source in the Data Source Manager.
Return value	`boolean`
Availability	**ActiveX**: ✓ | v1.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1
Usage notes	Use this method when you don't want to show the Data Source Selecting UI.
Example	

```
DWObject.SelectSourceByIndex(0);
DWObject.IfShowUI = false;
DWObject.Resolution = 300;
DWObject.AcquireImage();
```

SetOpenSourceTimeout()

Sets a timer which stops the source opening process once it expires.

Syntax	.SetOpenSourceTimeout(nMilliseconds);
Parameters	`number` `nMilliseconds` : The time, in milliseconds (thousandths of a second).
Return value	`boolean`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v11.0	✓ | v11.0	✓ | v11.0	✓ | v11.0	✗

Usage notes	`SetOpenSourceTimeout()` should be called before `OpenSource()` .
Example	

```
DWObject.SelectSource(function () {
DWObject.SetOpenSourceTimeout(3000);
DWObject.OpenSource();
DWObject.AcquireImage();
```

Properties

BitDepth

Returns or sets the pixel bit depth for the current value of `PixelType` .

Type	`number`
Accessors	`Get Set`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v3.0	✓ | v10.0	✓ | v11.0	✗	✗

Usage notes	Set this property after `OpenSource()` and before `AcquireImage()` . Set BitDepth based on the current value of `PixelType` . By default, the bit depth is 1 for 'TWPT_BW'(0), 8 for 'TWPT_GRAY'(1) and 24 for 'TWPT_RGB'(2).

IfAppendImage

Returns or sets whether newly acquired images are inserted or appended.

Type	`boolean`
Accessors	`Get Set`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.1	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	If the value of the property is set to `true`, the newly acquired images will be appended after the last image in buffer. If it's set to `false`, the images will be inserted before the current image.

IfDisableSourceAfterAcquire

Returns or sets whether to close the user interface after all images have been acquired.

Type	`boolean`
Accessors	`Get Set`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v3.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	Set this property after `OpenSource()` and before `AcquireImage()`. This property only makes sense when `IfShowUI` is set to `true`.

IfDuplexEnabled

Returns or sets whether to enable duplex scanning (in other words, whether to scan both sides of the paper).

Type	`boolean`
Accessors	`Get Set`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v3.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	Set this property after `OpenSource()` and before `AcquireImage()`. Not all scanners support duplex scanning. To confirm, check the user manual of the device or check the value of `Duplex` after `OpenSource()`.

IfFeederEnabled

Returns or sets whether a Data Source's Automatic Document Feeder (ADF) is enabled for scanning.

Type	`boolean`
Accessors	`Get Set`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v3.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	Set this property after `OpenSource()` and before `AcquireImage()`. If the property is set to `true`, the Data Source will try acquiring images from the document feeder first. If the Data Source doesn't have a document feeder, the flatbed will be used.

IfShowUI

Returns or sets whether the source displays the user interface when scanning.

Type	`boolean`

Accessors	`Get` `Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v3.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✓ | v11.0</td><td>✗</td></tr></table>
Usage notes	If the property is set to `true`, the Data Source will display its user interface when `AcquireImage()` is called. Otherwise, the UI will not be displayed and scanning will begin immediately.

ImageCaptureDriverType

Returns or sets whether to use TWAIN or ICA protocol on macOS.

Type	`number`
Accessors	`Get` `Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✗</td><td>✗</td><td>✓ | v11.0</td><td>✓ | v11.0</td><td>✗</td></tr></table>
Usage notes	Set this property right after the SDK is initialized or after calling `CloseSourceManager()` and `OpenSourceManager()`. This property only works on macOS. Allowed values are `EnumDWT_Driver.TWAIN: 0` `EnumDWT_Driver.ICA: 3` `EnumDWT_Driver.TWAIN_AND_ICA: 4` (added in v14.0, this is the default value since v14.0) When the property is set to `EnumDWT_Driver.TWAIN` or 0, only devices with TWAIN drivers can be used. When set to `EnumDWT_Driver.ICA` or 3, only devices with ICA drivers can be used which are typically listed here. When set to `EnumDWT_Driver.TWAIN_AND_ICA` or 4, both TWAIN and ICA devices can be used. Dynamsoft recommends the use of ICA drivers on macOS for the following reasons: 1. 1. Usually there is no need to install a driver for your scanner anymore, you can plug and play 2. 2. Generally, ICA drivers - which come with the Mac OS - works better

PageSize

Returns or sets the page size the source uses to acquire images.

Type	`EnumDWT_CapSupportedSizes`
Accessors	`Get` `Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v3.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✓ | v11.0</td><td>✗</td></tr></table>
Usage notes	Set this property after `OpenSource()` and before `AcquireImage()`.

PixelType

Returns or sets the pixel type used when acquiring images.

Type	`EnumDWT_PixelType`
Accessors	`Get Set`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v3.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	Set this property after `OpenSource()` and before `AcquireImage()`.

Resolution

Returns or sets the pixel type used when acquiring images.

Type	`number`
Accessors	`Get Set`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v3.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	Set this property after `OpenSource()` and before `AcquireImage()`.

SourceCount

Returns how many Data Sources are available on the local system.

Type	`number`
Accessors	`Get`
Usage notes	If `SourceCount` returns 0, it means there is no source available to use on the system.
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v1.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	If `SourceCount` returns 0, it means there is no source available to use on the system.

Events

OnPostAllTransfers

This event is triggered when all page(s) have been scanned and transferred.

Syntax	.RegisterEvent('OnPostAllTransfers',function(){...});

Arguments	• None
Example	`DWObject.RegisterEvent('OnPostAllTransfers', function() {` `  console.log(DWObject.HowManyImagesInBuffer);` `});`
Availability	ActiveX: ✓ | v2.0 / H5(Windows): ✓ | v10.0 / H5(macOS/TWAIN): ✓ | v11.0 / H5(macOS/ICA): ✓ | v11.0 / H5(Linux): ✓ | v12.1
Usage notes	This event fires after all pages in the document feeder have been scanned. This is a good place to upload the images, detect barcodes, discard blank pages, etc.

OnPostTransfer	
This event is triggered after each page has been scanned and transferred.	
Syntax	.RegisterEvent('OnPostTransfer',function(nImageIndex){...});
Arguments	• `number nImageIndex` : The index of the transferred image.
Example	`DWObject.RegisterEvent('OnPostTransfer', function(nImageIndex) {` `  console.log(nImageIndex);` `});`
Availability	ActiveX: ✓ | v1.0 / H5(Windows): ✓ | v10.0 / H5(macOS/TWAIN): ✓ | v11.0 / H5(macOS/ICA): ✓ | v11.0 / H5(Linux): ✓ | v12.1

Basic Edit

Methods		
AddText()	ChangeBitDepth()	ChangeImageSize()
ConvertToGrayScale()	CopyToClipboard()	CreateTextFont()
Crop()	CropToClipboard()	CutFrameToClipboard()
CutToClipboard()	Erase()	Flip()
Mirror()	MoveImage()	OverlayRectangle()
RemoveAllImages()	RemoveAllSelectedImages()	RemoveImage()
Rotate()	RotateEx()	RotateLeft()
RotateRight()	SetDPI()	SetImageWidth()
SetSelectedImageArea()	SetSelectedImageIndex()	ShowImageEditor()
SwitchImage()		

Properties
SelectionRectAspectRatio

Events	
OnImageAreaSelected	OnImageAreaDeSelected

Methods

AddText()	
Adds text on an image.	
Syntax	.AddText(nImageIndex, x, y, text, textColor, backgroundColor, backgroundRoundRadius, backgroundOpacity);
Parameters	`number` `nImageIndex` : Specifies the index of image in buffer. The index is 0-based. `number` `x` : Specifies the x-coordinate of the upper-left corner of the text. `number` `y` : Specifies the y-coordinate of the upper-left corner of the text. `string` `text` : Specifies the content of the text. `number` `textColor` : Specifies the color for the text. `number` `backgroundColor` :Specifies the background color. It is a value Specifying the 24-bit RGB value. Default is white (0xffffff). The byte-ordering of the 24-bit RGB value is RRGGBB. RR represents red, GG represents green and BB represents blue. `number` `backgroundRoundRadius` :Specifies the background round radius ranging from 0 to 0.5. `number` `backgroundOpacity` :Specifies the opacity of the color. 1.0 is 100% opaque and 0.0 is totally transparent.
Return value	`boolean`
Example	`DWObject.CreateTextFont(50, 30, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, "Arial");` `DWObject.AddText(0, 250, 600, 'Dynamic Web TWAIN', 0x0000ff, 0xff0000, 0.5, 0.5);`

Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v7.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✓ | v11.0</td><td>✗</td></tr></table>

ChangeBitDepth()	
Changes the bitdepth of a specified image. By doing this, you can change the pixel type of the image.	
Syntax	.ChangeBitDepth(nImageIndex, sBitDepth, bHighQuality);
Parameters	`number` `nImageIndex` : Specifies the index of the image to be converted. The index is 0-based. `number` `sBitDepth` : Specifies the target bit depth. `boolean` `bHighQuality` : Specifies whether or not to keep high quality while changing the bit depth. When it's true, it takes more time.
Return value	`boolean`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v10.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✓ | v11.0</td><td>✗</td></tr></table>
Usage notes	The allowed bit depths are 1, 4, 8, 24.

ChangeImageSize()	
Changes the size of an image specified by index.	
Syntax	.ChangeImageSize(nImageIndex, iNewwidth, iNewheight, newInterpolationMethod);
Parameters	`number` `nImageIndex` : Specifies the index of the image to be converted. The index is 0-based. `number` `iNewWidth` : Specifies the new width (in pixels) `number` `iNewHeight` : Specifies the new height (in pixels) `EnumDWT_InterpolationMethod` `newInterpolationMethod` : Specifies the algorithm used to do interpolation. <table><tr><th>Allowed Values</th><th>Interpolation Method</th></tr><tr><td>1</td><td>EnumDWT_InterpolationMethod.IM_NEARESTNEIGHBOUR</td></tr><tr><td>2</td><td>EnumDWT_InterpolationMethod.IM_BILINEAR</td></tr><tr><td>3</td><td>EnumDWT_InterpolationMethod.IM_BICUBIC</td></tr><tr><td>5</td><td>EnumDWT_InterpolationMethod.IM_BESTQUALITY</td></tr></table>
Return value	`boolean`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v4.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✓ | v11.0</td><td>✓ | v12.1</td></tr></table>

ConvertToGrayScale()

Converts a specified image to gray scale.

Syntax	.ConvertToGrayScale(nImageIndex);
Parameters	`number` `nImageIndex` : Specifies the index of the image to be converted. The index is 0-based.
Return value	`boolean`
Availability	**ActiveX**: ✓ | v10.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1
Usage notes	Typically, it only makes sense to convert a colored image to grey scale.

CopyToClipboard()

Specifies the index of image in buffer. The index is 0-based.

Syntax	.CopyToClipboard(nImageIndex);
Parameters	`number` `nImageIndex` : Specifies the index of image in buffer. The index is 0-based.
Return value	`boolean`
Availability	**ActiveX**: ✓ | v4.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1
Usage notes	This method makes a copy of the image to the clipboard. The copied image is in DIB format.

CreateTextFont()

Creates the font for adding text on an image using the method AddText().

Syntax	.CreateTextFont(height, width, escapement, orientation, weight, italic, underline, strikeOut, charSet, outputPrecision, clipPrecision, quality, pitchAndFamily, faceName);
Parameters	`number` `height` :Specifies the desired height of the font. `number` `width` : Specifies the average width of characters in the font. `number` `escapement` : Specifies the angle between the escapement vector and the x-axis of the display surface. The escapement vector is the line through the origins of the first and last characters on a line. The angle is measured counterclockwise from the x-axis. `number` `orientation` : Specifies the angle between the baseline of a character and the x-axis. The angle is measured counterclockwise from the x-axis for coordinate systems in which the y direction is down and clockwise from the x-axis for coordinate systems in which the y-direction is up. `number` `weight` : Specifies the font weight. `number` `italic` : Specifies an italic font if the value is not 0. `number` `underline` : Specifies an underlined font if the value is not 0. `number` `strikeOut` : Specifies an strikeout font if the value is not 0. `number` `charSet` : Specifies the font's character set. `number` `outputPrecision` : Specifies the desired output precision. The output precision defines how closely the output must match the requested font's height, width, character orientation, escapement, and pitch. `number` `clipPrecision` : Specifies the desired clipping precision. The clipping precision defines

	how to clip characters that are partially outside of the clipping region. `number` `quality` : Specifies the font's output quality which defines how carefully the GDI should attempt to match the logical-font attributes to those of an actual physical font. `number` `pitchAndFamily` : Specifies the pitch and family of the font. `string` `faceName` : Specifies the typeface name, the length of this string must not exceed 32 characters, including the terminating null character.
Return value	`boolean`
Example	`DWObject.CreateTextFont(50, 30, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, "Arial");` `DWObject.AddText(0, 250, 600, 'Dynamic Web TWAIN', 0x0000ff, 0xff0000, 0.5, 0.5);`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v7.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✗

Crop()

Crops an image of the specified index in buffer.

Syntax	.Crop(nImageIndex, left, top, right, bottom);
Parameters	`number` `nImageIndex` : Specifies the index of image in buffer. The index is 0-based. `number` `left` : Specifies the x-coordinate of the upper-left corner of the rectangle. The unit is pixel. `number` `top` : Specifies the y-coordinate of the upper-left corner of the rectangle. The unit is pixel. `number` `right` : Specifies the x-coordinate of the lower-right corner of the rectangle. The unit is pixel. `number` `bottom` : Specifies the y-coordinate of the lower-right corner of the rectangle. The unit is pixel.
Return value	`boolean`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v6.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	This method crops an image and replaces the original image with the cropped copy. If the application still needs the original image, use CropToClipboard() instead.

CropToClipboard()

Crops the image of a specified index in buffer to clipboard in DIB format.

Syntax	.CropToClipboard(nImageIndex, left, top, right, bottom);
Parameters	`number` `nImageIndex` : Specifies the index of image in buffer. The index is 0-based. `number` `left` : Specifies the x-coordinate of the upper-left corner of the rectangle. The unit is pixel. `number` `top` : Specifies the y-coordinate of the upper-left corner of the rectangle. The unit is pixel. `number` `right` : Specifies the x-coordinate of the lower-right corner of the rectangle. The unit is pixel. `number` `bottom` : Specifies the y-coordinate of the lower-right corner of the rectangle. The unit is pixel.

Return value	`boolean`
Availability	**ActiveX**: ✓ | v4.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1
Usage notes	This method crops an image and makes a copy of the cropped image to the clipboard.

CutFrameToClipboard()

Cuts the image data in the specified area to the system clipboard in DIB format.

Syntax	.CutFrameToClipboard(nImageIndex, left, top, right, bottom);
Parameters	`number` `nImageIndex` : Specifies the index of image in buffer. The index is 0-based. `number` `left` : Specifies the x-coordinate of the upper-left corner of the rectangle. The unit is pixel. `number` `top` : Specifies the y-coordinate of the upper-left corner of the rectangle. The unit is pixel. `number` `right` : Specifies the x-coordinate of the lower-right corner of the rectangle. The unit is pixel. `number` `bottom` : Specifies the y-coordinate of the lower-right corner of the rectangle. The unit is pixel.
Return value	`boolean`
Availability	**ActiveX**: ✓ | v6.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1
Usage notes	Use `BackgroundFillColor` to specify the fill color for the cut frame. Currently this method is only valid in Windows & Mac.

CutToClipboard()

Copies the image of a specified index in buffer to clipboard in DIB format.

Syntax	.CopyToClipboard(nImageIndex);
Parameters	`number` `nImageIndex` : Specifies the index of image in buffer. The index is 0-based.
Return value	`boolean`
Availability	**ActiveX**: ✓ | v4.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1

Erase()

Clears the specified area of a specified image and fill the area with the fill color.

Syntax	.Erase(nImageIndex, left, top, right, bottom);
Parameters	`number` `nImageIndex` : Specifies the index of image in buffer. The index is 0-based. `number` `left` : Specifies the x-coordinate of the upper-left corner of the rectangle. The unit is pixel. `number` `top` : Specifies the y-coordinate of the upper-left corner of the rectangle. The unit is pixel. `number` `right` : Specifies the x-coordinate of the lower-right corner of the rectangle. The unit is pixel. `number` `bottom` : Specifies the y-coordinate of the lower-right corner of the rectangle. The unit is pixel.
Return value	`boolean`
Availability	**ActiveX**: ✓ | v6.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1
Usage notes	The fill color is set by `BackgroundFillColor` .

Flip()

Flips the image of a specified index in buffer.

Syntax	.Flip(nImageIndex);
Parameters	`number` `nImageIndex` : Specifies the index of image in buffer. The index is 0-based.
Return value	`boolean`
Availability	**ActiveX**: ✓ | v6.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1

Mirror()

Mirrors the image of a specified index in buffer.

Syntax	.Mirror(nImageIndex);
Parameters	`number` `nImageIndex` : Specifies the index of image in buffer. The index is 0-based.
Return value	`boolean`
Availability	**ActiveX**: ✓ | v4.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1

MoveImage()

Moves a specified image.

Syntax	.MoveImage(nImageIndex, nTargetImageIndex);

Parameters	`number` `nImageIndex` : Specifies the index of the image to be converted. The index is 0-based. `number` `nTargetImageIndex` : Specifies the new index.
Return value	`boolean`
Availability	**ActiveX**: ✓ | v4.0 · **H5(Windows)**: ✓ | v10.0 · **H5(macOS/TWAIN)**: ✓ | v11.0 · **H5(macOS/ICA)**: ✓ | v11.0 · **H5(Linux)**: ✓ | v12.1

OverlayRectangle()

Decorates an image specified by index. The decoration is in the form of rectangles filled with transparent color.

Syntax	.OverlayRectangle(nImageIndex, left, top, right, bottom, color, nOpacity);
Parameters	`number` `nImageIndex` : Specifies the index of image in buffer. The index is 0-based. `number` `left` : Specifies the x-coordinate of the upper-left corner of the rectangle. `number` `top` : Specifies the y-coordinate of the upper-left corner of the rectangle. `number` `right` : Specifies the x-coordinate of the lower-right corner of the rectangle. `number` `bottom` : Specifies the y-coordinate of the lower-right corner of the rectangle. `number` `color` : Specifies the color. It is a value Specifying the 24-bit RGB value. Default is white (0xffffff). The byte-ordering of the 24-bit RGB value is RRGGBB. RR represent red, GG represents green and BB represents blue. `number` `nOpacity` : Specifies the opacity of the color. 1.0 is 100% opaque and 0.0 is totally transparent.
Return value	`boolean`
Example	`DWObject.OverlayRectangle(0, 50, 50, 300, 300, 0xff0000, 0.5);`
Availability	**ActiveX**: ✓ | v4.0 · **H5(Windows)**: ✓ | v10.1 · **H5(macOS/TWAIN)**: ✓ | v11.0 · **H5(macOS/ICA)**: ✓ | v11.0 · **H5(Linux)**: ✓ | v12.1

RemoveAllImages()

Removes all images in buffer.

Syntax	.RemoveAllImages();
Parameters	`None`
Return value	`boolean`
Availability	**ActiveX**: ✓ | v4.0 · **H5(Windows)**: ✓ | v10.0 · **H5(macOS/TWAIN)**: ✓ | v11.0 · **H5(macOS/ICA)**: ✓ | v11.0 · **H5(Linux)**: ✓ | v12.1

RemoveAllSelectedImages()

Removes all selected images.

Syntax	.RemoveAllSelectedImages();
Parameters	None
Return value	boolean
Example	`//This will remove the 2nd and 3rd images` `DWObject.SelectedImagesCount = 2;` `for(var i = 0; i < 2; i++){` `  DWObject.SetSelectedImageIndex(i,i + 1);` `}` `DWObject.RemoveAllSelectedImages();`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v4.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

RemoveImage()

Removes the image of a specified index in buffer.

Syntax	.RemoveImage(nImageIndex)
Parameters	number nImageIndex : Specifies the index of the image to be removed. The index is 0-based.
Return value	boolean

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v4.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Rotate()

Rotates the image of a specified index in buffer by a specified angle.

Syntax	.Rotate(nImageIndex, fAngle, bKeepSize);
Parameters	number nImageIndex : Specifies the index of image in buffer. The index is 0-based. number fAngle : Specifies the angle to rotate the image. Positive angle means clockwise. Negative value is counter-clockwise. boolean bKeepSize : Specifies whether to keep the original size of the image.
Return value	boolean

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v6.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	Rotate() rotates an image and replaces the original image with the rotated copy.

RotateEx()

Rotates the image of a specified index in buffer by a specified angle.

Syntax	.RotateEx(nImageIndex, fAngle, bKeepSize, newInterpolationMethod);
Parameters	`number nImageIndex` : Specifies the index of image in buffer. The index is 0-based. `number fAngle` : Specifies the angle to rotate the image. Positive angle means clockwise. Negative value is counter-clockwise. `boolean bKeepSize` : Specifies whether to keep the original size of the image. `EnumDWT_InterpolationMethod newInterpolationMethod` : Specifies the algorithm to do interpolation.

Allowed Values	Interpolation Method
1	EnumDWT_InterpolationMethod.IM_NEARESTNEIGHBOUR
2	EnumDWT_InterpolationMethod.IM_BILINEAR
3	EnumDWT_InterpolationMethod.IM_BICUBIC

Return value	`boolean`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v7.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	This method rotates an image and replaces the original image with the rotated copy.

RotateLeft()

Rotates the image of a specified index in buffer by 90 degrees counter-clockwise.

Syntax	.RotateLeft(nImageIndex);
Parameters	`number nImageIndex` : Specifies the index of image in buffer. The index is 0-based.
Return value	`boolean`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v4.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	RotateLeft() rotates an image and replaces the original image with the rotated copy.

RotateRight()

Rotates the image of a specified index in buffer by 90 degrees clockwise.

Syntax	.RotateRight(nImageIndex);
Parameters	`number nImageIndex` : Specifies the index of image in buffer. The index is 0-based.
Return value	`boolean`

Usage notes	This method rotates an image and replaces the original image with the rotated copy.
Availability	**ActiveX**: ✓ | v4.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1

SetDPI()

Changes the DPI (dots per inch) of an image specified by index.

Syntax	.SetDPI(nImageIndex, xResolution, yResolution, bResampleImage, newInterpolationMethod);
Parameters	`number nImageIndex` : Specifies the index of the image to be converted. The index is 0-based. `number xResolution` : Specifies horizontal resolution. `number yResolution` : Specifies vertical resolution. `boolean bResampleImage` : Specifies whether to resample the image. (The image size will change if this is set to true). `EnumDWT_InterpolationMethod newInterpolationMethod` : Specifies the algorithm to do interpolation.

Allowed Values	Interpolation Method
1	EnumDWT_InterpolationMethod.IM_NEARESTNEIGHBOUR
2	EnumDWT_InterpolationMethod.IM_BILINEAR
3	EnumDWT_InterpolationMethod.IM_BICUBIC

Return value	`boolean`
Availability	**ActiveX**: ✓ | v8.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1

SetImageWidth()

Changes the width of an image specified by index by adding an extra margin or removing part of the image.

Syntax	.SetImageWidth(ImageIndex, iNewWidth);
Parameters	`number nImageIndex` : Specifies the index of the image to be converted. The index is 0-based. `number iNewWidth` : Specifies the new width.
Return value	`boolean`
Availability	**ActiveX**: ✓ | v10.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1

SetSelectedImageArea()

Selects a rectangular area on an image specified by index.

Syntax	.SetSelectedImageArea(nImageIndex, left, top, right, bottom);
Parameters	`number` `nImageIndex` : Specifies the index of image in buffer. The index is 0-based. `number` `left` : Specifies the x-coordinate of the upper-left corner of the rectangle. `number` `top` : Specifies the y-coordinate of the upper-left corner of the rectangle. `number` `right` : Specifies the x-coordinate of the lower-right corner of the rectangle. `number` `bottom` : Specifies the y-coordinate of the lower-right corner of the rectangle.
Return value	`boolean`
Availability	**ActiveX** ✓ | v7.0 / **H5(Windows)** ✓ | v10.1 / **H5(macOS/TWAIN)** ✓ | v11.0 / **H5(macOS/ICA)** ✓ | v11.0 / **H5(Linux)** ✓ | v12.1
Usage notes	You should set Unit before using this method.

SetSelectedImageIndex()

Selects images programatically (instead of seleting by clicking on the images).

Syntax	.SetSelectedImageIndex(selectionArrayindex, newIndextobeSelected);
Parameters	`number` `selectionArrayindex` : Specifies the index of the selected Array to be used for storing the next image index to be selected. `number` `newIndextobeSelected` : Specifies the index of the image to be selected.
Return value	`boolean`
Example	`//This will remove the 2nd and 3rd images` `DWObject.SelectedImagesCount = 2;` `for(var i = 0; i < 2; i++){` `  DWObject.SetSelectedImageIndex(i,i + 1);` `}` `DWObject.RemoveAllSelectedImages();`
Availability	**ActiveX** ✓ | v7.0 / **H5(Windows)** ✓ | v10.0 / **H5(macOS/TWAIN)** ✓ | v11.0 / **H5(macOS/ICA)** ✓ | v11.0 / **H5(Linux)** ✓ | v12.1

ShowImageEditor()

Shows or hides the built-in image editor of Dynamic Web TWAIN.

Syntax	.ShowImageEditor([strDIVID, nDIVWidth, nDIVHeight]);
Parameters	The parameters are optional, you should either provide None or all 3. `string` `optionalDIVID` : A DIV to hold the editor on the page and its size. If the parameters are not provided, the editor will take the full window space. `number` `nDIVWidth` : The width of the DIV holding the editor. `number` `nDIVHeight` : The width of the DIV holding the editor.
Return value	`boolean`
Example	`DWObject.ShowImageEditor('divEditor', 500,700);`

Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v5.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✓ | v11.0</td><td>✓ | v12.1</td></tr></table>
Usage notes	Only one editor is allowed on one web page. The second time you try to call this method will close the editor that is already opened.

SwitchImage()	
Switches two images of specified indices in the buffer.	
Syntax	.SwitchImage(nImageIndex1, nImageIndex2);
Parameters	`number nImageIndex1` : Specifies the 1st index of the images to be switched. The index is 0-based. `number nImageIndex2` : Specifies the 2nd index of the images to be switched. The index is 0-based.
Return value	`boolean`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v5.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✓ | v11.0</td><td>✓ | v12.1</td></tr></table>

Properties

SelectionRectAspectRatio	
Specifies a fixed aspect ratio to be used when you use mouse to draw a rectanglar on an image to select an area.	
Type	`float`
Accessors	`Get Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v10.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✓ | v11.0</td><td>✓ | v12.1</td></tr></table>

Events

OnImageAreaDeSelected	
This event is triggered when user deselects an area (clicks outside of the drawn rectangle) on an image in Dynamic Web TWAIN viewer.	
Syntax	.RegisterEvent('OnImageAreaDeSelected',function(nImageIndex){...});
Arguments	• `number nImageIndex` : the index of the image. It should be >=0. When it is -1, it means the mouse is not on any image.

Example

```
DWObject.RegisterEvent('OnImageAreaDeSelected', function(nImageIndex){
    alert('The selected area on the image with index '
     + nImageIndex + 'has been deselected');
});
```

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

OnImageAreaSelected

This event is triggered when user selects an area (draws a rectangle) or move a selected area on an image in Dynamic Web TWAIN viewer.

Syntax

.RegisterEvent('OnImageAreaSelected',function(nImageIndex, left, top, right, bottom, sAreaIndex){...});

Arguments

- `number` `nImageIndex` : Specifies the index of image in buffer. The index is 0-based.
  `number` `left` : Specifies the x-coordinate of the upper-left corner of the rectangle.
  `number` `top` : Specifies the y-coordinate of the upper-left corner of the rectangle.
  `number` `right` : Specifies the x-coordinate of the lower-right corner of the rectangle.
  `number` `bottom` : Specifies the y-coordinate of the lower-right corner of the rectangle.
  `number` `sAreaIndex` : Specifies the index of the selected area. The index is 1-based. This is useful when you have multiple selected areas on one image.

Example

```
DWObject.RegisterEvent('OnImageAreaSelected', function(nImageIndex, left, top,
 right, bottom, sAreaIndex){
    alert(nImageIndex + 'left: ' + left);
});
```

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Display & UI

Methods		
SetViewMode()	CloseSource()	DisableSource()
EnableSource()	OpenSource()	SelectSource()
SelectSourceByIndex()	SetOpenSourceTimeout()	

Properties		
BackgroundColor	BackgroundFillColor	FitWindowType
Height	IfAutoScroll	IfFitWindow
ImageMargin	MaxImagesInBuffer	MouseShape
SelectionImageBorderColor	Width	Zoom
ShowPageNumber	MouseX	MouseY

Events	
OnTopImageInTheViewChanged	OnMouseClick
OnMouseDoubleClick	OnMouseMove
OnMouseRightClick	

Methods

SetViewMode()

Sets how the images are displayed in Dynamic Web TWAIN viewer.

Syntax	.SetViewMode(sHorizontalImageCount, sVerticalImageCount);
Parameters	`number sHorizontalImageCount` : specifies the number of columns. `number sVerticalImageCount` : specifies the number of rows.
Return value	`Void`
Example	`// Zoom is valid only when the view mode is set to -1 by -1.` `DWObject.SetViewMode(-1,-1);` `DWObject.Zoom = DWObject.Zoom * 1.2; // Zoom in` `DWObject.Zoom = DWObject.Zoom / 1.2; // Zoom out`
Availability	ActiveX: ✓ | v5.0 H5(Windows): ✓ | v10.1 H5(macOS/TWAIN): ✓ | v11.0 H5(macOS/ICA): ✓ | v11.0 H5(Linux): ✓ | v12.1
	The default view mode is 1 by 1.

Usage notes	When the view mode is set to -1 by -1, Dynamic Web TWAIN only shows the current image. N scroll bar is provided to navigate to other images. When the view mode is set to m (m>0) by n (n>0), a vertical scroll bar will be provided to navigate to other images. When the view mode is set to n (n>=1) by -1, a horizontal scroll bar will be provided to navigat to other images.

Properties

BackgroundColor

Returns or sets the background color of the built-in viewer.

Type	`number`
Accessors	`Get Set`
Usage notes	Default is white (0xffffff). The byte-ordering of the 24-bit RGB value is RRGGBB. RR represents red, GG represents green and BB represents blue.

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v6.0	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1

BackgroundFillColor

Returns or sets the fill color of the selected area of an image when it is cut, erased or rotated. It is a value specifying the 24-bit RGB value.

Type	`number`
Accessors	`Get Set`
Usage notes	Default is white (0xffffff). The byte-ordering of the 24-bit RGB value is RRGGBB. RR represents red, GG represents green and BB represents blue.

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v6.0	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1

FitWindowType

Returns or sets how the image is/should be resized to fit to the width or height of the Dynamic Web TWAIN viewer. To use the property, the view mode should be set to -1 by -1.

Type	`number`
Accessors	`Get Set`

Usage notes

Before change the value of FitWindowType, make sure you have set the view mode to -1 by -1 using SetViewMode.
Besides, if you want to change the value of FitWindowType after Zoom in/out, make sure set the property IfFitWindow to true.

Allowed Values	Fit Window Type
0 (default)	Fit the image to both the width and height of the viewer

1	Fit the image to the height of the viewer
2	Fit the image to the width of the viewer

When an error occurs, check ErrorCode or ErrorString for error information.

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v6.2	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1

Height

Returns or sets the viewer height (in px) of a Dynamic Web TWAIN instance on the page.

Type: `number`

Accessors: `Get Set`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v6.2	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1

IfAutoScroll

Returns or sets whether to display the newly added image or keep the current one after an image(s) is importec into the Dynamic Web TWAIN viewer.

Type: `boolean`

Accessors: `Get Set`

Usage notes:
This property is valid only in the ActiveX Edition and the HTML5 Edition from v12.0.
This property doesn't work when the view mode is set to -1 by -1.
The default value of the IfAutoScroll property is true.
If set to true, it will display the newly added image. If set to false, it will display the current one.
When an error occurs, check ErrorCode or ErrorString for error information.

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v12.0	✓ | v12.0	✓ | v12.0	✓ | v12.0	✓ | v12.1

IfFitWindow

Returns or sets whether to resize the image to fit both the width and height of the Dynamic Web TWAIN viewer. To use the property, the view mode should be set to -1 by -1.

Type: `boolean`

Accessors: `Get Set`

Usage notes:
Before change the value of IfFitWindow, make sure you have set the view mode to -1 by -1 using SetViewMode.
The default value of the IfFitWindow property is true.
When the value is 'false', the image will be displayed in its full size and scroll bars will appear if necessary (the width or height of the image is bigger than the viewer size).
When an error occurs, check ErrorCode or ErrorString for error information.

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v5.1	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1

ImageMargin

Returns or sets the margin between images when multiple images are displayed in Dynamic Web TWAIN viewer.

Type	`number`
Accessors	`Get Set`
Usage notes	The default value will auto judge. When an error occurs, check ErrorCode or ErrorString for error information.

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v5.0	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1

MaxImagesInBuffer

Returns or sets how many images are allowed to be acquired.

Type	`number`
Accessors	`Get Set`
Usage notes	The default value of this property is 32767. If you set MaxImagesInBuffer to be smaller than HowManyImagesInBuffer, the extra images in buffer will be removed. When image buffer is full, that is HowManyImagesInBuffer equals MaxImagesInBuffer, the newly acquired or loaded image will replace the existing ones from the first one on. When an error occurs, check ErrorCode or ErrorString for error information.

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v4.0	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1

MouseShape

Returns or sets the shape of the mouse (cursor).

Type	`boolean`
Accessors	`Get Set`
Usage notes	When the property is true, the cursor is set as 'hand'. If the width or height of the image is bigger than the Dynamic Web TWAIN viewer, scroll bars will appear and you can drag the image to adjust the position. When the property is false, the cursor is set as 'crosshair'. You can then select a rectangular area on Dynamic Web TWAIN viewer directly. After that, the event OnImageAreaSelected will fire and you can do things like 'crop the selected area'. When an error occurs, check ErrorCode or ErrorString for error information.

	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)

Availability	✓ | v5.1	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1

MouseX

Returns the X co-ordinate of the mouse. This is a read-only property.

Type	`number`
Accessors	`Get`

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v6.1	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1

MouseY

Returns the Y co-ordinate of the mouse. This is a read-only property.

Type	`number`
Accessors	`Get`

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v6.1	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1

SelectionImageBorderColor

Returns or sets the border color of the selected image. It is a value specifying the 24-bit RGB value.

Type	`number`
Accessors	`Get Set`
Usage notes	The default value is light blue 0x7DA2CE. Please NOTE that the byte-ordering of the 24-bit RGB value is RRGGBB. RR represents red, GG represents green and BB represents blue.

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v6.0	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1

ShowPageNumber

Returns or sets whether to show the page number at the top left corner of the viewer.

Type	`boolean`
Accessors	`Get Set`
Usage notes	This property only works in the HTML5 Edition from v10.2. If the property is true, the page number will be displayed at the top left corner of the viewer. If it is false, the page number will not be displayed.

	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)

Availability	✗	✓ | v10.2	✓ | v11.0	✓ | v11.0	✓ | v12.1

Width

Returns or sets the viewer width (in px) of a Dynamic Web TWAIN instance on the page.

Type	`number`
Accessors	`Get Set`

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v6.2	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1

Zoom

Returns or sets zoom factor for the image, only valid when the view mode is set to -1 by -1.

Type	`number`
Accessors	`Get Set`
Usage notes	The viewer will refresh to reflect the change of Zoom. Before change the value of Zoom, make sure you have set the view mode to -1 by -1 using SetViewMode. Also make sure you have set IfFitWindow to false. The zoom value is expressed as a percentage. The valid range is from 2 to 6500 percent (Value: 0.02 ~ 65.0); the default value is 100 percent (Value: 1.0). A zoom value can be specified before or after an image is displayed. When the zoom value is changed, the displayed image will refresh automatically. When an error occurs, check ErrorCode or ErrorString for error information.

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v5.2	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1

Events

OnMouseClick

This event is triggered when the mouse clicks on an image in Dynamic Web TWAIN viewer.

Syntax	.RegisterEvent('OnMouseClick',function(sImageIndex){...});
Arguments	• `number` `sImageIndex` : The index of the image.

Example

```
DWObject.RegisterEvent('OnMouseClick', function(sImageIndex){
    alert(sImageIndex);
});
```

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ |				

	v5.0

OnMouseDoubleClick

This event is triggered when the mouse double clicks on an image in Dynamic Web TWAIN viewer.

Syntax	.RegisterEvent('OnMouseDoubleClick',function(sImageIndex){...});
Arguments	• `number sImageIndex` : The index of the image.
Example	`DWObject.RegisterEvent('OnMouseDoubleClick', function(sImageIndex){ alert(sImageIndex); });`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.1	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1

OnMouseMove

This event is triggered when the mouse hovers over an image in Dynamic Web TWAIN viewer.

Syntax	.RegisterEvent('OnMouseMove',function(sImageIndex){...});
Arguments	• `number sImageIndex` : The index of the image.
Example	`DWObject.RegisterEvent('OnMouseMove', function(sImageIndex){ alert(sImageIndex); });`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.0	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1

OnMouseRightClick

This event is triggered when the mouse right clicks on an image in Dynamic Web TWAIN viewer.

Syntax	.RegisterEvent('OnMouseRightClick',function(sImageIndex){...});
Arguments	• `number sImageIndex` : The index of the image which was clicked upon.
Example	`DWObject.RegisterEvent('OnMouseRightClick', function(sImageIndex){ alert(sImageIndex); });`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.1	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1

	v5.1				

OnTopImageInTheViewChanged					
This event is triggered when the top image currently displayed in Dynamic Web TWAIN viewer changes.					
Syntax	.RegisterEvent('OnTopImageInTheViewChanged',function(sImageIndex){...});				
Arguments	• `number sImageIndex` : The index of the image.				
Example	`DWObject.RegisterEvent('OnTopImageInTheViewChanged', function(sImageIndex){ alert(sImageIndex); });`				
Availability	**ActiveX**	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✓ | v5.1	✓ | v10.1	✓ | v11.0	✓ | v11.0	✓ | v12.1
Usage notes	The returned value sImageIndex means the index of the top image. When sImageIndex equals -1, it indicates that there is no image in buffer.				

Load & Save

Methods	
FileExists()	LoadDibFromClipboard()
LoadImage()	LoadImageEx()
LoadImageFromBase64Binary()	SaveAllAsMultiPageTIFF()
SaveAllAsPDF()	SaveAsBMP()
SaveAsJPEG()	SaveAsPDF()
SaveAsPNG()	SaveAsTIFF()
SaveSelectedImagesAsMultiPagePDF()	SaveSelectedImagesAsMultiPageTIFF()
SaveSelectedImagesToBase64Binary()	ShowFileDialog()

Properties
IfShowFileDialog

Events	
OnGetFilePath	OnPostLoad

Code example

The following code example demonstrates how to use the APIs above to perform basic scanning.

```
var imagedata;
DWObject.SelectedImagesCount = 1;
DWObject.SetSelectedImageIndex(0,0);
DWObject.GetSelectedImagesSize(EnumDWT_ImageType.IT_JPG);
imagedata = DWObject.SaveSelectedImagesToBase64Binary();
DWObject.LoadImageFromBase64Binary(imagedata, EnumDWT_ImageType.IT_JPG, function(){
    console.log('success');
    },function(){
    console.log('failure');
});
```

Methods

FileExists()	
Checks whether a certain file exists on the local disk.	
Syntax	.FileExists(fileName);
Parameters	`string` `fileName` : the absolute path of the file to be checked.
Return value	`boolean`

Example	`DWObject.FileExists("C:\\DWT\\WebTWAIN.jpg");`
Availability	**ActiveX**: ✓ | v9.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v10.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1

LoadDibFromClipboard()

Loads a DIB image from the system clipboard into Dynamic Web TWAIN.

Syntax	.LoadDibFromClipboard([optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `OnSuccess Function optional optionalAsyncSuccessFunc` : callback function triggered when the file is loaded successfully. `OnFailure Function optional optionalAsyncFailureFunc` : callback function triggered when the file failed to be loaded.
Return value	`boolean` Only valid when used synchronously.
Availability	**ActiveX**: ✓ | v4.1 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v10.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1

LoadImage()

Loads local image(s) into the Dynamic Web TWAIN buffer.

Syntax	.LoadImage(fileName, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string fileName` : the absolute path to the file to be opened. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `OnSuccess Function optional optionalAsyncSuccessFunc` : callback function triggered when the file is loaded successfully. `OnFailure Function optional optionalAsyncFailureFunc` : callback function triggered when the file failed to be loaded.
Return value	`boolean` Only valid when used synchronously.
Example	`DWObject.LoadImage("C:\\DWT.jpg");`
Availability	**ActiveX**: ✓ | v4.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v10.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1
Usage notes	Dynamic Web TWAIN processes the image format according to the extension of `fileName` .

LoadImageEx()

Loads local image(s) into Dynamic Web TWAIN.

Syntax	.LoadImageEx(fileName, imageType, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]
Parameters	`string fileName` : the absolute path to open the file. `EnumDWT_ImageType (int) imageType` : the image format. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `OnSuccess Function optional optionalAsyncSuccessFunc` : callback function triggered when the file is loaded successfully. `OnFailure Function optional optionalAsyncFailureFunc` : callback function triggered when the file failed to be loaded.
Return value	`boolean` Only valid when used synchronously.
Example	`DWObject.IfShowFileDialog = true;` `DWObject.LoadImageEx("", EnumDWT_ImageType.IT_ALL);`
Availability	**ActiveX**: ✓ | v5.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v10.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1
Usage notes	If you would like to load/open images by showing the 'Open File' dialog box, you can set `IfShowFileDialog` to true.

LoadImageFromBase64Binary()

Load image(s) from a base64 string.

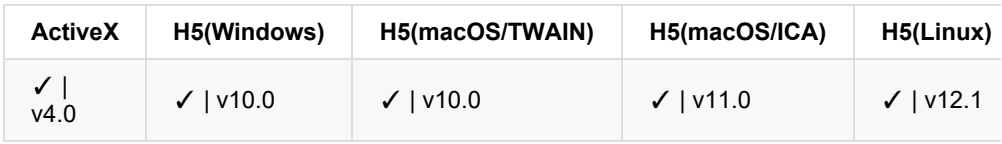
Syntax	.LoadImageFromBase64Binary(imageData, imageType, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string imageData` : the base64 string that represents the image. `EnumDWT_ImageType (int) imageType` : the image format. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `OnSuccess Function optional optionalAsyncSuccessFunc` : callback function triggered when the operation succeeds. `OnFailure Function optional optionalAsyncFailureFunc` : callback function triggered when the operation fails.
Return value	`boolean`
Example	`var imagedata;` `DWObject.SelectedImagesCount = 1;` `DWObject.SetSelectedImageIndex(0,0);` `DWObject.GetSelectedImagesSize(EnumDWT_ImageType.IT_JPG);` `imagedata = DWObject.SaveSelectedImagesToBase64Binary();` `DWObject.LoadImageFromBase64Binary(imagedata, EnumDWT_ImageType.IT_JPG, function(){` `  console.log('success');` `  },function(){` `  console.log('failure');` `});`
	ActiveX: ✓ | **H5(Windows)**: **H5(macOS/TWAIN)**: **H5(macOS/ICA)**: **H5(Linux)**:

	v6.2	✓ | v10.0	✓ | v10.0	✓ | v11.0	✓ | v12.1

SaveAllAsMultiPageTIFF()	
Saves all the images in the buffer as a Multipage TIFF file.	
Syntax	.SaveAllAsMultiPageTIFF(fileName, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` `fileName` : the absolute path to save the file. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `OnSuccess Function` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is saved successfully. `OnFailure Function` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be saved.
Return value	`boolean` Only valid when used synchronously.
Example	`DWObject.SaveAllAsMultiPageTIFF("C:\\DWT.tiff");`
Availability	**ActiveX** | **H5(Windows)** | **H5(macOS/TWAIN)** | **H5(macOS/ICA)** | **H5(Linux)** ✓ | v4.0 | ✓ | v10.0 | ✓ | v10.0 | ✓ | v11.0 | ✓ | v12.1
Usage notes	Multi-Page TIFF encoding is a built-in feature of Dynamic Web TWAIN, no extra dlls are required. If you would like to save images by showing the 'Save File' dialog box, you can set `IfShowFileDialog` to true.

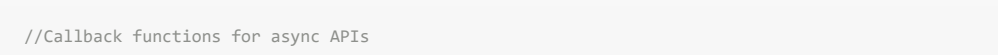
SaveAllAsPDF()	
Saves all the images in the buffer as a Multipage PDF file.	
Syntax	.SaveAllAsPDF(fileName, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` `fileName` : the absolute path to save the file. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `OnSuccess Function` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is saved successfully. `OnFailure Function` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be saved.
Return value	`boolean` Only valid when used synchronously.
Example	`DWObject.SaveAllAsPDF("C:\\DWT.pdf");`
Availability	**ActiveX** | **H5(Windows)** | **H5(macOS/TWAIN)** | **H5(macOS/ICA)** | **H5(Linux)** ✓ | v5.0 | ✓ | v10.0 | ✓ | v10.0 | ✓ | v11.0 | ✓ | v12.1
Usage	Multi-Page PDF encoding is a built-in feature of Dynamic Web TWAIN, no extra dlls are required. If you would like to save images by showing the 'Save File' dialog box, you can set

	`IfShowFileDialog` to true.

SaveAsBMP()	
Saves the image of a specified index in the buffer as a BMP file.	
Syntax	.SaveAsBMP(fileName, sImageIndex, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` `fileName` : the absolute path on the client machine to save the file. `number` `sImageIndex` : the index of the image in the buffer. The index is 0-based. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `OnSuccess Function` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is saved successfully. `OnFailure Function` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be saved.
Return value	`boolean` Only valid when used synchronously.
Example	
Availability	
Usage notes	If you would like to save images by showing the 'Save File' dialog box, you can set `IfShowFileDialog` to true.

```
//Callback functions for async APIs
function OnSuccess() {
    console.log('successful');
}
function OnFailure(errorCode, errorString) {
    alert(errorString);
}
function btn_SaveOnClick() {
    if (DWObject.HowManyImagesInBuffer == 0){
        alert("No images in buffer.");
        return;
    }
    DWObject.SaveAsBMP("D:\\test.bmp", 0, OnSuccess, OnFailure);
}
```

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v4.0	✓ | v10.0	✓ | v10.0	✓ | v11.0	✓ | v12.1

SaveAsJPEG()	
Saves the image of a specified index in the buffer as a JPEG file.	
Syntax	.SaveAsJPEG(fileName, sImageIndex, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` `fileName` : the absolute path on the client machine to save the file. `number` `sImageIndex` : the index of the image in the buffer. The index is 0-based. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `OnSuccess Function` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is saved successfully. `OnFailure Function` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be saved.
Return value	`boolean` Only valid when used synchronously.
	

```
//Callback functions for async APIs
```

Example

```
//Callback functions for async APIs
function OnSuccess() {
    console.log('successful');
}
function OnFailure(errorCode, errorString) {
        alert(errorString);
}
function btn_SaveOnClick() {
    if (DWObject.HowManyImagesInBuffer == 0){
        alert("No images in buffer.");
        return;
    }
    DWObject.SaveAsJPEG("D:\\test.jpg", 0, OnSuccess, OnFailure);
}
```

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v4.0	✓ | v10.0	✓ | v10.0	✓ | v11.0	✓ | v12.1

Usage notes

If you would like to save images by showing the 'Save File' dialog box, you can set `IfShowFileDialog` to true.

SaveAsPDF()

Saves the image of a specified index in the buffer as a PDF file.

Syntax

.SaveAsPDF(fileName, sImageIndex, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]

Parameters

`string fileName` : the absolute path on the client machine to save the file.
`number sImageIndex` : the index of the image in the buffer. The index is 0-based.
The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous.
`OnSuccess Function optional optionalAsyncSuccessFunc` : callback function triggered when the file is saved successfully.
`OnFailure Function optional optionalAsyncFailureFunc` : callback function triggered when the file failed to be saved.

Return value

`boolean`
Only valid when used synchronously.

Example

```
//Callback functions for async APIs
function OnSuccess() {
    console.log('successful');
}
function OnFailure(errorCode, errorString) {
    alert(errorString);
}
function btn_SaveOnClick() {
    if (DWObject.HowManyImagesInBuffer == 0){
        alert("No images in buffer.");
        return;
    }
    DWObject.SaveAsPDF("D:\\test.pdf", 0, OnSuccess, OnFailure);
}
```

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.0	✓ | v10.0	✓ | v10.0	✓ | v11.0	✓ | v12.1

Usage notes

If you would like to save images by showing the 'Save File' dialog box, you can set `IfShowFileDialog` to true.

SaveAsPNG()

Saves the image of a specified index in the buffer as a PNG file.

Syntax	.SaveAsPNG(fileName, sImageIndex, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` `fileName` : the absolute path on the client machine to save the file. `number` `sImageIndex` : the index of the image in the buffer. The index is 0-based. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `OnSuccess Function` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is saved successfully. `OnFailure Function` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be saved.
Return value	`boolean` Only valid when used synchronously.
Example	`//Callback functions for async APIs` `function OnSuccess() {` `  console.log('successful');` `}` `function OnFailure(errorCode, errorString) {` `  alert(errorString);` `}` `function btn_SaveOnClick() {` `  if (DWObject.HowManyImagesInBuffer == 0){` `  alert("No images in buffer.");` `  return;` `  }` `  DWObject.SaveAsPNG("D:\\test.png", 0, OnSuccess, OnFailure);` `}`
Availability	**ActiveX**: ✓ | v4.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v10.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1
Usage notes	If you would like to save images by showing the 'Save File' dialog box, you can set `IfShowFileDialog` to true.

SaveAsTIFF()

Saves the image of a specified index in the buffer as a TIFF file.

Syntax	.SaveAsTIFF(fileName, sImageIndex, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc
Parameters	`string` `fileName` : the absolute path on the client machine to save the file. `number` `sImageIndex` : the index of the image in the buffer. The index is 0-based. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `OnSuccess Function` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is saved successfully. `OnFailure Function` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be saved.
Return value	`boolean` Only valid when used synchronously.
	`//Callback functions for async APIs` `function OnSuccess() {` `  console.log('successful');` `}` `function OnFailure(errorCode, errorString) {`

Example	<pre>} function btn_SaveOnClick() { if (DWObject.HowManyImagesInBuffer == 0){ alert("No images in buffer."); return; } DWObject.SaveAsTIFF("D:\\test.tif", 0, OnSuccess, OnFailure); }</pre>
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v3.0</td><td>✓ | v10.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✓ | v12.1</td></tr></table>
Usage notes	If you would like to save images by showing the 'Save File' dialog box, you can set `IfShowFileDialog` to true.

SaveSelectedImagesAsMultiPagePDF()	
Saves the selected images in the buffer as a Multipage PDF file.	
Syntax	.SaveSelectedImagesAsMultiPagePDF(fileName, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string fileName` : the absolute path on the client machine for saving the file. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `OnSuccess Function optional optionalAsyncSuccessFunc` : callback function triggered when the file is saved successfully. `OnFailure Function optional optionalAsyncFailureFunc` : callback function triggered when the file failed to be saved.
Return value	`boolean` Only valid when used synchronously.
Example	<pre>DWObject.SelectedImagesCount = 1; DWObject.SetSelectedImageIndex(0,0); DWObject.SaveSelectedImagesAsMultiPagePDF("C:\\DWT.pdf");</pre>
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v6.0</td><td>✓ | v10.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✓ | v12.1</td></tr></table>
Usage notes	Multi-Page PDF encoding is a built-in feature of Dynamic Web TWAIN, no extra dlls are required. If you would like to save images by showing the 'Save File' dialog box, you can set `IfShowFileDialog` to true.

SaveSelectedImagesAsMultiPageTIFF()	
Saves the selected images in the buffer as a Multipage TIFF file.	
Syntax	.SaveSelectedImagesAsMultiPagePDF(fileName, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string fileName` : the absolute path to save the file. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `OnSuccess Function optional optionalAsyncSuccessFunc` : callback function triggered when the file is saved successfully.

	`OnFailure` `Function` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be saved.
Return value	`boolean` Only valid when used synchronously.
Example	``` DWObject.SelectedImagesCount = 1; DWObject.SetSelectedImageIndex(0,0); DWObject.SaveSelectedImagesAsMultiPageTIFF("C:\\DWT.tiff"); ```
Availability	**ActiveX**: ✓ | v6.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v10.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1
Usage notes	Multi-Page TIFF encoding is a built-in feature of Dynamic Web TWAIN, no extra dlls are required. If you would like to save images by showing the 'Save File' dialog box, you can set `IfShowFileDialog` to true.

SaveSelectedImagesToBase64Binary()

Saves the selected images in the buffer to a base64 string.

Syntax	.SaveSelectedImagesToBase64Binary();
Parameters	`None`
Return value	`string` The result string is the pure base64 string with no extra info. For example, `"/9j/4AAQSkZJRgABA..."` . To use the string in most circumstances, you need to add extra info like `"/9j/4AAQSkZJRgABA..."` .
Example	``` var imagedata; DWObject.SelectedImagesCount = 1; DWObject.SetSelectedImageIndex(0, 0); DWObject.GetSelectedImagesSize(EnumDWT_ImageType.IT_JPG); imagedata = DWObject.SaveSelectedImagesToBase64Binary(); newImage.src = "" + imagedata; ```
Availability	**ActiveX**: ✓ | v6.2 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v10.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1
Usage notes	Please invoke `GetSelectedImagesSize` before invoking this method to specify the target format. The default value is `IT_JPG` and it doesn't support black and white images.

ShowFileDialog()

Show the system's save-file dialog or open-file dialog.

Syntax	.ShowFileDialog(bSave, filter, filterIndex, defaultExtension, initialDir, allowMultiSelect, overwritePrompt, flags);
	`boolean` `bsave` : true -- show save-file dialog, false -- show open-file dialog. `string` `filter` : the filter name specifies the filter pattern (for example, "JPG | *.jpg"). To specify multiple filter patterns use a semicolon to separate the patterns (for example, "JPG, PNG and TIF | *.jpg;*.png;*.tif" or "JPG, PNG , TIF" for macOS). A pattern string can be a

Parameters	combination of valid file extensions with asterisk (*). Do not include spaces in the pattern string. To show all files, use the string "All Files | *.*". `int filterIndex` : determines the index for the filter string we set, in other words, where the filter string should appear in the dialog's filters drop-down box. By default, it's 0 and you don't need to change it. This parameter doesn't work on macOS. `string defaultExtension` : define the default extension which will be appended to the file name. Only useful when you try to save an image or images. If this member is NULL and the user fail to type an extension, no extension is appended. `string initialDir` : the initial directory. The algorithm for selecting the initial directory varies on different platforms. `boolean allowMultiSelect` : true -- allow multiple selections, false -- only allow single file selection. `boolean overwritePrompt` : true -- if a file already exists with the same name, the user needs to confirm before the existing file is overwritten, false -- if a file already exists with the same name it will be overwritten without further user confirmation. `int flags` : if this parameter equals 0, the program will be initiated with the custom settings. Otherwise it will be initiated with default flags, which means `allowMultiSelect` and `overwritePrompt` will not work.
Return value	`boolean`
Example	`DWObject.RegisterEvent('OnGetFilePath', function(bSave, filesCount, index, path, filename){ alert(" fileCount: " + filesCount + " index: " + index + " path: " + path + "\\" + filename); }); //On macOS var result = DWObject.ShowFileDialog(false, "TIF,TIFF,JPG,JPEG,PNG,PDF", 0, "", "", true, false, 0); //On Windows var result = DWObject.ShowFileDialog(false, "TIF,TIFF,JPG,JPEG,PNG,PDF", 0, "", "", true, true, 0) alert(result);`
Availability	ActiveX: ✓ | v8.0 H5(Windows): ✓ | v10.0 H5(macOS/TWAIN): ✓ | v10.0 H5(macOS/ICA): ✓ | v11.0 H5(Linux): ✓ | v12.1
Usage notes	For Mac Edition, the method performs differently. This method will trigger `OnGetFilePath` event even when it fails. If multiple files are selected, the event will be called multiple times.

Properties

IfShowFileDialog

Returns or sets whether to show open/save file dialog when saving scanned images or loading images from a local directory.

Type	`boolean`
Accessors	`Get Set`
Availability	ActiveX: ✓ | v6.2 H5(Windows): ✓ | v10.0 H5(macOS/TWAIN): ✓ | v10.0 H5(macOS/ICA): ✓ | v11.0 H5(Linux): ✓ | v12.1

Usage notes	When loading, the property only works for the method `LoadImageEx` .

Events

OnGetFilePath

This event is triggered when a file path is returned from the operating system this occurs when the `ShowFileDialog` method is called or when the `LoadImageEx` method is called with `IfShowFileDialog` set to true.

Syntax	.RegisterEvent('OnGetFilePath',function(bSave, filesCount, index, path, filename){...});
Arguments	• `boolean` `bSave` : 'true' -- show save file dialog, 'false' -- show open file dialog. • `number` `filesCount` : How many files were selected. 0 means no file was selected or the user closed/cancelled. • `number` `index` : The index of the currently selected file. 0-based. -1 means no file was selected. • `string` `path` : The parent path of currently selected file(s), "\\" is not included. If the methed `ShowFileDialog()` failed, the initial directory path set in the `ShowFileDialog` method is returned. • `string` `filename` : The current file name.
Example	`DWObject.RegisterEvent('OnGetFilePath', function(bSave, filesCount, index, path, filename) { alert("bSave:" + bSave + " fileCount: " + filesCount + " index: " + index + " path: " + path + "\\" + filename); });`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v8.0	✓ | v10.0	✓ | v10.0	✓ | v11.0	✓ | v12.1

OnPostLoad

This event is triggered when an image file from a local directory has been loaded into the control.

Syntax	.RegisterEvent('OnPostLoad',function(path, name, type){...});
Arguments	• `string` `path` : the local path of the loaded image. For example, C:\Users\[username]\Downloads\ • `string` `name` : the name of the loaded image. For example, image1.jpg • `EnumDWT_ImageType` `type` : Image format.
Example	`DWObject.RegisterEvent('OnPostLoad', function(path, name, type) { alert(path + '\\' + name); });`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v6.3	✓ | v10.0	✓ | v10.0	✓ | v11.0	✓ | v12.1

Usage	If multiple image files are loaded, this event will be triggered once for each image.

notes	If multiple image files are loaded, this event will be triggered once for each image.

Upload & Download

Methods	
ClearAllHTTPFormField()	FTPDownload()
FTPDownloadDirectly()	FTPDownloadEx()
FTPUpload()	FTPUploadAllAsMultiPageTIFF()
FTPUploadAllAsPDF()	FTPUploadAsMultiPagePDF()
FTPUploadAsMultiPageTIFF()	FTPUploadDirectly()
FTPUploadEx()	HTTPDownload()
HTTPDownloadDirectly()	HTTPDownloadEx()
HTTPUpload()	HTTPUploadAllThroughPostAsMultiPageTIFF()
HTTPUploadAllThroughPostAsPDF()	HTTPUploadThroughPost()
HTTPUploadThroughPostAsMultiPagePDF()	HTTPUploadThroughPostAsMultiPageTIFF()
HTTPUploadThroughPostDirectly()	HTTPUploadThroughPostEx()
SetHTTPFormField()	SetUploadSegment()
SetHTTPHeader()	

Properties	
FTPPassword	FTPPort
FTPUserName	HttpFieldNameOfUploadedImage
HTTPPort	HTTPPostResponseString
IfPASVMode	IfShowCancelDialogWhenImageTransfer
IfSSL	MaxUploadImageSize

Events	
OnInternetTransferPercentage	OnInternetTransferPercentageEx

Code example

The following code example demonstrates how to use the APIs above to perform basic uploading.

```
DWObject.HTTPUpload ('www.dynamsoft.com/SaveToFile.aspx?filename=001.pdf', [0,1],  EnumDWT_ImageType.IT_PDF, EnumDWT_
UploadDataFormat.Binary, 'test.pdf', OnHttpUploadSuccess, OnHttpUploadFailure);
function OnHttpUploadSuccess (httpResponse) {
    console.log("HTTPResponseString: " + httpResponse);
}
function OnHttpUploadFailure (errorCode, errorString, httpResponse) {
    alert("ErrorCode: " + errorCode+ "ErrorString: " + errorString + "HTTPResponseString: " + httpResponse);
}
```

Methods

ClearAllHTTPFormField()	
Clears all the web form fields which will be sent to the server with the images when uploading.	
Syntax	.ClearAllHTTPFormField();
Parameters	`None`
Return value	`boolean`
Example	`DWObject.ClearAllHTTPFormField();`
Usage notes	
Availability	v5.0+

FTPDownload()	
Downloads an image from a specified FTP server.	
Syntax	.FTPDownload(fTPServer, fTPRemoteFile, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` fTPServer: the FTP server. `string` fTPRemoteFile: the path of the file on the FTP server. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<OnSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is downloaded successfully. `<OnFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be downloaded. Please refer to the function prototype OnSuccess or OnFailure.
Return value	`Boolean, only valid when used synchronously.`
Example	
Usage notes	Dynamic Web TWAIN processes the image format according to the extension of fTPRemoteFile.
Availability	v4.0+

FTPDownloadDirectly()	
Downloads a file from a specified FTP server directly without opening it in Dynamic Web TWAIN.	
Syntax	.FTPDownloadDirectly(fTPServer, fTPRemoteFile, localFile, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` fTPServer: the FTP server. `string` fTPRemoteFile: the path of the file on the FTP server. `string` localFile: the local path for the downloaded file. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<OnSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is downloaded successfully. `<OnFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be downloaded. Please refer to the function prototype OnSuccess or OnFailure.

Return value	`Boolean, only valid when used synchronously.`
Example	
Usage notes	Technically this method can download any file from the FTP server.
Availability	v7.0+

FTPDownloadEx()	
Downloads an image from a specified FTP server.	
Syntax	.FTPDownloadEx(fTPServer, fTPRemoteFile, imageType, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` fTPServer: the FTP server. `string` fTPRemoteFile: the path of the file on the FTP server. `EnumDWT_ImageType (int)` imageType: the image format to be used. **Type** / **Numeric Value** IT_BMP 0 IT_JPG 1 IT_TIF 2 IT_PNG 3 IT_PDF 4 The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<OnSuccess function> optional optionalAsyncSuccessFunc` : callback function triggered when the file is downloaded successfully. `<OnFailure function> optional optionalAsyncFailureFunc` : callback function triggered when the file failed to be downloaded. Please refer to the function prototype OnSuccess or OnFailure.
Return value	`Boolean, only valid when used synchronously.`
Example	
Usage notes	
Availability	v5.0+

FTPUpload()	
Uploads the image of a specified index in the buffer to the FTP server in a specified image format.	
Syntax	.FTPUpload(fTPServer, sImageIndex, fTPRemoteFile, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` fTPServer: the FTP server. `number` sImageIndex: the index of the image in the buffer. The index is 0-based. `string` fTPRemoteFile: the file name, path included. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<OnSuccess function> optional optionalAsyncSuccessFunc` : callback function triggered when the file is uploaded successfully. `<OnFailure function> optional optionalAsyncFailureFunc` : callback function triggered when the file failed to be uploaded. Please refer to the function prototype OnSuccess or OnFailure.
Return value	`Boolean, only valid when used synchronously.`

Example	
Usage notes	Dynamic Web TWAIN processes the image format according to the extension of fTPRemoteFile.
Availability	v4.0+

FTPUploadAllAsMultiPageTIFF()	
Uploads all the images in the buffer to a specified FTP server as a TIFF file.	
Syntax	.FTPUploadAllAsMultiPageTIFF(fTPServer, fileName, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` fTPServer: the FTP server. `string` fileName: the file name, including the path. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<OnSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is uploaded successfully. `<OnFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be uploaded. Please refer to the function prototype OnSuccess or OnFailure.
Return value	`Boolean, only valid when used synchronously.`
Example	
Usage notes	
Availability	v4.0+

FTPUploadAllAsPDF()	
Uploads all the images in the buffer to a specified FTP server as a PDF file.	
Syntax	.FTPUploadAllAsPDF(fTPServer, fileName, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` fTPServer: the FTP server. `string` fileName: the file name, including the path to the file. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<OnSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is uploaded successfully. `<OnFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be uploaded. Please refer to the function prototype OnSuccess or OnFailure.
Return value	`Boolean, only valid when used synchronously.`
Example	
Usage notes	
Availability	v4.0+

FTPUploadAsMultiPagePDF()	
Uploads the selected images in buffer to a specified FTP server as a PDF file.	
Syntax	.FTPUploadAsMultiPagePDF(fTPServer, fileName, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
	`string` fTPServer: the FTP server.

Parameters	`string` fileName: the file name, path included. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<OnSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is uploaded successfully. `<OnFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be uploaded. Please refer to the function prototype OnSuccess or OnFailure.
Return value	`Boolean, only valid when used synchronously.`
Example	
Usage notes	
Availability	v6.0+

FTPUploadAsMultiPageTIFF()	
Uploads the selected images in the buffer to a specified FTP server as a TIFF file.	
Syntax	.FTPUploadAsMultiPageTIFF(fTPServer, fileName, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` fTPServer: the FTP server. `string` fileName: the file name, path included. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<OnSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is uploaded successfully. `<OnFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be uploaded. Please refer to the function prototype OnSuccess or OnFailure.
Return value	`Boolean, only valid when used synchronously.`
Example	
Usage notes	
Availability	v6.0+

FTPUploadDirectly	
Upload any type of file to the server via FTP.	
Syntax	.FTPUploadDirectly(fTPServer, strLocalFile, fTPRemoteFile, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` fTPServer: the FTP server. `string` strLocalFile: the path of the local file. `string` fTPRemoteFile: the file name, path included. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<OnSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is uploaded successfully. `<OnFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be uploaded. Please refer to the function prototype OnSuccess or OnFailure.
Return value	`Boolean, only valid when used synchronously.`
Example	
Usage notes	

Availability	v7.0+

FTPUploadEx()	
Uploads the image of a specified index in buffer to the FTP server in a specified image format.	
Syntax	.FTPUploadEx(fTPServer, sImageIndex, fTPRemoteFile, imageType, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` fTPServer: the FTP server. `number` sImageIndex: the index of image in buffer. The index is 0-based. `string` fTPRemoteFile: the file name, path included. `EnumDWT_ImageType (int)` imageType: the format. **Type** / **Numeric Value** IT_BMP 0 IT_JPG 1 IT_TIF 2 IT_PNG 3 IT_PDF 4 The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<OnSuccess function> optional optionalAsyncSuccessFunc` : callback function triggered when the file is uploaded successfully. `<OnFailure function> optional optionalAsyncFailureFunc` : callback function triggered when the file failed to be uploaded. Please refer to the function prototype OnSuccess or OnFailure.
Return value	`Boolean, only valid when used synchronously.`
Example	
Usage notes	
Availability	v5.0+

HTTPDownload()	
Downloads an image from a specified HTTP server.	
Syntax	.HTTPDownload(strHTTPServer, strHTTPRemoteFile, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` strHTTPServer: the HTTP server. `string` strHTTPRemoteFile: the file name, the path of the file on the HTTP server. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<OnSuccess function> optional optionalAsyncSuccessFunc` : callback function triggered when the file is downloaded successfully. `<OnFailure function> optional optionalAsyncFailureFunc` : callback function triggered when the file failed to be downloaded. Please refer to the function prototype onsuccess or OnFailure.
Return value	`Boolean, only valid when used synchronously.`
Example	`/****Get image01.png from: http://www.dynamsoft.com/images/image01.png****/` `DWObject.HTTPDownload('www.dynamsoft.com','/images/image01.png');`

Usage notes	If you want to use this method to upload / download files through HTTPS, please don't forget to set IfSSL to true and set the correct HTTPPort.
Availability	v4.0+

HTTPDownloadDirectly()	
Downloads a file from a specified HTTP server directly without opening it in Dynamic Web TWAIN.	
Syntax	.HTTPDownloadDirectly(strHTTPServer, strHTTPRemoteFile, localFile, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` strHTTPServer: the HTTP server. `string` strHTTPRemoteFile: the path of the file on the HTTP server, or the path to an action page (with necessary parameters) which gets and sends back the file stream to the client. `string` localFile: the local path for the downloaded file. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<OnSuccess function> optional optionalAsyncSuccessFunc` : callback function triggered when the file is downloaded successfully. `<OnFailure function> optional optionalAsyncFailureFunc` : callback function triggered when the file failed to be downloaded. Please refer to the function prototype OnSuccess or OnFailure.
Return value	`Boolean, only valid when used synchronously.`
Example	`/*Get ducument.txt from: http://www.dynamsoft.com/files/ducument.txt*/` `/*Save as 01.txt to: D:/*/` `DWObject.HTTPDownloadDirectly('www.dynamsoft.com','/files/ducument.txt','D:/01.txt');` `/*Get image02.jpg` `from: http://www.dynamsoft.com/images/GetImage.aspx?Index=2&Name=image02.jpg*/` `/**Save as 02.jpg to: D:/temp/02.jpg**/` `DWObject.HTTPDownloadDirectly('www.dynamsoft.com',` `'/images/GetImage.aspx?Index=2&Name=image02.jpg','D:/temp/02.jpg');`
Usage notes	Technically this method can download any file from the HTTP server. If you want to use this method to upload / download files through HTTPS, please don't forget to set IfSSL to true and set the correct HTTPPort.
Availability	v7.0+

HTTPDownloadEx()	
Downloads an image from a specified HTTP server.	
Syntax	.HTTPDownloadEx(strHTTPServer, strHTTPRemoteFile, imageType, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` strHTTPServer: the address of the HTTP server. For example: "www.dynamsoft.com", "localhost", "127.0.0.1". `string` strHTTPRemoteFile: the path of the file on the HTTP server, or the path to an action page (with necessary parameters) which gets and sends back the image stream to the client. `EnumDWT_ImageType (int)` imageType: the image format.

Type	Numeric Value
IT_BMP	0
IT_JPG	1
IT_TIF	2
IT_PNG	3

	IT_PDF: 4 The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<OnSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is downloaded successfully. `<OnFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be downloaded. Please refer to the function prototype OnSuccess or OnFailure.
Return value	`Boolean, only valid when used synchronously.`
Example	`/*Get image01.png from: http://www.dynamsoft.com/images/image01.png*/` `DWObject.HTTPDownloadEx('www.dynamsoft.com','/images/image01.png',3);` `/*Get image02.jpg from:` `http://www.dynamsoft.com/images/GetImage.aspx?Index=2&Name=image02.jpg*/` `DWObject.HTTPDownloadEx('www.dynamsoft.com',` `'/images/GetImage.aspx?Index=2&Name=image02.jpg',1);`
Usage notes	If you want to use this method to upload / download files through HTTPS, please don't forget to set IfSSL to true and set the correct HTTPPort.
Availability	v5.0+

HTTPUpload()

1. Uploads images of specified indices in the buffer to the HTTP server asynchronously.
1. Or uploads the form created by SetHTTPFormField.

Syntax	1. HTTPUpload(url, indices, enumImageType, dataFormat, fileName, asyncSuccessFunc, asyncFailureFunc); 2. HTTPUpload(url, indices, enumImageType, dataFormat, asyncSuccessFunc, asyncFailureFunc); 3. HTTPUpload(url, asyncSuccessFunc, asyncFailureFunc);
Parameters	`string` url: specifies the name of http server and action page and filename. For example "www.dynamsoft.com/ActionPageName?filename=FileName". `Array` indices: specifies the indices of images in buffer. The index is 0-based. If EnumDWT_ImageType is IT_BMP/IT_JPG/IT_PNG, since multi-page file is not possible, you can only specify one index. `EnumDWT_ImageType` imageType: specifies the format of the file you want to upload as.

Type	Numeric Value
IT_BMP	0
IT_JPG	1
IT_TIF	2
IT_PNG	3
IT_PDF	4

`string` fileName: specifies the file name.
`EnumDWT_UploadDataFormat` dataFormat: specifies the data format, either EnumDWT_UploadDataFormat.Binary or EnumDWT_UploadDataFormat.base64.

Type	Numeric Value
BINARY	0
BASE64	1

	`<onHttpUploadSuccess function>` `asyncSuccessFunc` : callback function triggered when the file is uploaded successfully. `<onHttpUploadFailure function>` `asyncFailureFunc` : callback function triggered when the file failed to be uploaded. Please refer to the function prototype onHttpUploadSuccess or onHttpUploadFailure.
Return value	`Boolean. This method is used asynchronously only. Even though it returns true doesn't mean that the upload is successful.`
Example	`DWObject.HTTPUpload('www.dynamsoft.com/SaveToFile.aspx?filename=001.pdf', [0,1], EnumDWT_ImageType.IT_PDF, EnumDWT_UploadDataFormat.Binary, 'test.pdf', OnHttpUploadSuccess, OnHttpUploadFailure);` `function OnHttpUploadSuccess (httpResponse) {` `  console.log("HTTPResponseString: " + httpResponse);` `}` `function OnHttpUploadFailure (errorCode, errorString, httpResponse) {` `  alert("ErrorCode: " + errorCode+ "ErrorString: " +` `  errorString + "HTTPResponseString: " + httpResponse);` `}`
Usage notes	This method is valid only in the HTML5 Edition from v12.0.
Availability	v12.0+

HTTPUploadAllThroughPostAsMultiPageTIFF()	
Uploads all the images in the buffer to a specified HTTP server as a single TIFF file.	
Syntax	.HTTPUploadAllThroughPostAsMultiPageTIFF(strHTTPServer, actionPage, fileName, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` strHTTPServer: the name of the HTTP server. For example: "www.dynamsoft.com", "localhost", "127.0.0.1". `string` actionPage: the relative path for the action page on the server. The action page will receive and process the uploaded image stream. `string` fileName: the file name. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<onHttpUploadSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is uploaded successfully. `<onHttpUploadFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be uploaded. Please refer to the function prototype onHttpUploadSuccess or onHttpUploadFailure.
Return value	`Boolean, only valid when used synchronously.`
	`function btnScan_onclick()` `{` `  if (DWObject) {` `  DWObject.SelectSource();` `  DWObject.OpenSource();` `  DWObject.IfDisableSourceAfterAcquire = true;` `  DWObject.AcquireImage();` `  }` `}` `// OnHttpUploadSuccess and OnHttpUploadFailure are callback functions.` `function OnHttpUploadSuccess() {` `  console.log('successful');` `}` `function OnHttpUploadFailure(errorCode, errorString, sHttpResponse) {` `  alert(errorString + sHttpResponse);` `}` `function btnUpload_onclick()` `{` `  var strstrHTTPServer = location.hostname; //The name of the HTTP server.` `  var CurrentPathName = unescape(location.pathname);` `  var CurrentPath = CurrentPathName.substring(0,` `  CurrentPathName.lastIndexOf("/") + 1);` `  var strActionPage = CurrentPath + "SaveToFile.aspx";`

Example	 ``` DWObject.IfSSL = false; // Set whether SSL is used DWObject.HTTPPort = location.port == "" ? 80 : location.port; // Upload all the images in the buffer to the // HTTP server as a TIFF file asynchronously DWObject.HTTPUploadAllThroughPostAsMultiPageTIFF(strstrHTTPServer, strActionPage, "imageData.tif", OnHttpUploadSuccess, OnHttpUploadFailure); } //SaveToFile.aspx: <%@ Page Language="C#" %> <% try{ String strImageName; HttpFileCollection files = HttpContext.Current.Request.Files; HttpPostedFile uploadfile = files["RemoteFile"]; strImageName = uploadfile.FileName; uploadfile.SaveAs(Server.MapPath(".") + "\\UploadedImages\\" + strImageName); } catch{ } %> ```
Usage notes	If you want to use this method to upload / download files through HTTPS, please don't forget to set IfSSL to true and set the correct HTTPPort.
Availability	v4.0+

HTTPUploadAllThroughPostAsPDF()	
Uploads all of the images in the buffer to a specified HTTP server as a PDF file.	
Syntax	.HTTPUploadAllThroughPostAsPDF(strHTTPServer, actionPage, fileName, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` strHTTPServer: the name of the HTTP server. For example: "www.dynamsoft.com", "localhost", "127.0.0.1". `string` actionPage: the relative path for the action page on the server. The action page will receive and process the uploaded image stream. `string` fileName: the file name to be saved on the server. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<onHttpUploadSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is uploaded successfully. `<onHttpUploadFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be uploaded. Please refer to the function prototype onHttpUploadSuccess or onHttpUploadFailure.
Return value	`Boolean, only valid when used synchronously.`
	 `` function btnScan_onclick() { if (DWObject) { DWObject.SelectSource(); DWObject.OpenSource(); DWObject.IfDisableSourceAfterAcquire = true; DWObject.AcquireImage(); } } // OnHttpUploadSuccess and OnHttpUploadFailure are callback functions. function OnHttpUploadSuccess() { console.log('successful'); } function OnHttpUploadFailure(errorCode, errorString, sHttpResponse) { ```

Example	<pre>alert(errorString + sHttpResponse); } function btnUpload_onclick() { var strstrHTTPServer = location.hostname; //The name of the HTTP server. var CurrentPathName = unescape(location.pathname); var CurrentPath = CurrentPathName.substring(0, CurrentPathName.lastIndexOf("/") + 1); var strActionPage = CurrentPath + "SaveToFile.aspx"; DWObject.IfSSL = false; // Set whether SSL is used DWObject.HTTPPort = location.port == "" ? 80 : location.port; // Upload all of the images in Dynamic Web TWAIN viewer // to the HTTP server as a PDF file asynchronously DWObject.HTTPUploadAllThroughPostAsPDF(strstrHTTPServer, strActionPage, "imageData.pdf", OnHttpUploadSuccess, OnHttpUploadFailure); } //SaveToFile.aspx: <%@ Page Language="C#" %> <% try{ String strImageName; HttpFileCollection files = HttpContext.Current.Request.Files; HttpPostedFile uploadfile = files["RemoteFile"]; strImageName = uploadfile.FileName; uploadfile.SaveAs(Server.MapPath(".") + "\\UploadedImages\\" + strImageName); } catch{ } %></pre>
Usage notes	If you want to use this method to upload / download files through HTTPS, please don't forget to set IfSSL to true and set the correct HTTPPort.
Availability	v5.0+

HTTPUploadThroughPost()	
Uploads the image of a specified index in the buffer to a specified HTTP server.	
Syntax	.HTTPUploadThroughPost(strHTTPServer, sImageIndex, actionPage, fileName, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` strHTTPServer: the name of the HTTP server. For example: "www.dynamsoft.com", "localhost", "127.0.0.1". `number` sImageIndex: the index of the image in the buffer. The index is 0-based. `string` actionPage: the relative path for the action page on the server. The action page will receive and process the uploaded image stream. `string` fileName: the file name to be saved on the server. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<onHttpUploadSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is uploaded successfully. `<onHttpUploadFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be uploaded. Please refer to the function prototype onHttpUploadSuccess or onHttpUploadFailure.
Return value	`Boolean, only valid when used synchronously.`
	<pre>function btnScan_onclick() { if (DWObject) { DWObject.SelectSource();</pre>

Example	<pre>
DWObject.OpenSource();
DWObject.IfDisableSourceAfterAcquire = true;
DWObject.AcquireImage();
}
}

// OnHttpUploadSuccess and OnHttpUploadFailure are callback functions.
function OnHttpUploadSuccess() {
console.log('successful');
}
function OnHttpUploadFailure(errorCode, errorString, sHttpResponse) {
alert(errorString + sHttpResponse);
}

function btnUpload_onclick()
{
var strstrHTTPServer = location.hostname; //The name of the HTTP server.
var CurrentPathName = unescape(location.pathname);
var CurrentPath = CurrentPathName.substring(0,
CurrentPathName.lastIndexOf("/") + 1);
var strActionPage = CurrentPath + "SaveToFile.aspx";
DWObject.IfSSL = false; // Set whether SSL is used
DWObject.HTTPPort = location.port == "" ? 80 : location.port;

// Upload the image of a specified index in
// Dynamic Web TWAIN viewer to the HTTP server asynchronously
if (document.getElementById("JPEG").checked) {
DWObject.HTTPUploadThroughPost(
strstrHTTPServer,
DWObject.CurrentImageIndexInBuffer,
strActionPage,
"imageData.jpg",
OnHttpUploadSuccess,
OnHttpUploadFailure
);
}
if (document.getElementById("PNG").checked) {
DWObject.HTTPUploadThroughPost(
strstrHTTPServer,
DWObject.CurrentImageIndexInBuffer,
strActionPage,
"imageData.png",
OnHttpUploadSuccess,
OnHttpUploadFailure
);
}
if (document.getElementById("BMP").checked) {
DWObject.HTTPUploadThroughPost(
strstrHTTPServer,
DWObject.CurrentImageIndexInBuffer,
strActionPage,
"imageData.bmp",
OnHttpUploadSuccess,
OnHttpUploadFailure
);
}
}

//SaveToFile.aspx:
<%@ Page Language="C#" %>
<%
try{
String strImageName;
HttpFileCollection files = HttpContext.Current.Request.Files;
HttpPostedFile uploadfile = files["RemoteFile"];
strImageName = uploadfile.FileName;
uploadfile.SaveAs(Server.MapPath(".") + "\\UploadedImages\\" + strImageName);
}
catch{
}
%></pre> |
| Usage notes | Dynamic Web TWAIN processes the image format according to the extension of fileName.
If you want to use this method to upload / download files through HTTPS, please don't forget to set IfSSL to true and set the correct HTTPPort. |

Availability	v4.0+

HTTPUploadThroughPostAsMultiPagePDF()

Uploads the selected images in the buffer to a specified HTTP server as a PDF file.

Syntax	.HTTPUploadThroughPostAsMultiPagePDF(strHTTPServer, actionPage, fileName, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` strHTTPServer: the name of the HTTP server. For example: "www.dynamsoft.com", "localhost", "127.0.0.1". `string` actionPage: the relative path for the action page on the server. The action page will receive and process the uploaded image stream. `string` fileName: the file name to be saved on the server. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<onHttpUploadSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is uploaded successfully. `<onHttpUploadFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be uploaded. Please refer to the function prototype onHttpUploadSuccess or onHttpUploadFailure.
Return value	`Boolean, only valid when used synchronously.`
Example	

```
function btnScan_onclick()
{
    if (DWObject) {
        DWObject.SelectSourceByIndex(document.getElementById("source").selectedIndex);
        DWObject.OpenSource();
        DWObject.IfDisableSourceAfterAcquire = true;
        DWObject.AcquireImage();
    }
}


// OnHttpUploadSuccess and OnHttpUploadFailure are callback functions.
function OnHttpUploadSuccess() {
    console.log('successful');
}
function OnHttpUploadFailure(errorCode, errorString, sHttpResponse) {
    alert(errorString + sHttpResponse);
}

function btnUpload_onclick()
{
    var strstrHTTPServer = location.hostname; //The name of the HTTP server.
    var CurrentPathName = unescape(location.pathname);
    var CurrentPath = CurrentPathName.substring(0,
    CurrentPathName.lastIndexOf("/") + 1);
    var strActionPage = CurrentPath + "SaveToFile.aspx";
    DWObject.IfSSL = false; // Set whether SSL is used
    DWObject.HTTPPort = location.port == "" ? 80 : location.port;

    DWObject.SelectedImagesCount = 3;
    DWObject.SetSelectedImageIndex(0, 0);
    // Set the 1st image as the first selected image.
    DWObject.SetSelectedImageIndex(1, 2);
    // Set the 3rd image as the second selected image.
    DWObject.SetSelectedImageIndex(2, 4);
    // Set the 5th image as the third selected image.
    DWObject.GetSelectedImagesSize(4);
    // 4 - PDF format. Calculate the size of selected images in PDF format.


    // Upload the selected images to the server asynchronously
    DWObject.HTTPUploadThroughPostAsMultiPagePDF(
      strstrHTTPServer,
      strActionPage,
      "imageData.pdf",
      OnHttpUploadSuccess,
      OnHttpUploadFailure
    );
}
```

	`` //SaveToFile.aspx: <%@ Page Language="C#" %> <% try{ String strImageName; HttpFileCollection files = HttpContext.Current.Request.Files; HttpPostedFile uploadfile = files["RemoteFile"]; strImageName = uploadfile.FileName; uploadfile.SaveAs(Server.MapPath(".") + "\\UploadedImages\\" + strImageName); } catch{ } %> ```
Usage notes	If you want to use this method to upload / download files through HTTPS, please don't forget to set IfSSL to true and set the correct HTTPPort.
Availability	v6.0+

HTTPUploadThroughPostAsMultiPageTIFF()	
Uploads selected images in the buffer to the HTTP server as a TIFF file.	
Syntax	.HTTPUploadThroughPostAsMultiPageTIFF(strHTTPServer, actionPage, fileName, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` strHTTPServer: the name of the HTTP server. For example: "www.dynamsoft.com", "localhost", "127.0.0.1". `string` actionPage: the relative path for the action page on the server. The action page will receive and process the uploaded image stream. `string` fileName: the file name to be saved on the server. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<onHttpUploadSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is uploaded successfully. `<onHttpUploadFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be uploaded. Please refer to the function prototype onHttpUploadSuccess or onHttpUploadFailure.
Return value	`Boolean, only valid when used synchronously.`
	`` function btnScan_onclick() { if (DWObject) { DWObject.SelectSourceByIndex(document.getElementById("source").selectedIndex); DWObject.OpenSource(); DWObject.IfDisableSourceAfterAcquire = true; DWObject.AcquireImage(); } } // OnHttpUploadSuccess and OnHttpUploadFailure are callback functions. function OnHttpUploadSuccess() { console.log('successful'); } function OnHttpUploadFailure(errorCode, errorString, sHttpResponse) { alert(errorString + sHttpResponse); } function btnUpload_onclick() { var strstrHTTPServer = location.hostname; //The name of the HTTP server. var CurrentPathName = unescape(location.pathname); var CurrentPath = CurrentPathName.substring(0, CurrentPathName.lastIndexOf("/") + 1); var strActionPage = CurrentPath + "SaveToFile.aspx"; DWObject.IfSSL = false; // Set whether SSL is used DWObject.HTTPPort = location.port == "" ? 80 : location.port; DWObject.SelectedImagesCount = 3; DWObject.SetSelectedImageIndex(0, 0); // Set the 1st image as the first selected image. ```

Example	 ``` DWObject.SetSelectedImageIndex(1, 2); // Set the 3rd image as the second selected image. DWObject.SetSelectedImageIndex(2, 4); // Set the 5th image as the third selected image. DWObject.GetSelectedImagesSize(2); // 2 - TIFF format. Calculate the size of selected images in TIFF format. // Upload the selected images to the server asynchronously DWObject.HTTPUploadThroughPostAsMultiPageTIFF(strstrHTTPServer, strActionPage, "imageData.tiff", OnHttpUploadSuccess, OnHttpUploadFailure); } //SaveToFile.aspx: <%@ Page Language="C#" %> <% try{ String strImageName; HttpFileCollection files = HttpContext.Current.Request.Files; HttpPostedFile uploadfile = files["RemoteFile"]; strImageName = uploadfile.FileName; uploadfile.SaveAs(Server.MapPath(".") + "\\UploadedImages\\" + strImageName); } catch{ } %> ```
Usage notes	If you want to use this method to upload / download files through HTTPS, please don't forget to set IfSSL to true and set the correct HTTPPort.
Availability	v6.0+

HTTPUploadThroughPostDirectly()	
Upload any type of file to the server.	
Syntax	.HTTPUploadThroughPostDirectly(strHTTPServer, strLocalFile, actionPage, fileName, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` strHTTPServer: the name of the HTTP server. For example: "www.dynamsoft.com", "localhost", "127.0.0.1". `string` strLocalFile: the path of the file to upload. `string` actionPage: the relative path for the action page on the server. The action page will receive and process the uploaded file stream. `string` fileName: the file name to be used when saving the file on the server. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<onHttpUploadSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is uploaded successfully. `<onHttpUploadFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be uploaded. Please refer to the function prototype onHttpUploadSuccess or onHttpUploadFailure.
Return value	`Boolean, only valid when used synchronously.`
	`` // OnHttpUploadSuccess and OnHttpUploadFailure are callback functions. function OnHttpUploadSuccess() { console.log('successful'); } function OnHttpUploadFailure(errorCode, errorString, sHttpResponse) { alert(errorString + sHttpResponse); } function btnUpload_onclick() { var strstrHTTPServer = location.hostname; //The name of the HTTP server. ```

Example

```
    var CurrentPathName = unescape(location.pathname);
    var CurrentPath = CurrentPathName.substring(0,
    CurrentPathName.lastIndexOf("/") + 1);
    var strActionPage = CurrentPath + "SaveToFile.aspx";
    DWObject.IfSSL = false; // Set whether SSL is used
    DWObject.HTTPPort = location.port == "" ? 80 : location.port;


    // Uploads a specific image in the buffer to
    // the HTTP server in a specified image format asynchronously
    DWObject.HTTPUploadThroughPostDirectly(
      strstrHTTPServer,
      "D:\\DWT.jpg",
      strActionPage,
      "imageData.jpg",
      OnHttpUploadSuccess,
      OnHttpUploadFailure
    );
}



//SaveToFile.aspx:
<%@ Page Language="C#" %>
<%
    try{
        String strImageName;
        HttpFileCollection files = HttpContext.Current.Request.Files;
        HttpPostedFile uploadfile = files["RemoteFile"];
        strImageName = uploadfile.FileName;
        uploadfile.SaveAs(Server.MapPath(".") + "\\UploadedImages\\" + strImageName);
    }
    catch{
    }
%>
```

Usage notes	If you want to use this method to upload / download files through HTTPS, please don't forget to set IfSSL to true and set the correct HTTPPort.
Availability	v7.0+

HTTPUploadThroughPostEx()

Uploads the image of a specified index in the buffer to the HTTP server in a specified image format.

Syntax	.HTTPUploadThroughPostEx(strHTTPServer, sImageIndex, actionPage, fileName, imageType, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` strHTTPServer: the name of the HTTP server. For example: "www.dynamsoft.com", "localhost", "127.0.0.1". `number` sImageIndex: the index of the image in the buffer. The index is 0-based. `string` actionPage: the relative path for the action page on the server. The action page will receive and process the uploaded image stream. `string` fileName: the file name to be saved on the server. `EnumDWT_ImageType` imageType: the format of the image.

Type	Numeric Value
IT_BMP	0
IT_JPG	1
IT_TIF	2
IT_PNG	3
IT_PDF	4

The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous.

`<onHttpUploadSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is uploaded successfully.

	`<onHttpUploadFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be uploaded. Please refer to the function prototype onHttpUploadSuccess or onHttpUploadFailure.
Return value	`Boolean, only valid when used synchronously.`
Example	 ``` function btnScan_onclick() { if (DWObject) { DWObject.SelectSourceByIndex(document.getElementById("source").selectedIndex); DWObject.OpenSource(); DWObject.IfDisableSourceAfterAcquire = true; DWObject.AcquireImage(); } } // OnHttpUploadSuccess and OnHttpUploadFailure are callback functions. function OnHttpUploadSuccess() { console.log('successful'); } function OnHttpUploadFailure(errorCode, errorString, sHttpResponse) { alert(errorString + sHttpResponse); } function btnUpload_onclick() { var strstrHTTPServer = location.hostname; //The name of the HTTP server. var CurrentPathName = unescape(location.pathname); var CurrentPath = CurrentPathName.substring(0, CurrentPathName.lastIndexOf("/") + 1); var strActionPage = CurrentPath + "SaveToFile.aspx"; DWObject.IfSSL = false; // Set whether SSL is used DWObject.HTTPPort = location.port == "" ? 80 : location.port; // Uploads a specific image in the buffer to the HTTP // server in a specified image format asynchronously DWObject.HTTPUploadThroughPostEx(strstrHTTPServer, DWObject.CurrentImageIndexInBuffer, strActionPage, "imageData.jpg", EnumDWT_ImageType.IT_JPG, OnHttpUploadSuccess, OnHttpUploadFailure); } //SaveToFile.aspx: <%@ Page Language="C#" %> <% try{ String strImageName; HttpFileCollection files = HttpContext.Current.Request.Files; HttpPostedFile uploadfile = files["RemoteFile"]; strImageName = uploadfile.FileName; uploadfile.SaveAs(Server.MapPath(".") + "\\UploadedImages\\" + strImageName); } catch{ } %> ```
Usage notes	If you want to use this method to upload / download files through HTTPS, please don't forget to set IfSSL to true and set the correct HTTPPort.
Availability	v5.0+

SetHTTPFormField()	
Sets up a field in the web form which will be sent to the server once an upload is initiated. You can set multiple fields by calling this method mutliple times.	
	.SetHTTPFormField(fieldName, fieldValue);

	.SetHTTPFormField(fieldName, blobValue, optional fileName);
Parameters	`string` fieldName: the name of the field which can later be used on the server to retrieve the field value. `string` fieldValue: the field value. `string` fieldName: the name of the field which can later be used on the server to retrieve the blob. `Blob` blobValue: the binary data to be uploaded. `string` fileName: the name to be used when saving the binary on the server.
Return value	`boolean`
Example	`DWObject.ClearAllHTTPFormField();` `DWObject.SetHTTPFormField("PDFFileName", "test.pdf");` `//Get action page path` `var CurrentPathName = unescape(location.pathname);` `var CurrentPath = CurrentPathName.substring(0, CurrentPathName.lastIndexOf("/") + 1);` `var strActionPage = CurrentPath + "SaveToFile.aspx";` `DWObject.HTTPUploadAllThroughPostAsMultiPageTIFF(` `"www.dynamsoft.com",` `strActionPage,` `"test.tiff"` `);` `//C# Action Page Part:` `System.Web.HttpContext.Current.Request.Form["PDFFileName"];`
Usage notes	It was improved to be able to set Blobs in HTTP Forms aside from strings from v13.1.
Availability	v5.0+

SetUploadSegment()	
Specifies when an upload should be segmented and the size of the segments.	
Syntax	.SetUploadSegment(segementUploadThreshold, moduleSize);
Parameters	`number` segementUploadThreshold: the threshold (in MB) over which segmented upload will be invoked. `number` moduleSize: the size of each segment (in KB).
Return value	`boolean`
Availability	v12.1+

SetHTTPHeader()	
Adds a header to an http(s) post request. This header will be cleared every time a HTTPUpload/HTTPDownload request is done.	
Syntax	.SetHTTPHeader(HeaderName, HeaderValue);
Parameters	`string` HeaderName: the name of the header which can later be used on the server to retrieve the header value. `string` HeaderValue: the header value.
Return value	`boolean`
Example	`DWObject.SetHTTPHeader('dwt_CustomHeader','dynamsoft');` `DWObject.HTTPUpload ('www.dynamsoft.com/SaveToFile.aspx?filename=001.pdf',` `[0,1], EnumImageType.PDF,` `EnumDWT_UploadDataFormat.BINARY, true, OnHttpUploadSuccess, OnHttpUploadFailure);` `function OnHttpUploadSuccess (httpResponse) {` `console.log("HTTPResponseString: " + httpResponse);` `}`

Example	``` } function OnHttpUploadFailure (errorCode, errorString, httpResponse) { alert("ErrorCode: " + errorCode+ "ErrorString: " + errorString + "HTTPResponseString: " + httpResponse); } ```
Usage notes	This method is valid only in the HTML5 Edition.
Availability	v12.0+

Properties

FTPPassword	
Returns or sets the password used to log into the FTP server.	
Type	`string`
Accessors	`Get Set`
Usage notes	If you are logging into a FTP server as "anonymous", then you don't need to set FTPUserName or FTPUserPassword.
Availability	v5.2+

FTPPort	
Returns or sets the port number of the FTP server to be used for uploading/downloading.	
Type	`number`
Accessors	`Get Set`
Usage notes	The default FTP port number is 21.
Availability	v5.2+

FTPUserName	
Returns or sets the username used to log into a FTP server.	
Type	`string`
Accessors	`Get Set`
Usage notes	If you are logging into a FTP server as "anonymous", then you don't need to set FTPUserName or FTPUserPassword.
Availability	v5.2+

HttpFieldNameOfUploadedImage	
Returns or sets the field name of the uploaded image when uploading through POST.	
Type	`string`
Accessors	`Get Set`
Usage notes	The default value is "RemoteFile".
Availability	v6.0+

HTTPPort	
Returns or sets the port number to be used for uploading via HTTP.	
Type	`number`
Accessors	`Get Set`
Usage notes	The default HTTP port number is 80. This property is typically used with (IfSSL) [#IfSSL].
Availability	v4.2.1+

HTTPPostResponseString	
Returns the response string from the HTTP server if an error occurred after calling one of the HTTPUploadThroughPost*** methods.	
Type	`string`
Accessors	`Get`
Usage notes	This is a read-only property.
Availability	v3.0.3+

IfPASVMode	
Returns or sets whether FTP passive mode is enabled.	
Type	`boolean`
Accessors	`Get Set`
Usage notes	The default value is true. When the value of the property is true, FTP passive mode is enabled. When the value of the property is set to false, FTP active mode is enabled.
Availability	v6.0+

IfShowCancelDialogWhenImageTransfer	
Returns or sets whether to show the cancel dialog when uploading images to server. The default value is true.	
Type	`boolean`
Accessors	`Get Set`
Usage	In the HTML5 Edition, this property only works with asynchronous uploading. The IfShowProgressBar property works for any image encoding/decoding related methods. For LoadImage,LoadImageEx, etc.

Property	Value	HTML5 Edition	Non-H Editions (Plug
IfShowCancelDialogWhenImageTransfer	true	Upload/Download: The progress bar will appear.	Upload/Dov The progres appear.
	false	Upload/Download: The progress bar will not appear.	Upload/Dov The progres not appear.
			Save/Load:

Usage notes	IfShowProgressBar	true	Save/Load: The progress bar will appear.	Save/Load: The progres appear. Upload/Dow (Encoding/D The progres appear.
		false	Save/Load: The progress bar will not appear.	Save/Load: The progres not appear. Upload/Dow (Encoding/D The progres not appear.
Availability	v5.2+			

IfSSL	
Returns or sets whether SSL is used when uploading or downloading images.	
Type	`number`
Accessors	`Get Set`
Usage notes	The default value is false. This property is typically used with HTTPPort. If IfSSL is set to true, make sure HTTPPort is set to a secure port like 443.
Availability	v5.2+

MaxUploadImageSize	
Returns or sets the size limit for uploading images.	
Type	`number`
Accessors	`Get Set`
Usage notes	The default value is -1 which indicates there is no limit over the upload size. The unit is 'Byte'.
Availability	v5.2+

Events

OnInternetTransferPercentage	
This event is triggered during image uploading or downloading.	
Syntax	.RegisterEvent('OnInternetTransferPercentage',function(sPercentage){...});
Arguments	• `number` `sPercentage` : the percentage that has been uploaded or downloaded.
Example	`DWObject.RegisterEvent('OnInternetTransferPercentage', function(sPercentage) {` `  console.log(sPercentage);` `});`

Usage notes	
Availability	v5.0+

Advanced Scan

Methods		
CancelAllPendingTransfers()	CloseSourceManager()	CloseWorkingProcess()
FeedPage()	GetCustomDSData()	GetCustomDSDataEx()
GetDeviceType()	GetSourceNameItems()	OpenSourceManager()
ResetImageLayout()	RewindPage()	SetCustomDSData()
SetCustomDSDataEx()	SetFileXferInfo()	SetImageLayout()

Properties		
Brightness	Contrast	CurrentSourceName
DataSourceStatus	DefaultSourceName	Duplex
IfAutoBright	IfAutoDiscardBlankpages	IfAutoFeed
IfAutomaticBorderDetection	IfAutomaticDeskew	IfAutoScan
IfFeederLoaded	IfPaperDetectable	IfShowIndicator
IfUIControllable	IfUseTwainDSM	PendingXfers
PixelFlavor	TransferMode	Unit
XferCount		

Events		
OnPreAllTransfers	OnPreTransfer	OnSourceUIClose

Methods

CancelAllPendingTransfers()

Cancels all pending transfers.

Syntax	.CancelAllPendingTransfers();
Parameters	`None`
Return value	`boolean`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes: This method is only valid in the events `OnPreAllTransfers`, `OnPreTransfer` and `OnPostTransfer`.

CloseSourceManager()

Closes and unloads Data Source Manager.

Syntax	.CloseSourceManager();
Parameters	`None`
Return value	`boolean`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	If this method is not called explicitly by the application, it'll be called automatically when the browser or tab closes.

CloseWorkingProcess()

Closes the scanning process to release resources on the machine.

Syntax	.CloseWorkingProcess();
Parameters	`None`
Return value	`boolean`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v11.2	✓ | v11.2	✓ | v11.2	✓ | v11.2	✓ | v12.1

Usage notes	In the HTML5 edition, Dynamic Web TWAIN uses a separate process to communicate with the scanners. When it's not scanning, you can choose to close this process to release the resources (CPU, memory, etc.) used on the machine.

FeedPage()

Ejects the current page and begins scanning the next page in the document feeder.

Syntax	.FeedPage();
Parameters	`None`
Return value	`boolean`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✗	✗

Usage notes	Use this method after `OpenSource()` is called and make sure IfFeederEnabled is `true`.

GetCustomDSData()

Gets custom DS data and saves the data in a specified file.

Syntax	.GetCustomDSData(fileName);
Parameters	`string filename`
Return value	`boolean`
Availability	ActiveX: ✓ | v10.0 H5(Windows): ✓ | v10.0 H5(macOS/TWAIN): ✓ | v11.0 H5(macOS/ICA): ✗ H5(Linux): ✗
Usage notes	The path of the file used for storing custom DS data. For example `DWObject.GetCustomDSData("C:\\customDSData");`

GetCustomDSDataEx()

Gets custom DS data and returns it in a base64 encoded string.

Syntax	.GetCustomDSDataEx();
Availability	ActiveX: ✓ | v10.0 H5(Windows): ✓ | v10.0 H5(macOS/TWAIN): ✓ | v11.0 H5(macOS/ICA): ✗ H5(Linux): ✗
Return value	A base64 encoded string which represents the custom DS data.

GetDeviceType()

Retrieve the device type of the currently selected Data Source, it might be a scanner, a web camera, etc.

Syntax	.GetDeviceType();
Parameters	`None`
Return value	`number`
Availability	ActiveX: ✓ | v7.0 H5(Windows): ✓ | v10.0 H5(macOS/TWAIN): ✓ | v11.0 H5(macOS/ICA): ✗ H5(Linux): ✗
Usage notes	Please find below all available numbers and the descriptions: 0: Failed, please check `ErrorString` for details. 1: Digital Camera 2: Flatbed-only scanner 3: Flatbed, feeder, auto feed 4: Feeder-only scanner, auto feed 5: Flatbed, feeder, without auto feed 6: Feeder-only scanner, without auto feed 7: Webcam

GetSourceNameItems()

Gets the name of a Data Source by its index in Data Source Manager Source list.

Syntax	.GetSourceNameItems(short index);
Parameters	`short index` The index of a Data Source in Data Source Manager Source list. For example `DWObject.GetSourceNameItems(0);`
Return value	`string`
Availability	**ActiveX** | **H5(Windows)** | **H5(macOS/TWAIN)** | **H5(macOS/ICA)** | **H5(Linux)** ✓ | v7.0 | ✓ | v10.0 | ✓ | v11.0 | ✓ | v11.0 | ✓ | v12.1
Usage notes	This method replaces `SourceNameItems` in old versions of Dynamic Web TWAIN.

OpenSourceManager()

Loads and opens Data Source Manager.

Syntax	.OpenSourceManager();
Parameters	`None`
Return value	`boolean`
Availability	**ActiveX** | **H5(Windows)** | **H5(macOS/TWAIN)** | **H5(macOS/ICA)** | **H5(Linux)** ✓ | v5.2 | ✓ | v10.0 | ✓ | v11.0 | ✓ | v11.0 | ✓ | v12.1
Usage notes	If application identification information needs to be set, it should be set before OpenSourceManager(). Dynamic Web TWAIN has built-in Wizard Mode. With Wizard Mode, OpenSourceManager() is called automatically when needed by methods like EnableSource() or SelectSource().

ResetImageLayout()

Reset the image layout in the Data Source back to default.

Syntax	.ResetImageLayout();
Parameters	`None`
Return value	`boolean`
Availability	**ActiveX** | **H5(Windows)** | **H5(macOS/TWAIN)** | **H5(macOS/ICA)** | **H5(Linux)** ✓ | v5.2 | ✓ | v10.0 | ✓ | v11.0 | ✗ | ✗
Usage notes	To set the image layout manually, you can use SetImageLayout().

RewindPage()

If IfFeederEnabled property is `true` , the Source will return the current page to the input area and return the last

page from the output area into the acquisition area.

Syntax	.RewindPage();
Parameters	`None`
Return value	`boolean`
Availability	**ActiveX**: ✓ | v5.2 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✗ **H5(Linux)**: ✗
Usage notes	Use this method after `OpenSource()` method and make sure IfFeederEnabled property is `true`.

SetCustomDSData()

Sets custom DS data to be used for scanning, the data is stored in a file. Custom DS data means a specific scanning profile.

Syntax	.SetCustomDSData(string fileName);
Parameters	`string filename` : The absolute path of the file where the custom Data Source data is stored.
Return value	`boolean`
Availability	**ActiveX**: ✓ | v10.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✗ **H5(Linux)**: ✗
Usage notes	If you want to get custom DS data and save the data in a local file, you can check `GetCustomDSData()` method or `GetCustomDSDataEx()` method.

SetCustomDSDataEx()

Sets custom DS data to be used for scanning, the input string is base64 encoded. Custom DS data means a specific scanning profile.

Syntax	.SetCustomDSDataEx(string strBase64);
Parameters	`string strBase64` : The input string which is base64 encoded.
Return value	`boolean`
Availability	**ActiveX**: ✓ | v10.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✗ **H5(Linux)**: ✗
Usage notes	If you want to get custom DS data and save the data in a local file, you can check `GetCustomDSData()` method or `GetCustomDSDataEx()` method.

SetFileXferInfo()

Sets the absolute file name (including its path) and file format to be used in File Transfer Mode. Note that this

method is valid only in File Transfer Mode. In File Transfer Mode, the image is transferred to the designated location on the disk directly instead of Dynamic Web TWAIN.

Syntax	.SetFileXferInfo(string fileName, EnumDWT_FileFormat fileFormat);
Parameters	`string` `fileName` The name of the file to be used in the transfer. For example: "C:\\webtwain.jpg". When you writ the name like this string + %d + string, then when you scan multiple files, the name will automatically change. For example: "C:\\webtwain" + <> + ".jpg" will result in "C:\\webtwain0.jpg","C:\\webtwain1.jpg", "C:\\webtwain2.jpg", etc. By doing this, you are now able to scan multiple files to the disk. `EnumDWT_FileFormat` `fileFormat` : for all available values, please check this page
Return value	`boolean`
Example	`DWObject.OpenSource();` `DWObject.TransferMode = EnumDWT_TransferMode.TWSX_FILE;` `if(DWObject.TransferMode == EnumDWT_TransferMode.TWSX_FILE){` `  if(DWObject.SetFileXferInfo("C:\\Temp\\WebTWAIN<>.bmp",` `  EnumDWT_FileFormat.TWFF_BMP)){` `  DWObject.IfShowUI = true;` `  DWObject.AcquireImage();` `  }` `}`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✗	✗

Usage notes	Make sure the format you set is supported by the Data Source (your scanner).

SetImageLayout()

Sets the left, top, right, and bottom sides of the image layout rectangle for the current Data Source. In other words, sets which area should be scanned. The image layout rectangle defines what portion of the Data Source's scanning area is acquired. Note that you should set Unit property before using this method.

Syntax	.SetImageLayout(left, top, right, bottom);
Parameters	`number` `left` : specifies the left side of the image layout rectangle . `number` `top` : specifies the top side of the image layout rectangle . `number` `right` : specifies the right side of the image layout rectangle . `number` `bottom` : specifies the bottom side of the image layout rectangle .
Return value	`boolean`
Example	`DWObject.SelectSource();` `DWObject.OpenSource();` `DWObject.IfShowUI = false;` `DWObject.Unit=EnumDWT_UnitType.TWUN_PIXELS;` `DWObject.SetImageLayout(50, 50, 100, 100);` `DWObject.AcquireImage();`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✗	✗

Usage notes	This method uses Capability Negotiation underneath, thus is device-dependent. If a device doesn't support the customization of the scan area, this method might not work correctly.

Properties

Brightness

Returns or sets Brightness to be used for scanning by the Data Source.

Type	`number`
Accessors	`Get Set`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✗	✓ | 12.1

Usage notes: Set this property after `OpenSource()` and before `AcquireImage()`.
Typically the value to be set is -1000 ~ 1000.

Contrast

Returns or sets Contrast to be used for scanning by the Data Source.

Type	`number`
Accessors	`Get Set`
Usage notes	Set this property after `OpenSource()` and before `AcquireImage()`. Typically the value to be set is -1000 ~ 1000.

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✗	✓ | 12.1

Usage notes: Set this property after `OpenSource()` and before `AcquireImage()`.
Typically the value to be set is -1000 ~ 1000.

CurrentSourceName

Returns the device name of current source. This is a read-only property.

Type	`string`
Accessors	`Get`
Usage notes	If no source is currently selected, CurrentSourceName property returns "".

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	If no source is currently selected, CurrentSourceName property returns "".

DataSourceStatus

Returns a value which indicates the Data Source status. This is a read-only property.

Type	`number`
Accessors	`Get`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes

Allowed values :
0: The Data Source is closed
1: The Data Source is opened
2: The Data Source is enabled
3: The Data Source is acquiring images.

DefaultSourceName

Returns the name of the default source. This is a read-only property.

Type	`string`
Accessors	`Get`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✗	✗

Duplex

Returns whether the source supports duplex. If yes, it further returns the level of duplex the Source supports (one pass or two pass duplex). This is a read-only property.

Type	`EnumDWT_DUPLEX (int)`
Accessors	`Get`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✗	✗

Usage notes

Available values.
TWDX_NONE (0): duplex is not supported TWDX_1PASSDUPLEX (1): 1-pass duplex
TWDX_2PASSDUPLEX (2): 2-pass duplex

IfAutoBright

Returns or sets whether to enable the Source's Auto-brightness feature.

Type	`boolean`
Accessors	`Get Set`

Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v5.2</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✗</td><td>✗</td></tr></table>
Usage notes	The default value is `false`.

IfAutoDiscardBlankpages

Returns or sets whether the Data Source (the scanner) discards blank images during scanning.

Type	`boolean`
Accessors	`Get Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v7.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✗</td><td>✗</td></tr></table>
Usage notes	The property works only if the device and its driver supports discarding blank pages. You can find whether your device supports this capbility from its user manual. Or, you can use the built-in methods of Dynamic Web TWAIN to detect blank images: IsBlankImageExpress.

IfAutoFeed

Returns or sets whether to enable the Data Source's automatic document feeding process.

Type	`number`
Accessors	`Get Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v7.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✗</td><td>✗</td></tr></table>
Usage notes	If the value of this property is `true`, the Data Source will automatically feed the next page from the document feeder after the previous page is acquired.

IfAutomaticBorderDetection

Returns or sets whether to enable the Data Source's automatic border detection feature.

Type	`boolean`
Accessors	`Get Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v7.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✗</td><td>✗</td></tr></table>
Usage notes	The property works only if the device and its driver support detecting the border automatically. You can find whether your device supports this capbility from its user manual. If the value of this property is `true`, the Data Source (scanner) will automatically detect the border of the document on the flatbed and will only scan the area within the border.

IfAutomaticDeskew

Returns or sets whether to enable the Data Source's automatic skew correction feature.

Type	`boolean`
Accessors	`Get Set`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v7.0	✓ | v10.0	✓ | v11.0	✗	✗

Usage notes

The property works only if the device and its driver supports deskewing automatically. You can find whether your device supports this capbility from its user manual.
If this property is set to `true`, the Data Source (scanner) will automatically correct the skew of the scanned images.

IfAutoScan

Returns or sets whether to enable the Data Source's automatic document scanning process.

Type	`boolean`
Accessors	`Get Set`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✗	✗

Usage notes

Please Make sure IfFeederEnabled property is `true` before use this property.

IfFeederLoaded

Returns whether or not there are documents loaded in the Data Source's feeder when IfFeederEnabled and IfPaperDetectable are `true`. This is a read-only property.

Type	`boolean`
Accessors	`Get`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✗	✗

Usage notes

Make sure `IfPaperDetectable` is `true` before use this property.

IfPaperDetectable

Returns whether the Source has a paper sensor that can detect documents on the ADF or Flatbed. This is a read-only property.

Type	`boolean`
Accessors	`Get`

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v5.2	✓ | v10.0	✓ | v11.0	✗	✗

IfShowIndicator

Returns or sets whether the Source displays a progress indicator during acquisition and transfer. This property works only when IfShowUI is set to `false` .

Type	`boolean`
Accessors	`Get Set`

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v5.2	✓ | v10.0	✓ | v11.0	✗	✗

Usage notes	If you set both IfShowUI and IfShowIndicator to `false` , The progress indicator will not appear; If you set IfShowUI to `false` but IfShowIndicator to `true` , The progress indicator will appear; If you set IfShowUI to `true` , this property will no longer work. The progress indicator always appears.

IfUIControllable

Returns whether the Data Source supports acquisitions with the UI (User Interface) disabled. ' `false` ' indicates that the Data Source only supports acquisitions with the UI enabled. This is a read-only property.

Type	`boolean`
Accessors	`Get`

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v5.2	✓ | v10.0	✓ | v11.0	✗	✗

Usage notes	Set this property after `OpenSource()` and before `AcquireImage()` . Typically the value to be set is -1000 ~ 1000.

IfUseTwainDSM

Returns or sets whether the new TWAIN DSM (Data Source Manager) is used for acquisitions. The new TWAIN DSM is a .dll file called 'TWAINDSM.dll'. By default, the old DSM is used ('twain_32.dll' under 'C:\Windows').

Type	`boolean`
Accessors	`Get Set`
Usage notes	Set this property before any TWAIN related methods or properties are used.

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v6.2	✓ | v10.0	✓ | v11.0	✗	✗

PendingXfers

Returns the number of transfers the Data Source is ready to supply upon demand. This is a read-only property.

Type	`Short`
Accessors	`Get Set`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✗	✗

Usage notes

This property is only valid in the event OnPostTransfer.
The Data Source returns -1 if it is not sure how many transfers are pending. Scanners with ADF (Automatic Document Feeder) typically return -1 if the current image is not the last one.

PixelFlavor

Returns or sets the pixel flavor to be used for acquiring images.

Type	`Short`
Accessors	`Get Set`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✗	✗

Usage notes

Available values:
0: Chocolate. Zero pixel represents darkest shade
1: Vanilla. Zero pixel represents lightest shade.

TransferMode

Returns or sets the Data Source's transfer mode.

Type

`enumTW_TRANSFERMODE`
TWSX_NATIVE: NATIVE mode is the default mode. In this mode, the whole image is transfered in a single memory block.
TWSX_FILE: DISK FILE mode is not required by TWAIN specification. In this mode, the image is transfered to a specified file directly. The disk file mode is ideal when transferring large images that might encounter memory limitations with Native mode.
TWSX_MEMORY: MEMORY mode is also required by TWAIN, like NATIVE mode. Although this mode is the most complex, Dynamic TWAIN handles the tranfer details, making it as simple as NATIVE mode.

Accessors	`Get Set`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✗	✗

Usage notes

The default value for TransferMode is TWSX_NATIVE (0).
For DISK FILE mode - TWSX_FILE (1), since it is not required by TWAIN, the application needs to make sure it is supported by the current Source. One way to do this is to check the TransferMode property after `OpenSource()` and see if it is still TWSX_FILE (1).

Unit

Returns or sets the unit of measure for all quantities.

Type	`EnumDWT_UnitType (short)`
Accessors	`Get Set`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✓ | v11.0	✗

Usage notes

Available values:
EnumDWT_UnitType.TWUN_INCHES: inches(Default)
EnumDWT_UnitType.TWUN_CENTIMETERS: centimeters
EnumDWT_UnitType.TWUN_PICAS: picas
EnumDWT_UnitType.TWUN_POINTS: points
EnumDWT_UnitType.TWUN_TWIPS: twips
EnumDWT_UnitType.TWUN_PIXELS: pixels
EnumDWT_UnitType.TWUN_MILLIMETERS: millimeters

XferCount

Returns and sets the number of images your application is willing to accept.

Type	`Short`
Accessors	`Get Set`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✗	✓ | v12.1

Usage notes

Allowed values are between -1 and 215. '-1' indicate multiple images.

Events

OnPreAllTransfers

This event is triggered when all images are scanned and ready to be transferred.

Syntax	.RegisterEvent('OnPreAllTransfers',function(){...});
Arguments	`None`

Example

```
DWObject.RegisterEvent('OnPreAllTransfers', function() {
    DWObject.CancelAllPendingTransfers();
});
```

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	Multiple transfers may occur when ADF(Auto Document Feeder) is enabled or there are more than one frames in a page. In those cases, `OnPreTransfer` is triggered multiple times but `OnPreAllTransfers` is triggerred just once. This is the place to call `CancelAllPendingTransfers()`.

OnPreTransfer

This event is triggered when a page has been scanned and ready to be transferred.

Syntax	.RegisterEvent('OnPreTransfer',function(){...});
Arguments	None
Example	`DWObject.RegisterEvent('OnPreTransfer', function() {` `  DWObject.CancelAllPendingTransfers();` `});`
Usage notes	This is where to check PendingXFERs, ImageLayoutDocumentNumber ImageLayoutFrameLeft ImageLayoutFrameTop, ImageLayoutFrameRight, ImageLayoutFrameBottom, ImageLayoutPageNumber, ImageLayoutFrameNumber. This is also the place to call CancelAllPendingTransfers().

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.2	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes	This is where to check PendingXFERs, ImageLayoutDocumentNumber ImageLayoutFrameLeft ImageLayoutFrameTop, ImageLayoutFrameRight, ImageLayoutFrameBottom, ImageLayoutPageNumber, ImageLayoutFrameNumber. This is also the place to call CancelAllPendingTransfers().

OnSourceUIClose

This event is triggered when the User Interface of the scanner source is closed manually by the user.

Syntax	.RegisterEvent('OnSourceUIClose',function(){...});
Arguments	None
Example	`DWObject.RegisterEvent('OnSourceUIClose', function() {` `  ...;` `});`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v8.0.1	✓ | v10.0	✓ | v11.0	✗	✗

Capability Negotiation

Methods		
CapGet()	CapGetCurrent()	CapGetDefault()
CapGetFrameBottom()	CapGetFrameLeft()	CapGetFrameRight()
CapGetFrameTop()	CapGetHelp()	CapGetLabel()
CapGetLabels()	CapIfSupported()	CapReset()
CapSet()	CapSetFrame()	GetCapItems()
GetCapItemsString()	SetCapItems()	SetCapItemsString()

Properties		
Capability	CapCurrentIndex	CapCurrentValue
CapDefaultIndex	CapDefaultValue	CapMaxValue
CapMinValue	CapNumItems	CapMinValue
CapStepSize	CapType	CapValue
CapValueString	CapValueType	

Code example

```
function getACapability(capability){// Specify a Capability with capability
    STR_CapValueType = [
        'TWTY_INT8', 'TWTY_INT16', 'TWTY_INT32', 'TWTY_UINT8', 'TWTY_UINT16', 'TWTY_int', 'TWTY_BOOL',
        'TWTY_FIX32', 'TWTY_FRAME', 'TWTY_STR32', 'TWTY_STR64', 'TWTY_STR128', 'TWTY_STR255'];

    DWObject.SelectSource();
    DWObject.OpenSource();
    DWObject.Capability = capability; // Specify a Capability
    DWObject.<code>CapGet()</code>;
    var i, nCapType = DWObject.CapType;
    switch (nCapType) {
        case EnumDWT_CapType.TWON_ARRAY/*3*/:
            console.log('Available Values:');
            for (i = 0; i < DWObject.CapNumItems; i++) {
                if (DWObject.CapValueType > 8) /* >8 is string*/
                /*STR*/console.log( DWObject.GetCapItemsString(i));
                else
                /*NUM*/console.log(DWObject.GetCapItems(i));
            }
            break;
        case EnumDWT_CapType.TWON_ENUMERATION/*4*/:
            console.log('Available Values:');
            for (i = 0; i < DWObject.CapNumItems; i++) {
                if (DWObject.CapValueType > 8)
                /*STR*/console.log(DWObject.GetCapItemsString(i));
                else
                /*NUM*/console.log(DWObject.GetCapItems(i));
            }
            if (DWObject.CapValueType > 8) {
```

```
                console.log('Current Index = ' + DWObject.CapCurrentIndex + ' (Value: ' + DWObject.GetCapItemsString(
DWObject.CapCurrentIndex) + ')');
                console.log('Default Index = ' + DWObject.CapDefaultIndex + ' (Value: ' + DWObject.GetCapItemsString(
DWObject.CapDefaultIndex) + ')');
            }
            else {
                console.log('Current Index = ' + DWObject.CapCurrentIndex + ' (Value: ' + DWObject.GetCapItems(DWObje
ct.CapCurrentIndex) + ')');
                console.log('Default Index = ' + DWObject.CapDefaultIndex + ' (Value: ' + DWObject.GetCapItems(DWObje
ct.CapDefaultIndex) + ')');
            }
            break;
        case EnumDWT_CapType.TWON_ONEVALUE/*5*/:
            var tempValue = '';
            if (DWObject.CapValueType > 8)
            /*STR*/tempValue = DWObject.CapValueString;
            else
            /*NUM*/tempValue = DWObject.CapValue;
            /*
            * Special for BOOL
            */
            if (DWObject.CapValueType == EnumDWT_CapValueType.TWTY_BOOL) {
                if (tempValue == 0) tempValue = 'FALSE'; else tempValue = 'TRUE';
            }
            console.log('ItemType = ' + STR_CapValueType[DWObject.CapValueType]);
            console.log('Value = ' + tempValue);
            break;
        case EnumDWT_CapType.TWON_RANGE/*6*/:
            console.log('ItemType = ' + STR_CapValueType[DWObject.CapValueType]);
            console.log('Min = ' + DWObject.CapMinValue);
            console.log('Max = ' + DWObject.CapMaxValue);
            console.log('StepSize = ' + DWObject.CapStepSize);
            console.log('Default = ' + DWObject.CapDefaultValue);
            console.log('Current = ' + DWObject.CapCurrentValue);
            break;
        default: console.log('This Capability is not supported');
    }
    var supportLevel = [];
    if (DWObject.CapIfSupported(EnumDWT_MessageType.TWQC_GET)) supportLevel.push('GET');/*TWQC_GET*/
    if (DWObject.CapIfSupported(EnumDWT_MessageType.TWQC_SET)) supportLevel.push('SET');/*TWQC_SET*/
    if (DWObject.CapIfSupported(EnumDWT_MessageType.TWQC_RESET)) supportLevel.push('RESET');/*TWQC_RESET*/
    if (supportLevel.length > 0) {
        console.log('Supported operations: ');
        console.log(supportLevel.join(' / '));
    }
}

// Set A Capability
function setACapability(capability, valueToSet, indexToSet) {
    var tempValue = '', i, valueToShow;
    DWObject.SelectSource();
    DWObject.OpenSource();
    DWObject.Capability = capability;
    DWObject.<code>CapGet()</code>;
    nCapType = DWObject.CapType;
    switch (nCapType) {
        case EnumDWT_CapType.TWON_ARRAY/*3*/:
            console.log('Setting an Array is not implemented');
            break;
        case EnumDWT_CapType.TWON_ENUMERATION/*4*/:
            DWObject.CapValue = valueToSet;
            DWObject.CapCurrentIndex = indexToSet;
            DWObject.<code>CapSet()</code>;
            console.log('CapSet: ' + DWObject.ErrorString);
            DWObject.<code>CapGet()</code>;
            console.log('After Setting:');
```

```
            if (DWObject.CapValueType > 8) {
                console.log('Current Index = ' + DWObject.CapCurrentIndex + ' (Value: ' + DWObject.GetCapItemsString(
DWObject.CapCurrentIndex) + ')');
                console.log('Default Index = ' + DWObject.CapDefaultIndex + ' (Value: ' + DWObject.GetCapItemsString(
DWObject.CapDefaultIndex) + ')');
            }
            else {
                console.log('Current Index = ' + DWObject.CapCurrentIndex + ' (Value: ' + DWObject.GetCapItems(DWObje
ct.CapCurrentIndex) + ')');
                console.log('Default Index = ' + DWObject.CapDefaultIndex + ' (Value: ' + DWObject.GetCapItems(DWObje
ct.CapDefaultIndex) + ')');
            }
            break;
        case EnumDWT_CapType.TWON_ONEVALUE/*5*/:
            DWObject.CapValue = valueToSet;
            DWObject.<code>CapSet()</code>;
            console.log('CapSet: ' + DWObject.ErrorString);
            DWObject.<code>CapGet()</code>;
            console.log('Value after setting: ' + DWObject.CapValue);
            break;
        case EnumDWT_CapType.TWON_RANGE/*6*/:
            DWObject.CapCurrentValue = valueToSet;
            DWObject.<code>CapSet()</code>;
            console.log('CapSet: ' + DWObject.ErrorString);
            DWObject.<code>CapGet()</code>;
            console.log('Value after setting: ' + DWObject.CapCurrentValue);
            break;
        default: console.log('This Capability is not supported');
    }
}
```

Methods

`CapGet()`					
Gets the detailed information of the capability specified by (Capability)[#Capability].					
Syntax	`.CapGet()`				
Parameters	`None`				
Return value	`boolean`				
Availability	**ActiveX**	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗

CapGetCurrent()					
Gets the current value of the capability specified by (Capability)[#Capability].					
Syntax	.CapGetCurrent()				
Parameters	`None`				
Return value	`boolean`				
	ActiveX	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**

Availability	✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗
Usage notes	Currently this method is only valid in Windows & Mac.				

CapGetDefault()					
Gets the default value of the capability specified by (Capability)[#Capability].					
Syntax	.CapGetDefault()				
Parameters	`None`				
Return value	`boolean`				
Availability	**ActiveX**	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗

CapGetFrameBottom()					
Returns the value of the bottom edge of the specified frame.					
Syntax	.CapGetFrameBottom(index)				
Parameters	`number optionalAsyncFailureFunc` : specifies which frame to check. The index is 0-based.				
Return value	`number`				
Availability	**ActiveX**	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗
Usage notes	The default Unit is assumed to be "inches" unless it has been otherwise negotiated between th application and Data Source.				

CapGetFrameLeft()					
Returns the value of the Left edge of the specified frame.					
Syntax	.CapGetFrameLeft(index)				
Parameters	`number optionalAsyncFailureFunc` : specifies which frame to check. The index is 0-based.				
Return value	`number`				
Availability	**ActiveX**	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗
Usage notes	The default Unit is assumed to be "inches" unless it has been otherwise negotiated between th application and Data Source.				

CapGetFrameLeft()

Returns the value of the Left edge of the specified frame.

Syntax	.CapGetFrameLeft(index)
Parameters	`number` `optionalAsyncFailureFunc` : specifies which frame to check. The index is 0-based.
Return value	`number`
Availability	ActiveX: ✓ | v1.0 H5(Windows): ✓ | v10.0 H5(macOS/TWAIN): ✓ | v11.0 H5(macOS/ICA): ✗ H5(Linux): ✗
Usage notes	The default Unit is assumed to be "inches" unless it has been otherwise negotiated between th application and Data Source.

CapGetFrameRight()

Returns the value of the Right edge of the specified frame.

Syntax	.CapGetFrameRight(index)
Parameters	`number` `optionalAsyncFailureFunc` : specifies which frame to check. The index is 0-based.
Return value	`number`
Availability	ActiveX: ✓ | v1.0 H5(Windows): ✓ | v10.0 H5(macOS/TWAIN): ✓ | v11.0 H5(macOS/ICA): ✗ H5(Linux): ✗
Usage notes	The default Unit is assumed to be "inches" unless it has been otherwise negotiated between th application and Data Source.

CapGetFrameTop()

Returns the value of the Top edge of the specified frame.

Syntax	.CapGetFrameTop(index)
Parameters	`number` `optionalAsyncFailureFunc` : specifies which frame to check. The index is 0-based.
Return value	`number`
Availability	ActiveX: ✓ | v1.0 H5(Windows): ✓ | v10.0 H5(macOS/TWAIN): ✓ | v11.0 H5(macOS/ICA): ✗ H5(Linux): ✗
Usage notes	The default Unit is assumed to be "inches" unless it has been otherwise negotiated between th application and Data Source.

CapGetHelp()

Get help information of the capability specified by Capability.

Syntax	.CapGetHelp(index)
Parameters	`None`
Return value	`boolean`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v1.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✗</td><td>✗</td></tr></table>
Usage notes	Call this method after calling `CapGet()` method.

CapGetLabel()

Get Label information of the capability specified by Capability.

Syntax	.CapGetLabel()
Parameters	`None`
Return value	`boolean`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v1.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✗</td><td>✗</td></tr></table>
Usage notes	Call this method after calling `CapGet()` method.

CapGetLabels()

Get Labels information of the capability specified by Capability.

Syntax	.CapGetLabels()
Parameters	`None`
Return value	`boolean`
Example	
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v1.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✗</td><td>✗</td></tr></table>
Usage notes	Call this method after calling `CapGet()` method.

CapIfSupported()

Returns the Data Source's support level of the capability specified by Capability.

Syntax	.CapIfSupported(messageType)
Parameters	`<EnumDWT_MessageType (number)>` `messageType`
Return value	`boolean`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗

CapReset()

Resets the the capability specified by Capability.

Syntax	.CapReset()
Parameters	`None`
Return value	`boolean`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗

`CapSet()`

Sets the the capability specified by Capability.

Syntax	`. CapSet()`
Parameters	`None`
Return value	`boolean`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗

CapSetFrame()

Sets the values of the specified frame.

Syntax	.CapSetFrame(index, left, top, right, bottom);
Parameters	`number` index: specifies which frame to set. The index is 0-based. `number` left: specifies the value of the left edge of the specified frame. `number` top: specifies the value of the top edge of the specified frame. `number` right: specifies the value of the right edge of the specified frame. `number` bottom: specifies the value of the bottom edge of the specified frame.
Return value	`boolean`

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)

Availability	✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗
Usage notes	The frame is specified by values in a specific unit. The default unit is "inches" and can be configured by the property Unit				

GetCapItems()

Gets the cap item value of the capability specified by Capability.

Syntax	.GetCapItems(index)
Parameters	`number` `index` :specifies the index of a cap item. The index is 0-based.
Return value	`string`

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗

Usage notes	Valid only when CapType is `TWON_ARRAY (3)` or `TWON_ENUMERATION (4)` .

GetCapItemsString()

Gets the cap item value of the capability specified by Capability.

Syntax	.GetCapItemsString(index)
Parameters	`number` `index` :specifies the index of a cap item. The index is 0-based.
Return value	`string`

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗

Usage notes	Valid only when CapType is `TWON_ARRAY (3)` or `TWON_ENUMERATION (4)` .

SetCapItems()

Sets the cap item value of the capability specified by Capability.

Syntax	.SetCapItems(index, newVal)
Parameters	`number` `index` :specifies the index of a cap item. The index is 0-based. `number` `newVal` :specifies the value.
Return value	`string`

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗

Usage notes	Valid only when CapType is `TWON_ARRAY (3)` or `TWON_ENUMERATION (4)` .

SetCapItemsString()					
Sets the cap item value of the capability specified by Capability.					
Syntax	.SetCapItemsString(index, newVal)				
Parameters	`number index` :specifies the index of a cap item. The index is 0-based. `number newVal` :specifies the value.				
Return value	`None`				
Availability	**ActiveX**	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗
Usage notes	Valid only when CapType is `TWON_ARRAY (3)` or `TWON_ENUMERATION (4)` .				

Properties

Capability					
Specifies the capabiltiy to be negotiated with the Data Source.					
Type	`EnumDWT_Cap (int)`				
Accessors	`Get Set`				
Availability	**ActiveX**	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗

CapCurrentIndex					
Returns the index (0-based) of the current value of a capability. Then you can find the current value by this index in `GetCapItemsString()` or `GetCapItemsCapValueType()` .					
Type	`number`				
Accessors	`Get Set`				
Availability	**ActiveX**	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗
Usage notes	Check this property after calling `CapGet()` . Valid only when the capability has the CapType of `TWON_ENUMERATION (4)` .				

CapCurrentValue

Returns the current value of a capability.

Type	`number`
Accessors	`Get Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v1.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✗</td><td>✗</td></tr></table>
Usage notes	Check this property after calling `CapGet()`. Valid only when the capability has the CapType of `TWON_RANGE (6)`.

CapDefaultIndex

Returns the index (0-based, read-only) of the default value of a capability. Then you can find the default value by this index in `GetCapItemsString` or `GetCapItems`.

Type	`number`
Accessors	`Get Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v1.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✗</td><td>✗</td></tr></table>
Usage notes	Check this property after calling `CapGet()`. Valid only when the capability has the CapType of `TWON_ENUMERATION (4)`. Default value reflects the Data Source's power-on value. It can NOT be set.

CapDefaultValue

Returns the default value of a capability.

Type	`number`
Accessors	`Get Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v1.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✗</td><td>✗</td></tr></table>
Usage notes	Check this property after calling `CapGet()`. Valid only when the capability has the CapType of `TWON_RANGE (6)`. Default value reflects the Data Source's power-on value. It can NOT be set.

CapMaxValue

Returns the maximum value of a capability.

Type	`number`
Accessors	`Get Set`
	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr></table>

Availability	✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗
Usage notes	Check this property after calling `CapGet()`. Valid only when the capability has the CapType of `TWON_RANGE (6)`. Default value reflects the Data Source's power-on value. It can NOT be set.				

CapMinValue

Returns the Minimum value of a capability.

Type	`number`				
Accessors	`Get Set`				
Availability	**ActiveX**	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗
Usage notes	Check this property after calling `CapGet()`. Valid only when the capability has the CapType of `TWON_RANGE (6)`. Default value reflects the Data Source's power-on value. It can NOT be set.				

CapNumItems

Returns or sets the number of values of capability.

Type	`number`				
Accessors	`Get Set`				
Availability	**ActiveX**	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗
Usage notes	Check this property after calling `CapGet()`. Set this property before calling `CapSet()`. Valid only when the capability has the CapType of `TWON_ARRAY (3)` or `TWON_ENUMERATION (4)`.				

CapStepSize

Returns or sets the step size of the values of a capability.

Type	`number`				
Accessors	`Get Set`				
Availability	**ActiveX**	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✓ | v1.0	✓ | v10.0	✓ | v11.0	✗	✗
Usage notes	Check this property after calling `CapGet()`. Set this property before calling `CapSet()`. Valid only when the capability has the CapType of `TWON_RANGE (6)`.				

CapType

Returns or sets the type of capability container used to exchange capability information between application and source.

Type	`EnumDWT_CapType (number)`
Accessors	`Get Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v1.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✗</td><td>✗</td></tr></table>
Usage notes	Set CapType when you try to set a capability with `CapSet()` . Check CapType after you read a capability by `CapGet()` .

CapValue

Returns or sets the value of the capability specified by Capability.

Type	`number`
Accessors	`Get Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v1.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✗</td><td>✗</td></tr></table>
Usage notes	Set CapType when you try to set a capability with `CapSet()` . Check CapType after you read a capability by `CapGet()` . To check a capability, read this property after calling `CapGet()` method. When setting a capability, set this property before calling `CapSet()` to actually set the value. Valid only when the capability has the CapType of `TWON_ONEVALUE (5)` . Use this property to get/set a capability that has a value type of number or even Boolean (1 means 'true', 0 means 'false'). For `string` type, use CapValueString instead.

CapValueString

Returns or sets the string value of a capability specified by Capability.

Type	`string`
Accessors	`Get Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v1.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✗</td><td>✗</td></tr></table>
Usage notes	To check a capability, read this property after calling `CapGet()` method. When setting a capability, set this property before calling `CapSet()` to actually set the value. Valid only when the capability has the CapType of `TWON_ONEVALUE (5)` . Use this property to get/set a capability that has a value type of `string` , otherwise, use CapValue instead.

CapValueType

Returns or sets the value type for reading the value of a capability.

<table>
<tr><td>Type</td><td></td></tr>
<tr><td>Accessors</td><td><code>Get Set</code></td></tr>
<tr><td>Availability</td><td>
<table>
<tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr>
<tr><td>✓ | v1.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✗</td><td>✗</td></tr>
</table>
</td></tr>
<tr><td>Usage notes</td><td>When you try to read a capability value, check CapValueType after you call <code>CapGet()</code>.
Valid only when the capability has the CapType of <code>TWON_ONEVALUE (5)</code>.
When you try to write a capability value, set CapValueType before you call <code>CapSet()</code>.</td></tr>
</table>

Encode & Decode

Methods		
ClearTiffCustomTag()	ConvertToBase64()	ConvertToBlob()
SetTiffCustomTag()		

Properties		
IfOpenImageWithGDIPlus	IfTiffMultiPage	JPEGQuality
PDFAuthor	PDFCompressionType	PDFCreationDate
PDFCreator	PDFKeywords	PDFModifiedDate
PDFProducer	PDFSubject	PDFTitle
PDFVersion	TIFFCompressionType	

Methods

ClearTiffCustomTag()

Clears the content of all custom tiff tags.

Syntax	.ClearTiffCustomTag()
Parameters	`None`
Return value	`None`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v10.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

ConvertToBase64()

Converts the images of specified indices in buffer to a base64 string.

Syntax	.ConvertToBase64(indices, enumImageType, asyncSuccessFunc, asyncFailureFunc);

Parameters

`number indices` : specifies the indices of images in buffer. The index is 0-based.
`EnumDWT_ImageType enumImageType` : specifies the format.

Type	Numeric Value
IT_BMP	0
IT_JPG	1
IT_TIF	2
IT_PNG	3

IT_PDF	4

`OnSuccess function asyncSuccessFunc` : callback function triggered when the file is converted successfully.
`OnFailure function asyncFailureFunc` : callback function triggered when the file fails to be converted.

Return value

`Base64Result` is returned in the callback `asyncSuccessFunc()`
Methods available in `Base64Result` object

Data Type	Method Name	Description
`number`	getLength()	Returns the length of the Base64 String
`string`	getData(offset, length)	Returns the string containing the extracted part of the Base64 string. The parameter `offset` means the position where to start the extraction. The parameter `length` means the number of characters to extract.
`string`	getMD5()	Returns the MD5 of the Base64 String.

The result string returned from `getData` is the pure base64 string with no extra info. For example, `"/9j/4AAQSkZJRgABA..."`
To use the string in most circumstances, you need to add extra info like `"/9j/4AAQSkZJRgABA..."`
Because this method is async only, the returned value doesn't mean if the convert succeeds o not. When conversion fails, check ErrorCode or ErrorString for error information.

Example

```
DWObject.ConvertToBase64 ([1,3], EnumDWT_ImageType.IT_PDF, asyncSuccessFunc,
asyncFailureFunc);
function asyncSuccessFunc (result) {
     var length=result.getLength();
     console.log(result.getData(0,length));
     console.log(result);
}
function asyncFailureFunc (errorCode, errorString) {
     alert("ErrorCode: "+errorCode +"\r"+"ErrorString:"+ errorString);
}
```

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✗	✓ | v12.0	✓ | v12.0	✓ | v12.0	✓ | v12.1

ConvertToBlob()

Converts the images of specified indices in buffer to a blob.

Syntax

.ConvertToBlob(indices, enumImageType, asyncSuccessFunc, asyncFailureFunc);

Parameters

`Number[] indices` : specifies the indices of images in buffer. The index is 0-based.
`EnumDWT_ImageType enumImageType` : specifies the format.

Type	Numeric Value
IT_BMP	0
IT_JPG	1
IT_TIF	2
IT_PNG	3

	IT_PDF | 4 `OnSuccess function` `asyncSuccessFunc` : callback function triggered when the file is converted successfully. `OnFailure function` `asyncFailureFunc` : callback function triggered when the file fails to be converted.
Return value	`Blob` is returned in the callback `asyncSuccessFunc()` Because this method is asynchronous, whether there is a returned value doesn't mean the success/failure of the operation. When the conversion fails, check ErrorCode or ErrorString for error information.
Example	`DWObject.ConvertToBlob ([1,3], EnumDWT_ImageType.IT_PDF, asyncSuccessFunc, asyncFailureFunc);` `function asyncSuccessFunc (result) {` `  console.log(result.size);` `}` `function asyncFailureFunc (errorCode, errorString) {` `  alert("ErrorCode: "+errorCode +"\r"+"ErrorString:"+ errorString);` `}`
Availability	ActiveX: ✗ | H5(Windows): ✓ | v13.0 | H5(macOS/TWAIN): ✓ | v13.0 | H5(macOS/ICA): ✓ | v13.0 | H5(Linux): ✓ | v13.0

SetTiffCustomTag()

Sets a custom tiff tag (up to 32 tags). The string to be set in a tag can be base64 encoded.

Syntax	.SetTiffCustomTag(tagIdentifier, content, UseBase64Encoding);
Parameters	`number` `tagIdentifier` : specifies the identify number of custom tag in the tiff image. `string` `content` : the string to be set for this tag. The string will be written to the .tiff file when you save/upload it. If the string is base64 encoded, it'll be decoded when creating the TIFF file `Boolean` `UseBase64Encoding` : specifies whether the content is base64 encoded.
Return value	`boolean`
Example	`DWObject.ClearTiffCustomTag();` `DWObject.SetTiffCustomTag(700, "Custom Tiff Tag Value", true);` `DWObject.SaveAsTIFF("C:\\DWT.tiff", 0);`
Availability	ActiveX: ✓ | 10.0 | H5(Windows): ✓ | v10.0 | H5(macOS/TWAIN): ✓ | v11.0 | H5(macOS/ICA): ✓ | v11.0 | H5(Linux): ✓ | v12.1

Properties

IfOpenImageWithGDIPlus

Returns or sets whether to use GDI+ library to decode images when loading them.

Type	`boolean`
Accessors	`Get Set`

Usage notes	Windows GDI+ is a class-based API for C/C++ programmers. It enables applications to use graphics and formatted text on both the video display and the printer. Applications based on the Microsoft Win32 API do not access graphics hardware directly. Instead, GDI+ interacts with device drivers on behalf of applications. GDI+ is also supported by Microsoft Win64. With GDI+, you can load some images which are not supported otherwise. However, please note that GDI+ library might be different on different OS. So an image might load on Win 7 but not Win XP.
Availability	**ActiveX**: ✓ | v10.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✗ **H5(macOS/ICA)**: ✗ **H5(Linux)**: ✗

IfTiffMultiPage

Return or sets whether it's allowed to save many images as one TIFF file.

Type	`boolean`
Accessors	`Get Set`
Usage notes	When you save a new image in the same name of an existing TIFF file • If this property is true, the new image will be added to the existing file • If this property is false, the new image will replace the existing file
Availability	**ActiveX**: ✓ | v3.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1

JPEGQuality

Returns or sets the quality of JPEG files or JPEG-encoded PDF files.

Type	`number`
Accessors	`Get Set`
Usage notes	The default value of JPEGQuality property is 80. The valid range is 0-100. The higher the JPEGQuality property, the better the quality and the bigger the size of the file.
Availability	**ActiveX**: ✓ | v5.0 **H5(Windows)**: ✓ | v10.0 **H5(macOS/TWAIN)**: ✓ | v11.0 **H5(macOS/ICA)**: ✓ | v11.0 **H5(Linux)**: ✓ | v12.1

TIFFCompressionType

Returns or sets the compression type for TIFF files.

Type	`number`
Accessors	`Get Set`

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)

Availability	✓ | v4.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes

Allowed Values	EnumDWT_TIFFCompressionType	Description	Supported image format
0	TIFF_AUTO	Auto mode	1, 4, 8, 24 bit
1	TIFF_NONE	Dump mode	1, 4, 8, 24 bit
2	TIFF_RLE	CCITT modified Huffman RLE	1 bit
3	TIFF_FAX3	CCITT Group 3 fax encoding	1 bit
	TIFF_T4	CCITT T.4 (TIFF 6 name)	1 bit
4	TIFF_FAX4	CCITT Group 4 fax encoding	1 bit
	TIFF_T6	CCITT T.6 (TIFF 6 name)	1 bit
5	TIFF_LZW	Lempel-Ziv & Welch	1, 4, 8, 24 bit
7	TIFF_JPEG	JPEG encoding	24 bit
32773	TIFF_PACKBITS	Macintosh RLE	1, 4, 8, 24 bit

When set to `TIFF_AUTO (0)` , 1-bit images will be compressed in `TIFF_T6 (4)` while images with other bit depth will be compressed in `TIFF_LZW (5)` .

When set to `TIFF_JPEG (7)` , 1-bit images will be compressed in `TIFF_T6 (4)` , color images or grey images (8-bit or higher) in `TIFF_JPEG (7)` standard, and other images by `TIFF_LZW (5)` .

`TIFF_T4 (3)` and `TIFF_FAX3 (3)` are two names for the same compression type. So are `TIFF_T6 (4)` and `TIFF_FAX4 (4)`

When `TIFF_JPEG (7)` is used, you can use JPEGQuality to further reduce the size of the TIFF file.

As of version v14.2, `TIFF_JPEG (7)` and `TIFF_PACKBITS (32773)` are only supported on Windows.

PDFAuthor

Returns or sets the name of the author that creates the PDF document.

Type	`string`
Accessors	`Get Set`

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)

Availability	✓ | v5.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

PDFCompressionType

Returns or sets the compression type for PDF files.

Type	`string`
Accessors	`Get Set`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Usage notes

Allowed Values	EnumDWT_PDFCompressionType	Description	Supported image format
0	PDF_AUTO	Auto mode	1, 4, 8, 24 bit
1	PDF_FAX3	CCITT Group 3 fax encoding	1 bit
2	PDF_FAX4	CCITT Group 4 fax encoding	1 bit
3	PDF_LZW	Lempel-Ziv & Welch	1, 4, 8, 24 bit
4	PDF_RLE	CCITT modified Huffman RLE	1 bit
5	PDF_JPEG	JPEG encoding	8, 24 bit

PDFCreationDate

Returns or sets the date when the PDF document is created.

Type	`string`
Accessors	`Get Set`
Usage notes	The default value is current date.

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

PDFCreator

Returns or sets the name of the application that created the original document (used when the PDF document is converted from another form);

Type	`string`
Accessors	`Get Set`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

PDFKeywords

Returns or sets the keywords associated with the PDF document.

Type	`string`
Accessors	`Get Set`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

PDFModifiedDate

Returns or sets the date when the PDF document is last modified.

Type	`string`
Accessors	`Get Set`
Usage notes	The default value is current date.
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

PDFProducer

Returns or sets the name of the application that generated the PDF document.

Type	`string`
Accessors	`Get Set`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

PDFSubject

Returns or sets the subject of the PDF document.

Type	`string`
Accessors	

Accessors	
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

PDFTitle

Returns or sets the title of the PDF document.

Type	`string`
Accessors	`Get Set`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

PDFVersion

Returns or sets the version number for PDF files.

Type	`string`
Accessors	`Get Set`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v5.0	✓ | v10.0	✓ | v11.0	✓ | v11.0	✓ | v12.1

Runtime Information

Methods		
GetImageBitDepth()	GetImageHeight()	GetImageSize()
GetImageSizeWithSpecifiedType()	GetImageWidth()	GetImageXResolution()
GetImageYResolution()	GetSelectedImageIndex()	GetSelectedImagesSize()
GetSkewAngle()	GetSkewAngleEx()	IsBlankImageExpress()

Properties		
BlankImageCurrentStdDev	BlankImageMaxStdDev	BlankImageThreshold
CurrentImageIndexInBuffer	HowManyImagesInBuffer	ImageLayoutDocumentNumber
ImageLayoutFrameBottom	ImageLayoutFrameLeft	ImageLayoutFrameNumber
ImageLayoutFrameRight	ImageLayoutFrameTop	ImageLayoutPageNumber
ImagePixelType	MagData	MagType
SelectedImagesCount		

Methods

GetImageBitDepth()	
Returns the pixel bit depth of the selected image.	
Syntax	.GetImageBitDepth(sImageIndex);
Parameters	`number` `sImageIndex` : the index of the image in the buffer. The index is 0-based.
Return value	`number` If Dynamic Web TWAIN fails to get the pixel bit depth of the image, -1 is returned.
Availability	v6.2+

GetImageHeight()	
Returns the height (in pixels) of the selected image.	
Syntax	.GetImageHeight(sImageIndex);
Parameters	`number` `sImageIndex` : the index of the image in the buffer. The index is 0-based.
Return value	`number` If Dynamic Web TWAIN fails to get the height of the image, -1 is returned.
Availability	v6.2+

GetImageSize()	
Calculates the file size of a new image resized from an image of a specified index in buffer.	
Syntax	.GetImageSize(sImageIndex, iWidth, iHeight);

Parameters	`number` `sImageIndex` : the index of the image in the buffer. The index is 0-based. `number` `iWidth` : the width for the new image. `number` `iHeight` : the height for the new image.
Return value	`number` : returns the size in bytes. If Dynamic Web TWAIN fails to get the size of the image, -1 is returned.
Availability	All versions.

GetImageSizeWithSpecifiedType()	
Calculates the file size of a specified image based on its format.	
Syntax	.GetImageSizeWithSpecifiedType(sImageIndex, sImageType);
Parameters	`number` `sImageIndex` : the index of the image in the buffer. The index is 0-based. `EnumDWT_ImageType` `sImageType` : the image format to be used.
Return value	`number` : returns the size in bytes. If Dynamic Web TWAIN fails to calculate the size, -1 is returned.
Availability	All versions.

GetImageWidth()	
Returns the width (in pixels) of the selected image.	
Syntax	.GetImageWidth(sImageIndex);
Parameters	`number` `sImageIndex` : the index of the image in the buffer. The index is 0-based.
Return value	`number` If Dynamic Web TWAIN fails to get the width of the image, -1 is returned.
Availability	v6.2+

GetImageXResolution()	
Returns the horizontal resolution of the specified image.	
Syntax	.GetImageXResolution(sImageIndex);
Parameters	`number` `sImageIndex` : the index of the image in the buffer. The index is 0-based.
Return value	`number` If Dynamic Web TWAIN fails to get the resolution of the image, -1 is returned.
Availability	v8.0+

GetImageYResolution()	
Returns the vertical resolution of the specified image.	
Syntax	.GetImageYResolution(sImageIndex);
Parameters	`number` `sImageIndex` : the index of the image in the buffer. The index is 0-based.
Return value	`number`
Availability	v8.0+
Usage notes	If Dynamic Web TWAIN fails to get the resolution of the image, -1 is returned.

GetSelectedImageIndex()

Returns the index of a selected image in the index array of selected images.	
Syntax	.GetSelectedImageIndex(selectedIndex);
Parameters	`number` `selectedIndex` : the index of the array which holds all the indices of selected images.
Return value	`number`
Availability	v7.0+
Usage notes	If three images are selected in a 10 image buffer (indexes 0,5,9) GetSelectedImageIndex(1) = 5; as it is the second image selected.

GetSelectedImagesSize()	
Calculates the file size in the specified format for the selected images.	
Syntax	.GetSelectedImagesSize(imageType);
Parameters	`EnumDWT_ImageType` `sImageType` : the image format.
Return value	`number` If Dynamic Web TWAIN fails to calculate the size, -1 is returned.
Availability	v6.0+
Usage notes	If the sImageType is IT_TIF or IT_PDF, `GetSelectedImagesSize()` returns the total size of the selected images. Otherwise, the method returns the size of the first selected image.

GetSkewAngle()	
Returns the skew angle of an image specified by index.	
Syntax	.GetSkewAngle(sImageIndex);
Parameters	`number` `sImageIndex` : the index of the image in the buffer. The index is 0-based.
Return value	`number`
Availability	v9.0+
Usage notes	This method can be combined with a rotation API to correct a skewed document.

GetSkewAngleEx()	
Returns the skew angle of a rectangular portion of an image specified by index.	
Syntax	.GetSkewAngleEx(sImageIndex, left, top, right, bottom);
Parameters	`number` `sImageIndex` : the index of the image in the buffer. The index is 0-based. `number` `left` : the x-coordinate of the upper-left corner of the rectangle. `number` `top` : the y-coordinate of the upper-left corner of the rectangle. `number` `right` : the x-coordinate of the lower-right corner of the rectangle. `number` `bottom` : the y-coordinate of the lower-right corner of the rectangle.
Return value	`number`
Availability	v9.0+

IsBlankImageExpress()	
Detects whether the image of a specified index is blank.	
Syntax	.IsBlankImageExpress(sImageIndex);

Parameters	`number sImageIndex` : the index of the image in the buffer. The index is 0-based.
Return value	`boolean` : 'true' means the image is blank.
Availability	v10.0+

Properties

BlankImageCurrentStdDev	
Returns the deviation of the pixels in the current image. This is a read-only property.	
Type	`number`
Accessors	`Get`
Availability	v7.0+
Usage notes	This property is only valid after `IsBlankImageExpress` is called.

BlankImageMaxStdDev	
Returns or sets the deviation of the pixels in an image.	
Type	`number`
Accessors	`Get Set`
Availability	All versions.
Usage notes	[0, 100] is the interval of allowed values, inclusive. 0 gives a single-color image. The default value is 1. This property is only valid after `IsBlankImageExpress` is called.

BlankImageThreshold	
Returns or sets the dividing line between black and white.	
Type	`number`
Accessors	`Get Set`
Availability	All versions.
Usage notes	[0, 255] is the interval of allowed values, inclusive. The default value is 128. This property is only valid after `IsBlankImageExpress` is called.

CurrentImageIndexInBuffer	
Returns the index (0-based) of the current image selected by Dynamic Web TWAIN or sets the image with the specified index as the current image.	
Type	`number`
Accessors	`Get Set`
Availability	All versions.
	By changing `CurrentImageIndexInBuffer` , you can enumerate all the images in buffer. When `CurrentImageIndexInBuffer` changes, the control will be redrawn to reflect the change. When image buffer is full, that is `HowManyImagesInBuffer = MaxImagesInBuffer` , any

Usage notes	acquired or loaded image will replace the existing one positioned by `CurrentImageIndexInBuffer` . For example, if 1. `MaxImagesInBuffer` is set to 4 2. `HowManyImagesInBuffer` returns 4 3. `CurrentImageIndexInBuffer` is 3 When a new image is acquired, the `CurrentImageIndexInBuffer` is set back to 0, and the first image is replaced by the newly acquired one . If another image is acquired, `CurrentImageIndexInBuffer` is set to 1 and the second image is replaced by the new one.

HowManyImagesInBuffer	
Returns how many images are currently loaded in Dynamic Web TWAIN. This is a read-only property.	
Type	`number`
Accessors	`Get`
Availability	All versions.

ImageLayoutDocumentNumber	
Returns the document number of the current image. This is a read-only property.	
Type	`number`
Accessors	`Get`
Availability	All versions.
Usage notes	The document number is assigned by the source and is useful for grouping pages together. Usually a physical representation, this could just as well be a logical construct. Initial value is 1. Increment when a new document is placed into the document feeder (usually tell this has happened when the feeder empties). Resets when no longer acquiring from the feeder. `ImageLayoutDocumentNumber` property, along with other properties about the current image information, is valid only in `OnPreTransfer` and `OnPostTransfer` event.

ImageLayoutFrameBottom	
Returns the value of the bottom edge of the current image frame (in `Unit`). This is a read-only property.	
Type	`number`
Accessors	`Get`
Availability	All versions.
Usage notes	The unit of `ImageLayoutFrameBottom` property is determined by `Unit` property. The default unit is assumed to be "inches" unless it has been otherwise negotiated between the application and Data Source. `ImageLayoutFrameBottom` property, along with other properties about the current image information, is valid only in `OnPreTransfer` and `OnPostTransfer` event. The frame information here is only about the current frame. To get the information about all the frames to be transferred in an acquire session, please use capability negotiation. The capability to be negotiated is ICAP_FRAMES (4372).

ImageLayoutFrameLeft	
Returns the value of the left-most edge of the current image frame (in `Unit`). This is a read-only property.	
Type	`number`
Accessors	`Get`

Availability	All versions.
Usage notes	The unit of `ImageLayoutFrameLeft` property is determined by `Unit` property. The default unit is assumed to be "inches" unless it has been otherwise negotiated between the application and Data Source. `ImageLayoutFrameLeft` property, along with other properties about the current image information, is valid only in `OnPreTransfer` and `OnPostTransfer` event. The frame information here is only about the current frame. To get the information about all the frames to be transferred in an acquire session, please use capability negotiation. The capability to be negotiated is ICAP_FRAMES (4372).

ImageLayoutFrameNumber

Returns the frame number of the current image. This is a read-only property.

Type	`number`
Accessors	`Get`
Availability	All versions.
Usage notes	Usually a chronological index of the acquired frames, these frames are related to one another in some way. Usually, they were acquired from the same page. The source assigns these values. Initial value is 1. Reset when a new page is acquired. `ImageLayoutFrameNumber` property, along with other properties about the current image information, is valid only in `OnPreTransfer` and `OnPostTransfer` event. The frame information here is only about the current frame. To get the information about all the frames to be transferred in an acquire session, please use capability negotiation. The capability to be negotiated is ICAP_FRAMES (4372).

ImageLayoutFrameRight

Returns the value of the right-most edge of the current image frame (in `Unit`). This is a read-only property.

Type	`number`
Accessors	`Get`
Availability	All versions.
Usage notes	The unit of `ImageLayoutFrameRight` property is determined by `Unit` property. The default unit is assumed to be "inches" unless it has been otherwise negotiated between the application and Data Source. `ImageLayoutFrameRight` property, along with other properties about the current image information, is valid only in `OnPreTransfer` and `OnPostTransfer` event. The frame information here is only about the current frame. To get the information about all the frames to be transferred in an acquire session, please use capability negotiation. The capability to be negotiated is ICAP_FRAMES (4372).

ImageLayoutFrameTop

Returns the value of the top-most edge of the current image frame (in `Unit`). This is a read-only property.

Type	`number`
Accessors	`Get`
Availability	All versions.
Usage notes	The unit of `ImageLayoutFrameTop` property is determined by `Unit` property. The default unit is assumed to be "inches" unless it has been otherwise negotiated between the application and Data Source. `ImageLayoutFrameTop` property, along with other properties about the current image information, is valid only in `OnPreTransfer` and `OnPostTransfer` event. The frame information here is only about the current frame. To get the information

	about all the frames to be transferred in an acquire session, please use capability negotiation. The capability to be negotiated is ICAP_FRAMES (4372).

ImageLayoutPageNumber	
Returns the page number of the current image. This is a read-only property.	
Type	`number`
Accessors	`Get`
Availability	All versions.
Usage notes	Increment for each page fed from the feeder. `ImageLayoutPageNumber` property, along with other properties about the current image information, is valid only in `OnPreTransfer` and `OnPostTransfer` event.

ImagePixelType	
Returns the pixel type of the current image. This is a read-only property.	
Type	`EnumDWT_PixelType`
Accessors	`Get`
Availability	All versions.
Usage notes	Please note the property is only valid in `OnPreTransfer` and `OnPostTransfer` event.

MagData	
Returns the magnetic data if the data source supports magnetic data recognition. This is a read-only property.	
Type	`number`
Accessors	`Get`
Availability	v8.0+
Usage notes	Whether it works depends on the scanner's capability. Binary information is not supported.

MagType	
Returns the magnetic type if the data source supports magnetic data recognition. This is a read-only property.	
Type	`EnumDWT_MagType`
Accessors	`Get`
Availability	v8.0+

SelectedImagesCount	
Returns or sets how many scanned images are, or will be, selected.	
Type	`number`
Accessors	`Get Set`
Availability	v6.0+

General Utilities

Methods		
GetImagePartURL()	GetImageURL()	Print()
RegisterEvent()	UnregisterEvent()	SetLanguage()

Properties		
BufferMemoryLimit	ErrorCode	ErrorString
IfAllowLocalCache	IfShowProgressBar	LogLevel
Manufacturer	ProductFamily	ProductKey
ProductName	VersionInfo	

Events
OnBitmapChanged

Methods

GetImagePartURL()	
Gets the internal url of an image in buffer specified by index.	
Syntax	.GetImagePartURL(index);
Parameters	`number` `index` : specifies the index of image in buffer. The index is 0-based.
Return value	`string`
Availability	ActiveX: ✗ / H5(Windows): ✓ | v13.0 / H5(macOS/TWAIN): ✓ | v13.0 / H5(macOS/ICA): ✓ | v13.0 / H5(Linux): ✓ | v13.0
Usage notes	The returned URL will be like "dwt://dwt_trial_13000404/img?id=306159652&index=0&t=1502184632022".

GetImageURL()	
Gets the url of the new image resized from the image of a specified index in buffer.	
Syntax	.GetImageURL(index, [optional nWidth, optional nHeight]);
Parameters	`number` `index` : specifies the index of image in buffer. The index is 0-based. `optional` `number` nWidth: specifies the new width which should be no smaller than 150. `optional` `number` nHeight: specifies the new height which should be no smaller than 150.
Return value	`string`
	ActiveX / H5(Windows) / H5(macOS/TWAIN) / H5(macOS/ICA) / H5(Linux)

Availability	✗	✓ | v12.0	✓ | v12.0	✓ | v12.0	✓ | v12.1
Usage notes	The returned URL will be like "http://127.0.0.1:18622/dwt/dwt_trial_14100828/img?id=780003506&index=0&t=1539674113432". If nWidth and nHeight are not specified, the original image will be returned.				

Print()					
Opens the print dialog of the browser to print images.					
Syntax	.Print()				
Parameters	`boolean` `PrintOption` : By default, it uses the browser print UI. If set it to true, it will open a separate process for the print operation.				
Return value	`boolean`				
Availability	**ActiveX**	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✓ | v6.0	✓ | v10.0	✓ | v10.0	✓ | v11.0	✓ | v12.1
Usage notes	Currently this method is only valid in Windows & Mac. This method opens the browser print dialog to print documents. This dialog will appear in a new browser window. Typically it is blocked by the browser at the first time, you can choose to 'always allow' the dialog.				

RegisterEvent()					
Adds an event listener to a built-in Dynamic Web TWAIN event.					
Syntax	.RegisterEvent()				
Parameters	`string` `eventName` : the name of a built-in event.				
Return value	`boolean`				
Availability	**ActiveX**	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✓ | v9.2	✓ | v10.0	✓ | v10.0	✓ | v11.0	✓ | v12.1
Usage notes	For the same event, the 2nd listener will override the 1st one. So there is always just one event listener for one event.				

UnRegisterEvent()					
Adds an event listener to a built-in Dynamic Web TWAIN event.					
Syntax	.UnRegisterEvent(eventName,evt)				
Parameters	`string` `eventName` : the name of a built-in event. `Function` `evt` : a function as the event listener.				
Return value	`boolean`				
	ActiveX	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**

Availability	✓ | v9.2	✓ | v10.0	✓ | v10.0	✓ | v11.0	✓ | v12.1

SetLanguage()

Sets the language for the authorization dialogs.

Syntax	.SetLanguage()
Parameters	`boolean` `PrintOption` : By default, it uses the browser print UI. If set it to true, it will open a separate process for the print operation.
Return value	`boolean`

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✗	✓ | v13.0	✓ | v13.0	✓ | v13.0	✓ | v13.0

Usage notes	Available values are English: 0, French: 1, Arabic: 2, Spanish: 3, Portuguese: 4, German: 5, Italian: 6, Russian: 7, Chinese: 8

Properties

BufferMemoryLimit

Returns or sets how much physical memory (caching threshold) is allowed for storing images currently loaded in Dynamic Web TWAIN. Once the limit is reached, images will be cached on the hard disk.

Type	`number`
Accessors	`Gets Set`
Usage notes	Set this property only when you have a very small physical memory (< 2GB) or a very big one (>4GB). The more memory is allowed, the better the performance will be. The default value is set to 800 (MB), anything beyond 800MB gets compressed, encrypted and cached on the local disk. All cached data is encrypted and can only be read by Dynamic Web TWAIN and it will be destroyed when it is no longer used.

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v10.1	✓ | v10.1	✓ | v10.1	✓ | v11.0	✓ | v12.1

ErrorCode

Returns the error code.

Type	`number`
Accessors	`Get`
Usage notes	Check out the Complete Error List

	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)

Availability	✓ | v1.0	✓ | v10.0	✓ | v10.0	✓ | v11.0	✓ | v12.1

ErrorString

Returns the error string.

Type	`number`
Accessors	`Get`
Usage notes	Check out the Complete Error List

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v1.0	✓ | v10.0	✓ | v10.0	✓ | v11.0	✓ | v12.1

IfAllowLocalCache

Returns or sets whether the feature of disk caching is enabled.

Type	`boolean`
Accessors	`Get Set`
Usage notes	The default value of IfAllowLocalCache is true. When the property is true, you can scan as many images as you want as long as you have a big enough disk. The default threshold is set to 800 (MB), anything beyond 800MB gets compressed, encrypted and cached on the local disk. If neccessary, you can set the threshold using BufferMemoryLimit for better performance. All cached data is encrypted and can only be read by Dynamic Web TWAIN and it will be destroyed when it is no longer used.

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v10.0	✓ | v10.0	✓ | v10.0	✓ | v11.0	✓ | v12.1

IfShowProgressBar

Returns or sets whether the progress bar is/should be displayed during encoding or decoding.
It works for any image encoding/decoding related methods. For example: LoadImage, LoadImageEx, ConvertToBlob, etc.

Type	`boolean`
Accessors	`Get Set`

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v8.0	✓ | v10.0	✓ | v10.0	✓ | v11.0	✓ | v12.1

LogLevel

Returns or sets the log level for debugging.

Type	Number
Accessors	`Get, Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v6.3</td><td>✓ | v10.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✓ | v12.1</td></tr></table>
Usage notes	The default value for LogLevel is 0 which means the extra information for debugging won't be logged. To log the information for debugging, you can set it to 1.

Manufacturer

Returns or sets the Manufacturer string for the application identity.

Type	string
Accessors	`Get, Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v1.0</td><td>✓ | v10.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✓ | v12.1</td></tr></table>
Usage notes	The Manufacturer property is a part of the application identity. The application identity consists of the Manufacturer, ProductFamily, ProductName and VersionInfo. All the application identity information should be set prior to invoking OpenSourceManager(). Since with built-in Wizard Mode, Dynamic TWAIN manages the transition of TWAIN state intelligently, OpensourceManager() may be called automatically by other TWAIN related functions, such as OpenSource() or AcquireImage(). It is recommended that application identity information be set prior to any other TWAIN functions.

ProductFamily

Returns or sets the ProductFamily string for the application identity.

Type	string
Accessors	`Get, Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v1.0</td><td>✓ | v10.0</td><td>✓ | v10.0</td><td>✓ | v11.0</td><td>✓ | v12.1</td></tr></table>
Usage notes	The ProductFamily property is a part of the application identity. The application identity consists of the Manufacturer, ProductFamily, ProductName and VersionInfo. All the application identity information should be set prior to invoking OpenSourceManager(). Since with built-in Wizard Mode, Dynamic TWAIN manages the transition of TWAIN state intelligently, OpensourceManager() may be called automatically by other TWAIN related functions, such as OpenSource() or AcquireImage(). It is recommended that application identity information be set prior to any other TWAIN functions.

ProductKey

Returns or sets the ProductKey string for the application identity.

Type	string

Accessors	`Get, Set`
Availability	**ActiveX**: ✓ | v9.0 — **H5(Windows)**: ✓ | v10.0 — **H5(macOS/TWAIN)**: ✓ | v10.0 — **H5(macOS/ICA)**: ✓ | v11.0 — **H5(Linux)**: ✓ | v12.1
Usage notes	Each product key is generated with one or a set of licenses. It's not allowed to run your application which based on Dynamic Web TWAIN without a product key (the licenses).

ProductName

Returns or sets the ProductName string for the application identity.

Type	string
Accessors	`Get, Set`
Availability	**ActiveX**: ✓ | v1.0 — **H5(Windows)**: ✓ | v10.0 — **H5(macOS/TWAIN)**: ✓ | v10.0 — **H5(macOS/ICA)**: ✓ | v11.0 — **H5(Linux)**: ✓ | v12.1
Usage notes	The ProductName property is a part of the application identity. The application identity consists of the Manufacturer, ProductFamily, ProductName and VersionInfo. All the application identity information should be set prior to invoking OpenSourceManager(). Since with built-in Wizard Mode, Dynamic TWAIN manages the transition of TWAIN state intelligently, OpensourceManager() may be called automatically by other TWAIN related functions, such as OpenSource() or AcquireImage(). It is recommended that application identity information be set prior to any other TWAIN functions.

VersionInfo

Returns or sets the VersionInfo string for the application identity.

Type	string
Accessors	`Get, Set`
Availability	**ActiveX**: ✓ | v1.0 — **H5(Windows)**: ✓ | v10.0 — **H5(macOS/TWAIN)**: ✓ | v10.0 — **H5(macOS/ICA)**: ✓ | v11.0 — **H5(Linux)**: ✓ | v12.1
Usage notes	The VersionInfo property is a part of the application identity. The application identity consists of the Manufacturer, ProductFamily, ProductName and VersionInfo. All the application identity information should be set prior to invoking OpenSourceManager(). Since with built-in Wizard Mode, Dynamic TWAIN manages the transition of TWAIN state intelligently, OpensourceManager() may be called automatically by other TWAIN related functions, such as OpenSource() or AcquireImage(). It is recommended that application identity information be set prior to any other TWAIN functions.

Events

OnWebTwainReady

This event is triggered as soon as Dynamic Web TWAIN is successfully loaded and initialized on the page.

Syntax	Dynamsoft.WebTwainEnv.RegisterEvent('OnWebTwainReady', Dynamsoft_OnReady); functior Dynamsoft_OnReady() {...} or Dynamsoft.WebTwainEnv.RegisterEvent('OnWebTwainReady', function() {...});
Arguments	None
Example	

```
Dynamsoft.WebTwainEnv.RegisterEvent('OnWebTwainReady', Dynamsoft_OnReady);

function Dynamsoft_OnReady() {
    var DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
    DWObject.Width = 270;
    DWObject.Height = 350;
    //Add event listeners to DWObject
    DWObject.RegisterEvent("OnPostTransfer", function(){...});
    DWObject.RegisterEvent("OnPostLoad", function(){...});
    DWObject.RegisterEvent("OnMouseClick", function(){...});
}
```

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v8.0	✓ | v10.0	✓ | v10.0	✓ | v11.0	✓ | v12.1

Usage notes	This is the best place to set up Dynamic Web TWAIN (change the size of it, add event listeners etc.) as shown below in the example.

OnBitmapChanged

This event is triggered when the current image in buffer is changed like flipped, cropped, rotated, etc. or a new image has been acquired.

Syntax	.RegisterEvent('OnBitmapChanged',function(){...});
Arguments	None
Example	

```
DWObject.RegisterEvent('OnBitmapChanged', function() {
    alert('image is changed');
});
```

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v8.0	✓ | v10.0	✓ | v10.0	✓ | v11.0	✓ | v12.1

Addons

1. PDF Raterizer
2. Webcam Capture
3. File Uploader
4. Barcode Reader
5. OCR Basic
6. OCR Pro

PDF Rasterizer

Methods	
Addon.PDF.Download()	Addon.PDF.SetResolution()
Addon.PDF.SetPassword()	Addon.PDF.SetConvertMode()

Code example

The following code example demonstrates how to use the APIs above to perform basic scanning.

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

    function OnFailure(errorCode, errorString) {
        alert(errorString);
    }

    function LoadImage() {
        if (DWObject) {
            //Please NOTE that the PDF Rasterizer doesn't work for Chrome/Firefox 26-
            // Call DWObject.Addon.PDF.Download(url) to download PDF Rasterizer module to local.
            DWObject.Addon.PDF.SetResolution(200);
            DWObject.Addon.PDF.SetConvertMode(EnumDWT_ConvertMode.CM_RENDERALL);

            DWObject.IfShowFileDialog = true; // Open the system's file dialog to load image
            DWObject.LoadImageEx("", EnumDWT_ImageType.IT_PDF, OnSuccess, OnFailure);
            // Load images in all supported formats (.bmp, .jpg, .tif, .png, .pdf). OnSuccess or OnFailure will be ca
lled after the operation
        }
    }
```

Methods

Addon.PDF.Download()	
Downloads and installs the PDF add-on (it's typically a zipped dll file) on the local system.	
Syntax	.Addon.PDF.Download(remoteFile, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string remoteFile` : specifies the url path of the add-on. E.g. "http://www.dynamsoft.com/DWT/Resources/PDF.zip". `OnSuccess function optional optionalAsyncSuccessFunc` : callback function triggered when the file is downloaded successfully. `OnFailure function optional optionalAsyncFailureFunc` : callback function triggered when the file failed to be downloaded.
Return value	`boolean` Only valid when used synchronously.
Usage notes	The server version number info is defined in the addon JS file. As long as the defined version number is not the same as the local addon dll, the download method will auto

	get the dll from server deployed to local.
Availability	v11.2+

Addon.PDF.SetResolution()	
Sets the output image resolution of the PDF Rasterizer.	
Syntax	.Addon.PDF.SetResolution(resolution)
Parameters	`number resolution` : specifies the resolution of the output images.
Return value	`boolean`
Usage notes	The default value is 200. We recommend that you set a value smaller than 300, otherwise it might slow down the program or cause the process to fail.
Availability	v11.2+

Addon.PDF.SetPassword()	
Specifies the password needed for rasterizing a password-protected PDF file.	
Syntax	.Addon.PDF.SetPassword(password)
Parameters	`string password` : Specifies the PDF password.
Return value	`boolean`
Usage notes	This API is only available in the HTML5 edition for Windows.
Availability	v11.2+

Addon.PDF.SetConvertMode()	
Sets the image convert mode for the PDF Rasterizer.	
Syntax	.Addon.PDF.SetConvertMode(EnumDWT_ConvertMode.CM_RENDERALL);
Parameters	`EnumDWT_ConvertMode` `EnumDWT_ConvertMode.CM_DEFAULT` : It's the default mode. In this mode, the PDF Rasterizer is turned off. `EnumDWT_ConvertMode.CM_RENDERALL` : All the content in the target PDF file will converted in a set resolution in this mode. The value of the resolution is 200 by default but can be set via the method Addon.PDF.SetResolution.
Return value	`boolean`
Usage notes	Use this method before you import a PDF into the control with methods such as LoadImage() and FTPDownload().
Availability	v11.2+

Webcam Capture

Methods	
Addon.Webcam.CaptureImage()	Addon.Webcam.CloseSource()
Addon.Webcam.Download()	Addon.Webcam.GetCameraControlPropertyMoreSet
Addon.Webcam.GetCameraControlPropertySetting()	Addon.Webcam.GetFrameRate()
Addon.Webcam.GetFramePartUrl()	Addon.Webcam.GetFrameUrl()
Addon.Webcam.GetMediaType()	Addon.Webcam.GetResolution()
Addon.Webcam.GetSourceList()	Addon.Webcam.GetVideoPropertyMoreSetting()
Addon.Webcam.GetVideoPropertySetting()	Addon.Webcam.PauseVideo()
Addon.Webcam.PlayVideo()	Addon.Webcam.SelectSource()
Addon.Webcam.SetCameraControlPropertySetting()	Addon.Webcam.SetFrameRate()
Addon.Webcam.SetMediaType()	Addon.Webcam.SetResolution()
Addon.Webcam.SetVideoPropertySetting()	Addon.Webcam.SetVideoRotateMode()
Addon.Webcam.StopVideo()	

Methods

Addon.Webcam.CaptureImage()	
Captures an image from the current Webcam.	
Syntax	.Addon.Webcam.CaptureImage(OnCaptureSuccess, OnCaptureError);
Parameters	`Function` `OnCaptureSuccess` : callback function fired when capturing suceeds. `Function` `OnCaptureError` : callback function fired when capturing fails.
Return value	`Void`
Availability	**ActiveX**: ✓ | v14.3.1 — **H5(Windows)**: ✓ | v14.3 — **H5(macOS/TWAIN)**: ✗ — **H5(macOS/ICA)**: ✗ — **H5(Linux)**: ✗

Addon.Webcam.CloseSource()	
Closes the current Webcam Source.	
Syntax	.Addon.Webcam.CloseSource();
Parameters	`None`
Return value	`Boolean`

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v14.3.1	✓ | v14.3	✗	✗	✗

Usage notes	When you close the camera, the video stream will also stop at the last frame.

Addon.Webcam.Download()

Downloads and installs Webcam add-on (it's typically a zipped dll file) on the local system.

Syntax	.Addon.Webcam.Download(strFilePath, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	`string` strFilePath: the path of the .zip file that contains the Webcam add-on. The following two parameters are optional. If either one exists or both exist, the method is asynchronous, otherwise it's synchronous. `<OnSuccess function>` `optional` `optionalAsyncSuccessFunc` : callback function triggered when the file is downloaded successfully. `<OnFailure function>` `optional` `optionalAsyncFailureFunc` : callback function triggered when the file failed to be downloaded. Please refer to the function prototype OnSuccess or OnFailure.
Return value	`boolean` , only valid when used synchronnously.
Usage notes	The download will occur when the Webcam add-on doesn't exist on the local machine or the local add-on is of a different version.

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v14.3.1	✓ | v14.3	✗	✗	✗

Addon.Webcam.GetCameraControlPropertyMoreSetting()

Returns the range and default values of a specified Webcam property.

Syntax	.Addon.Webcam.GetCameraControlPropertyMoreSetting(nProperty);

`number` `nProperty` : specifies the property to query. Allowed values are

Values	EnumDWT_CameraControlProperty	Description
0	`CCP_PAN`	Specifies the camera's pan setting, in degrees. Values range from –180 to +180, with the default set to zero. Positive values are clockwise from the origin (the camera rotates clockwise when viewed from above), and negative values are counterclockwise from the origin.
1	`CCP_TILT`	Specifies the camera's tilt setting, in degrees. Values range from –180 to +180, with the default set to zero. Positive values point the imaging plane up, and negative values point the imaging plane down.

Parameters

2	`CCP_ROLL`	Specifies the camera's roll setting, in degrees. Values range from –180 to +180, with the default set to zero. Positive values cause a clockwise rotation of the camera along the image-viewing axis, and negative values cause a counterclockwise rotation of the camera.
3	`CCP_ZOOM`	Specifies the camera's zoom setting, in millimeters. Values range from 10 to 600, and the default is specific to the device.
4	`CCP_EXPOSURE`	Specifies the exposure setting, in log base 2 seconds. In other words, for values less than zero, the exposure time is 1/2^n seconds, and for values zero or above, the exposure time is 2^n seconds. For example: Value Seconds -3 1/8 -2 1/4 -1 1/2 0 1 1 2 2 4
5	CCP_IRIS	Specifies the camera's iris setting, in units of f_{stop}* 10.
6	CCP_FOCUS	Specifies the camera's focus setting, as the distance to the optimally focused target, in millimeters. The range and default value are specific to the device.

Return value

An object of the type `CameraControlMoreSetting` is returned. Methods available in the object are

Data Type	Method Name	Description
`<long>`	GetMinValue()	Returns the minimum value of the property.
`<long>`	GetMaxValue()	Returns the maximum value of the property.
`<long>`	GetSteppingDelta()	Returns the step size for the property. The step size is the smallest increment by which the property can change.
`<long>`	GetDefaultValue()	Returns the default value of the property.
`<Boolean>`	GetIfAuto()	Returns a value that indicates whether the setting is controlled manually or automatically.

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v14.3.1	✓ | v14.3	✗	✗	✗

Addon.Webcam.GetCameraControlPropertySetting()

Returns the basic settings of a camera property.

Syntax

.Addon.Webcam.GetCameraControlPropertySetting(nProperty);

Parameters

`number _nProperty` : specifies the property to query. Allowed values are

Values	EnumDWT_CameraControlProperty	Description
0	`CCP_PAN`	Specifies the camera's pan setting, in degrees. Values range from –180 to +180, with the default set to zero. Positive values are clockwise from the origin (the camera rotates clockwise when viewed from above), and negative values are counterclockwise from the origin.
1	`CCP_TILT`	Specifies the camera's tilt setting, in degrees. Values range from –180 to +180, with the default set to zero. Positive values point the imaging plane up, and negative values point the imaging plane down.
2	`CCP_ROLL`	Specifies the camera's roll setting, in degrees. Values range from –180 to +180, with the default set to zero. Positive values cause a clockwise rotation of the camera along the image-viewing axis, and negative values cause a counterclockwise rotation of the camera.
3	`CCP_ZOOM`	Specifies the camera's zoom setting, in millimeters. Values range from 10 to 600, and the default is specific to the device.
4	`CCP_EXPOSURE`	Specifies the exposure setting, in log base 2 seconds. In other words, for values less than zero, the exposure time is 1/2^n seconds, and for values zero or above, the exposure time is 2^n seconds. For example: Value Seconds -3 1/8 -2 1/4 -1 1/2 0 1 1 2 2 4
5	CCP_IRIS	Specifies the camera's iris setting, in units of f_{stop}* 10.
6	CCP_FOCUS	Specifies the camera's focus setting, as the distance to the optimally focused target, in millimeters. The range and default value are specific to the device.

<table>
<tr><td>Return value</td><td>An object of the type <code>CameraControlSetting</code> is returned. Methods available in the object are
<table>
<tr><th>Data Type</th><th>Method Name</th><th>Description</th></tr>
<tr><td><code><Number></code></td><td>GetValue()</td><td>Returns the value of the property.</td></tr>
<tr><td><code><Boolean></code></td><td>GetIfAuto()</td><td>Returns a value that indicates whether the setting is controlled manually or automatically.</td></tr>
</table></td></tr>
<tr><td>Availability</td><td>
<table>
<tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr>
<tr><td>✓ | v14.3.1</td><td>✓ | v14.3</td><td>✗</td><td>✗</td><td>✗</td></tr>
</table></td></tr>
</table>

Addon.Webcam.GetFrameRate()

Retrieves the frame rates supported by the current Webcam.

<table>
<tr><td>Syntax</td><td>.Addon.Webcam.GetFrameRate();</td></tr>
<tr><td>Parameters</td><td><code>None</code></td></tr>
<tr><td>Return value</td><td>An object of the type <code>WebcamFrameRate</code> is returned which contains a list of all available frame rates. Methods available in the object are
<table>
<tr><th>Data Type</th><th>Method Name</th><th>Description</th></tr>
<tr><td><code><Number></code></td><td>GetCount()</td><td>Returns the count of available frame rates.</td></tr>
<tr><td><code><String></code></td><td>Get(Number index)</td><td>Returns a frame rate from the list.</td></tr>
<tr><td><code><String></code></td><td>GetCurrent()</td><td>Returns the current frame rate.</td></tr>
</table></td></tr>
<tr><td>Usage notes</td><td>This method should be called after <code>Addon.Webcam.SelectSource()</code> . If there is only one data source available, it will be selected automatically.</td></tr>
<tr><td>Availability</td><td>
<table>
<tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr>
<tr><td>✗</td><td>✓ | v14.3.1</td><td>✗</td><td>✗</td><td>✗</td></tr>
</table></td></tr>
</table>

Addon.Webcam.GetFramePartUrl()

Gets the internal URL (dwt://) of the latest frame of the video stream.

<table>
<tr><td>Syntax</td><td>.Addon.Webcam.GetFramePartUrl();</td></tr>
<tr><td>Parameters</td><td><code>None</code></td></tr>
<tr><td>Return value</td><td><code>String</code></td></tr>
<tr><td>Usage notes</td><td>The returned URL will be like "dwt://dwt_trial_13000404/img?id=306159652&index=0&t=1502184632022".</td></tr>
<tr><td>Availability</td><td>
<table>
<tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr>
<tr><td>✓ | v14.3.1</td><td>✓ | v14.3.1</td><td>✗</td><td>✗</td><td>✗</td></tr>
</table></td></tr>
</table>

Addon.Webcam.GetFrameUrl()

Gets the URL (http(s)://) of the latest frame of the video stream.

Syntax	.Addon.Webcam.GetFrameUrl();
Parameters	`None`
Return value	`String`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✗	✓ | v14.3.1	✗	✗	✗

Addon.Webcam.GetMediaType()

Retrieves the media types supported by the current Webcam.

Syntax	.Addon.Webcam.GetMediaType();
Parameters	`None`

Return value

An object of the type `WebcamMediaType` is returned which contains a list of all available media types. Methods available in the object are

Data Type	Method Name	Description
`<Number>`	GetCount()	Returns the count of available media types.
`<String>`	Get(Number index)	Returns a media type from the list.
`<String>`	GetCurrent()	Returns the current media type.

Usage notes

This method should be called after `Addon.Webcam.SelectSource()` . If there is only one data source available, it will be selected automatically.

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v14.3.1	✓ | v14.3	✗	✗	✗

Addon.Webcam.GetResolution();

Retrieves the resolutions supported by the current Webcam.

Syntax	.Addon.Webcam.GetResolution()
Parameters	`None`

Return value

An object of the type `WebcamResolution` is returned which contains a list of all available resolutions. Methods available in the object are

Data Type	Method Name	Description
`<Number>`	GetCount()	Returns the count of available resolutions.
`<String>`	Get(Number index)	Returns a resolution from the list.
`<String>`	GetCurrent()	Returns the current resolution.

Usage notes	This method should be called after `Addon.Webcam.SelectSource()` . If there is only one data source available, it will be selected automatically.
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v14.3.1	✓ | v14.3	✗	✗	✗

Addon.Webcam.GetSourceList()

Gets a list of all available Webcams.

Syntax	.Addon.Webcam.GetSourceList();
Parameters	`None`
Return value	`String[]` An array of strings containing all Webcam names e.g. ["HD Webcam", "HP Webcam 123", "USB video driver"].
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v14.3.1	✓ | v14.3	✗	✗	✗

Usage notes	When you call this method, the camera addon will be restored. In other words, the video strear will be cut off and the selected camera will be deselected and previous settings will be lost.

Addon.Webcam.GetVideoPropertyMoreSetting()

Returns the range and default values of a specified video property.

Syntax	.Addon.Webcam.GetVideoPropertyMoreSetting(nProperty);
Parameters	`number` `nProperty` : Specifies the property to query, allowed values are:

Values	EnumDWT_VideoProperty	Description
0	VP_BRIGHTNESS	Specifies the brightness, also called the black level. For NTSC, the value is expressed in IRE units * 100. For non-NTSC sources, the units are arbitrary, with zero representing blanking and 10,000 representing pure white. Values range from –10,000 to 10,000.
1	VP_CONTRAST	Specifies the contrast, expressed as gain factor * 100. Values range from zero to 10,000.
2	VP_HUE	Specifies the hue, in degrees * 100. Values range from -180,000 to 180,000 (-180 to +180 degrees).
3	VP_SATURATION	Specifies the saturation. Values range from 0 to 10,000.
4	VP_SHARPNESS	Specifies the sharpness. Values range from 0 to 100.
5	VP_GAMMA	Specifies the gamma, as gamma * 100. Values range from 1 to 500.

	6	VP_COLORENABLE	Specifies the color enable setting. The possible values are 0 (off) and 1 (on).
	7	VP_WHITEBALANCE	Specifies the white balance, as a color temperature in degrees Kelvin. The range of values depends on the device.
	8	VP_BACKLIGHTCOMPENSATION	Specifies the backlight compensation setting. Possible values are 0 (off) and 1 (on).
	9	VP_GAIN	Specifies the gain adjustment. Zero is normal. Positive values are brighter and negative values are darker. The range of values depends on the device.

Return value

An object of the type `VideoPropertyMoreSetting` is returned. Methods available in the object are

Data Type	Method Name	Description
`<Number>`	GetMinValue()	Returns the minimum value of the property.
`<Number>`	GetMaxValue()	Returns the maximum value of the property.
`<Number>`	GetSteppingDelta()	Returns the step size for the property. The step size is the smallest increment by which the property can change.
`<Number>`	GetDefaultValue()	Returns the default value of the property.
`<Boolean>`	GetIfAuto()	Returns a value that indicates whether the setting is controlled manually or automatically.

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v14.3.1	✓ | v14.3	✗	✗	✗

Addon.Webcam.GetVideoPropertySetting()

Returns the basic settings of a video property.

Syntax

.Addon.Webcam.GetVideoPropertySetting(nProperty);

`number nProperty` : Specifies the property to query, allowed values are:

Values	EnumDWT_VideoProperty	Description
0	VP_BRIGHTNESS	Specifies the brightness, also called the black level. For NTSC, the value is expressed in IRE units * 100. For non-NTSC sources, the units are arbitrary, with zero representing blanking and 10,000 representing pure white. Values range from –10,000 to 10,000.
1	VP_CONTRAST	Specifies the contrast, expressed as gain factor * 100. Values range from

<table>
<tr><td>Parameters</td><td>
<table>
<tr><td></td><td></td><td>zero to 10,000.</td></tr>
<tr><td>2</td><td>VP_HUE</td><td>Specifies the hue, in degrees * 100. Values range from -180,000 to 180,000 (-180 to +180 degrees).</td></tr>
<tr><td>3</td><td>VP_SATURATION</td><td>Specifies the saturation. Values range from 0 to 10,000.</td></tr>
<tr><td>4</td><td>VP_SHARPNESS</td><td>Specifies the sharpness. Values range from 0 to 100.</td></tr>
<tr><td>5</td><td>VP_GAMMA</td><td>Specifies the gamma, as gamma * 100. Values range from 1 to 500.</td></tr>
<tr><td>6</td><td>VP_COLORENABLE</td><td>Specifies the color enable setting. The possible values are 0 (off) and 1 (on).</td></tr>
<tr><td>7</td><td>VP_WHITEBALANCE</td><td>Specifies the white balance, as a color temperature in degrees Kelvin. The range of values depends on the device.</td></tr>
<tr><td>8</td><td>VP_BACKLIGHTCOMPENSATION</td><td>Specifies the backlight compensation setting. Possible values are 0 (off) and 1 (on).</td></tr>
<tr><td>9</td><td>VP_GAIN</td><td>Specifies the gain adjustment. Zero is normal. Positive values are brighter and negative values are darker. The range of values depends on the device.</td></tr>
</table>
</td></tr>
<tr><td>Return value</td><td>An object of the type <code>VideoPropertySetting</code> is returned. Methods available in the object are
<table>
<tr><th>Data Type</th><th>Method Name</th><th>Description</th></tr>
<tr><td><code><Number></code></td><td>GetValue()</td><td>Returns the current value of the property.</td></tr>
<tr><td><code><Boolean></code></td><td>GetIfAuto()</td><td>Returns a value that indicates whether the setting is controlled manually or automatically.</td></tr>
</table>
</td></tr>
<tr><td>Availability</td><td>
<table>
<tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr>
<tr><td>✓ | v14.3.1</td><td>✓ | v14.3</td><td>✗</td><td>✗</td><td>✗</td></tr>
</table>
</td></tr>
</table>

Addon.Webcam.PauseVideo()

Pauses the video stream.

Syntax	.Addon.Webcam.PauseVideo();
Parameters	`None`
Return value	`Boolean`
Availability	

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ |				

	v14.3.1	✓ | v14.3	✗	✗	✗
Usage notes	This method only pauses the video for the HTML5 edition. Also, when you capture an image, it be based on the actual frame at that point from the camera, not the paused frame.				

Addon.Webcam.PlayVideo()

Starts to play the video stream in a specified container.

Syntax	.Addon.Webcam.PlayVideo(DWObject, nQuality ,onFrameCaptured);
Parameters	`Object` `DWObject` : Specifies the object to hold the video stream. `number` `nQuality` : Specifies the quality of each frame in the video stream. Only valid for the HTML5 edition. `optional` `Function` `onFrameCaptured` : Specifies the callback function for each showing frame in the video stream.
Return value	`None`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v14.3.1	✓ | v14.3	✗	✗	✗

Addon.Webcam.SelectSource()

Selects an available Webcam.

Syntax	.Addon.Webcam.SelectSource(strWebcamName);
Parameters	`string` `strWebcamName` : specifies the Webcam to select.
Return value	`Boolean`

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v14.3.1	✓ | v14.3	✗	✗	✗

Addon.Webcam.SetCameraControlPropertySetting()

Sets a property for the current camera.

Syntax	.Addon.Webcam.SetCameraControlPropertySetting(nProperty, nValue, bAuto);

`number` `nProperty` : specifies the property to change. Allowed values are

Values	EnumDWT_CameraControlProperty	Description
0	`CCP_PAN`	Specifies the camera's pan setting, in degrees. Values range from –180 to +180, with the default set to zero. Positive values are clockwise from the origin (the camera rotates clockwise when viewed from above), and negative values are counterclockwise from the origin.

Parameters

1	`CCP_TILT`	Specifies the camera's tilt setting, in degrees. Values range from –180 to +180, with the default set to zero. Positive values point the imaging plane up, and negative values point the imaging plane down.
2	`CCP_ROLL`	Specifies the camera's roll setting, in degrees. Values range from –180 to +180, with the default set to zero. Positive values cause a clockwise rotation of the camera along the image-viewing axis, and negative values cause a counterclockwise rotation of the camera.
3	`CCP_ZOOM`	Specifies the camera's zoom setting, in millimeters. Values range from 10 to 600, and the default is specific to the device.
4	`CCP_EXPOSURE`	Specifies the exposure setting, in log base 2 seconds. In other words, for values less than zero, the exposure time is 1/2^n seconds, and for values zero or above, the exposure time is 2^n seconds. For example: Value Seconds -3 1/8 -2 1/4 -1 1/2 0 1 1 2 2 4
5	CCP_IRIS	Specifies the camera's iris setting, in units of f_{stop}* 10.
6	CCP_FOCUS	Specifies the camera's focus setting, as the distance to the optimally focused target, in millimeters. The range and default value are specific to the device.

`number` `nValue` : Specifies the value to set to the property. `boolean` `bAuto` : Specifies whether the setting is controlled manually or automatically.

Return value

`Boolean`

Usage notes

This method should be called after `Addon.Webcam.SelectSource()` . If there is only one data source available, it will be selected automatically.

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v14.3.1	✓ | v14.3	✗	✗	✗

Addon.Webcam.SetFrameRate()

Sets the frame rate of the current Webcam.

Sets the frame rate of the current Webcam.

Syntax	.Addon.Webcam.SetFrameRate(nValue);
Parameters	`number` `nValue` : Specifies the frame rate.
Return value	`Boolean`
Usage notes	This method should be called after `Addon.Webcam.SelectSource()` . If there is only one data source available, it will be selected automatically.

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v14.3.1	✓ | v14.3	✗	✗	✗

Addon.Webcam.SetMediaType()

Sets the media type of the current Webcam.

Syntax	.Addon.Webcam.SetMediaType(strMediaType);
Parameters	`string` `strMediaType` : Specifies the media type.
Return value	`Boolean`
Usage notes	This method should be called after `Addon.Webcam.SelectSource()` . If there is only one data source available, it will be selected automatically. Make sure you only set a supported media type which you can get using the method `Addon.Webcam.GetMediaType()` .

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v14.3.1	✓ | v14.3	✗	✗	✗

Addon.Webcam.SetResolution()

Sets the resolution of the current Webcam.

Syntax	.Addon.Webcam.SetResolution(strResolution)
Parameters	`string` `strResolution` : Specifies the resolution.
Return value	`Boolean`
Usage notes	This method should be called after `Addon.Webcam.SelectSource()` . If there is only one data source available, it will be selected automatically. Make sure you only set a supported resolution whicl you can get using the method `Addon.Webcam.GetResolution()` .

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v14.3.1	✓ | v14.3	✗	✗	✗

Addon.Webcam.SetVideoPropertySetting()

Sets a property for the video stream.

Parameters

`number` `nProperty` : Specifies the property to change, allowed values are:

Values	EnumDWT_VideoProperty	Description
0	VP_BRIGHTNESS	Specifies the brightness, also called the black level. For NTSC, the value is expressed in IRE units * 100. For non-NTSC sources, the units are arbitrary, with zero representing blanking and 10,000 representing pure white. Values range from –10,000 to 10,000.
1	VP_CONTRAST	Specifies the contrast, expressed as gain factor * 100. Values range from zero to 10,000.
2	VP_HUE	Specifies the hue, in degrees * 100. Values range from -180,000 to 180,000 (-180 to +180 degrees).
3	VP_SATURATION	Specifies the saturation. Values range from 0 to 10,000.
4	VP_SHARPNESS	Specifies the sharpness. Values range from 0 to 100.
5	VP_GAMMA	Specifies the gamma, as gamma * 100. Values range from 1 to 500.
6	VP_COLORENABLE	Specifies the color enable setting. The possible values are 0 (off) and 1 (on).
7	VP_WHITEBALANCE	Specifies the white balance, as a color temperature in degrees Kelvin. The range of values depends on the device.
8	VP_BACKLIGHTCOMPENSATION	Specifies the backlight compensation setting. Possible values are 0 (off) and 1 (on).
9	VP_GAIN	Specifies the gain adjustment. Zero is normal. Positive values are brighter and negative values are darker. The range of values depends on the device.

`number` `nValue` : Specifies the value to set to the property. `boolean` `bAuto` : Specifies whether the setting is controlled manually or automatically.

Return value

`Boolean`

Usage notes

This method should be called after `Addon.Webcam.SelectSource()` . If there is only one data source available, it will be selected automatically.

Availability

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v14.3.1	✓ | v14.3	✗	✗	✗

Addon.Webcam.SetVideoRotateMode()

Sets video rotate mode of the current Webcam.

Syntax	.Addon.Webcam.SetVideoRotateMode(nVideoRotateMode);
Parameters	`number` `nVideoRotateMode` : Specifies the video rotate mode on a video capture device, allowed values are:

Values	EnumDWT_VideoRotateMode enumeration	Description
0	VRM_NONE	Don't rotate
1	VRM_90_DEGREES_CLOCKWISE	90 deg Clockwise
2	VRM_180_DEGREES_CLOCKWISE	180 deg Clockwise
3	VRM_270_DEGREES_CLOCKWISE	270 deg Clockwise
4	VRM_FLIP_VERTICAL	Flip
5	VRM_FLIP_HORIZONTAL	Mirror

Return value	`Boolean`
Usage notes	This method should be called after `Addon.Webcam.SelectSource()` . If there is only one data source available, it will be selected automatically.

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v14.3.1	✓ | v14.3	✗	✗	✗

Addon.Webcam.StopVideo()

Stops the video stream.

Syntax	.Addon.Webcam.StopVideo();
Parameters	`None`
Return value	`Boolean`

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v14.3.1	✓ | v14.3	✗	✗	✗

File Uploader

> The `FileUploader` is an independent module that handles the uploading of files. It's released with Dynamic Web TWAIN version 14.0. Once an upload job starts, it no longer relies on Dynamic Web TWAIN which means the upload carries on even after you have closed the browser. The files/data to upload is stored temporarily on the local disk (typically in the path `C:\Windows\SysWOW64\Dynamsoft\DynamsoftService\upload`) before they're uploaded and will be purged as soon as they're uploaded or when the Dynamsoft Service restarts.

Methods		
Init()	CreateJob()	Run()
Cancel()	CancelAllUpload()	GenerateURLForUploadData()

Properties		
ServerUrl	HttpHeader	SourceValue
FormField		

Events		
OnUploadTransferPercentage	OnRunSuccess	OnRunFailure

Code example

The following code example demonstrates how to use the APIs above to perform uploading.

```
var fileUploaderManager;

function onInitSuccess(objFileUploader) {
    fileUploaderManager = objFileUploader;
}

function onInitFailure(errorCode, errorString) {
    alert('Init failed: ' + errorString);
};

Dynamsoft.FileUploader.Init("", onInitSuccess, onInitFailure);

function UploadFile() {
    var job = fileUploaderManager.CreateJob();
    job.ServerUrl ='http://yourserver/youractionpage.aspx';
    job.FileName = "sample.jpg";
    job.ImageType = EnumDWT_ImageType.IT_JPG;
    DWObject.GenerateURLForUploadData([0],EnumDWT_ImageType.IT_JPG, function(result) {
        job.SourceValue.Add(result, "sample.jpg");
        job.FormField.Add('customField', 'FormFieldValue');
        job.OnUploadTransferPercentage = FileUpload_OnUploadTransferPercentage;
        job.OnRunSuccess = FileUpload_OnRunSuccess;
        job.OnRunFailure = FileUpload_OnRunFailure;
        fileUploaderManager.Run(job);
    }, function(errorCode, errorString) {
        console.log(errorString);
    });
```

```
}

function FileUpload_OnUploadTransferPercentage(job, sPercentage) {
    console.log(sPercentage);
}

function FileUpload_OnRunFailure(job, errorCode, errorString) {
    alert(errorString);
}

function FileUpload_OnRunSuccess(job) {
    alert(' upload completed! ');
}
```

Methods

Init()	
This is a global API that initiates the FileUploader module.	
Syntax	Dynamsoft.FileUploader.Init(remoteFile, asyncSuccessFunc, asyncFailureFunc);
Parameters	`string` remoteFile: an URL that specifies the path of the FileUploader library on the server. In v14.0, the library is installed together with the core|scan module of Dynamic Web TWAIN, so you can set this parameter to be an empty string. `OnSuccess function` `asyncSuccessFunc` : callback function triggered when initiation succeeds, this function has a primitive parameter which refers to an UploadManager object that has just been initiated. `OnFailure function` `asyncFailureFunc` : callback function triggered when initiation fails. When the FileUploader library (.dll) is not installed or the installed file is a different version, Init will try to download and install the library from the path specified by remoteFile. If that fails, this callback function is triggered. Please refer to the function prototype OnSuccess or OnFailure.
Example	`var dsUploadManager;` `Dynamsoft.FileUploader.Init('', function(obj) {` `  dsUploadManager = obj;` `}, function(){});`
Availability	**ActiveX** | **H5(Windows)** | **H5(macOS/TWAIN)** | **H5(macOS/ICA)** | **H5(Linux)** ✗ | ✓ | v14.0 | ✗ | ✗ | ✗

CreateJob()	
Create a upload job.	
Syntax	.CreateJob();
Parameters	`None`
Return value	`Object` A job object. For instance: `{HttpHeader: {}, documents: Array(0), Version: "1.0", HttpVersion: "1.1", ServerUrl: "", …}`
Example	`var job = dsUploadManager.CreateJob();`
	ActiveX | **H5(Windows)** | **H5(macOS/TWAIN)** | **H5(macOS/ICA)** | **H5(Linux)**

Availability	✗	✓ | v14.0	✗	✗	✗

Run()					
Starts to execute an upload job.					
Syntax	.Run(obj);				
Parameters	`Object` obj: A job object.				
Return value	`boolean`				
Example	`var job = dsUploadManager.CreateJob();` `dsUploadManager.Run(job);`				
Availability	**ActiveX**	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✗	✓ | v14.0	✗	✗	✗

Cancel()					
Cancels a upload job after it has started and before it completes. Normally this is done in the event OnUploadTransferPercentage.					
Syntax	.Cancel(obj);				
Parameters	`Object` obj: A job object.				
Return value	`boolean`				
Example	`var job = dsUploadManager.CreateJob();` `job.OnUploadTransferPercentage= FileUpload_OnUploadTransferPercentage;` `dsUploadManager.Run(job);` `function FileUpload_OnUploadTransferPercentage(job, iPercentage)` `{` `  console.log('job cancelled!')` `  dsUploadManager.Cancel(job);` `}`				
Availability	**ActiveX**	**H5(Windows)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✗	✓ | v14.0	✗	✗	✗

CancelAllUpload()	
Cancels all upload jobs.	
Syntax	.CancelAllUpload();
Parameters	`None`
Return value	`boolean`
Example	`dsUploadManager.CancelAllUpload();`

Example	

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✗	✓ | v14.0	✗	✗	✗

GenerateURLForUploadData()

Generates a URL that will be used by the upload module to fetch the file/data to upload.

Syntax	DWObject.GenerateURLForUploadData(indices, enumImageType, asyncSuccessFunc, asyncFailureFunc);
Parameters	`Number[]` indices: the indices of the images in the buffer. The index is 0-based. `EnumDWT_ImageType` enumImageType: the format in which you'd like the images to be uploaded. `OnSuccess function` `asyncSuccessFunc` :callback function triggered when the operation succeeds. This function will return the result URL. `OnFailure function` `asyncFailureFunc` : callback function triggered when the operation fails. Please refer to the function prototype OnSuccess or OnFailure.
Example	

```
Dynamsoft.FileUploader.Init('', function(obj){dsUploadManager=obj}, function(){});
DWObject.GenerateURLForUploadData([0,1], EnumDWT_ImageType.IT_PDF,
    function(result){
        var serverurl= "https://yoursite/yourserverurl.aspx";
        var jobtemp = dsUploadManager.CreateJob();
        jobtemp.ServerUrl = serverUrl;
        jobtemp.SourceValue.Add(result, "uploadedFile.pdf");
        dsUploadManager.Run(jobtemp);
    }, function(){});
```

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✗	✓ | v14.0	✗	✗	✗

Properties

ServerUrl

Specifies the target of the HTTP Post Request of the upload job. This typically is a file on the server. For example: job.ServerUrl = 'http://www.dynamsoft.com/ScanAndUpload/Actions/SaveToFile.aspx';

Type	`string`
Accessors	`Get Set`

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✗	✓ | v14.0	✗	✗	✗

HttpHeader

Specifies headers in the the HTTP Post Request of the upload job. For example: job.HttpHeader["Content-Type"] = "text/plain";

Type	`Object`

Accessors	`Set`
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✗</td><td>✓ | v14.0</td><td>✗</td><td>✗</td><td>✗</td></tr></table>
Usage notes	By default, HttpHeader is an empty object. If left as it is, default headers are used. Otherwise, the headers set by this property will be added to the HTTP Post Request or replace existing ones with the same names.

SourceValue

Specifies the files to be uploaded and the name for it. The files are specified by URLs which can be created with the method GenerateURLForUploadData. This object has a method Add to add file to the job.

Type	`Object`
Accessors	`Set`
Usage notes	Use the `Add(string urltoFetchFileData, string fileName)` method of the Object to add data for uploading, check out the sample code for more information.
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✗</td><td>✓ | v14.0</td><td>✗</td><td>✗</td><td>✗</td></tr></table>

FormField

Specifies extra fields to be uploaded in the same HTTP post.

Type	`Object`
Accessors	`Set`
Usage notes	Use the `Add(string fieldName, string fieldValue)` method of the Object to add fields for uploading, check out the sample code for more information.
Availability	<table><tr><th>ActiveX</th><th>H5(Windows)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✗</td><td>✓ | v14.0</td><td>✗</td><td>✗</td><td>✗</td></tr></table>

Events

OnUploadTransferPercentage

The event is triggered during the execution of an upload job. It has a parameter which specifies the percentage of the completion of the job.

Syntax	.OnUploadTransferPercentage = function(){...};
Arguments	• `Object obj` : A job object. • `number sPercentage` : The percentage of the completion of the job.
Example	<pre>job.OnUploadTransferPercentage = FileUpload_OnUploadTransferPercentage; function FileUpload_ OnUploadTransferPercentage (obj, sPercentage){ console.log(sPercentage); }</pre>

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✗	✓ | v14.0	✗	✗	✗

OnRunSuccess

The event is triggered when an upload job completes successfully.

Syntax	.OnRunSuccess = function(){...};
Arguments	• `Object obj` : A job object.
Example	`job.OnRunSuccess = FileUpload_OnRunSuccess;` `function FileUpload_OnRunSuccess(obj) {` `  alert(' upload completed ');` `}`

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✗	✓ | v14.0	✗	✗	✗

OnRunFailure

The event is triggered when an upload job fails.

Syntax	.OnRunFailure = function(){...};
Arguments	• `Object obj` : A job object. • `number errorCode` : The error code. • `string errorString` : The error string.
Example	`job.OnRunFailure = FileUpload_OnRunFailure;` `function FileUpload_OnRunFailure(obj, errorCode, errorString) {` `  alert(errorString);` `}`

Availability	ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✗	✓ | v14.0	✗	✗	✗

Barcode Reader

Constructor
dynamsoft.BarcodeReader()

Methods		
decode()	decodeBase64String()	getRuntimeSettings()
updateRuntimeSettings()	resetRuntimeSettings()	
dynamsoft.BarcodeReader.initServiceConnection()		

Properties	
productKey	bAutoConnectService
resourcesPath	ifCheck64bitServiceFirst

Events	
onAutoConnectServiceSuccess	onAutoConnectServiceError

Others	
Enumerations	Errors

Code example

The following code example demonstrates how to use the APIs above to perform barcode reading.

```
var reader = new dynamsoft.BarcodeReader("<Put your license key here>");
var runtimeSettings = reader.getRuntimeSettings();
runtimeSettings.mBarcodeFormatIds = 1023; // 1D Barcodes
reader.updateRuntimeSettings(runtimeSettings);
var idx = DWObject.GetSelectedImageIndex(0);
var url = DWObject.GetImagePartURL(idx);
reader.decode(url).then(function(results){
    for(var i = 0; i < results.length; ++i){
        console.log(results[i].BarcodeText);
        console.log(results[i].LocalizationResult.ExtendedResultArray[0].Confidence);
    }
});
```

Constructor

dynamsoft.BarcodeReader()	
Creates an instance of BarcodeReader.	
Syntax	dynamsoft.BarcodeReader(); or dynamsoft.BarcodeReader(licenceKeys);

Parameters	`string licenceKeys (optional)` : The license key for the Barcode Reader add-on.
Return value	`dynamsoft.BarcodeReader`
Example	`var reader = new dynamsoft.BarcodeReader();`
Usage notes	If the parameter licenceKeys is not set, the SDK will get the license key from dynamsoft.dbrEnv.productKey
Availability	lo

ActiveX	H5(Win)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v14.1	✓ | v14.1	✗	✗	✗

Methods

decode()

Decodes barcodes from an image.

Syntax	.decode(source);
Parameters	The image to be decoded. It supports png, jpeg, bmp and tiff files. The parameter `source` supports the following types: `string source (dcsUrl)` `string source (dwtUrl)`
Return value	`Promise(resolve(array TextResult), reject(ex))` Methods available in the `TextResult` object:

Members	Description
ResultType emResultType	The barcode format.
pszBarcodeFormatString	Barcode type in string.
pszBarcodeText	The barcode text, ends by '\0'.
pBarcodeBytes	The barcode content in a byte array.
nBarcodeBytesLength	The length of the byte array.
SLocalizationResult pLocalizationResult	The corresponding localization result.

Example

```
// dwtUrl: HTML5 Edition only
reader.decode('dwt://dwt_trial_13000404/img?id=306159652&index=0&t=1502184632022').then(
    results=>{
    for(var i = 0; i < results.length; ++i){
        console.log(results[i].BarcodeText);
        // Confidence >= 30 is reliable
        console.log(results[i].LocalizationResult.ExtendedResultArray[0].Confidence);
    }
});
// dcsUrl
reader.decode('dcs://dcs_trial_6110531/img?id=306159652&index=0&t=1502184632022').then(
    function(results){
    // ie6-7 does not support console.log
    var messageArr = [];
    for(var i = 0; i < results.length; ++i){
        messageArr.push(results[i].BarcodeText);
        // Confidence >= 30 is reliable
```

```
            messageArr.push(results[i].LocalizationResult.ExtendedResultArray[0].Confidence);
        }
        alert(messageArr.join(''));
    })['catch'](function(ex){
        // ie6-9 does not support '.catch(function(ex){...})'
        if(ex){alert(ex.message||ex);}
    });
```

Availability

ActiveX	H5(Win)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✗	✓ | v14.1	✗	✗	✗

decodeBase64String()

Decodes barcodes from a base64 image.

Syntax

.decodeBase64String(base64Str);

Parameters

`string base64Str` : The barcode image to be decoded.

Return value

`Promise(resolve(array TextResult), reject(ex))`

Methods available in the `TextResult` object:

Members	Description
ResultType emResultType	The barcode format.
pszBarcodeFormatString	Barcode type in string.
pszBarcodeText	The barcode text, ends by '\0'.
pBarcodeBytes	The barcode content in a byte array.
nBarcodeBytesLength	The length of the byte array.
SLocalizationResult pLocalizationResult	The corresponding localization result.

Example

```
var base64str = '';
//with mime
reader.decodeBase64String(base64str).then(results=>{
    for(var i = 0; i < results.length; ++i){
        console.log(results[i].BarcodeText);
    }
});
//without mime
reader.decodeBase64String(base64str.substring(base64str.indexOf(',') + 1)).then(results=>{
    for(var i = 0; i < results.length; ++i){
        console.log(results[i].BarcodeText);
    }
});
```

Availability

ActiveX	H5(Win)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✓ | v14.1	✓ | v14.1	✗	✗	✗

getRuntimeSettings()

Gets the current barcode reading settings.

Syntax

.getRuntimeSettings();

Parameters	None.
Return value	`PlainObject`
Example	`var mysettings = reader.getRuntimeSettings();`
Availability	ActiveX: ✓ | v14.1 / H5(Win): ✓ | v14.1 / H5(macOS/TWAIN): ✗ / H5(macOS/ICA): ✗ / H5(Linux): ✗

updateRuntimeSettings()

Updates the current barcode reading settings.

Syntax	.updateRuntimeSettings(settings);
Parameters	`PublicRuntimeSettings Object settings` : A struct that represents barcode reader settings. `typedef struct PublicRuntimeSettings { number mTimeout; number mBarcodeFormatIds; number mTextureDetectionSensitivity; number mDeblurLevel; number mAntiDamageLevel; number mMaxBarcodesCount; number mScaleDownThreshold; number mGrayEqualizationSensitivity; number mExpectedBarcodesCount; };` For more info, please refer to PublicParameterSettings.
Return value	`undefined`
Example	`//get the barcode reading settings` `var settings = reader.getRuntimeSettings();` `//change the settings` `settings.mBarcodeFormatIds = 1023; // 1D barcodes` `settings.mExpectedBarcodesCount = 10;` `settings.mDeblurLevel = 9;` `settings.mAntiDamageLevel = 9;` `settings.mScaleDownThreshold = 3000;` `//update the settings` `reader.updateRuntimeSettings(settings);`
Availability	ActiveX: ✓ | v14.1 / H5(Win): ✓ | v14.1 / H5(macOS/TWAIN): ✗ / H5(macOS/ICA): ✗ / H5(Linux): ✗

resetRuntimeSettings()

Resets all barcode reading settings to default values.

Syntax	.resetRuntimeSettings();
Parameters	None.
Return value	`undefined`

Example	`reader.resetRuntimeSettings();`
Availability	<table><tr><th>ActiveX</th><th>H5(Win)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v14.1</td><td>✓ | v14.1</td><td>✗</td><td>✗</td><td>✗</td></tr></table>

dynamsoft.BarcodeReader.initServiceConnection()	
Initializes the connection to the Dynamsoft Service.	
Syntax	dynamsoft.BarcodeReader.initServiceConnection();
Parameters	None.
Return value	`Promise(resolve(null), reject(ex))`
Availability	<table><tr><th>ActiveX</th><th>H5(Win)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v14.1</td><td>✓ | v14.1</td><td>✗</td><td>✗</td><td>✗</td></tr></table>

Properties

productKey	
Returns or sets the license key for Barcode Reader add-on.	
Type	`string`
Accessors	`Get Set`
Usage notes	The default value of bAutoConnectService is true. If you want to connect to the service manually, please set it to false before loading "dynamsoft.barcodereader.min.js" and call dynamsoft.BarcodeReader.initServiceConnection when needed.
Availability	<table><tr><th>ActiveX</th><th>H5(Win)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr><tr><td>✓ | v14.1</td><td>✓ | v14.1</td><td>✗</td><td>✗</td><td>✗</td></tr></table>

bAutoConnectService	
Returns or sets whether to connect to the Dynamsoft Service automatically.	
Type	`boolean`
Accessors	`Get Set`
Usage notes	If you don't pass a license or the license has expired, the barcode reader add-on will continue to function normally but the last three characters of the barcode result will be masked with "***".
	<table><tr><th>ActiveX</th><th>H5(Win)</th><th>H5(macOS/TWAIN)</th><th>H5(macOS/ICA)</th><th>H5(Linux)</th></tr></table>

Availability	✓ | v14.1	✓ | v14.1	✗	✗	✗

resourcesPath					
Returns or sets where the barcode reader related dependencies are placed. This is a relative path to the current web page.					
Type	`string`				
Accessors	`Get Set`				
Usage notes	The path is very important as it points to all the JavaScript files, MSI file, images, etc. necessary for the barcode reader add-on to work correctly.				
Availability	**ActiveX**	**H5(Win)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✓ | v14.1	✓ | v14.1	✗	✗	✗

ifCheck64bitServiceFirst					
Returns or sets whether to use Dynamsoft Service 64-bit first.					
Type	`string`				
Accessors	`Get Set`				
Usage notes	The default value of ifCheck64bitServiceFirst is false.				
Availability	**ActiveX**	**H5(Win)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✓ | v14.1	✓ | v14.1	✗	✗	✗

Events

onAutoConnectServiceSuccess					
The success callback function of the service connection.					
Syntax	dynamsoft.dbrEnv.onAutoConnectServiceSuccess = function(){ ... };);				
Arguments	None.				
Example	`dynamsoft.dbrEnv.onAutoConnectServiceSuccess = function(){ console.log("success"); };`				
Availability	**ActiveX**	**H5(Win)**	**H5(macOS/TWAIN)**	**H5(macOS/ICA)**	**H5(Linux)**
	✓ | v14.1	✓ | v14.1	✗	✗	✗

onAutoConnectServiceError

The failure callback function of the service connection.

Syntax	dynamsoft.dbrEnv.onAutoConnectServiceError = function(status){ ... };
Arguments	• `status` : The status of the service.
Example	`dynamsoft.dbrEnv.onAutoConnectServiceError = function(status){ console.log("error"); };`

Availability	ActiveX	H5(Win)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
	✓ | v14.1	✓ | v14.1	✗	✗	✗

Error

`Error` dynamsoft.BarcodeReader.BarcodeReaderException

Member	Type	Description
code	`number` dynamsoft.BarcodeReader.EnumErrorCode	The error code.
message	`String`	The error string.

Environment APIs (Optional)

The following code example demonstrates how to set the Barcode Reader environment.

```
// All settings are optional, including dynamsoft and dynamsoft.dbrEnv.
dynamsoft = self.dynamsoft || {};
dynamsoft.dbrEnv = dynamsoft.dbrEnv || {};
dynamsoft.dbrEnv.productKey = "<a license key>",
dynamsoft.dbrEnv.bAutoConnectService = true;
dynamsoft.dbrEnv.resourcesPath = 'DBRResources';
dynamsoft.dbrEnv.ifCheck64bitServiceFirst = true;
dynamsoft.dbrEnv.onAutoConnectServiceSuccess = function(){
    console.log("success");
};
dynamsoft.dbrEnv.onAutoConnectServiceError = function(status){
    console.log("error");
};
```

Enumerations

`enum` dynamsoft.BarcodeReader.EnumBarcodeFormat

```
dynamsoft.BarcodeReader.EnumBarcodeFormat.All = 503317503;
dynamsoft.BarcodeReader.EnumBarcodeFormat.OneD = 0x3FF;
```

```
dynamsoft.BarcodeReader.EnumBarcodeFormat.CODE_39 = 0x1;
dynamsoft.BarcodeReader.EnumBarcodeFormat.CODE_128 = 0x2;
dynamsoft.BarcodeReader.EnumBarcodeFormat.CODE_93 = 0x4;
dynamsoft.BarcodeReader.EnumBarcodeFormat.CODABAR = 0x8;
dynamsoft.BarcodeReader.EnumBarcodeFormat.ITF = 0x10;
dynamsoft.BarcodeReader.EnumBarcodeFormat.EAN_13 = 0x20;
dynamsoft.BarcodeReader.EnumBarcodeFormat.EAN_8 = 0x40;
dynamsoft.BarcodeReader.EnumBarcodeFormat.UPC_A = 0x80;
dynamsoft.BarcodeReader.EnumBarcodeFormat.UPC_E = 0x100;
dynamsoft.BarcodeReader.EnumBarcodeFormat.INDUSTRIAL_25 = 0x200;
dynamsoft.BarcodeReader.EnumBarcodeFormat.PDF417 = 0x2000000;
dynamsoft.BarcodeReader.EnumBarcodeFormat.QR_CODE = 0x4000000;
dynamsoft.BarcodeReader.EnumBarcodeFormat.DATAMATRIX = 0x8000000;
dynamsoft.BarcodeReader.EnumBarcodeFormat.AZTEC = 0x10000000;
```

`enum` dynamsoft.BarcodeReader.EnumErrorCode

```
dynamsoft.BarcodeReader.EnumErrorCode.DBR_SYSTEM_EXCEPTION = 1;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_SUCCESS = 0;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_UNKNOWN = -10000;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_NO_MEMORY = -10001;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_NULL_REFERENCE = -10002;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_LICENSE_INVALID = -10003;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_LICENSE_EXPIRED = -10004;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_FILE_NOT_FOUND = -10005;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_FILETYPE_NOT_SUPPORTED = -10006;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_BPP_NOT_SUPPORTED = -10007;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_INDEX_INVALID = -10008;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_BARCODE_FORMAT_INVALID = -10009;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_CUSTOM_REGION_INVALID = -10010;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_MAX_BARCODE_NUMBER_INVALID = -10011;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_IMAGE_READ_FAILED = -10012;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_TIFF_READ_FAILED = -10013;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_QR_LICENSE_INVALID = -10016;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_1D_LICENSE_INVALID = -10017;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_DIB_BUFFER_INVALID = -10018;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_PDF417_LICENSE_INVALID = 10019;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_DATAMATRIX_LICENSE_INVALID = -10020;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_PDF_READ_FAILED = -10021;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_PDF_DLL_MISSING = -10022;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_PAGE_NUMBER_INVALID = -10023;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_CUSTOM_SIZE_INVALID = -10024;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_CUSTOM_MODULESIZE_INVALID = -10025;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_RECOGNITION_TIMEOUT = -10026;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_JSON_PARSE_FAILED = -10030;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_JSON_TYPE_INVALID = -10031;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_JSON_KEY_INVALID = -10032;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_JSON_VALUE_INVALID = -10033;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_JSON_NAME_KEY_MISSING = -10034;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_JSON_NAME_VALUE_DUPLICATED = -10035;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_TEMPLATE_NAME_INVALID = -10036;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_JSON_NAME_REFERENCE_INVALID = -10037;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_PARAMETER_VALUE_INVALID = 10038;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_DOMAIN_NOT_MATCHED = -10039;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_RESERVEDINFO_NOT_MATCHED = -10040;
dynamsoft.BarcodeReader.EnumErrorCode.DBR_DBRERR_AZTEC_LICENSE_INVALID = -10041;
```

OCR Pro

For Server Side OCR, check out Server-Side OCR

Client-Side

Methods

Download()	IsModuleInstalled()	Recognize()
RecognizeFile()	RecognizeRect()	RecognizeSelectedImages()

Properties

Settings

OCR Pro Result Object

OCRResult	PageResult
LetterResult	ErrorInfo

Code example

The following code example demonstrates how to use the APIs above to perform basic scanning.

```
function DoOCR() {
    if (DWObject) {
        if (DWObject.HowManyImagesInBuffer == 0) {
            alert("Please scan or load an image first.");
            return;
        }
        var settings = Dynamsoft.WebTwain.Addon.OCRPro.NewSettings();
        var bMultipage = false;
        settings.RecognitionModule = EnumDWT_OCRProRecognitionModule.OCRPM_AUTO;
        settings.Languages = "eng";
        settings.OutputFormat = EnumDWT_OCRProOutputFormat.OCRPFT_TXTS;
        settings.LicenseChecker = "LicenseChecker.aspx";
        DWObject.Addon.OCRPro.Settings = settings;
        //Get ocr result.
        var i, nCount = DWObject.HowManyImagesInBuffer;
        DWObject.SelectedImagesCount = nCount;
        for (i = 0; i < nCount; i++) {
            DWObject.SetSelectedImageIndex(i, i);
        }
        DWObject.Addon.OCRPro.RecognizeSelectedImages(function(result) {
                if (result == null)
                    return null;
                var bRet = "", pageCount = result.GetPageCount();
                if (pageCount == 0) {
                    alert("OCR result is Null.");
                    return;
                } else {
                    for (i = 0; i < pageCount; i++) {
                        var page = result.GetPageContent(i);
                        var letterCount = page.GetLettersCount();
```

```
                    for (var n = 0; n < letterCount; n++) {
                        var letter = page.GetLetterContent(n);
                        bRet += letter.GetText();
                    }
                }
                console.log(bRet);
            }
        }, function(errorcode, errorstring, result) {
            if (errorcode != -2600 && errorcode != -2601) //-2600:LicenseChecker cannot be empty.  -2601:Cannot c
onnect to the LicenseChecker, please check and make sure it's set correctly.
                alert(errorstring);
            var strErrorDetail = "";
            var aryErrorDetailList = result.GetErrorDetailList();
            for (var i = 0; i < aryErrorDetailList.length; i++) {
                if (i > 0)
                    strErrorDetail += ";";
                strErrorDetail += aryErrorDetailList[i].GetMessage();
            }
            if (strErrorDetail.length > 0 && errorstring != strErrorDetail)
                alert(strErrorDetail);
        }
    );
  }
}
```

Methods

Download()	
Downloads and installs the OCRPro add-on (it's typically a zipped dll file) on the local system.	
Syntax	.Addon.OCRPro.Download(sDLLPath, [optionalAsyncSuccessFunc, optionalAsyncFailureFunc]);
Parameters	• `string sDLLPath` : specifies the url path of the add-on. E.g. `"http://www.dynamsoft.com/DWT/Resources/OCR.zip"` . • `<OnSuccess function> optional optionalAsyncSuccessFunc` : callback function triggered when the download succeeds. • `<OnFailure function> optional optionalAsyncFailureFunc` : callback function triggered when the download fails.
Return value	`boolean`
Availability	ActiveX: ✗ | H5(Windows): ✓ | v11.3.2 | H5(macOS/TWAIN): ✗ | H5(macOS/ICA): ✗ | H5(Linux): ✗
Usage notes	This method will attempt to download and install the dll from the server if the dll doesn't exist already or the exising one is of a different version. The method `IsModuleInstalled()` can be use to determine whether it's necessary to call `Download()` .

IsModuleInstalled()	
Returns whether the OCRPro module is already installed on the local system.	
Syntax	.Addon.OCRPro.IsModuleInstalled();
Parameters	• None

Return value	`boolean`
Availability	Versions: `v14.1+`
Usage notes	If the method returns `false`, you can use the method `Download()` to download and install the dll from the server.

Recognize()

Performs OCR on a given image.

Syntax	.Addon.OCRPro.Recognize(nImageIndex, asyncSuccessFunc, asyncFailureFunc);
Parameters	• `number nImageIndex` : Specifies the index of the image in buffer. The index is 0-based. • `<OnOCRSuccess function> asyncSuccessFunc` : Callback function triggered when the OCR executed successfully. The arguments are `number nImageIndex` — The index of the image. `OCRResult Result` — The OCR result. • `<OnOCRFailure function> asyncFailureFunc` : Callback function triggered when the OCR operation fails. The arguments are `number nErrorCode` — The error code. `string strErrorString` — The error string. `OCRResult Result` — The OCR result for the image.
Return value	`None`
Availability	ActiveX: ✗ | H5(Windows): ✓ | v11.3.2 | H5(macOS/TWAIN): ✗ | H5(macOS/ICA): ✗ | H5(Linux): ✗
Usage notes	Check out the sample code and the details for `OCRResult`.

RecognizeFile()

Performs OCR on local files directly without loading them in the viewer.

Syntax	.Addon.OCRPro.RecognizeFile(strFileNames, asyncSuccessFunc, asyncFailureFunc);
Parameters	• `string strFileNames` : Specifies the local paths of the target files. If multiple files are given, they should be separated by the '|' character. • `<OnOCRSuccess function> asyncSuccessFunc` : Callback function triggered when the OCR executed successfully. The arguments are `string strFileNames` — The file paths. `OCRResult Result` — The OCR result for the image. • `<OnOCRFailure function> asyncFailureFunc` : Callback function triggered when the OCR operation fails. The arguments are `number nErrorCode` — The error code. `string strErrorString` — The error string.

	`OCRResult` `Result` — The OCR result for the image.
Return value	`boolean`
Availability	ActiveX: ✗ | H5(Windows): ✓ | v11.3.2 | H5(macOS/TWAIN): ✗ | H5(macOS/ICA): ✗ | H5(Linux): ✗
Usage notes	Check out the sample code and the details for `OCRResult`.

RecognizeRect()

Peforms OCR on the given rectangle on a specified image.

Syntax	.Addon.OCRPro.RecognizeRect(nImageIndex, aryZones, asyncSuccessFunc, asyncFailureFunc);
Parameters	• `number` `nImageIndex` : Specifies the index of the image in buffer. The index is 0-based. • `Array` `aryZones` : An array of OCRZone's created by the method `Dynamsoft.WebTwain.Addon.OCRPro.NewOCRZone` to specify the coordinates of the rectangle(s) for OCR. • `<OnOCRSuccess function>` `asyncSuccessFunc` : Callback function triggered when the OCR executed successfully. The arguments are `number` `nImageIndex` — The index of the image. `OCRResult` `Result` — The OCR result. • `<OnOCRFailure function>` `asyncFailureFunc` : Callback function triggered when the OCR operation fails. The arguments are `number` `nErrorCode` — The error code. `string` `strErrorString` — The error string. `OCRResult` `Result` — The OCR result for the image.
Return value	`boolean`
Availability	ActiveX: ✗ | H5(Windows): ✓ | v11.3.2 | H5(macOS/TWAIN): ✗ | H5(macOS/ICA): ✗ | H5(Linux): ✗
Usage notes	Check out the sample code and the details for `OCRResult`.

RecognizeSelectedImages()

Performs OCR on the currently selected images in buffer.

Syntax	.Addon.OCRPro.RecognizeSelectedImages(asyncSuccessFunc, asyncFailureFunc);
Return value	`boolean`

<table>
<tr><td>Parameters</td><td>

- `<OnOCRSelectedImagesSuccess function>` `asyncSuccessFunc` : Callback function triggered when the OCR executed successfully. The only argument is

`OCRResult` `Result`	The OCR result for the image.

- `<OnOCRFailure function>` `asyncFailureFunc` : Callback function triggered when the OCR operation fails. The arguments are

`number` `nErrorCode`	The error code.
`string` `strErrorString`	The error string.
`OCRResult` `Result`	The OCR result for the image.

</td></tr>
<tr><td>Availability</td><td>

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✗	✓ | v11.3.2	✗	✗	✗

</td></tr>
<tr><td>Usage notes</td><td>Check out the sample code and the details for <code>OCRResult</code>.</td></tr>
</table>

Properties

<table>
<tr><th colspan="2">Settings</th></tr>
<tr><td colspan="2">Returns or sets the parameters for OCR.</td></tr>
<tr><td>Type</td><td><code>Dynamsoft.WebTwain.Addon.OCRPro.NewSettings</code></td></tr>
<tr><td>Accessors</td><td><code>Get Set</code></td></tr>
<tr><td>Availability</td><td>

ActiveX	H5(Windows)	H5(macOS/TWAIN)	H5(macOS/ICA)	H5(Linux)
✗	✓ | v11.3.2	✗	✗	✗

</td></tr>
<tr><td>Usage notes</td><td>Check out Settings</td></tr>
</table>

New Settings

Name	Type	Description
`Languages`	`EnumDWT_OCRLanguage`	The language to OCR. E.g., "eng" or `EnumDWT_OCRLanguage.OCRL_ENG`
`LicenseChecker`	`string`	A string that specify the url for the license checker
`OutputFormat`	`EnumDWT_OCROutputFormat`	Specify the output format. E.g., `EnumDWT_OCROutputFormat.OCROF_PDFIMAGEOVERTEXT`
`PDFAVersion`	`EnumDWT_OCRProPDFAVersion`	Specify the PDFA version. E.g., `EnumDWT_OCRProPDFAVersion.OCRPPDFAV_1A`
`PDFVersion`	`EnumDWT_OCRProPDFVersion`	Specify the PDF version. E.g., `EnumDWT_OCRProPDFVersion.OCRPPDFV_0`

RecognitionModule	EnumDWT_OCRProRecognitionModule	Specify the recognition module. E.g., EnumDWT_OCRProRecognitionModule.OCRPM_AUTO
Redaction	Redaction	An object that specifies the redaction

Redaction

Name	Type	Description
FindText	string	The text to find. E.g., "twain".
FindTextFlags	EnumDWT_OCRFindTextFlags	Specify how the finding works.
FindTextAction	EnumDWT_OCRFindTextAction	Specify the action for redaction.

OCRResult

An object of the type `OCRResult` is returned which contains the OCR result. Methods of the object are

Name	Description
Get()	Returns a base64 encoded string that contains the result of the OCR operation.
GetErrorCode()	Returns the OCR error code.
GetErrorString()	Returns the OCR error string.
GetErrorDetailList()	Returns an array which contains detailed error information for each page that was OCR'd. Check out ErrorInfo.
GetInput()	Returns the input information of the OCR processing methods. The input could be the indices of the images in buffer or the local file paths.
Save()	Saves the OCR result as a file (.txt, .pdf, etc.) on the local disk.
GetOCRTotalCount()	Returns how many pages are allowed to be OCR'd by the current license. E.g. 300000.
GetAlreadyOCRCount()	Returns how many pages have been OCR'd.
GetPageCount()	Returns how many pages there are in the OCR result.
GetPageContent(nPageIndex)	Returns the content (`PageResult`) of the page specified by `nPageIndex` .

PageResult

An object of the type `PageResult` is returned by `GetPageContent(nPageIndex)` . Methods of the object are

Name	Description
GetLetterCount()	Returns how many letters are recognized on the specified page.
GetLetterContent(nLetterIndex)	Returns the content (`LetterResult`) of the specified letter.

LetterResult

An object of the type `LetterResult` is returned by `GetLetterContent(nLetterIndex)` . Methods of the object are

Name	Description

`GetText()`	Returns the text of the specified word in the OCR result.
`GetLetterRect()`	Returns the coordinates for the rectangle that contains a specified letter. The coordinates string is in the format of "left,top,right,bottom".

ErrorInfo

The following are the methods in each `ErrorInfo` object.

Name	Description
`GetInput()`	Returns the file path or the index of the Input.
`GetMessage()`	Returns the error message.
`GetPage()`	Returns the number of the page on which the error was thrown. If `GetInput()` returns a file path, then this returns the index of the page in that file. If `GetInput()` returns an index, then `GetPage()` is always "0".

Server-Side

The OCR Pro engine runs as a service. The process of server-side OCR is

1. The image(s) is uploaded to the server and saved
2. The path(s) of the saved image(s) is sent to the OCR pro service as part of the OCRPro.ServerSide.Request
3. The service returns the OCR result in a HTTP Response

The following demonstrates the structures of the Request and the Response

OCRPro.ServerSide.Request

```
{
    productKey: "***",
    inputFile: ["d:\\input\\1.tif"],
    outputFile: " d:\\temp\\ocrresult.pdf",
    zones: [[100, 100, 200, 300]],
    settings:
    {
        recognitionModule: "auto", /*optional*/
        languages: "eng,arabic",
        recognitionMethod: "File",
        threadCount: "2", /*optional*/
        outputFormat: "IOTPDF",
        pdfVersion: "1.7", /*optional*/
        pdfAVersion: "pdf/a-2a", /*optional*/
        redaction:
            {
            "findText": "AAA",
            "findTextFlags": 1,
            "findTextAction": 0
            }
    }
}
```

API	Description

<table>
<tr><td>productKey</td><td>The product key which is generated from an OCR license.</td></tr>
<tr><td>inputFile</td><td>Specifies the files to be OCR'd. This is an array of strings which are absolute paths of the files. The supported formats are BMP, JPG, TIF, PDF, PNG, JBIG2, JPEG2000, PCX, etc. Please note the use of '\\' instead of just '\'.</td></tr>
<tr><td>outputFile</td><td>Specifies where the output file is saved. If the input includes more than one file, all of them will be merged into one file. Otherwise, the result will only be returned in the OCRPro.ServerSide.Response.</td></tr>
<tr><td>zones</td><td>Specifies which zones are to be OCR'd on an image. There can be multiple zones but it works only when the <code>recognitionMethod</code> is <code>Page</code> . The coordinates are in the sequence of [[left, top, right, bottom]].</td></tr>
<tr><td>settings</td><td></td></tr>
<tr><td>.recognitionModule</td><td>Specifies which Module is to be used for this OCR. Allowed values are:

<table><tr><td><code>mostaccurate</code></td><td>Most accurate but time consuming</td></tr><tr><td><code>fastest</code></td><td>Takes the least time but not very accurate</td></tr><tr><td><code>balanced</code></td><td>Maintains a balance between accuracy and performance</td></tr><tr><td><code>auto</code></td><td>Automatically use one of the aboove 3 modules, this is the default value</td></tr></table></td></tr>
<tr><td>.languages</td><td>Specifies the language for this OCR. For example, English: "eng", Arabic :"arabic". You can also set multiple languages like this "eng,arabic". The supported languages are "ara / arabic; ces / czech; dan / danish; deu / german; ell / greek; eng / english; fra / french; fin / finnish; hun / hungar; ita / italian; nld / duntch; nor / norsk; por / port; pol / polish; rus / russian; swe / swedish; spa / spanish; tur / turkish";</td></tr>
<tr><td>.recognitionMethod</td><td>Specifies how the OCR is performed. There are two methods: <code>Page</code> is the default value and it means the OCR is performed on one page at a time, the other method is <code>File</code> which means the OCR is performed on one file at a time. The method <code>File</code> is faster and it supports multiple threads. But only the method <code>Page</code> supports zonal OCR and returning detailed results like the coordinates for each recognized letter.</td></tr>
<tr><td>.threadCount</td><td>Specifies the maximum number of threads to be used for this OCR. The default value is -1 which means all possible threads will be used. This setting is only valid when <code>recognitionMethod</code> is set to <code>File</code> .</td></tr>
<tr><td>.outputFormat</td><td>Specifies the file type for outputting the OCR result. Allowed values are

<table><tr><th>Format</th><th>Description</th></tr><tr><td>TXTS</td><td>Standard text file.</td></tr><tr><td>TXTCSV</td><td>CSV text file.</td></tr><tr><td>TXTF</td><td>Formatted Text file.</td></tr><tr><td>XML</td><td>Simple XML file.</td></tr><tr><td>IOTPDF</td><td>Image over text PDF file.</td></tr><tr><td>IOTPDF_MRC</td><td>Image over text PDF with MRC technology.</td></tr></table></td></tr>
<tr><td>.pdfVersion</td><td>Specifies the version of the PDF file if the <code>outputFormat</code> is set to either <code>IOTPDF</code> or <code>IOTPDF_MRC</code> . The version number allowed are 1.0 to 1.7 and by default it is 1.5.</td></tr>
<tr><td>.pdfAVersion</td><td>Specifies the version of the PDF file if the <code>outputFormat</code> is set to either <code>IOTPDF</code> or <code>IOTPDF_MRC</code> . The version number allowed are "pdf/a-1a","pdf/a-1b","pdf/a-2a ","</td></tr>
</table>

	pdf/a-2b "," pdf/a-2u ", " pdf/a-3a ","pdf/a-3b","pdf/a-3u".
`.redaction`	Specifies how the redaction is done. **Option** / **Description** findText: A string to specify what to find. findTextFlags: Specifies how the text is found. The allowed values are 1 (WHOLEWORD), 2 (MATCHCASE), 4 (FUZZYMATCH). findTextAction: Specifies the action once the text is found. The allowed values are 0 (HIGHLIGHT), 1 (STRIKEOUT) or 2 (MARKFORREDACT).

OCRPro.ServerSide.Response

```
{
    "Request": {
        inputFile: ["d:\\input\\1.tif"],
        settings: {...},
        outputFile: ...
    }
    ocrTotalCount: 300000,
    alreadyOCRCount: 80,
    code: 0,
    message: "Recognize succeeded.",
    errorList:
        [
            {
                "input": "d:\\input\\1.tif",
                "message": "Image file format error.",
                "page": "1"
            }
        ]
    resultFile: "***", //base64-encoded file content
    resultDetail:
        [
            [//page 0
                {//letter 0
                    "letter": "Aa",
                    "boundary": [0,0,18,18]
                },
                {//letter 1
                    ...
                },
                ...
            ],
            [//page 1
                ...
            ],
            ...
        ]
}
```

API	Description
`inputFile`	Check out OCRPro.ServerSide.Request for more info.
`settings`	Check out OCRPro.ServerSide.Request for more info.
`outputFile`	

`outputFile`	Check out OCRPro.ServerSide.Request for more info.
`ocrTotalCount`	Returns how many pages are allowed to be OCR'd by the current license.
`alreadyOCRCount`	Returns how many pages have already been OCR'd.
`code`	Returns the error code for the OCR. If it's not `0`, check `errorList` for more details.
`message`	Returns the overall error message.
`errorList`	Returns the detailed error messages for each of the OCR'd files.
`resultFile`	Returns the result file encoded as a base64 string. It only works when OCRPro.ServerSide.Request doesn't specify an output file path.
`resultDetail`	Returns detailed OCR result down to each found letter in JSON format. This is only valid when the `recognitionMethod` is set to `Page`.

Appendix

Editions

Dynamic Web TWAIN

Name	Description
HTML5 for Windows	This edition supports Chrome 27+, Firefox 27+, IE 10+ and Edge on Windows
HTML5 for Mac	This edition supports Chrome 27+, Firefox 27+ and Safari 7+ on macOS
HTML5 for Linux	This edition supports Chrome 27+, Firefox 27+ on Linux
HTML5	This refers to all three editions above
ActiveX for Windows	This edition supports IE 6 ~ 9 and can be configured to work in IE 10/11 as well on Windows

Mobile Browser Capture

Name	Description
Mobile Browser Capture	This edition supports Safari 11+ on `iOS` and Chrome 58+ on `Android`

Operating Systems

Windows

> Windows XP/7/8/2008/2012/2016 and 10; 32-bit and 64-bit

macOS

> OS X 10.6.8 and later

Linux

> Ubuntu 12.0.4+, Debian 8+, Fedora 24+, mint 18.3; 64-bit

Function Prototypes

> Description The following are all the function prototypes used in Dynamic Web TWAIN.

OnSuccess()

Parameters	Type	Description
`None`	`N/A`	`N/A`

OnFailure(errorCode, errorString)

Parameters	Type	Description
errorCode	`Number`	The error code
errorString	`String`	The error string

OnHttpUploadSuccess(sHttpResponse)

Parameters	Type	Description
sHttpResponse	`String`	The information returned from the server after the upload

OnHttpUploadFailure(errorCode, errorString, sHttpResponse)

Parameters	Type	Description
errorCode	`Number`	The error code
errorString	`String`	The error string
sHttpResponse	`String`	The detailed error information returned from the server

Enumerations

EnumDWT_PixelType	
TWPT_BW	0
TWPT_GRAY	1
TWPT_RGB	2
TWPT_PALLETE	3
TWPT_CMY	4
TWPT_CMYK	5
TWPT_YUV	6
TWPT_YUVK	7
TWPT_CIEXYZ	8
TWPT_LAB	9
TWPT_SRGB	10
TWPT_SCRGB	11
TWPT_INFRARED	16

EnumDWT_BorderStyle	
TWBS_NONE	0
TWBS_SINGLEFLAT	1
TWBS_SINGLE3D	2

EnumDWT_MessageType	
TWQC_GET	1
TWQC_SET	2
TWQC_GETDEFAULT	4
TWQC_GETCURRENT	8
TWQC_RESET	16

EnumDWT_Cap	
CAP_NONE	0
CAP_XFERCOUNT	1
ICAP_COMPRESSION	256
ICAP_PIXELTYPE	257
ICAP_UNITS	258

ICAP_XFERMECH	259
CAP_AUTHOR	4096
CAP_CAPTION	4097
CAP_FEEDERENABLED	4098
CAP_FEEDERLOADED	4099
CAP_TIMEDATE	4100
CAP_SUPPORTEDCAPS	4101
CAP_EXTENDEDCAPS	4102
CAP_AUTOFEED	4103
CAP_CLEARPAGE	4104
CAP_FEEDPAGE	4105
CAP_REWINDPAGE	4106
CAP_INDICATORS	4107
CAP_SUPPORTEDCAPSEXT	4108
CAP_PAPERDETECTABLE	4109
CAP_UICONTROLLABLE	4110
CAP_DEVICEONLINE	4111
CAP_AUTOSCAN	4112
CAP_THUMBNAILSENABLED	4113
CAP_DUPLEX	4114
CAP_DUPLEXENABLED	4115
CAP_ENABLEDSUIONLY	4116
CAP_CUSTOMDSDATA	4117
CAP_ENDORSER	4118
CAP_ALARMS	4120
CAP_ALARMVOLUME	4121
CAP_AUTOMATICCAPTURE	4122
CAP_TIMEBEFOREFIRSTCAPTURE	4123
CAP_TIMEBETWEENCAPTURES	4124
CAP_CLEARBUFFERS	4125
CAP_MAXBATCHBUFFERS	4126
CAP_DEVICETIMEDATE	4127
CAP_POWERSUPPLY	4128
CAP_CAMERAPREVIEWUI	4129

CAP_SERIALNUMBER	4132
CAP_PRINTER	4134
CAP_PRINTERENABLED	4135
CAP_PRINTERINDEX	4136
CAP_PRINTERMODE	4137
CAP_PRINTERSTRING	4138
CAP_PRINTERSUFFIX	4139
CAP_LANGUAGE	4140
CAP_FEEDERALIGNMENT	4141
CAP_FEEDERORDER	4142
CAP_REACQUIREALLOWED	4144
CAP_BATTERYMINUTES	4146
CAP_BATTERYPERCENTAGE	4147
CAP_CAMERASIDE	4148
CAP_SEGMENTED	4149
CAP_CAMERAENABLED	4150
CAP_CAMERAORDER	4151
CAP_MICRENABLED	4152
CAP_FEEDERPREP	4153
CAP_FEEDERPOCKET	4154
CAP_AUTOMATICSENSEMEDIUM	4155
CAP_CUSTOMINTERFACEGUID	4156
ICAP_AUTOBRIGHT	4352
ICAP_BRIGHTNESS	4353
ICAP_CONTRAST	4355
ICAP_CUSTHALFTONE	4356
ICAP_EXPOSURETIME	4357
ICAP_FILTER	4358
ICAP_FLASHUSED	4359
ICAP_GAMMA	4360
ICAP_HALFTONES	4361
ICAP_HIGHLIGHT	4362
ICAP_IMAGEFILEFORMAT	4364
ICAP_LAMPSTATE	4365

ICAP_LIGHTSOURCE	4366
ICAP_ORIENTATION	4368
ICAP_PHYSICALWIDTH	4369
ICAP_PHYSICALHEIGHT	4370
ICAP_SHADOW	4371
ICAP_FRAMES	4372
ICAP_XNATIVERESOLUTION	4374
ICAP_YNATIVERESOLUTION	4375
ICAP_XRESOLUTION	4376
ICAP_YRESOLUTION	4377
ICAP_MAXFRAMES	4378
ICAP_TILES	4379
ICAP_BITORDER	4380
ICAP_CCITTKFACTOR	4381
ICAP_LIGHTPATH	4382
ICAP_PIXELFLAVOR	4383
ICAP_PLANARCHUNKY	4384
ICAP_ROTATION	4385
ICAP_SUPPORTEDSIZES	4386
ICAP_THRESHOLD	4387
ICAP_XSCALING	4388
ICAP_YSCALING	4389
ICAP_BITORDERCODES	4390
ICAP_PIXELFLAVORCODES	4391
ICAP_JPEGPIXELTYPE	4392
ICAP_TIMEFILL	4394
ICAP_BITDEPTH	4395
ICAP_BITDEPTHREDUCTION	4396
ICAP_UNDEFINEDIMAGESIZE	4397
ICAP_EXTIMAGEINFO	4399
ICAP_MINIMUMHEIGHT	4400
ICAP_MINIMUMWIDTH	4401
ICAP_AUTODISCARDBLANKPAGES	4404
ICAP_FLIPROTATION	4406

ICAP_BARCODEDETECTIONENABLED	4407
ICAP_SUPPORTEDBARCODETYPES	4408
ICAP_BARCODEMAXSEARCHPRIORITIES	4409
ICAP_BARCODESEARCHPRIORITIES	4410
ICAP_BARCODESEARCHMODE	4411
ICAP_BARCODEMAXRETRIES	4412
ICAP_BARCODETIMEOUT	4413
ICAP_ZOOMFACTOR	4414
ICAP_PATCHCODEDETECTIONENABLED	4415
ICAP_SUPPORTEDPATCHCODETYPES	4416
ICAP_PATCHCODEMAXSEARCHPRIORITIES	4417
ICAP_PATCHCODESEARCHPRIORITIES	4418
ICAP_PATCHCODESEARCHMODE	4419
ICAP_PATCHCODEMAXRETRIES	4420
ICAP_PATCHCODETIMEOUT	4421
ICAP_FLASHUSED2	4422
ICAP_IMAGEFILTER	4423
ICAP_NOISEFILTER	4424
ICAP_OVERSCAN	4425
ICAP_AUTOMATICBORDERDETECTION	4432
ICAP_AUTOMATICDESKEW	4433
ICAP_AUTOMATICROTATE	4434
ICAP_JPEGQUALITY	4435
ICAP_FEEDERTYPE	4436
ICAP_ICCPROFILE	4437
ICAP_AUTOSIZE	4438
ICAP_AUTOMATICCROPUSESFRAME	4439
ICAP_AUTOMATICLENGTHDETECTION	4440
ICAP_AUTOMATICCOLORENABLED	4441
ICAP_AUTOMATICCOLORNONCOLORPIXELTYPE	4442
ICAP_COLORMANAGEMENTENABLED	4443
ICAP_IMAGEMERGE	4444
ICAP_IMAGEMERGEHEIGHTTHRESHOLD	4445
ICAP_SUPPORTEDEXTIMAGEINFO	4446

EnumDWT_CapType	
TWON_NONE	0
TWON_ARRAY	3
TWON_ENUMERATION	4
TWON_ONEVALUE	5
TWON_RANGE	6

EnumDWT_TransferMode	
TWSX_NATIVE	0
TWSX_FILE	1
TWSX_MEMORY	2

EnumDWT_FileFormat	
TWFF_TIFF	0
TWFF_PICT	1
TWFF_BMP	2
TWFF_XBM	3
TWFF_JFIF	4
TWFF_FPX	5
TWFF_TIFFMULTI	6
TWFF_PNG	7
TWFF_SPIFF	8
TWFF_EXIF	9
TWFF_PDF	10
TWFF_JP2	11
TWFF_JPN	12
TWFF_JPX	13
TWFF_DEJAVU	14
TWFF_PDFA	15
TWFF_PDFA2	16

EnumDWT_TIFFCompressionType	
TIFF_AUTO	0
TIFF_NONE	1
TIFF_RLE	2
TIFF_FAX3	3

TIFF_T4	3
TIFF_FAX4	4
TIFF_T6	4
TIFF_LZW	5
TIFF_JPEG	7
TIFF_PACKBITS	32773

EnumDWT_InterpolationMethod	
IM_NEARESTNEIGHBOUR	1
IM_BILINEAR	2
IM_BICUBIC	3
IM_BESTQUALITY	5

EnumDWT_ImageType	
IT_BMP	0
IT_JPG	1
IT_TIF	2
IT_PNG	3
IT_PDF	4
IT_ALL	5
IT_GIF	6

EnumDWT_PDFCompressionType	
PDF_AUTO	0
PDF_FAX3	1
PDF_FAX4	2
PDF_LZW	3
PDF_RLE	4
PDF_JPEG	5

EnumDWT_ShowMode	
SW_ACTIVE	0
SW_MAX	1
SW_MIN	2
SW_CLOSE	3
SW_IFLIVE	4

EnumDWT_CapValueType	
TWTY_INT8	0
TWTY_INT16	1
TWTY_INT32	2
TWTY_UINT8	3
TWTY_UINT16	4
TWTY_int	5
TWTY_BOOL	6
TWTY_FIX32	7
TWTY_FRAME	8
TWTY_STR32	9
TWTY_STR64	10
TWTY_STR128	11
TWTY_STR255	12

EnumDWT_UnitType	
TWUN_INCHES	0
TWUN_CENTIMETERS	1
TWUN_PICAS	2
TWUN_POINTS	3
TWUN_TWIPS	4
TWUN_PIXELS	5
TWUN_MILLIMETERS	6

EnumDWT_DUPLEX	
TWDX_NONE	0
TWDX_1PASSDUPLEX	1
TWDX_2PASSDUPLEX	2

EnumDWT_CapLanguage	
TWLG_DAN	0
TWLG_DUT	1
TWLG_ENG	2
TWLG_FCF	3
TWLG_FIN	4
TWLG_FRN	5

TWLG_GER	6
TWLG_ICE	7
TWLG_ITN	8
TWLG_NOR	9
TWLG_POR	10
TWLG_SPA	11
TWLG_SWE	12
TWLG_USA	13
TWLG_USERLOCALE	-1
TWLG_AFRIKAANS	14
TWLG_ALBANIA	15
TWLG_ARABIC	16
TWLG_ARABIC_ALGERIA	17
TWLG_ARABIC_BAHRAIN	18
TWLG_ARABIC_EGYPT	19
TWLG_ARABIC_IRAQ	20
TWLG_ARABIC_JORDAN	21
TWLG_ARABIC_KUWAIT	22
TWLG_ARABIC_LEBANON	23
TWLG_ARABIC_LIBYA	24
TWLG_ARABIC_MOROCCO	25
TWLG_ARABIC_OMAN	26
TWLG_ARABIC_QATAR	27
TWLG_ARABIC_SAUDIARABIA	28
TWLG_ARABIC_SYRIA	29
TWLG_ARABIC_TUNISIA	30
TWLG_ARABIC_UAE	31
TWLG_ARABIC_YEMEN	32
TWLG_BASQUE	33
TWLG_BYELORUSSIAN	34
TWLG_BULGARIAN	35
TWLG_CATALAN	36
TWLG_CHINESE	37
TWLG_CHINESE_HONGKONG	38

TWLG_CHINESE_PRC	39
TWLG_CHINESE_SINGAPORE	40
TWLG_CHINESE_SIMPLIFIED	41
TWLG_CHINESE_TAIWAN	42
TWLG_CHINESE_TRADITIONAL	43
TWLG_CROATIA	44
TWLG_CZECH	45
TWLG_DANISH	0
TWLG_DUTCH	1
TWLG_DUTCH_BELGIAN	46
TWLG_ENGLISH	2
TWLG_ENGLISH_AUSTRALIAN	47
TWLG_ENGLISH_CANADIAN	48
TWLG_ENGLISH_IRELAND	49
TWLG_ENGLISH_NEWZEALAND	50
TWLG_ENGLISH_SOUTHAFRICA	51
TWLG_ENGLISH_UK	52
TWLG_ENGLISH_USA	13
TWLG_ESTONIAN	53
TWLG_FAEROESE	54
TWLG_FARSI	55
TWLG_FINNISH	4
TWLG_FRENCH	5
TWLG_FRENCH_BELGIAN	56
TWLG_FRENCH_CANADIAN	3
TWLG_FRENCH_LUXEMBOURG	57
TWLG_FRENCH_SWISS	58
TWLG_GERMAN	6
TWLG_GERMAN_AUSTRIAN	59
TWLG_GERMAN_LUXEMBOURG	60
TWLG_GERMAN_LIECHTENSTEIN	61
TWLG_GERMAN_SWISS	62
TWLG_GREEK	63
TWLG_HEBREW	64

TWLG_HUNGARIAN	65
TWLG_ICELANDIC	7
TWLG_INDONESIAN	66
TWLG_ITALIAN	8
TWLG_ITALIAN_SWISS	67
TWLG_JAPANESE	68
TWLG_KOREAN	69
TWLG_KOREAN_JOHAB	70
TWLG_LATVIAN	71
TWLG_LITHUANIAN	72
TWLG_NORWEGIAN	9
TWLG_NORWEGIAN_BOKMAL	73
TWLG_NORWEGIAN_NYNORSK	74
TWLG_POLISH	75
TWLG_PORTUGUESE	10
TWLG_PORTUGUESE_BRAZIL	76
TWLG_ROMANIAN	77
TWLG_RUSSIAN	78
TWLG_SERBIAN_LATIN	79
TWLG_SLOVAK	80
TWLG_SLOVENIAN	81
TWLG_SPANISH	11
TWLG_SPANISH_MEXICAN	82
TWLG_SPANISH_MODERN	83
TWLG_SWEDISH	12
TWLG_THAI	84
TWLG_TURKISH	85
TWLG_UKRANIAN	86
TWLG_ASSAMESE	87
TWLG_BENGALI	88
TWLG_BIHARI	89
TWLG_BODO	90
TWLG_DOGRI	91
TWLG_GUJARATI	92

TWLG_HARYANVI	93
TWLG_HINDI	94
TWLG_KANNADA	95
TWLG_KASHMIRI	96
TWLG_MALAYALAM	97
TWLG_MARATHI	98
TWLG_MARWARI	99
TWLG_MEGHALAYAN	100
TWLG_MIZO	101
TWLG_NAGA	102
TWLG_ORISSI	103
TWLG_PUNJABI	104
TWLG_PUSHTU	105
TWLG_SERBIAN_CYRILLIC	106
TWLG_SIKKIMI	107
TWLG_SWEDISH_FINLAND	108
TWLG_TAMIL	109
TWLG_TELUGU	110
TWLG_TRIPURI	111
TWLG_URDU	112
TWLG_VIETNAMESE	113

EnumDWT_CapSupportedSizes	
TWSS_NONE	0
TWSS_A4LETTER	1
TWSS_B5LETTER	2
TWSS_USLETTER	3
TWSS_USLEGAL	4
TWSS_A5	5
TWSS_B4	6
TWSS_B6	7
TWSS_USLEDGER	9
TWSS_USEXECUTIVE	10
TWSS_A3	11
TWSS_B3	12

TWSS_A6	13
TWSS_C4	14
TWSS_C5	15
TWSS_C6	16
TWSS_4A0	17
TWSS_2A0	18
TWSS_A0	19
TWSS_A1	20
TWSS_A2	21
TWSS_A4	1
TWSS_A7	22
TWSS_A8	23
TWSS_A9	24
TWSS_A10	25
TWSS_ISOB0	26
TWSS_ISOB1	27
TWSS_ISOB2	28
TWSS_ISOB3	12
TWSS_ISOB4	6
TWSS_ISOB5	29
TWSS_ISOB6	7
TWSS_ISOB7	30
TWSS_ISOB8	31
TWSS_ISOB9	32
TWSS_ISOB10	33
TWSS_JISB0	34
TWSS_JISB1	35
TWSS_JISB2	36
TWSS_JISB3	37
TWSS_JISB4	38
TWSS_JISB5	2
TWSS_JISB6	39
TWSS_JISB7	40
TWSS_JISB8	41

TWSS_JISB9	42
TWSS_JISB10	43
TWSS_C0	44
TWSS_C1	45
TWSS_C2	46
TWSS_C3	47
TWSS_C7	48
TWSS_C8	49
TWSS_C9	50
TWSS_C10	51
TWSS_USSTATEMENT	52
TWSS_BUSINESSCARD	53
TWSS_MAXSIZE	54

EnumDWT_CapFeederAlignment	
TWFA_NONE	0
TWFA_LEFT	1
TWFA_CENTER	2
TWFA_RIGHT	3

EnumDWT_CapFeederOrder	
TWFO_FIRSTPAGEFIRST	0
TWFO_LASTPAGEFIRST	1

EnumDWT_CapPrinter	
TWPR_IMPRINTERTOPBEFORE	0
TWPR_IMPRINTERTOPAFTER	1
TWPR_IMPRINTERBOTTOMBEFORE	2
TWPR_IMPRINTERBOTTOMAFTER	3
TWPR_ENDORSERTOPBEFORE	4
TWPR_ENDORSERTOPAFTER	5
TWPR_ENDORSERBOTTOMBEFORE	6
TWPR_ENDORSERBOTTOMAFTER	7

EnumDWT_CapPrinterMode	
TWPM_SINGLESTRING	0
TWPM_MULTISTRING	1

TWPM_COMPOUNDSTRING	2

EnumDWT_CapBitdepthReduction	
TWBR_THRESHOLD	0
TWBR_HALFTONE	1
TWBR_CUSTHALFTONE	2
TWBR_DIFFUSION	3

EnumDWT_CapBitOrder	
TWBO_LSBFIRST	0
TWBO_MSBFIRST	1

EnumDWT_CapFilterType	
TWFT_RED	0
TWFT_GREEN	1
TWFT_BLUE	2
TWFT_NONE	3
TWFT_WHITE	4
TWFT_CYAN	5
TWFT_MAGENTA	6
TWFT_YELLOW	7
TWFT_BLACK	8

EnumDWT_CapFlash	
TWFL_NONE	0
TWFL_OFF	1
TWFL_ON	2
TWFL_AUTO	3
TWFL_REDEYE	4

EnumDWT_CapFlipRotation	
TWFR_BOOK	0
TWFR_FANFOLD	1

EnumDWT_CapImageFilter	
TWIF_NONE	0
TWIF_AUTO	1
TWIF_LOWPASS	2

TWIF_BANDPASS	3
TWIF_HIGHPASS	4
TWIF_TEXT	3
TWIF_FINELINE	4

EnumDWT_CapLightPath	
TWLP_REFLECTIVE	0
TWLP_TRANSMISSIVE	1

EnumDWT_CapLightSource	
TWLS_RED	0
TWLS_GREEN	1
TWLS_BLUE	2
TWLS_NONE	3
TWLS_WHITE	4
TWLS_UV	5
TWLS_IR	6

EnumDWT_MagType	
TWMD_MICR	0
TWMD_RAW	1
TWMD_INVALID	2

EnumDWT_CapNoiseFilter	
TWNF_NONE	0
TWNF_AUTO	1
TWNF_LONEPIXEL	2
TWNF_MAJORITYRULE	3

EnumDWT_CapORientation	
TWOR_ROT0	0
TWOR_ROT90	1
TWOR_ROT180	2
TWOR_ROT270	3
TWOR_PORTRAIT	0
TWOR_LANDSCAPE	3
TWOR_AUTO	4

TWOR_AUTOTEXT	5
TWOR_AUTOPICTURE	6

EnumDWT_CapOverscan	
TWOV_NONE	0
TWOV_AUTO	1
TWOV_TOPBOTTOM	2
TWOV_LEFTRIGHT	3
TWOV_ALL	4

EnumDWT_CapPixelFlavor	
TWPF_CHOCOLATE	0
TWPF_VANILLA	1

EnumDWT_CapPlanarChunky	
TWPC_CHUNKY	0
TWPC_PLANAR	1

EnumDWT_DataSourceStatus	
TWDSS_CLOSED	0
TWDSS_OPENED	1
TWDSS_ENABLED	2
TWDSS_ACQUIRING	3

EnumDWT_FitWindowType	
enumFitWindow	0
enumFitWindowHeight	1
enumFitWindowWidth	2

EnumDWT_PlatformType	
enumWindow	0
enumMac	1
enumLinux	2

EnumDWT_UploadDataFormat	
Binary	0
Base64	1

EnumDWT_MouseShape	

Default	0
Hand	1
Crosshair	2
Zoom	3
NWResize	4
EResize	5
NResize	6
Resize	7
Move	8

EnumDWT_Language	
English	0
French	1
Arabic	2
Spanish	3
Portuguese	4
German	5
Italian	6
Russian	7
Chinese	8

EnumDWT_InitMsg	
Info	1
Error	2
NotInstalledError	3
DownloadError	4
DownloadNotRestartError	5

EnumDWT_Driver	
TWAIN	0
ICA	3
SANE	3
TWAIN_AND_ICA	4

EnumDWT_OCRDownloadType	
OCRDT_Dll	0
OCRDT_LANGUAGE	1

EnumDWT_OCRLanguage		
Code	**Language**	**Value**
OCRL_ARA	Arabic	ara
OCRL_BEN	Bengali	ben
OCRL_CHI_SIM	Chinese_Simplified	chi_sim
OCRL_CHI_TRA	Chinese_Traditional	chi_tra
OCRL_DEU	German	deu
OCRL_ENG	English	eng
OCRL_FAS	Persian	fas
OCRL_FRA	French	fra
OCRL_HIN	Hindi	hin
OCRL_IND	Indonesian	ind
OCRL_ITA	Italian	ita
OCRL_JAV	Javanese	jav
OCRL_JPN	Japanese	jpn
OCRL_KOR	Korean	kor
OCRL_MAR	Marathi	mar
OCRL_MSA	Malay	msa
OCRL_PAN	Panjabi	pan
OCRL_POR	Portuguese	por
OCRL_RUS	Russian	rus
OCRL_SPA	Spanish	spa
OCRL_SWA	Swahili	swa
OCRL_TAM	Tamil	tam
OCRL_TEL	Telugu	tel
OCRL_THA	Thai	tha
OCRL_TUR	Turkish	tur
OCRL_URD	Urdu	urd
OCRL_VIE	Vietnamese	vie

EnumDWT_OCROutputFormat		
OCROF_TEXT	0	Outputs in a plain text format with a .txt extension if saved as a file.
OCROF_PDFPLAINTEXT	1	Outputs the OCR text results to a PDF. Any images from the original scanned image are lost.
		Outputs the OCR text results to a PDF, with the

		original scanned image printed overtop.
OCROF_PDFPLAINTEXT_PDFX	3	Outputs the OCR text results to a PDF/A. Any images from the original scanned image are lost.
OCROF_PDFIMAGEOVERTEXT_PDFX	4	Outputs the OCR text results to a PDF/A, with the original scanned image printed overtop.

EnumDWT_OCRPageSetMode		
OCRPSM_OSD_ONLY	0	Script detection only(OSD).
PSM_AUTO_OSD	1	Automatic page segmentation with orientation and script detection. (OSD)
PSM_AUTO_ONLY	2	Automatic page segmentation, but no OSD, or OCR.
PSM_AUTO	3	Fully automatic page segmentation, but no OSD. (Default)
PSM_SINGLE_COLUMN	4	Assume a single column of text of variable sizes.
PSM_SINGLE_COLUMN	5	Assume a single uniform block of vertically aligned text.
PSM_SINGLE_BLOCK	6	Assume a single uniform block of text.
PSM_SINGLE_LINE	7	Treat the image as a single text line.
PSM_SINGLE_WORD	8	Treat the image as a single word.
PSM_CIRCLE_WORD	9	Treat the image as a single word in a circle.
PSM_SINGLE_CHAR	10	Treat the image as a single character.

EnumDWT_OCRProRecognitionModule	
OCRPM_AUTO	AUTO
OCRPM_MOSTACCURATE	MOSTACCURATE
OCRPM_BALANCED	BALANCED
OCRPM_FASTEST	FASTEST

EnumDWT_OCRProOutputFormat	
OCRPFT_TXTS	TXTS
OCRPFT_TXTCSV	TXTCSV
OCRPFT_TXTF	TXTF
OCRPFT_XML	XML
OCRPFT_IOTPDF	IOTPDF
OCRPFT_IOTPDF_MRC	IOTPDF_MRC

EnumDWT_OCRProPDFVersion	
OCRPPDFV_0	1.0
OCRPPDFV_1	1.1
OCRPPDFV_2	1.2

OCRPPDFV_3	1.3
OCRPPDFV_4	1.4
OCRPPDFV_5	1.5
OCRPPDFV_6	1.6
OCRPPDFV_7	1.7

EnumDWT_OCRProPDFAVersion	
OCRPPDFAV_1A	pdf/a-1a
OCRPPDFAV_1B	pdf/a-1b
OCRPPDFAV_2A	pdf/a-2a
OCRPPDFAV_2B	pdf/a-2b
OCRPPDFAV_2U	pdf/a-2u
OCRPPDFAV_3A	pdf/a-3a
OCRPPDFAV_3B	pdf/a-3b
OCRPPDFAV_3U	pdf/a-3u

EnumDWT_OCRProType	
OCRDT_File	0
OCRDT_Index	1

EnumDWT_OCRFindTextFlags	
OCRFT_WHOLEWORD	1
OCRFT_MATCHCASE	2
OCRFT_FUZZYMATCH	4

EnumDWT_OCRFindTextAction	
OCRFT_HIGHLIGHT	0
OCRFT_STRIKEOUT	1
OCRFT_MARKFORREDACT	2

Error List

Error Code	Error String
0	Successful
-1001	General failure
-1002	Not enough memory to perform operation
-1003	Source Manager unable to find the specified Source

-1004	Source is connected to maximum supported number of applications
-1005	Source or Source Manager reported an error to the user and handled the error
-1006	Capability not supported by Source or operation is not supported on capability, or capability had dependencies on other capabilities and cannot be operated upon at this time
-1009	Unrecognized operation triplet
-1010	Data parameter out of supported range
-1011	Operation out of expected sequence
-1012	Unknown destination in DSM_Entry
-1013	Capability not supported by source
-1014	Operation not supported by capability
-1015	Capability has dependency on other capability and cannot be operated upon at this time
-1016	File System operation is denied (file is protected)
-1017	Operation failed because file already exists
-1018	File not found
-1019	Operation failed because directory is not empty
-1020	The feeder is jammed
-1021	The feeder detected multiple pages
-1022	Error writing file
-1023	The device went offline prior to or during this operation
-1030	Can not open Source Manager "TWain_32.dll" is missing or is in use by another application
-1031	Sequence error. The operation can not be performed upon the current Source Manager or Source state
-1032	User cancelled the operation
-1033	Invalid index
-1034	Invalid value
-1035	There is no image
-1036	Error reading file
-1070	BMP file or format error
-1071	JPEG file or format error
-1073	Only 24-bit true color and 8-bit gray-scaled images are supported for JPEG compression
-1080	General TIFF error
-1081	TIFF format error or not supported
-1090	BMP format error or not supported
-1100	PNG format error or not supported
-1110	Unrecognized file extension

-1200	PDF format error or not supported
-2000	Can not initiate the internet session
-2001	HTTP request error
-2002	HTTP server error
-2003	HTTP process error
-2004	FTP download file is too large
-2007	The system is busy, some operations are not completed. Please try later
-2207	The dynamsoft service installed on your computer is outdated and no longer works with the JavaScript code on the website
-2208	The connection with the local dynamsoft service encountered a problem and has been reset
-2209	The HTML5 (Chrome&Firefox) edition does not support this method or property
-2300	Http upload error: the HTTP Server cannot empty
-2301	Network error
-2302	The result format is invalid
-2303	Upload cancelled
-2304	Http download error: the url is invalid
-2305	User cancelled the operation
-2306	Upload Error: the upload file cannot be empty
-2307	The width or height you entered is invalid
-2308	The local dynamsoft service has been stopped
-2309	The LocalFile is emtpy in the Function
-2310	The index is out of range
-2311	The RemoteFile is emtpy in Barcode Download Function
-2312	The file length is emtpy
-2313	The size of the images you are about to upload has exceeded the allowed size
-2314	The parameter cannot be empty
-2315	The index is out of range
-2316	The RemoteFile is emtpy in Webcam Download Function
-2317	The RemoteFile is emtpy in Pdf Download Function
-2318	Invalid destination file
-2319	Invalid source file
-2320	Invalid file
-2321	The index is out of range
-2322	The left or top or right or bottom you entered is invalid
-2323	The OCR output format is not supported

-2324	The OCR page set mode is not supported
-2325	The current product key is empty or invalid, please contact the site administrator
-2326	The current product key has expired, please contact the site administrator
-2327	The current product key does not support Chrome, please contact the site administrator
-2328	The current product key does not support Firefox, please contact the site administrator
-2329	The current product key does not support IE, please contact the site administrator
-2330	The current product key does not support Edge, please contact the site administrator
-2331	The current product key is a trial version key but your local dynamsoft service is in full version, please uninstall your local version first and access this page again to install the correct version. If the issue persists, please contact the site administrator
-2332	The current product key is a full version key but your local dynamsoft service is in trial version, please uninstall your local version first and access this page again to install the correct version. If the issue persists, please contact the site administrator
-2333	The current product key is missing the core license, please contact the site administrator
-2334	The current product key does not include a license for reading 1D barcode, please contact the site administrator
-2335	The current product key does not include a license for reading QRcode barcode, please contact the site administrator
-2336	The current product key does not include a license for reading PDF417 barcode, please contact the site administrator
-2337	The current product key does not include a license for reading DataMatrix barcode, please contact the site administrator
-2338	The current product key does not support Webcam, please contact the site administrator
-2339	The current product key does not support pdf rasterizer, please contact the site administrator
-2340	The current product key does not support OCR, please contact the site administrator
-2341	The current product key does not support OCR pro, please contact the site administrator
-2342	The domain of your current site does not match the domain bound in the current product key, please contact the site administrator
-2343	The current product key does not support your browser, please contact the site administrator
-2344	The current product key does not support Windows OS, please contact the site administrator
-2345	The current product key does not support MAC OS, please contact the site administrator
-2346	The current product key does not support Linux OS, please contact the site administrator
-2347	The current product key does not support your OS, please contact the site administrator
-2348	The current product key is invalid because it's generated with the licenses of a different major version
-2349	The current product key does not include a license for reading barcode, please contact the site administrator
-2350	The indices cannot be empty
-2351	You cannot upload more than one image when the format is BMP, JPG or PNG
-2352	The indices are out of range

-2353	The header name being used is a protected keyword and is not allowed
-2354	The header name cannot be empty
-2355	The header name cannot be null
-2356	The header name cannot be undefined
-2357	The header name you entered is invalid
-2358	The type of the parameter indices must be an Array
-2359	The index is out of range
-2360	The index is null or undefined
-2361	You cannot convert more than one image to base64 string when the format is BMP, JPG or PNG
-2362	Convert to base64 failed
-2367	Invalid value for the parameter segmentUploadThreshold
-2368	Invalid value for the parameter moduleSize
-2369	The module for Dynamic Web TWAIN has failed to download
-2370	The current product key is invalid, please contact the site administrator
-2372	You cannot convert to binary more than one image when the format is BMP, JPG or PNG
<= -3000	See ErrorString property for details

Install & Upgrade

Upgrade From Trial or an Old Version

To upgrade Dynamic Web TWAIN from trial or an old version to the latest full version, please follow the steps below.

For the developer

Basic Steps

- Step 1 Update `Resources` on the Development machine
  - a) Uninstall the trial/old version

    > **Windows**: Search Dynamic Web TWAIN in `Control Panel -> Programs and Features`, and remove all the relevant components there.
    >
    > **macOS**: Execute `Applications > Dynamsoft > Dynamic Web TWAIN SDK {Version Number} > Uninstall.pkg`

  - b) Install the latest full version

    The download link of the full version can be found in the purchasing email that was sent to the registered email/purchaser's email. If you purchased the SDK but lost the download link of the full version, please request the download again.

  - c) Replace the whole `Resources` folder of Dynamic Web TWAIN in your application with the `Resources` folder of the full version. Typically, you can find it at the following path

    > **Windows**: `C:\Program Files (x86)\Dynamsoft\Dynamic Web TWAIN SDK {Version Number}`
    >
    > **macOS**: `Applications > Dynamsoft > Dynamic Web TWAIN SDK {Version Number}`

- Step 2 Update the License
  - a) Replace the product key In the file `dynamsoft.webtwain.config.js`, search for `Dynamsoft.WebTwainEnv.ProductKey`. Input the ProductKeys you received.

    ```
    Dynamsoft.WebTwainEnv.Containers = [{ContainerId:'dwtcontrolContainer', Width:270, Height:350}];
    // Please note that the license key is not the same thing as the Product Key.
    // You need to use your license keys to generate a Product Key.
    // For more info about how to generate a product key, please check the reference below.
    Dynamsoft.WebTwainEnv.ProductKey = '88FFAA09C42D5DE******;t0076lQAAAGNcO61He******';
    ```

  - b) Set `Dynamsoft.WebTwainEnv.Trial` to `false` if it's `true`

    In the file `dynamsoft.webtwain.config.js`, search for `Dynamsoft.WebTwainEnv.Trial`, and set it to `false` manually.

- Step 3 Deploy the full version to the server

  After you have finished the testing on your dev machines, you can update your application on the server accordingly.

For silent upgrade

For version 14.3 and above, if you'd like the upgrade to be silent for the end users. The following extra steps are needed before you deploy your application to the server

- a) Download the following files and put them in the correct directory (create the directory if it doesn't exist)

> For Windows users, put the following files under `/Resources/dist/win/`
> https://tst.dynamsoft.com/libs/dwt/14.3/dist/win/WinDWT_14.2.0.1025.zip
> https://tst.dynamsoft.com/libs/dwt/14.3/dist/win/Pdf.zip
>
> For macOS users, put the following files under `/Resources/dist/mac/`
> https://tst.dynamsoft.com/libs/dwt/14.3/dist/mac/MacDWT_14.1.0.0828.zip
> https://tst.dynamsoft.com/libs/dwt/14.3/dist/mac/MacPdf.zip
>
> For Linux users, put the following files under `/Resources/dist/linux/`
> https://tst.dynamsoft.com/libs/dwt/14.3/dist/linux/LinuxDWT_14.1.0.0828.zip
> https://tst.dynamsoft.com/libs/dwt/14.3/dist/linux/LinuxPdf.zip

- b) Add the configuration `Dynamsoft.WebTwainEnv.IfInstallDWTModuleWithZIP = true` in the file `dynamsoft.webtwain.config.js`

For the end users

Download and install the new version when you see the prompt to do so. This only needs to be done once.

For silent upgrade, the end users don't need to do anything.

> **NOTE** If you are upgrading from a very old version (at least 2 major versions apart like from v12 to v14), more steps may be needed, feel free to contact Dynamsoft Support Team for more information.

Install on the Client Machines

Dynamic Web TWAIN is a 100% client-side SDK, every client machine needs to install its components in order to use the SDK. Over the years, Dynamsoft has made many efforts to make the process as smooth as possible. Here we'll talk about how the installation is done in version 14.3 (latest as of November, 2018).

HTML5 editions

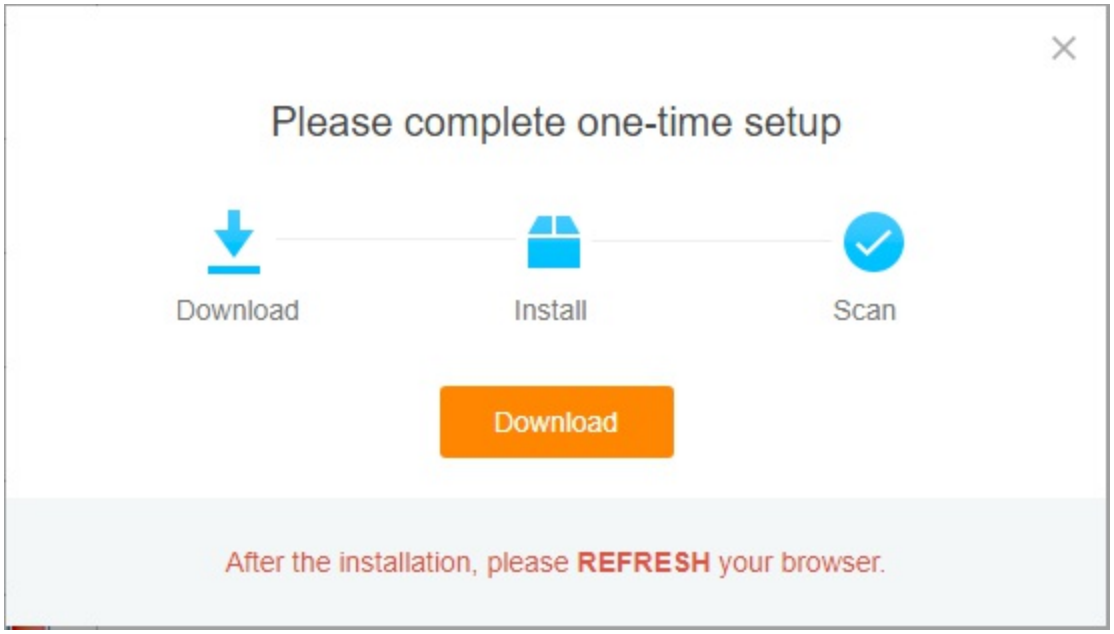
- Install Dynamsoft Service

  When the client first visit the web page which has Dynamic Web TWAIN implemented, the automatic initialization of the SDK which is built in its JavaScript library will try to establish connection with the Dynamsoft Service which is expected to be installed locally.

  > NOTE: the initialization happens after the DOMContentLoaded event has been fired.

  The connection will be attempted **twice for each** of three pre-defined ports. Depending on whether the protocol is HTTP or HTTPS, the ports are 18625, 18993, 18627 or 18626, 18994, 18628.

  Should the connection fails, it means the service is not installed and the following prompt will come up and ask the end user to download and install the service.

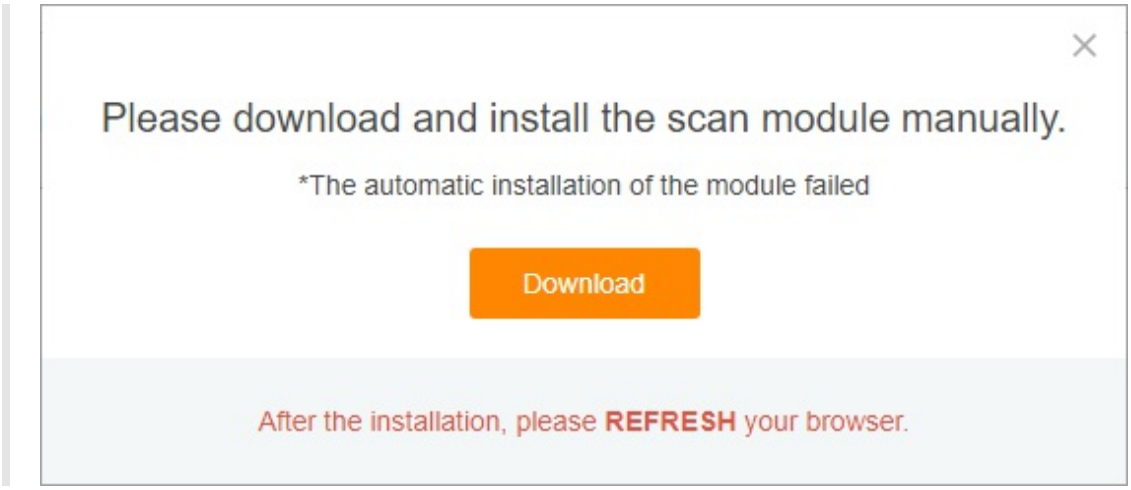


> NOTE: the same prompt will appear no matter whether the client OS is Windows, macOS or Linux. But the file you download differs on different Systems. On Windows and macOS, the users can double click the downloaded installer to install the SDK. On Linux, however, the users will need to run either one of the following command to install it
>
> Debian / Ubuntu
>
> ```
> dpkg -i DynamsoftServiceSetup.deb
> ```
>
> Fedora
>
> ```
> rpm -ivh DynamsoftServiceSetup.rpm
> ```

- Install Dynamic Web TWAIN

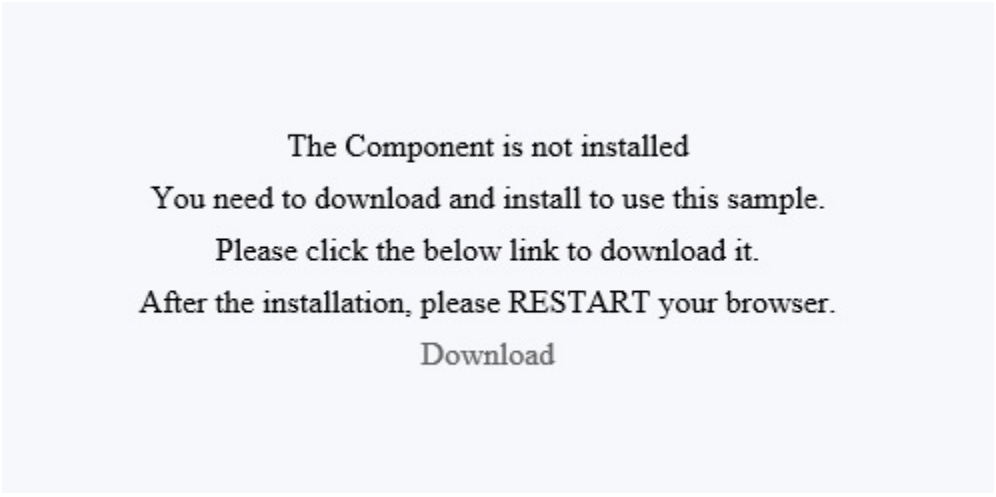
  In version 14.2 and above, Dynamic Web TWAIN and its PDF Rasterizer add-on are installed together with the Dynamsoft Service. But in versions 13.0 ~ 14.1, Dynamic Web TWAIN is installed separately. In these versions, once the Service is installed, the JavaScript library will continue to check whether the library file for Dynamic Web TWAIN is installed (`.dll/.bundle/.so` for Windows/macOS/Linux). The pre-defined ports for Dynamic Web TWAIN are 18622/18995/18620 (HTTP) or 18623/18996/18621 (HTTPS). If the library isn't found, the service will attempt to download the file (a `.zip` file that contains the library) and put it in place. No user-interaction needed for this step.

  > NOTE: On **Windows**, should the downloading or installing of the `.zip` file fails, an extra prompt will come up to allow the user to download and install the library manually.

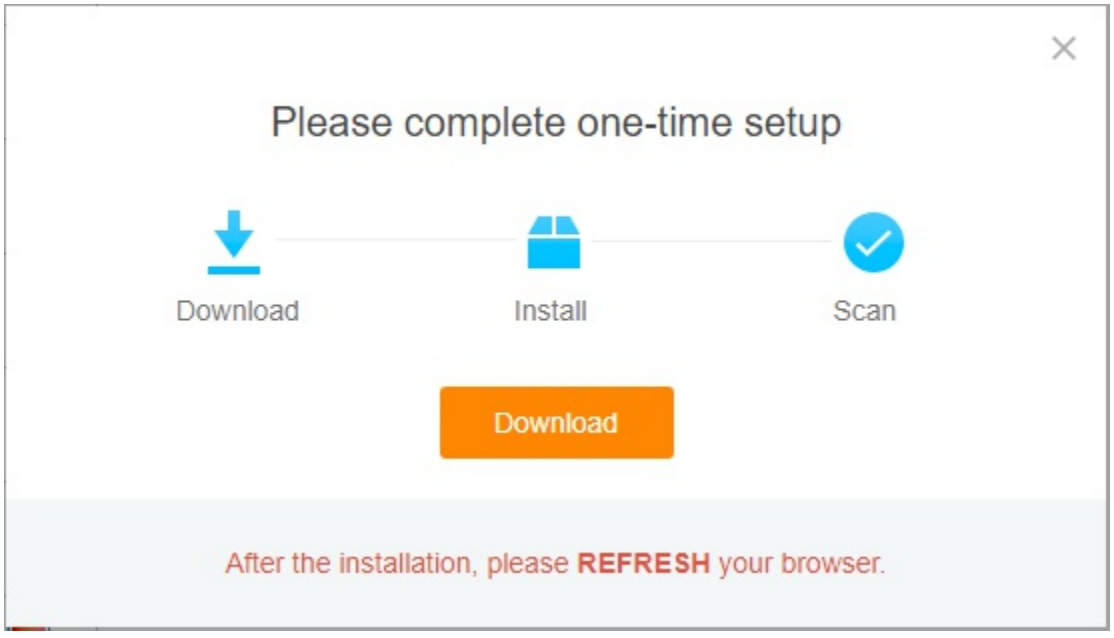


ActiveX Edition

For IE 6~9 on Windows, ActiveX is used. In v14.1 and v14.2, if it's not installed, the following message will appear where the component is expected to be on the page to allow the user to download and install the ActiveX.



For v14.3 and above, the prompt will be the same as the HTML5 edition



> NOTE In v14.2 and above, the ActiveX edition is installed together with the Dynamsoft Service and the HTML5 edition.

For more information about the installation files in version 14.1. Check out About the Distribution Files

---

Install/Uninstall Silently

To silently install Dynamsoft Service via Command Line, you need to first have the file `DynamsoftServiceSetup.msi` which can be found in the installation folder of Dynamic Web TWAIN which is typically located at

`C:\Program Files (x86)\Dynamsoft\Dynamic Web TWAIN SDK {Version No.} {Trial}\Resources\dist\win`

Please run the CMD as administrator, and enter the following line of code with the relevant `.msi` name.

```
C:\Users\{User}\{DWT Path}>msiexec /i DynamsoftServiceSetup.msi /quiet
```

To silently uninstall Dynamic Web TWAIN, you can replace the `/i` parameter with `/x`.

```
C:\Users\{User}\{DWT Path}>msiexec /x DynamsoftServiceSetup.msi /quiet
```

If you want to create an install/uninstall log, you can use a command line which looks like this:

```
C:\Users\{User}\{DWT Path}>msiexec /x DynamsoftServiceSetup.msi /quiet /L*V "C:\log\example.log"
```

If you can't find the `.msi`, feel free to contact Dynamsoft Support Team.

Note: if you are a system administrator, you can also use the `.msi` to remotely install the software to all the end-user machines through Group Policy. Check out the article from Microsoft.

Uninstall on the Client Machines

The step one is always to close the browsers to make sure the SDK is not in use.

On Windows

ActiveX

1. Uninstall the DynamicWebTWAINActiveX through Control Panel. The names for the program are `Dynamic Web TWAIN Trial x86` & `Dynamic Web TWAIN Trial x64`
2. Remove it by using the tool called `DWTICSActiveXClearTool.exe`

HTML5

1. Remove the Dynamsoft Service through Control Panel. The name is `Dynamsoft Service`.
2. Remove the folder `C:\Windows\SysWOW64\Dynamsoft\DynamsoftService` and all files under it

On macOS

Run the file `Uninstall.pkg` to uninstall it. The file can be found in `Go > Applications > Dynamsoft > WebTwain > {installed version No.} >`

On Linux

Run the file `uninstall.sh` to uninstall it. The file can be found in `opt/dynamsoft/DynmasoftService`

Deploy & Distribute

Environmental Requirements

Sever Side

Dynamic Web TWAIN runs 100% on the client-side, thus it doesn't matter what operating system runs on the server. The role of the server is to hold the Resources of the SDK and supply them when needed. No matter what web server is being used, you must make sure that it has the following MIME types set up correctly

Extension	MIME Type	Required by
.css	text/css	All Editions
.js	application/javascript	All Editions
.zip	application/x-zip-compressed	All Editions
.cab	application/vnd.ms-cab-compressed	ActiveX
.exe	application/octet-stream	ActiveX
.msi	application/octet-stream	HTML5 for Windows
.pkg	application/pkg-mac	HTML5 for macOS
.rpm	audio/x-pn-realaudio-plugin	HTML5 for Linux
.deb	application/x-debian-package	HTMl5 for Linux

Client Side

Check out About Editions

About the Running Services

On Windows

> There are three always-running processes by default. All of them are called `Dynamsoft Service` and uses the same file `DynamsoftService.exe`. However they are started with different arguments.

The main process is started without any argument.

```
`C:\Windows\SysWOW64\Dynamsoft\DynamsoftService\DynamsoftService.exe`
```

Then there is a monitor process which is meant to monitor the main process and automatically start it in case it crashes

```
`C:\Windows\SysWOW64\Dynamsoft\DynamsoftService\DynamsoftService.exe -asmonitor Global\Dynamsoft_1.4_73769109_stop_service_event Global\Dynamsoft_1.4_73769078_certcheck_event`
```

The last always-running process is meant to support the SSL certificate for Firefox

`C:\Windows\SysWOW64\Dynamsoft\DynamsoftService\DynamsoftService.exe" "-certcheck" "1160" "Global\Dynamsoft_1.4_73770140_1_certcheck_event`

Then each time you open a browser tab to use Dynamic Web TWAIN, two more processes will appear which are

- Dynamsoft Service, also using the file `DynamsoftService.exe`. It's started like this

```
 "-scan" "\\.\pipe\dynamsoftscan_14.1_1160_52" "0" "Global\ss342249531_53_1160" "0"
"C:\Windows\SysWOW64\Dynamsoft\DynamsoftService\dwt_trial_14.1.0.0828.dll"
```

- Dynamsoft Scanning New Module which uses the file `DSSCN.exe`. It's started like this

```
 "-scan" "\\.\pipe\dynamsoftscan_14.1_1160_50" "1" "Global\ss342249250_51_1160" "0"
"C:\Windows\SysWOW64\Dynamsoft\DynamsoftService\dwt_trial_14.1.0.0828.dll"
```

About the Distribution Files

> Like any other libraries, you need to include the resource files of Dynamic Web TWAIN to use it in your project. The library is distributed as multiple JavaScript files, a CSS file and multiple binary files. From version 14.1, the files are organized as shown below

Resources
addon
dynamsoft.upload.js
dynamsoft.webtwain.addon.pdf.js
dist
linux
DynamsoftServiceSetup.deb
DynamsoftServiceSetup.rpm
LinuxDWT_Trial_14.1.0.0828.zip
LinuxPdf.zip
mac
DynamsoftServiceSetup.pkg
MacDWT_Trial_14.1.0.0828.zip
MacPdf.zip
win
ActiveX
DynamicWebTWAIN.cab
DynamicWebTWAINx64.cab
DynamicWebTWAINActiveX.exe
WebTwainMSITrialX64.msi
WebTwainMSITrialX86.msi
DynamsoftServiceSetup.msi
WinDWT_Trial_14.1.0.0828.zip
WinDWT_Trial_14.1.0.0828_x64.zip
DynamicWebTWAINModuleTrial.msi
DynamsoftPDFRasterizerModule.msi
Pdf.zip
Pdfx64.zip
serviceupdate
LinuxDSUpdate_14.1.0.0828.zip
MacDSUpdate_14.1.0.0828.zip
WinDSUpdate_14.1.0.0828.zip
WinDSUpdate_14.1.0.0828_x64.zip
LICENSE

Readme.txt
dynamsoft.webtwain.config.js
dynamsoft.webtwain.css
dynamsoft.webtwain.initiate.js
dynamsoft.webtwain.install.js
As you can see, there are more than 20 files. Depending on how you design your application, you can remove some of them to keep only the necessary files. The purpose of these files is detailed below to help you better understand them.

JavaScript, CSS, etc.

- `Readme.txt`

  This file contains information about the resources files.

- `dynamsoft.webtwain.config.js`

  This file is used to make basic configuration of Dynamic Web TWAIN. It's where you enter the product key for the product and to change the initial viewer size, etc.

- `dynamsoft.webtwain.initiate.js`

  This file is the core of the Dynamic Web TWAIN JavaScript Library. You're not supposed to change it without consulting the Dynamsoft Support Team.

- `dynamsoft.webtwain.install.js`

  This file is used to configure the dialogs which are shown when Dynamic Web TWAIN is not installed or needs to be upgraded. This file is already referenced inside the `dynamsoft.webtwain.initiate.js`

- `dynamsoft.webtwain.css`

  This file contains the style definitions for all the elements of built-in image viewer, progress bar, dialogs, etc.

- `addon/dynamsoft.webtwain.addon.pdf.js`

  This file contains the functionalities of the PDF Rasterizer addon. You're not supposed to change it without consulting the Dynamsoft Support Team.

- `addon/dynamsoft.upload.js`

  This file defines the interfaces of the Dynamsoft Upload Module.

Binary files

Under `dist/win/`

> The following files are for end users who use IE 10/11, Edge, Chrome or Firefox on Windows (Windows XP/7/8/2008/2012/2016 and 10; 32-bit and 64-bit)

- `DynamsoftServiceSetup.msi`

  This Dynamsoft Service needs to be manually installed on end-user machine. For controlled environment, you can also use it to silently deploy the service to all end-user machines.

- `WinDWT_*_*.*.*.*.zip`
- `WinDWT_*-*.*.*.*_x64.zip`

These `.zip` files contain the core scanning library which is TWAIN-based. Please keep it in the Resources folder on your HTTP server. The file will be automatically and silently deployed to the end-user machine once the Dynamsoft Service is installed. You must make sure that your HTTP server is able to serve `.zip` files.

- `DynamicWebTWAINModule.msi` or `DynamicWebTWAINModuleTrial.msi`

  In case the automatic deplyment of the `.zip` files fails, a prompt will come up and provide this `.msi` file for end users to download and install the core scanning library.

> The following files are for Windows end users who use IE 6/7/8/9. As these old versions of IE don't support HTML5 and still rely on the old ActiveX technology. We still provide the ActiveX to support them.

- `ActiveX/DynamicWebTWAINActiveX.exe`

  This is the default package to be downloaded in an automatic prompt, the end user needs to install it manually.

- `ActiveX/WebTwainMSIX64.msi` or `WebTwainMSITrialX64.msi`
- `ActiveX/WebTwainMSIX86.msi` or `WebTwainMSITrialX86.msi`

  These `.msi` files are mainly designed for silent group deployment in a controlled environment.

- `ActiveX/DynamicWebTWAIN.cab` and `ActiveX/DynamicWebTWAINx64.cab`

  These `.cab` files are Microsoft's legacy way to install ActiveX in Internet Explorer. If you prefer using them, you can set `ActiveXInstallWithCAB` to `true` in `dynamsoft.webtwain.config.js` .

- `Pdf.zip` and `Pdfx64.zip`

  These files are used to install the PDF Rasterizer on the client machine.

Under dist/mac/

> The following files are for end users who use Safari, Chrome or Firefox on mac OS (OS X 10.6.8+)

- `DynamsoftServiceSetup.pkg`

  This Dynamsoft Service needs to be manually installed on end-user machine.

- `MacDWT_*.*.*.*.zip`

  These `.zip` files contain the core scanning library which is TWAIN|ICA-based. Please keep it in the Resources folder on your HTTP server. The file will be automatically and silently deployed to the end-user machine once the Dynamsoft Service is installed. You must make sure that your HTTP server is able to serve `.zip` files.

- `MacPdf.zip`

  This file is used to install the PDF Rasterizer on the client machine.

Under dist/linux

> The following files are for end users who use Chrome or Firefox on Linux (Ubuntu 12.0.4+, Debian 8+, Fedora 24+, mint 18.3; 64-bit)

- `DynamsoftServiceSetup.deb` or `DynamsoftServiceSetup.rpm`

  This Dynamsoft Service needs to be manually installed on Debian/Ubuntu/mint or Fedora end-user machine.

- `LinuxDWT_*.*.*.*.zip`

  These `.zip` files contain the core scanning library which is SANE-based. Please keep it in the Resources folder on your HTTP server. The file will be automatically and silently deployed to the end-user machine once the Dynamsoft Service is installed. You must make sure that your HTTP server is able to serve `.zip` files.

- `LinuxPdf.zip`

  This file is used to install the PDF Rasterizer on the client machine.

Under dist/serviceupdate

- `WinDSUpdate_14.1.0.0828.zip`
- `WinDSUpdate_14.1.0.0828_x64.zip`
- `MacDSUpdate_14.1.0.0828.zip`
- `LinuxDSUpdate_14.1.0.0828.zip`

  These files are used to update the Dynamsoft Service. The update is disabled by default but can be enabled by setting `IfUpdateService` to `true` in `dynamsoft.webtwain.config.js`.

License & Price

Update the Product Key

To update your application to use a new trial key or a full version key. You only need to find the file `dynamsoft.webtwain.config.js` and replace the key in it

```
Dynamsoft.WebTwainEnv.ProductKey = '{the new key}';
```

For the full version, you also need to make sure that you have set the following

```
Dynamsoft.WebTwainEnv.Trial = false;
```

If you are using the trial version but doesn't have a valid key, you can request a trial key here.

If you can't find the file `dynamsoft.webtwain.config.js`, you might be using a file called `dynamsoft.webtwain.min.js` instead. In this case, you can simply write the following line in your own javascript code before using any APIs of the Dynamic Web TWAIN SDK.

```
Dynamsoft.WebTwainEnv.ProductKey = '{the new key}';
```

Decide On Required Licenses

Dynamic Web TWAIN is licensed on a per-Server per-Application basis and it comes in several different editions. To determine which licenses you need for your application, you basically need to be sure of the following. For more info, check out the License Agreement.

1. What operating sytems and browsers your clients use. More info>>
2. How many web servers (physical or virtual) will the application be deployed to.

> NOTE:
>
> It doesn't matter what operating system the server runs.

The license is perpetual by default. However, we also offer other options like annual license, revenue share, etc.. Feel free to contact Dynamsoft Sales Team for more information.

TroubleShooting

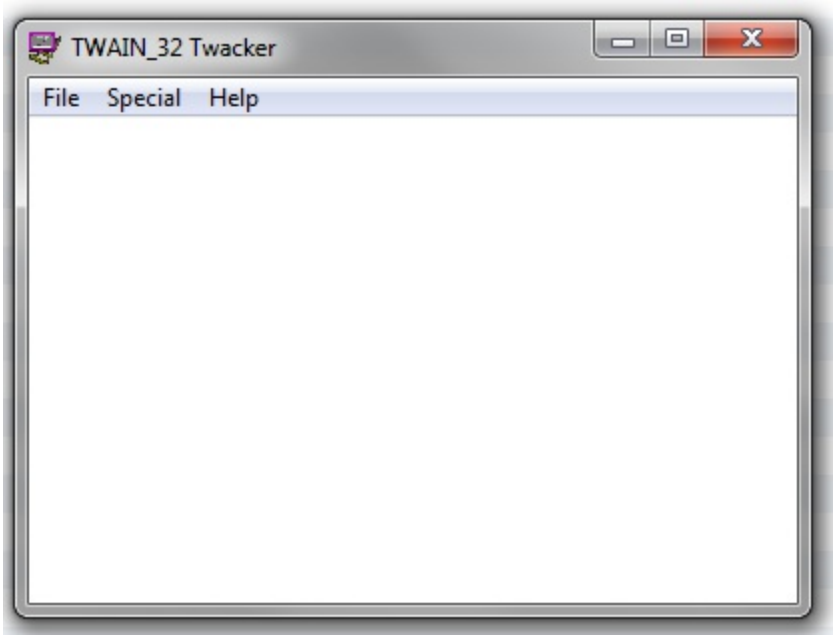
Confirm Hardware Compatibility

In order to support the widest range of scanners, Dynamic Web TWAIN is designed strictly by the following standards

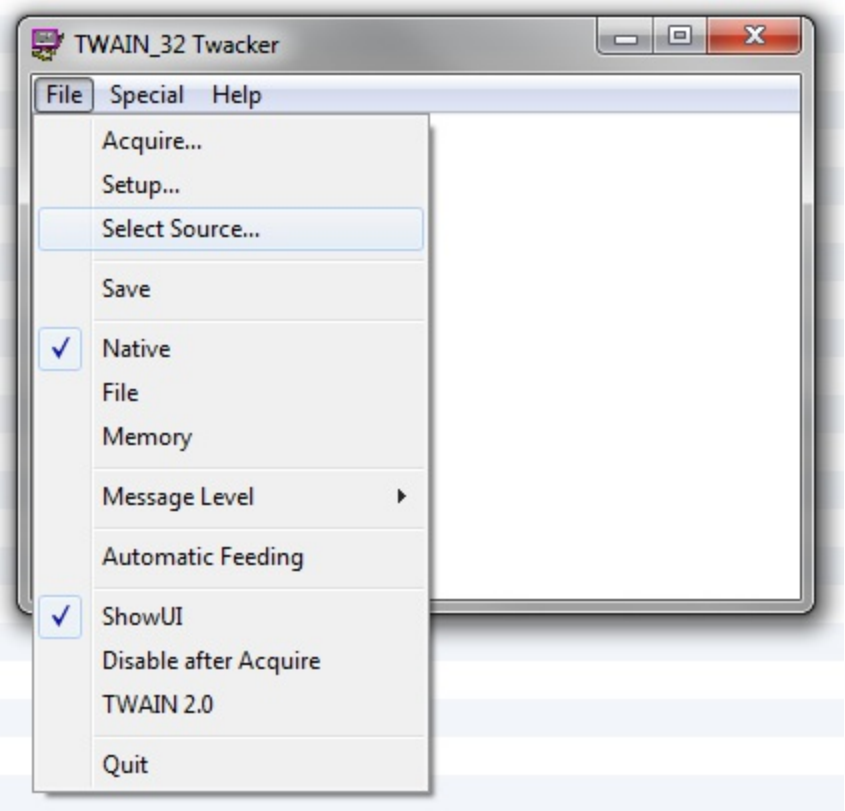
- Windows: `TWAIN Specification`
- macOS: `TWAIN Specification` & `Image Capture Class`
- Linux: `SANE`

Therefore, as long as your scanner is compliant with the above standard on the specific OS, it is supported. Here we'll talk a bit on how to find a supported device.

- macOS: Apple maintains a list themselves. Check out Printer and scanner software for macOS High Sierra, Sierra, El Capitan, Yosemite, and Mavericks
- Linux: Check out SANE: Supported Devices
- Windows: The TWAIN specification is a 25+ years old standard widely accepted by scanner vendors. So it is very easy to find a compatible scanner for your Windows clients because almost all models of all brands of scanners in the market nowadays support TWAIN. If you want to verify if a scanner driver is TWAIN compatible, you can refer to twain-certified-drivers. However, this list is not complete which means you might not find your device there. In this case, you can also use a tool to help verify whether your device is TWAIN-compliant. The tool is called Twacker and it's developed by the TWAIN Working Group. To use it to verify your device, the steps are
- Download and install the tool
- Open the installed program

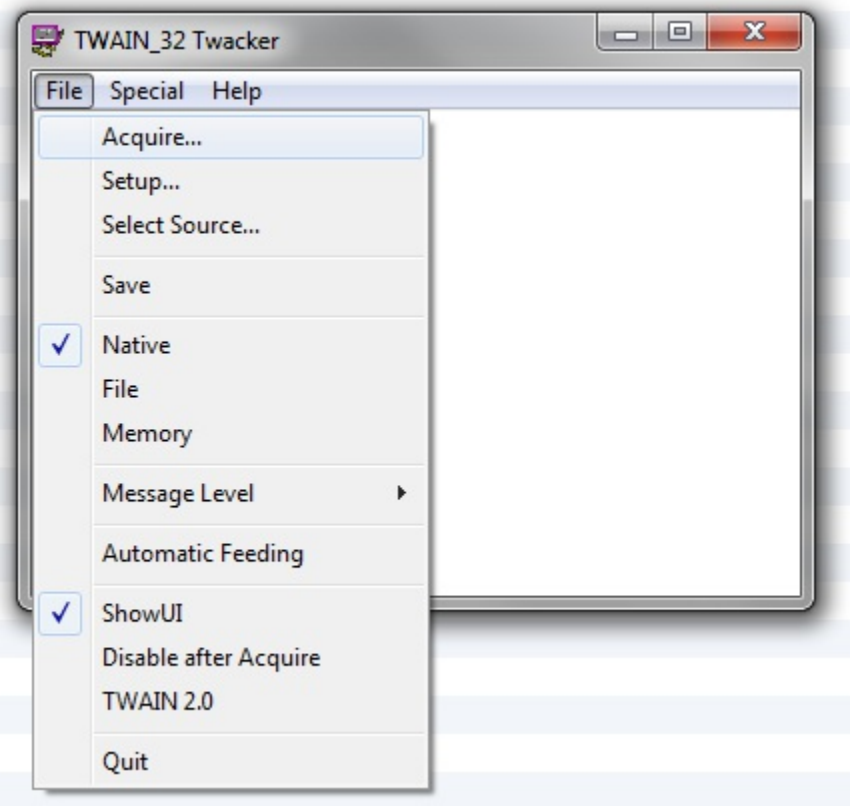


- Select your device



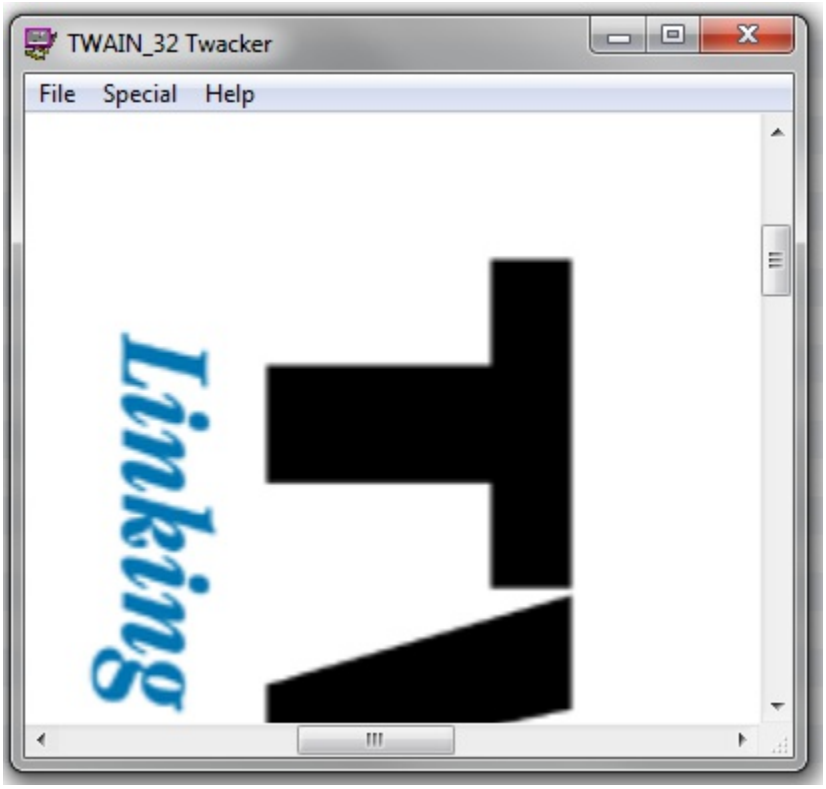
> If your device is not listed, please check if the driver is installed. Or, you can try running Twacker as "Admin" since you may not have permission to access the data source.

- Choose the settings and try scanning



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

try other commands and see how it works



> If your scanner doesn't work with TWACKER, please check your scanner model online and make sure you have installed the (latest) TWAIN driver.

Get Detailed Info of Issues

To catch more detailed information for troubleshooting issues with Dynamic Web TWAIN, there are a few things to try.

Collect Logs

For HTML5 edition for Windows/macOS/Linux, the log files are located in

Windows:

```
C:\Windows\SysWOW64\Dynamsoft\DynamsoftService\log
```

macOS:

```
Go > Applications > Dynamsoft > DynamsoftService > {installed version No.} > log
```

Linux:

```
/opt/dynamsoft/DynamsoftService/log
```

To get logs for a particular issue, the recommended steps are

1. Delete old logs
2. Change `LogLevel` to `14` in the file `C:\Windows\SysWOW64\Dynamsoft\DynamsoftService\log\DSConfiguration.ini`
3. Reproduce the issue

> NOTE:

> 1. Enabling the debug mode will slow down the performance of the scan page. Don't forget to change `LogLevel` back to `6` .
> 2. On macOS, also collect the `system.log` file. To locate it,type `/var/log` in `Go > Go to Folder…`

For ActiveX, the steps to collect the log are different

1. Download the tool DebugView, unzip and open `Dbgview.exe`
2. Set `LogLevel` to `1` in JavaScript

```
DWObject.LogLevel = 1;
```

3. In `Dbgview.exe` , click `Ctrl + X` to clear display
4. Reproduce the issue in Internet Explorer
5. In `Dbgview.exe` , click `Ctrl + S` to save the log file

Check out How to Read Log on how to read the log file.

Collect Dump files

On Windows, if the HTML5 edition of Dynamic Web TWAIN crashes, it'll generate a dump file automatically under `C:\Windows\SysWOW64\Dynamsoft\DynamsoftService\dump` . To troubleshoot with the dump file, please send it to Dynamsoft Support Team and describe the issue in details (including steps to reproduce the issue if possible).

How to Read Log

Question 1 : What are these log files for?

> **Answer** :
>
> dss.log --> For Dynamsoft Service
>
> nw.log --> Details about the network traffic
>
> wts.log --> For the client part of Dynamic Web TWAIN HTMl5 edition
>
> wtss.log --> For the server part of Dynamic Web TWAIN HTMl5 edition

Question 2 : What info do we see if Dynamic Web TWAIN SDK is not doing anything?

> **Answer** :
>
> There is a regular polling going on every 30 seconds, in the log, it looks like this

```
Client-side:

[Process:  1716 Thread: 17652] [05/23/2018 17:43:45.075] [Debug-0]: CClientProxy::Send begin task=DefaultSourceName s
eq=18975 status=8 event=11
[Process:  1716 Thread: 17652] [05/23/2018 17:43:45.090] [Debug-0]: CClientProxy::Send end 0
[Process:  1716 Thread: 17652] [05/23/2018 17:44:15.150] [Debug-0]: CClientProxy::Send begin task=DefaultSourceName s
eq=18976 status=8 event=11
[Process:  1716 Thread: 17652] [05/23/2018 17:44:15.156] [Debug-0]: CClientProxy::Send end 0

Server-side:

[Process:  5364 Thread:  7536] [05/23/2018 18:01:20.930] [Debug-0]: CTwainServer::receive1 task=DefaultSourceName seq
=18478 status=8 event=11
```

Question 3 : What does a command to the service look like? >

> **Answer** : It looks like the following with "id" used to identify which **client** sent the command

```
cmd = [{
"id" : "467653534",
"method" : "GetImageByIndex",
"parameter" : [ 0, 581, 511 ]
}].
```

Question 4 : What basic infomration can we get from the log? >

> **Answer** :
>
> - The Operating system

```
Windows info: 6.2.9200 Pack: 0.0 Other:PID=2 Type=1 Mask=256
```

> - The version of Dynamsoft Web TWAIN (wts.log)

```
Activex Version info:32c0048, Dynamic Web TWAIN 14.0 Trial, 14, 0, 0, 0618, x64:0
```

> - The version of Dynamosft Service (dss.log)

```
Current version info: Dynamsoft Service 1, 4, 0, 0618.
```

> - The current LogLevel

```
Log Level = 14, 0.
```

> - Websocket listening ports

```
[Debug-0]: dwt_command, Websocket connection initialized.
[Info-0]: Create websocket context succeed at default 127.0.0.1:18625, use_ssl = false!
[Info-0]: Start websocket service succeed. port = [18625], use_ssl = false.
[Process: 16784 Thread: 15400] [05/23/2018 17:56:44.554] [Info-0]: Websocket Listening starts at port = 18625, use_ss
l = false
[Debug-0]: Get the port number and try creating websocket listening.
[Debug-0]: dwt_command, Websocket connection initialized.
[Info-0]: Create websocket context succeed at default 127.0.0.1:18626, use_ssl = true!
[Info-0]: Add https service succeed! Succeed port = 18626, use_ssl = true.
[Process: 16784 Thread: 10740] [05/23/2018 17:56:46.794] [Info-0]: Websocket Listening starts at port = 18626, use_ss
l = true
[Debug-0]: Get http service parameter
[Debug-0]: Get the port number and try creating websocket listening.
[Info-0]: Add http service succeed! Succeed port = 18622, use_ssl = false.
[Debug-0]: dwt_command, Websocket connection initialized.
[Info-0]: Create websocket context succeed at default 127.0.0.1:18622, use_ssl = false!
[Info-0]: Start websocket service succeed. port = [18622], use_ssl = false.
[Process: 16784 Thread:  9508] [05/23/2018 17:56:48.071] [Info-0]: Websocket Listening starts at port = 18622, use_ss
l = false
```

> - Requesting origin

```
origin:http://127.0.0.1:100
```

- How a command worked

```
result json = [{
    "description" : "User cancelled the operation.",
    "exception" : -1032,
    "id" : "667465648",
    "method" : "SelectSource",
    "result" : [ false ],
    "cmdId" : ""
}].
```

- Sequence in which commands are being called

Develop & Customize

Insert Images To a Specified Index

> By default, when you scan or load images, they are appended. In other words, they are added to the very end of the image array in buffer. However, in some business scenarios, the user might want to insert these new images to a specified index. Unfortunately, Dynamic Web TWAIN doesn't provide a native method for that. The following code snippet shows how it can be done

Insert when acquiring

```
function acquireToIndex(index) {
    DWObject.IfAppendImage = false;
    DWObject.CurrentImageIndexInBuffer = index;
    DWObject.RegisterEvent('OnBitmapChanged', function (strUpdatedIndex, operationType, sCurrentIndex) {
        if (operationType == 2) { //inserting
            DWObject.CurrentImageIndexInBuffer ++;
        }
    });
    DWObject.AcquireImage();
}
```

Insert when loading

```
var bPostLoad = false, newIndices = [];
function loadToIndex(index) {
    bPostLoad = false;
    newIndices = [];
    DWObject.IfAppendImage = false;
    DWObject.CurrentImageIndexInBuffer = index;
    DWObject.RegisterEvent('OnPostLoad', function () {
        bPostLoad = true;
        for (var j = 0; j < newIndices.length / 2; j++)
            if (newIndices[j] != newIndices[newIndices.length - j - 1])
                DWObject.SwitchImage(newIndices[j], newIndices[newIndices.length - j - 1]);
        DWObject.IfAppendImage = true;
    });
    DWObject.RegisterEvent('OnBitmapChanged', function (strUpdatedIndex, operationType, sCurrentIndex) {
        if (operationType == 2) { //inserting
            for (var i = 0; i < newIndices.length; i++)
                newIndices[i] += 1;
            newIndices.push(parseInt(strUpdatedIndex[0]));
        }
    });
    DWObject.LoadImageEx('', 5);
}
```

Reuse TWAIN Configurations

> Custom DataSource Data or CDD refers to all configurations of a device. Many devices support exporting CDD to a file or a base64-encoded string to be used later. By using the same CDD, the same device can always carry out scan jobs with the same configurations. CDD can also be shared among multiple devices

> to ensure all of them are scanning with the same settings. To use this feature, we have four methods: `SetCustomDSData()`, `SetCustomDSDataEx()`, `GetCustomDSData()` and `GetCustomDSDataEx()`. The typical steps are

1. Show the scanner's User Interface and change all the settings necessary.
2. Perform a scan and remember/save the CDD in the callback of the event `OnPostAllTransfers` using either `GetCustomDSData()` or `GetCustomDSDataEx()`.
3. Later on, when you or any other user who needs to scan with the same settings on the same device (or device of the same model), you can hide the scanner's own interface and simply use the method `SetCustomDSDataEx()` or `SetCustomDSData()` to pass the CDD to the device.

Code Snippet

```
var DWObject = null;
var Base64EncodedDSData = "";
function Dynamsoft_OnReady() {
    DWObject = Dynamsoft.WebTwainEnv.GetWebTwain('dwtcontrolContainer');
    DWObject.RegisterEvent("OnPostAllTransfers", function () {
        Base64EncodedDSData = DWObject.GetCustomDSDataEx();
    });
}
function AcquireImage() {
    DWObject.IfDisableSourceAfterAcquire = true;
    DWObject.SelectSource(function () {
        var OnAcquireImageSuccess, OnAcquireImageFailure;
        OnAcquireImageSuccess = OnAcquireImageFailure = function () {
            DWObject.CloseSource();
        };
        DWObject.OpenSource();
        if (Base64EncodedDSData != "") {
            DWObject.IfShowUI = false;
            DWObject.SetCustomDSDataEx(Base64EncodedDSData);
        } else {
            DWObject.IfShowUI = true;
        }
        DWObject.AcquireImage(OnAcquireImageSuccess, OnAcquireImageFailure);
    }, function () { console.log("Failed to Select A Source!"); });
}
```

TWAIN Capability Negotiation

How can I set resolution in the X and Y direction separately

> With capability negotiation, you can use the capabilities ICAP_XRESOLUTION and ICAP_YRESOLUTION to set resolution in the X and Y direction separately.

Code Snippet

```
DWObject.SelectSource();
DWObject.OpenSource();
//Set X-RESOLUTION current value.
DWObject.Capability = EnumDWT_Cap.ICAP_XRESOLUTION;
DWObject.CapType = EnumDWT_CapType.TWON_RANGE;
DWObject.CapValue = 300;
if (DWObject.CapSet())
    alert("Successful");
else
    alert("Source doesn't support this capability");
//Set Y-RESOLUTION current value.
```

```
DWObject.Capability = EnumDWT_Cap.ICAP_YRESOLUTION;
DWObject.CapType = EnumDWT_CapType.TWON_RANGE;
DWObject.CapValue = 200;
if (DWObject.CapSet())
    alert("Successful");
else
    alert("Source doesn't support this capability");
DWObjet.AcquireImage();
```

For more information, please refer to How to Perform Capability Negotiation.

How to detect and discard blank pages automatically

> If the TWAIN driver of your device supports discarding blank pages, you can use the driver's built-in feature.
>
> You can set the `IfShowUI` property to true to display the User Interface (UI) of the source and you can check the option there (it normally reads 'discard blank')
>
> If you don't want to show the user interface of the source, you can set `IfAutoDiscardBlankpages` to true or negotiate the `ICAP_AUTODISCARDBLANKPAGES` capability in code to discard blank page automatically. Please NOTE that this property or capability only works if the scanner itself supports the feature (on the hardware level).

Code Snippet

```
DWObject.SelectSource();
DWObject.OpenSource;
DWObject.IfShowUI = false;
//*Use the property
DWObject.IfAutoDiscardBlankpages = true;
//*Use capability negotiation
DWObject.Capability = EnumDWT_Cap.ICAP_AUTODISCARDBLANKPAGES;
DWObject.CapType = EnumDWT_CapType.TWON_ONEVALUE;
DWObject.CapValue = -1;//Auto
if(DWObject.CapSet){
   alert("Successful!");
}
DWObject.AcquireImage();
```

> If the scanner itself doesn't support discarding blank pages, you can also use the method `IsBlankImageExpress()` to do this as a workaround. To detect and discard blank pages automatically, you can do it in the `OnPostTransfer` event which fires after each transfer.

Code Snippet

```
function DWObject_OnPostTransfer() {
    DWObject.BlankImageMaxStdDev = 20;
    if (DWObject.IsBlankImageExpress(DWObject.CurrentImageIndexInBuffer)) {
        DWObject.RemoveImage(DWObject.CurrentImageIndexInBuffer);
    }
}
```

> NOTE: In many cases, the scanned blank image may come with some noises which would affect the result returned by IsBlankImageExpress. To improve the result, you may adjust the value of BlankImageMaxStdDev Property. The default value is 1 (0 means single-color image). Thus, by increasing the value a little bit (e.g. to 20), noises on images are ignored so a blank image can be detected faster.

For more information, please refer to How to Perform Capability Negotiation.

How to rotate the scanned image data prior to transfer

> With capability negotiation, you can use the capability ICAP_ROTATION to rotate the scanned image data before it's transferred.
>
> Note: Before using these methods, please make sure that the driver of your device supports automatic rotating.

Code Snippet

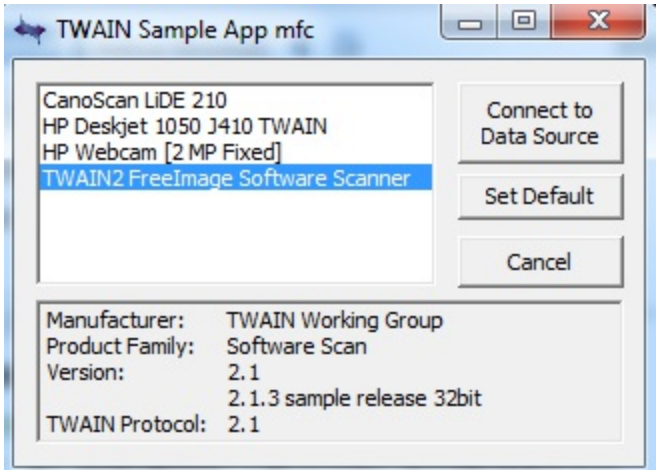
```
function btnScan()
{
    DWObject.SelectSource();
    DWObject.OpenSource();
    DWObject.MaxImagesInBuffer = 4;

   /*============== rotate ==============*/
    DWObject.Capability = EnumDWT_Cap.ICAP_ROTATION;
    DWObject.CapType  =  EnumDWT_CapType.TWON_ONEVALUE;
    DWObject.CapValue = 270; //270 degree rotation
        if (DWObject.CapSet())
            alert("Successful");
        else
            alert("Source doesn't support this capability");
    DWObject.IfDisableSourceAfterAcquire = true;
    DWObject.AcquireImage();
}
```

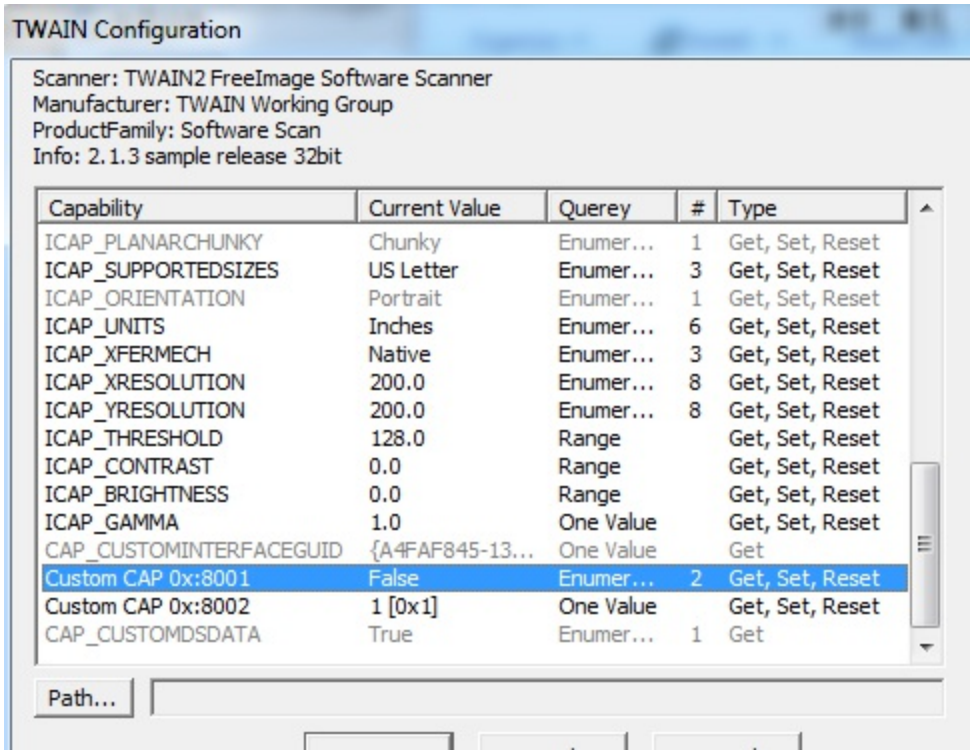
For more information, please refer to How to Perform Capability Negotiation.

How to use custom capabilities of your TWAIN device

> To use a custom capability of your TWAIN driver, you need to know what the capability code is first. You can follow the steps below:

- Install the TWAIN sample application.
- Use the TWAIN Sample App to open the source and then check what the hexadecimal value of the capability is.





- As an example, the code 0x8001 is for the highlighted custom capability above. Now we can use the following code to negotiate the capability.

Code Snippet

```
DWObject.SelectSource();
DWObject.OpenSource();
DWObject.Capability = 0x8001;
DWObject.CapType = 5; //TWON_ONEVALUE
DWObject.CapValue = 1;
if (DWObject.CapSet())
    alert("successful");
else
    alert("Source doesn't support this capatiblity");
```

For more information, please refer to How to Perform Capability Negotiation.

Details on the Upload Feature

How to upload images directly to Database

> To upload images to the server and save them directly into database, you simply need to rewrite the action page on the server. The following is an example written in `C#` and uses MSSQL as the database

```
try {
    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);
    inputStream.Close();
    string sql_insertData = "INSERT INTO " + tableName + " (document_name, document_data) VALUES (@document_name, @do
cument_data)";
    using(System.Data.SqlClient.SqlCommand sqlcmd_insertData = new System.Data.SqlClient.SqlCommand(sql_insertData, C
onnection)) {
        sqlcmd_insertData.Parameters.Add("@document_data", System.Data.SqlDbType.Binary, iFileLength).Value = inputBu
ffer;
        sqlcmd_insertData.Parameters.Add("@document_name", System.Data.SqlDbType.VarChar, 255).Value = strImageName;
        sqlcmd_insertData.ExecuteScalar();
    }
    Connection.Close();
} catch (Exception) {}
```

If you use languges like PHP, JSP, etc., you can check out sample scripts as part of the official upload sample.

How to upload images through methods like AJAX instead of built-in methods

Basically, the workflow would be like this:

1. Scan images to the control.
2. Use the method `ConvertToBase64()` or `ConvertToBlob()` to convert the images in the buffer to a Base64-encoded string or a Blob object.
3. Upload the string or the Blob to the server

We'll use the method `ConvertToBase64()` as an example in the code sample below. Note that `C#` is used on the server-side.

Client-Side

```
// Select all images
var aryIndices = [];
for (var i = 0; i < DWObject.HowManyImagesInBuffer; i++) {
    aryIndices.push(i);
}

DWObject.ConvertToBase64(aryIndices, EnumDWT_ImageType.IT_PDF, asyncSuccessFunc, asyncFailureFunc);

function asyncSuccessFunc(result) {
    var param = encodeURIComponent("ImageName") + "=" + encodeURIComponent("uploadedImages.pdf") + "&" + encodeURICom
ponent("RemoteFile") + "=" + encodeURIComponent(result);
    //You can upload the image data from here
    makeRequest(strActionPage, param, false);
}
```

```
function asyncFailureFunc(errorCode, errorString) {
    alert("ErrorCode: " + errorCode + "\r" + "ErrorString:" + errorString);
}

//AJAX
function makeRequest(url, parameter, flg) {
    if (window.XMLHttpRequest) {
        xhr = new XMLHttpRequest(); // Set up the request.
    } else {
        if (window.ActiveXObject) {
            try {
                xhr = new ActiveXObject("MSXML2.XMLHTTP.3.0");
            } catch (ex) {
                alert(ex);
            }
        }
    }
    if (xhr) {
        if (flg) {
            xhr.open("GET", url, true); // Open the connection.
            xhr.onreadystatechange = requestresultCat;
            xhr.setRequestHeader("If-Modified-Since", "0");
            xhr.send(null);
        } else {
            xhr.open("POST", url, false);
            if (parameter != null) {
                xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
                xhr.send(parameter); // Send the Data.
            } else {
                xhr.send(null);
            }
        }
    } else {
        AppendMessage("&lt;b&gt;Sorry, but I couldn't create an XMLHttpRequest!&lt;/b&gt; ");
    }
}

function requestresult() {
    if (xhr.readyState == 4) {
        if (xhr.status == 200) { // File(s) uploaded.
            var outMsg;
            if (xhr.responseXML & amp; & amp; xhr.responseXML.documentElement) {
                outMsg = xhr.responseXML;
                alert(xhr.responseText);
            }
        }
    }
}
```

Server-Side

```
try {
    string imageName = HttpContext.Current.Request.Form["ImageName"];
    string imageData = HttpContext.Current.Request.Form["RemoteFile"];
    string Path = System.Web.HttpContext.Current.Request.MapPath(".") + "/ImageScanned/";
    if (!Directory.Exists(Path)) {
        Directory.CreateDirectory(Path);
    }
    string outputFileName = Path + imageName;
    // Convert the Base64 UUEncoded input into binary output.
    byte[] binaryData;
    try {
        binaryData = System.Convert.FromBase64String(imageData);
```

```
    } catch (System.ArgumentNullException) {
        System.Console.WriteLine("Base 64 string is null.");
        return;
    } catch (System.FormatException) {
        System.Console.WriteLine("Base 64 string length is not " +
            "4 or is not an even multiple of 4.");
        return;
    }

    // Write out the decoded data.
    System.IO.FileStream outFile;
    try {
        outFile = new System.IO.FileStream(outputFileName,
            System.IO.FileMode.Create,
            System.IO.FileAccess.Write);
        outFile.Write(binaryData, 0, binaryData.Length);
        outFile.Close();
    } catch (System.Exception exp) {}
} catch (Exception exc) {}
```

How to upload multiple files at a time

Scenario

After scanning multiple files, you might want to upload them one by one as individual images. Before version 13.1, you have to call the upload method(s) multiple times. In v13.1+, you can do this in one go.

Solution

Use the methods `ConvertToBlob()` and `HTTPUpload()`.

Steps:

- In JavaScript, write code similar to the following

```
var i = 0;
DWObject.ClearAllHTTPFormField();
DWObject.SetHTTPFormField("UploadedImagesCount", DWObject.HowManyImagesInBuffer);

function asyncFailureFunc(errorCode, errorString) {
    alert("ErrorCode: " + errorCode + "\r" + "ErrorString:" + errorString);
}

var convertImage = function (_index) {
    DWObject.ConvertToBlob([_index], EnumDWT_ImageType.IT_JPG,
        function (result) {
            DWObject.SetHTTPFormField('image_' + _index, result, 'JPG_image_' + _index);
            i++;
            if (i < DWObject.HowManyImagesInBuffer) {
                convertImage(i);
            } else {
                DWObject.HTTPUpload("http://yourserver/youractionpage.aspx", function () {
                    console.log('success')
                }, function () {
                    console.log('failure');
                });
            }
        }, asyncFailureFunc);
};

convertImage(0);
```

- On the server, add an action page to process the uploaded data

```
<%@ Page Language="C#" %>

<%
    try
    {
        String strImageName;
        String strInfo = HttpContext.Current.Request["UploadedImagesCount"];
        short uploadedImagesCount = Convert.ToInt16(strInfo);
        HttpFileCollection files = HttpContext.Current.Request.Files;
        for (short i = 0; i < uploadedImagesCount; i++)
        {
            HttpPostedFile uploadfile = files["image_" + i.ToString()];
            strImageName = uploadfile.FileName;
            uploadfile.SaveAs(Server.MapPath(".") + "\\UploadedImages\\" + strImageName + ".jpg");
        }
    }
    catch
    {}
%>
```

How to use FileUploader to do the upload

In Dynamic Web TWAIN Version 14.0, we added a new upload module called `FileUploader` to handle upload jobs. The following shows how to use this new module to do the upload.

Step:

- Reference the `dynamsoft.upload.js` file

```
<script type="text/javascript" src="Resources/addon/dynamsoft.upload.js">
```

- Initiate the FileUploader module

```
var dsUploadManager;
Dynamsoft.FileUploader.Init('', function(obj) {
    dsUploadManager = obj;
    },
    function(errorcode, errorstring){
        alert(errorstring);
    }
);
```

- Create an job and execute it

```
var job = dsUploadManager.CreateJob();
dsUploadManager.Run(job);
```

Full sample

```
var fileUploaderManager;

Dynamsoft.FileUploader.Init("", onInitSuccess, onInitFailure);

function onInitSuccess(objFileUploader) {
    fileUploaderManager = objFileUploader;
}
```

```
function onInitFailure(errorCode, errorString) {
    alert('Init failed: ' + errorString);
}

function UploadFile() {
    var job = fileUploaderManager.CreateJob();
    job.ServerUrl = 'http://yourserver/youractionpage.aspx';
    DWObject.GenerateURLForUploadData([0], EnumDWT_ImageType.IT_JPG, function(res) {
        job.SourceValue.Add(res, "sample.jpg");
        job.SourceValue.Add("D:\\test1.jpg", "test.jpg");
        job.OnUploadTransferPercentage = FileUpload_OnUploadTransferPercentage;
        job.OnRunSuccess = FileUpload_OnRunSuccess;
        job.OnRunFailure = FileUpload_OnRunFailure;
        fileUploaderManager.Run(job);
    }, function(errorCode, errorString) {console.log(errorString);});
}

function FileUpload_OnUploadTransferPercentage(job, sPercentage) {
    console.log(sPercentage);
}

function FileUpload_OnRunFailure(job, errorCode, errorString) {
    alert(errorString);
}

function FileUpload_OnRunSuccess(job) {
    alert(' upload completed! ');
}
```

Details on the Download Feature

How to download images directly from Database

Scenario

Sometimes you may need to download an image file from the server but that file exists only in the database.

Solution

With a proper action page `DownloadFromDB.aspx`, the method `HTTPDownloadEx()` can fulfill the requirement.

Steps

- On the server, add an action page to fetch the image data from the database and send it back.

Here we take `C#` and `MSSQL` as an example and assume the image to download is a JPEG image and can be fetched by its ID (`iImageID`).

```
string strConnString = "Server=***\\SQLEXPRESS;Database=" + "DemoWebTwain" + ";Integrated Security=sspi;Persist Secur
ity Info=False";
string strExc = "";
string strImageExtName = "jpg";
try {
    String strImageID = request["iImageIndex"];
    //Get the image data from the database
    HttpRequest request = HttpContext.Current.Request;
    byte[] byFileData = null;
    if (strImageID != null && strImageID != "") {
        int iImageID = 0;
        try {
            iImageID = Convert.ToInt32(strImageID);
```

```
        } catch {
            return;
        }
        System.Data.SqlClient.SqlConnection sqlConnection = new System.Data.SqlClient.SqlConnection(strConnString);
        System.Data.SqlClient.SqlCommand sqlCmdObj = new System.Data.SqlClient.SqlCommand("SELECT imgImageData FROM "
 + "tblWebTwain" + " WHERE ID = " + iImageID.ToString(), sqlConnection);
        sqlConnection.Open();
        System.Data.SqlClient.SqlDataReader sdrRecordset = sqlCmdObj.ExecuteReader();
        if (sdrRecordset.Read()) {
            long iByteLength;
            iByteLength = sdrRecordset.GetBytes(0, 0, null, 0, int.MaxValue);
            byFileData = new byte[iByteLength];
            sdrRecordset.GetBytes(0, 0, byFileData, 0, Convert.ToInt32(iByteLength));
        }
        sdrRecordset.Close();
        sqlConnection.Close();
        sdrRecordset = null;
        sqlConnection = null;
    }
    Response.Clear();
    Response.Buffer = true;
    if (byFileData != null) {
        string fileNameEncode;
        fileNameEncode = HttpUtility.UrlEncode("downloadImg.jpg", System.Text.Encoding.UTF8);
        fileNameEncode = fileNameEncode.Replace("+", "%20");
        string appendedheader = "attachment;filename=" + fileNameEncode;
        Response.AppendHeader("Content-Disposition", appendedheader);
        Response.ContentType = "image/jpg";
        Response.OutputStream.Write(byFileData, 0, byFileData.Length);
    }
} catch (Exception ex) {}
```

- On the client side, use the method `HTTPDownloadEx` to download the image. Put the action page `DownloadFromDB.aspx` as the source of the image (as the 2nd parameter `String HTTPRemoteFile`):

```
//downloadsource should be the correct path for
//the page DownloadFromDB.aspx plus any necessary
//parameters needed for fetching the image(s)
var downloadsource = DownloadFromDB.aspx + "?iImageIndex=0";
//You should know the type of the image you are
//downloading because it is downloading as a stream
//and Dynamic Web TWAIN needs the correct type
//in order to process the stream
DWObject.HTTPDownloadEx(strHTTPServer, downloadsource, EnumDWT_ImageType.IT_JPG);
```

Hide or Change the Progress Bar

To hide the progress bar during time-consuming operations like uploading/downloading, etc., set `IfShowProgressBar` and `IfShowCancelDialogWhenImageTransfer` to `false`.

For upload and download, you can use the event `OnInternetTransferPercentage` to customize your own progress bar. This event will return the percentage of the upload/download process

```
DWObject.RegisterEvent('OnInternetTransferPercentage', function(sPercentage){
    console.log(sPercentage); //Your code goes here.
});
```

Customize Prompts

To make the SDK easy-to-use, UI elements like prompts to download Dynamsoft Service are built-in. By default, the prompts come in English. To change it, the steps are

1. Find and open the file `dynamsoft.webtwain.install.js` which can be found under the Resources folder.
2. In this file you can change the wording and formatting of the prompts which is defined in the function `_show_install_dialog` and `OnWebTWAINModuleDownloadManually`.

Customize Display Language

The default display language for Dynamic Web TWAIN is English. From version 14.0, it's possible to config the language. The steps are

1. Find and open the file `dynamsoft.webtwain.config.js`.
2. Search for `Dynamsoft.WebTwainEnv.CustomizableDisplayInfo` and you can find the information which may appear when using the SDK. You can then change the words/sentences to your own native/target language which will then replace the default information.

Customize Built-in Image Editor

Dynamic Web TWAIN has a built-in image editor which shows up when you call the method `ShowImageEditor`. By default, the editor shows all of its buttons in its toolbar and it also takes up the full screen. From version 14.0, it's possible to configure the size of the toolbar as well as where it shows up.

To configure the toolbar, the steps are

1. Find and open the file `dynamsoft.webtwain.config.js`.
2. Search for `buttons: {` and you can find the place where you can change the titles of the buttons as well as whether certain buttons are hidden.

> NOTE: To make sure the buttons show up nicely, the editor itself will hide buttons by group in case the editor window is not wide enough. Therefore, if the window is too narrow, some buttons might be hidden even if you have set them visible.

To configure where the editor shows up on the web page, use the method `ShowImageEditor` and specify the ID of the DIV to show the editor and its size.

Make Saved Images Small

To reduce the size of scanned/imported images when saving or uploading, you can try the following things

1. Scan images in `Grey` or `B&W` by setting the property `PixelType`.
2. Use a lower resolution by seeting the property `Resolution`. For existing images which were imported, you can also use the method `SetDPI` to change its DPI if it's too big.
3. Choose a proper compression type for TIFF/PDF. The properties are `TIFFCompressionType` and `PDFCompressionType`.
4. If you are uploading images as JPEG or JPEG-encoded PDF/TIFF, you can also set the property `JPEGQuality` to a lower value. The following is the sample code

```
DWObject.TIFFCompressionType = EnumDWT_TIFFCompressionType.TIFF_JPEG;
DWObject.JPEGQuality = 60;  // default value is 80
```