Dynamic Web TWAIN

Developer's Guide

Dynamsoft OCR Professional Module

Based on DWT Version 11.3.2

July, 2016

Dynamsoft

12 Years of Experience in TWAIN SDKs and Version Control Solution

Using the OCR Add-on

Description

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

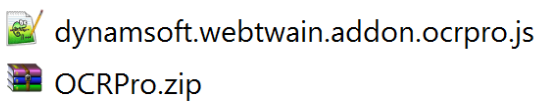
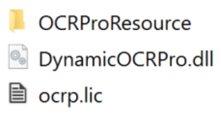
First of all, Dynamsoft offers 2 different OCR solutions:

Professional OCR *<Recommended>*

Dynamsoft developed her own OCR solution based on the Tesseract Open Source OCR Engine which is now called the Basic OCR. While it proves to work just fine for some of our customers, we do get more and more requests for a better OCR solution to integrate with our Dynamic Web TWAIN SDK. After carefully evaluating the options, Dynamsoft chose to work with the industry leader Nuance® to provide an extra OCR solution which is simply called the **Dynamsoft OCR Professional** Module.

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

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

 dynamsoft.webtwain.addon.ocrpro.js OCRPro.zip *Figure 1 **Client Side** OCR Professional Resources*	 OCRProResource DynamicOCRPro.dll ocrp.lic *Figure 2 **Server Side** OCR Professional Resources*

Basic OCR

As mentioned above, Dynamsoft developed the Basic OCR based on Tesseract Open Source OCR. This engine generates decent results at a modest cost. In order to maintain the simplicity of the product, the developers redesigned the APIs to keep only the essential few. More info>>>

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

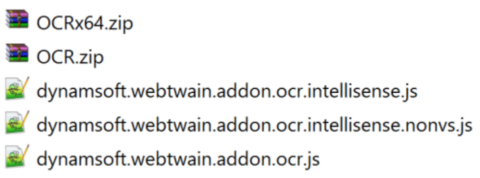

OCRx64.zip
OCR.zip
dynamsoft.webtwain.addon.ocr.intellisense.js
dynamsoft.webtwain.addon.ocr.intellisense.nonvs.js
dynamsoft.webtwain.addon.ocr.js

Figure 3 Client-Side OCR Basic Resources

How to use the Professional OCR Add-on

The professional OCR add-on is designed to work both on the client-side and on the server-side.

Client-side:

You need to first make sure you have referenced the file dynamsoft.webtwain.addon.*ocrpro*.js in your code.

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

Now, let's look at a code snippet

```
function DoOCR() {
    if (DWObject) {
        DWObject.Addon.OCRPro.Download(
            "http://localhost/dwt/addon/OCRPro.zip", function () {
                DWObject.Addon.OCRPro.Recognize(DWObject.CurrentImageIndexInBuffer,
                    function (index,result) { GetOCRProInfoInner(result);},
                    function (errorCode, errorString, result) {alert(errorString);});
            }, function (errorCode, errorString) {alert(errorString); });
    }
}
```

NOTE: You can download a working sample [here](#).

As seen in the snippet above, both methods **Download** and **Recognize** are **asynchronous**.

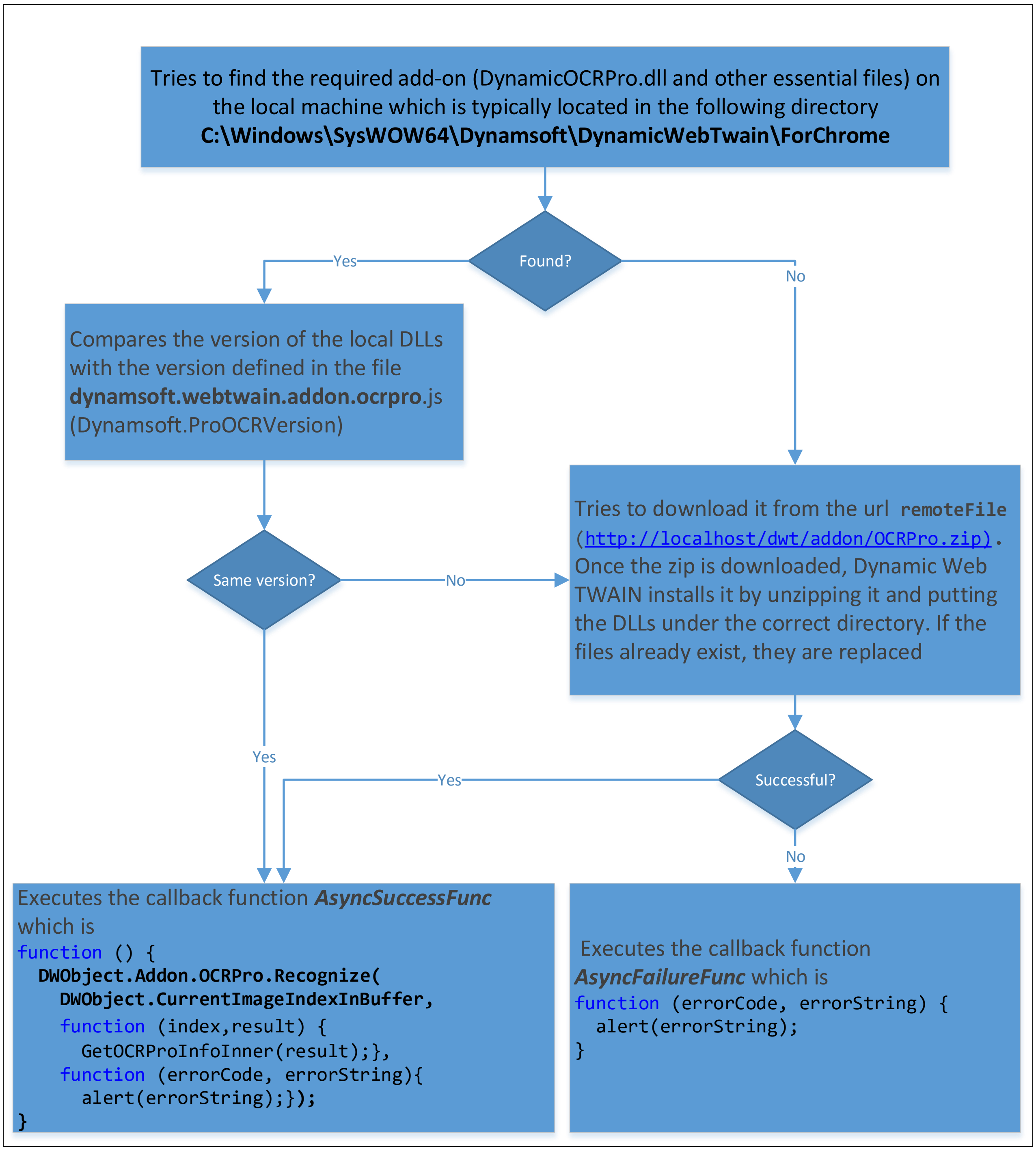
Here is how the 'download' method works

Syntax

```
.Addon.OCRPro.Download(remoteFile, AsyncSuccessFunc, AsyncFailureFunc)
```

Workflow

See the diagram on the next page


Tries to find the required add-on (DynamicOCRPro.dll and other essential files) on the local machine which is typically located in the following directory **C:\Windows\SysWOW64\Dynamsoft\DynamicWebTwain\ForChrome**
Found?
Yes
No
Compares the version of the local DLLs with the version defined in the file **dynamsoft.webtwain.addon.ocrpro**.js (Dynamsoft.ProOCRVersion)
Same version?
No
Tries to download it from the url `remoteFile` (http://localhost/dwt/addon/OCRPro.zip). Once the zip is downloaded, Dynamic Web TWAIN installs it by unzipping it and putting the DLLs under the correct directory. If the files already exist, they are replaced
Successful?
Yes
Yes
No
Executes the callback function ***AsyncSuccessFunc*** which is
Executes the callback function ***AsyncFailureFunc*** which is

```
function () {
  DWObject.Addon.OCRPro.Recognize(
    DWObject.CurrentImageIndexInBuffer,
    function (index,result) {
      GetOCRProInfoInner(result);},
    function (errorCode, errorString){
      alert(errorString);});
}
```

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

Figure 4 Asynchronous workflow for OCRPro.Download

When the necessary files are in place, the OCR begins. Here is how the ‘Recognize’ method works

Syntax

```
.Addon.OCR.Recognize(sImageIndex, AsyncSuccessFunc, AsyncFailureFunc);
```

Workflow

See the diagram on the next page

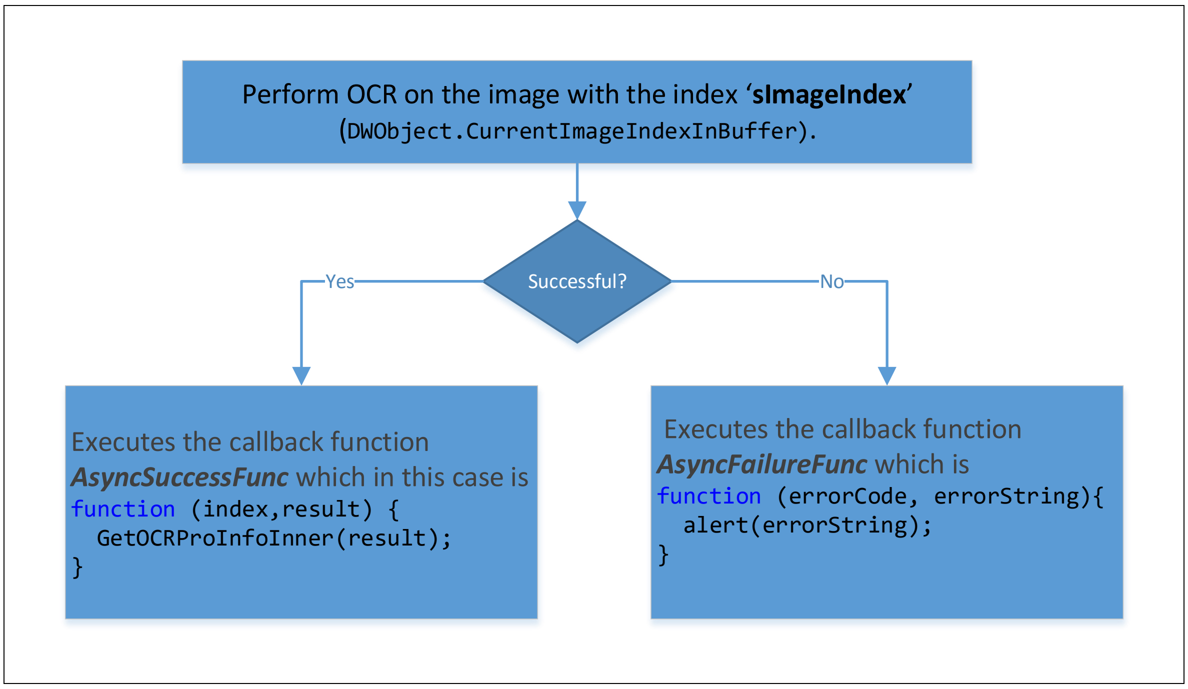

Perform OCR on the image with the index 'sImageIndex' (DWObject.CurrentImageIndexInBuffer).
Successful?
Yes
No
Executes the callback function AsyncSuccessFunc which in this case is function (index,result) { GetOCRProInfoInner(result); }
Executes the callback function AsyncFailureFunc which is function (errorCode, errorString){ alert(errorString); }

Figure 5 Asynchronous workflow for OCRPro.Recognize

After the OCR is done, we need to process the result in the successful callback `function GetOCRProInfoInner`. The following is how it's done

```
function GetOCRProInfoInner(result) {
  if (result == null)
    return null;
  var pageCount = result.GetPageCount();
  if (pageCount == 0) {
    alert("OCR result is Empty.");
    return;
  } else {
    var bRet = "";
    for (var 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);  //Print out the OCR result as plain text
  }
  if (savePath.length > 1)
    //Save the file on the local machine, savePath is an absolute path on the local disk
    result.Save(savePath);
}
```

Server-side:

Most OCR solutions on the market are running on the server-side. So how should you choose the best way for your solution? Below is a simple comparison to help you decide

	Server-Side	Client-Side
Client-side Installation	Not needed	One-time. 90M download
Resources Consumption	All OCRs are done on the server	OCR is done on individual PCs

Basically, if you have a powerful dedicated server which is capable of processing multiple (like tens or hundreds of) documents at the same time, you can choose the server-side OCR. This also means that your users only need to upload the documents to the server. They don't need to spend time downloading the OCR component to their PCs or have their PCs running the OCR each time they need it done to a document.

On the other hand, client-side OCR means you don't need a very powerful server as this is a distributed OCR system. Also the users can OCR and save the result without having to access the server. This will save the time needed for uploading the original documents and downloading the searchable documents after the OCR is done.

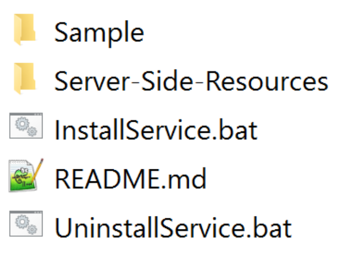
How does the server-side OCR work

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

To implement this solution, please follow the steps below

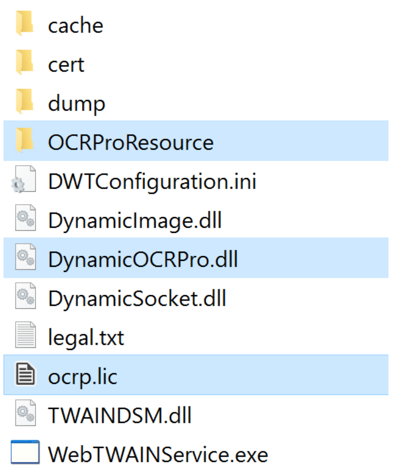
1. Download and **unzip** the all-in-one sample from Dynamsoft Code Gallery. You'll see the structure as shown in Figure 6 on the next page.

2. **[Installation]**
If you have installed Dynamic Web TWAIN 11.3.2 or later on the server already, you need to **manually** copy the following highlighted files shown in Figure 7 and paste them in the directory **C:\Windows\SysWOW64\Dynamsoft\DynamicWebTwain\ForChrome**.



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

Figure 6 Demo structure



cache
cert
dump
OCRProResource
DWTConfiguration.ini
DynamicImage.dll
DynamicOCRPro.dll
DynamicSocket.dll
legal.txt
ocrp.lic
TWAINDSM.dll
WebTWAINService.exe

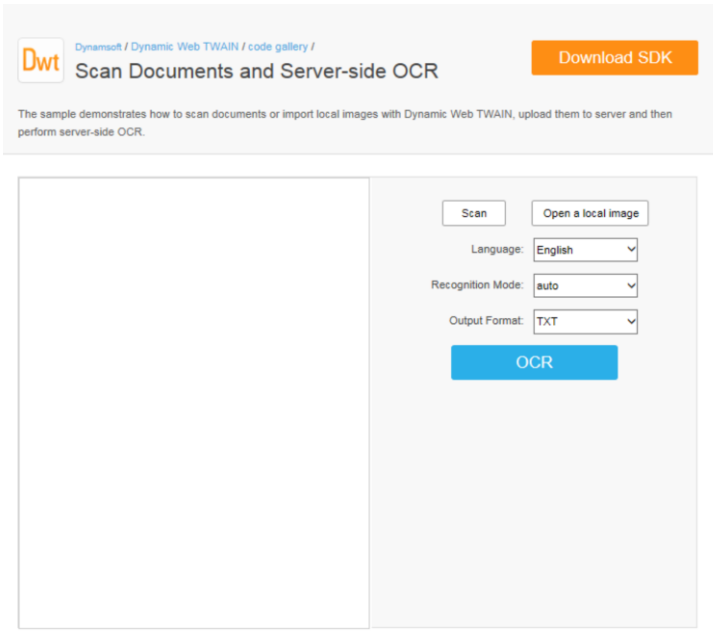
Figure 7 Resources for OCR Professional

3. If you have never installed Dynamic Web TWAIN v11.3.2 or later on the server

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

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

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


Dynamsoft / Dynamic Web TWAIN / code gallery /

Scan Documents and Server-side OCR

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

Language: English

Recognition Mode: auto

Output Format: TXT

Figure 8 The UI for DWT OCR Professional Demo

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

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

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

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

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

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

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

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

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

Should you need any help with implementing the code into your own solution, please feel free to contact Dynamsoft Support team at **support [at] dynamsoft.com** or give us a call toll-free at **1-877-605-5491**.

How to use the Basic OCR Add-on

The basic OCR add-on is designed to work on the client-side and **it works very much like how you use the professional OCR add-on** on the client-side.

The first step is to make sure you have referenced the file dynamsoft.webtwain.addon.ocr.js in your code.

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

And a very similar code snippet

```
function DoOCR() {
    if (DWObject) {
        DWObject.Addon.OCR.DownloadLangData("http://localhost/dwt/addon/OCR.zip");
        DWObject.Addon.OCR.Download(
            "http://localhost/dwt/addon/OCR.zip", function () {
                DWObject.Addon.OCR.Recognize(DWObject.CurrentImageIndexInBuffer,
                    function (index,result) { SaveOCRResult(index, result);},
                    function (errorCode, errorString) {alert(errorString);});
            }, function (errorCode, errorString) {alert(errorString); });
    }
}
```

For explanation of the code above, please refer to how to use OCR professional on the client side.