WINGSCAN WHITEPAPER: APPENDING

APPENDING ADDITIONAL SCANS

This Whitepaper is going to be "laser focused" on implementing one specific use case with WingScan: To allow one append the results of a second and subsequent scan to an existing document / previous scan present in the WebDocumentViewer / WebDocumentThumbnailer.

The "out of the box" experience with WingScan is that when you open or scan a document, you are replacing any existing document that is present. So if you have say 2 pages loaded and you begin a scan with page.. you will end up with just the new 1 page scan open and the former document is lost/closed. This then is about "what if I want to add more pages to the existing open document".

This is not a "getting started with WingScan" tutorial. For that, please see INFO: WingScan Whitepaper - Getting Started with Web Scanning (in .NET Framework).

.For other White Papers please see the Whitepaper category on our KB.

TECHNICAL OVERVIEW

In some ways, the append versus open operation is a WebDocumentViewer / WebDocumentThumbnailer operation. We could easily write a KB on "appending a document in WDV" ... in fact, we may do that at a future date. We will be taking advantage of the WDV/WDT viewer.document.insertPages() feature. The difficulty is that in order to insert multiple pages, one must know how many pages one will be inserting. There is no "just insert all the pages in this document".. (An enhancement request for this is officially on file but has not yet been implemented).

An additional complication is that you don't necessarily know in advance before you scan how many pages will be part of the scan. If you are using blank page detection/removal (VRS License required) then the number of pages may not match the number of pages you have inserted.

In order to solve this problem, what we will be doing is implementing a simple count incrementor during the imageAcquired event so that we track in real time how many pages

have been uploaded. This way, we know the moment the scan has finished how many pages need to be inserted. This saves us the ugly step of having to go back to the server side and implement some form of "count number of pages in this file" (perhaps fodder for a WDV whitepaper in the future).

We will be starting with the final result project from INFO: WingScan Whitepaper - Getting Started with Web Scanning (in .NET Framework). You can download the sources for the starter project here: WingScan_withViewer.zip (11.3.0.3).

We will then modify it to add handlers for image acquired, we will add a new function to decide if we are appending or adding and add buttons to trigger that functionality, and finally adjusting the uploadCompleted handler to perform the append (or open as directed).

DOCUMENT STANDARDS AND ASSUMPTIONS

The technical portions of this document assume that you are a developer with access to MS Visual Studio 2013 or later. We will be providing examples in C#. Our SDK works with VB.NET as well, but for the sake of simplicity any .NET code will be provided in C# You can use a converter tool such as Telerik Code converter to convert examples to VB.NET

This document and our SDK assume that you're a developer familiar with basic use of Visual Studio, IIS Express or IIS, and general web application development practices. The samples will be using an HTML 5 approach in a non-MVC application targeting .NET framework 4.5.2 in a 64 bit application.

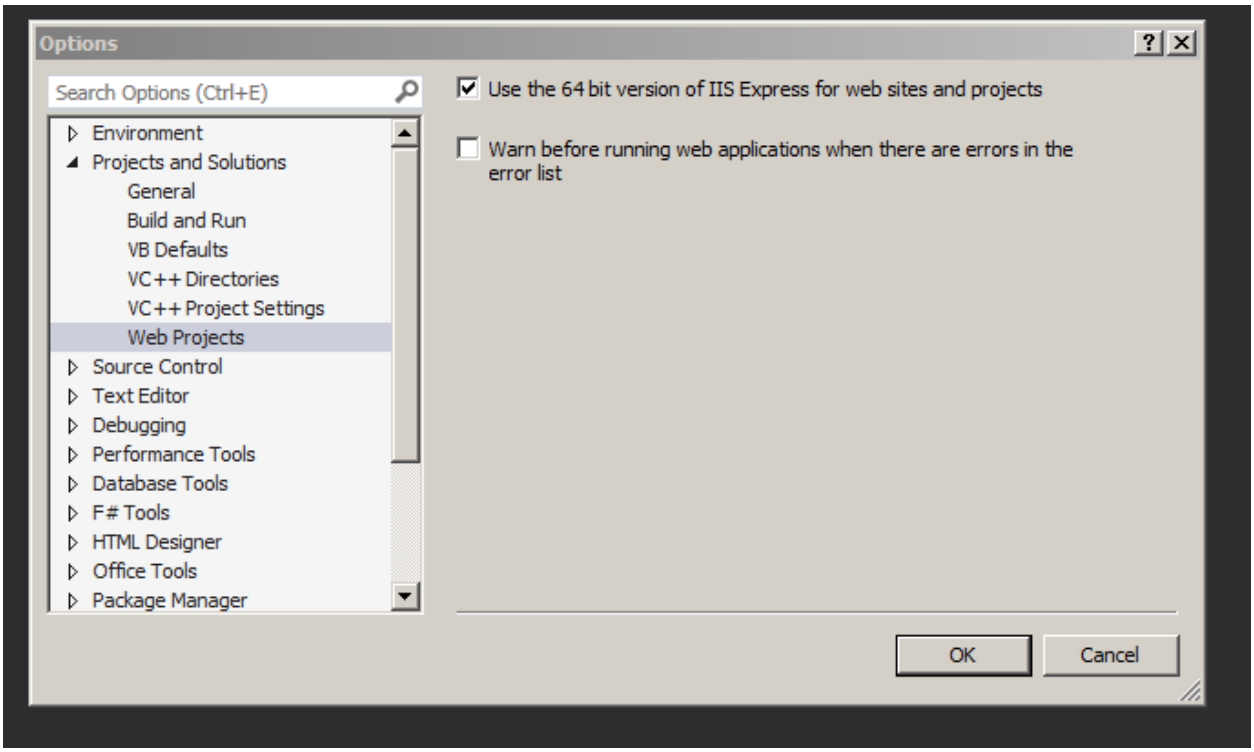
Please note that due to changes in Google Chrome and the Chromium engine, WingScan will not work if you're not on at least 11.4.0.4. Also note you must target at least .NET Framework 4.5.2 with DotImage 11.2 or greater - but since MS has end of lifed < 4.6.1 we suggest at least that.

Also note that this document covers a standard ASP.NET web app, not ASP.NET Core. We do have support for WingScan with .NET Core 8+ and WingScan with .NET Core 10+, but that is beyond the scope of this document. The fundamental concepts of this document apply whether you are going to use ASP.NET or ASP.NET Core the client-side code is the same - the difference is in the handler (used for ASP.NET) versus the startup middleware (used by ASP.NET Core).

INFO: WingScan Whitepaper - Appending Additional Scans

A brief set of instructions for downloading our SDK and activating a license will be given, but the rest of the document assumes you've installed the latest DotImage SDK (The current version when this was written was 11.4.0.14 in March 2024)

Examples herein will use IIS Express (built in web server for Visual Studio) set to run in x64 mode



We do provide NuGet packages for our components, but this example will assume you've installed and activated our SDK locally.

If you're a licensed developer, or are actively evaluating our SDK and run into problems/questions, you are welcome to contact support. You can make a support case in writing [in our support portal]

You may also call in to support during our normal business hours

Support Hours: M - F from 8AM - 5PM EDT (New York Time)

Call us: 1-781-743-2119

STANDARD TROUBLESHOOTING

Web applications with WingScan have a lot of "moving parts" and troubleshooting can seem a daunting task.

There are two tools (one ours, one a free third party tool) that can greatly enhance troubleshooting and assist you in reporting issues to support

If you are having issues with WingScan

Please download and run the diagnostic logging tool

Use it to collect a log while you reproduce the issue.

These diagnostic logs will often show the root cause of an issue. Although they may be a bit daunting to look at at first, rest-assured, they're a valuable diagnostic tool. So, if you're contacting support about a WingScan issue, please take a moment to collect this log then save it to a text file, then zip up the log file and attach it to your support case

In addition to the WingScan log, problems related to the client-side web components are best diagnosed by looking at a log of the network traffic. Most commonly a faulting application will clearly show a 500 internal server error being thrown and the full response body of that error will offer a great deal of insight into the root cause (licensing issues, and errors in the WebCaptureHandler and WebDocumentRequestHandler will most often show up in the 500 error response body)

We recommend Fiddler if you need to collect a log and report it to support.

Please download, install, and run Fiddler web logging

Use it to collect a log while you reproduce the issue, then save the log as a native .saz file and and attach it to your support case as a file attachment/ (please do not save as CSV or text log. The native SAZ file gives us much better access to the tools we need to help diagnose your issue)

PLEASE NOTE: you need to capture a session while NOT using SSL/HTTPS. Fiddler logs cannot see into HTTPS without enabling a special certificate which we do not recommend.. if

your capture is of an HTTPS session we will not get useful diagnostic information from the log

INSTALLATION AND LICENSING

If you have not done so already, please download the latest version of Atalasoft DotImage SDK

NOTE: The current version is 11.4.0.14 as of the creation of this document (March 2024) updates of the SDK happen with regularity. We work hard not to make breaking changes so this guide should be compatible with future releases. We will update the specific technical details as needed as changes arise

Some of the images here may be of older versions - but you need to be on at least 11.4.0.4 in order to work with WingScan and modern Chromium based browsers.

If you have not done so, please create an account with Atalasoft so that you can download and activate our SDK

- Download and install the Atalasoft DotImage SDK
- Run the activation wizard
- If you have SDK serials for DotImage and/or WingScan use the "Activate an SDK serial for use on this machine or a server license" option and activate your serials for version 11.4
- If you do not yet have licenses, you can select the "request 30 day evaluation"
- If you run into issues activating or require assistance, please contact support

NOTE for IIS users: The licensing above should be sufficient if you're using the built in web server in Visual Studio (IIS Express) but if you are using a local copy of IIS, then the IIS process does not run as your user name so it will not pick up your licenses. You'll need to take the extra step of copying your licenses into the application bin directory.

Assuming you have an app named WingScanTestApp hosted in IIS under:

c:\inetpub\wwwroot\WingScan_withViewer\

Then you'd copy your licenses from:

C:\users\YOUR_USERNAME\AppData\Local\Atalasoft\DotImage 11.4\

to

C:\inetpub\wwwroot\WingScan_withViewer\bin\

MODIFYING THE SAMPLE APP

As mentioned above, we will be basing this project on the completed getting started with WingScan project from another whitepaper.

- Download BasicWebCapture_withViewer.zip (11.5.0.10)
- Unzip it to a known location
- Open the solution with Visual Studio 2022 (or later)
- Add vars for keeping track of state
  - open Default.aspx and edit the script section where we initialize viewers... look for where it has

    ```
    <script type="text/javascript"> var _viewer = null; var _thumbs = null; //
    Initialize Web Scanning and Web Viewing... this will run at startup $(function () {
    ...
    ```

  - just after the var _thumbs line, add this:

    ```
    // _append and _scanCount are needed for scan new vs scan add... var _append =
    false; var _scanCount = 0;
    ```

- Create utility function makeSrcArray

  This is needed due to an oddity in how the insertPages works - we need to build the array of page indices it expects when given just a page count.
  - Within the JavaScript block, just after you initialize the viewer and web scanning, add this makeSrcArray function

    ```
    // This is needed because the insertPages call needs an array of indices // so we
    pass in a number of pages to generate the needed src indicies array function
    makeSrcArray(count) { var returnArray = new Array(); for (var i = 0; i < count; i++)
    { returnArray.push(i); } return returnArray; }
    ```

- Modify scanWithSelectedScanner function

  The WingScan_withViewer sample app has a scanWithSelectedScanner method. We need to modify it to handle control of append versus open

- In the default.aspx page, find the original scanWithSelectedScanner() function:

```
// this is needed because if you do not use this code, then only the system default
scanner will ever be used function scanWithSelectedScanner() {
Atalasoft.Controls.Capture.WebScanning.scanningOptions.scanner =
$('.atala-scanner-list').val(); Atalasoft.Controls.Capture.WebScanning.scan(); }
```

- Replace it with this:

```
// this is needed because if you do not use this code, then only the system default
scanner will ever be used // We will be adding the append value to set here function
scanWithSelectedScanner(append) { // setting the _append var to true/false to
determine if we will scan new (false) or append to existing scan (true) if(append) {
_append = true; } else { _append = false; } // this is the standard scan with
selected scanner code Atalasoft.Controls.Capture.WebScanning.scanningOptions.scanner
= $('.atala-scanner-list').val(); Atalasoft.Controls.Capture.WebScanning.scan(); }
```

  What we have done here is modified it to take an argument for append.. if there's anything not-null passed in, then we set append to true, otherwise, we set to false. Then we just do the same thing we did in the original code.. scan wit the selected scanner.

- Add event handlers for onScanStared and onImageAquired in support of append feature

  Now, we need to modify the web capture initialization. We will be adding handlers for two events that we did not handle at all in the WingScan_withViewer sample.
  - Between handlerUrl and onUploadCompleted, add onScanStarted handler

    Original code looks like this:

```
try { Atalasoft.Controls.Capture.WebScanning.initialize({ handlerUrl:
'WebCaptureHandler.ashx', onUploadCompleted: function (eventName, eventObj) { ...
```

  - We will add just after the handlerUrl: 'WebCaptureHandler.ashx',

```
onScanStarted: function () { // When we start a new scan, we want to make this count
zeroed out so that we get an accurate count _scanCount = 0; },
```

  - After the new onScanStarted, add the onImageAquired handler

```
onImageAcquired: function (index, image) { // this just increments the count each
time an image is acquired.. // this is how we will know how many pages were scanned
so we can properly // insert later // NOTE THIS EVENT DOES NOT FIRE IF
discardBlankPages was true and the page scanned was discarded // this allows us to
have a correct count for number of pages scanned _scanCount++; },
```

- Modify the onUploadCompleted to do the actual append versus open

  Now, we need to modify the original onUploadCompleted event. We need to decide if we are appending or opening and then take the appropriate action
  - The original onUploadCompleted() will look like this:

```
onUploadCompleted: function (eventName, eventObj) { if (eventObj.success) {
_thumbs.OpenUrl("atala-capture-upload/" + eventObj.documentFilename);
Atalasoft.Controls.Capture.CaptureService.documentFilename =
eventObj.documentFilename; } },
```

  - We need to replace it with this:

```
onUploadCompleted: function (eventName, eventObj) { // we only do this if the upload
completed successfully if (eventObj.success) { // You will need to adjust this to
point to the actual location where // the files are being uploaded/scanned into //
atala-capture-upload is simply the component's default location var
folderWhereFilesUploadTo = "/atala-capture-upload/"; // we need to know the page
count so that we can avoid trying to append (add) when // there are no pages..
attempting the _thumbs.document.insertPages var pageCount =
_thumbs.getDocumentInfo().count; // we can not insert if there are no pages existing
so we will check // and switch to "new scan" if there aren't pages if (_append &&
pageCount > 0) { var destIndex = pageCount var srcIndices =
makeSrcArray(_scanCount); //alert("Inserting " + _scanCount + " pages at index " +
destIndex); _thumbs.document.insertPages(folderWhereFilesUploadTo +
eventObj.documentFilename,

srcIndices,

destIndex); } else { _thumbs.OpenUrl(folderWhereFilesUploadTo +
eventObj.documentFilename); } } },
```

- Finally modify/replace the original scan button with 2 buttons - one to scan and open and one to scan with append
  - in the Default.aspx page, find the original HTML for the scan button. It will look like this

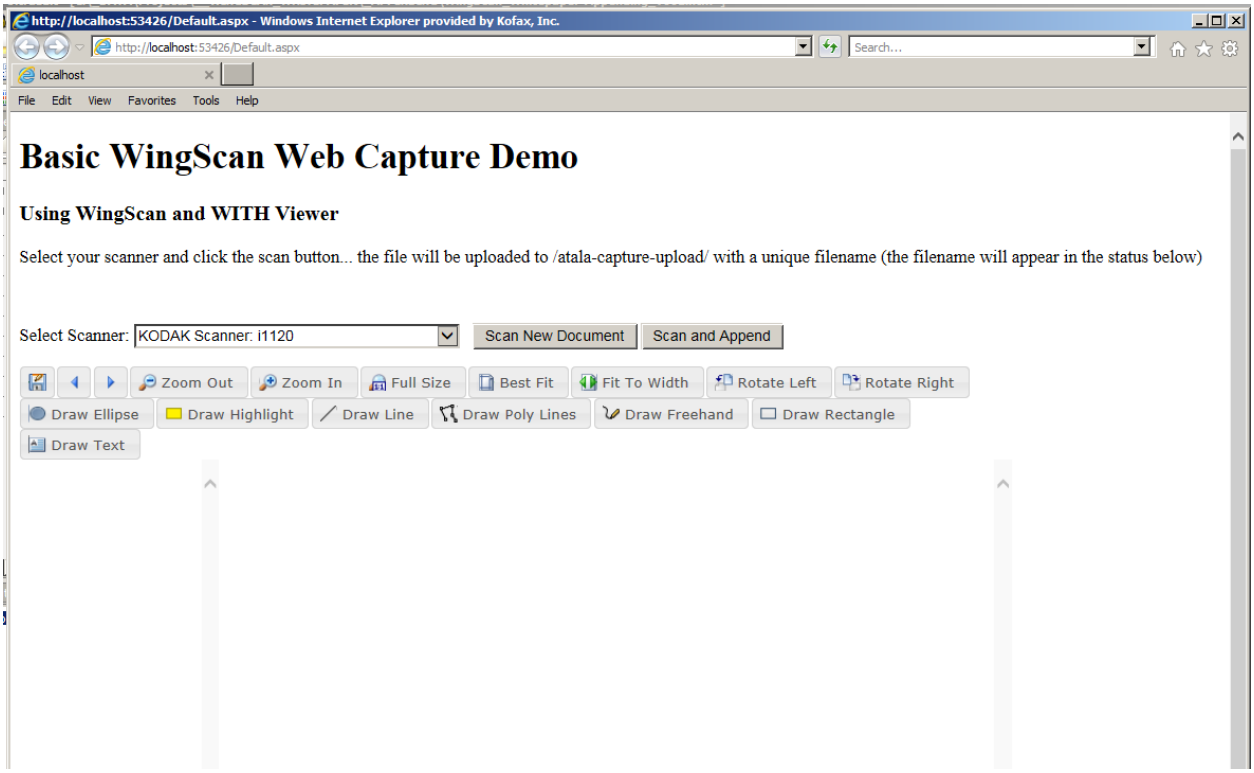
```
<p>Select Scanner: <select class="atala-scanner-list" disabled="disabled"
name="scannerList" style="width:22em"> <option selected="selected">(no scanners
available)</option> </select>   <!-- DO NOT USE <input type="button"
class="atala-scan-button" value="Scan" /> --> <input type="button" id="scanNOW"
value="Scan" onclick="scanWithSelectedScanner(); return false;" /> </p>
```

  - Replace it with this:

```
<p>Select Scanner: <select class="atala-scanner-list" disabled="disabled"
name="scannerList" style="width:22em"> <option selected="selected">(no scanners
available)</option> </select>   <!-- DO NOT USE <input type="button"
class="atala-scan-button" value="Scan" /> --> <input type="button" id="scan_new"
```

```
value="Scan New Document" onclick="scanWithSelectedScanner(false); return false;" />
<input type="button" id="scan_append" value="Scan and Append"
onclick="scanWithSelectedScanner(true); return false;" /> </p>
```

- That's it - you're done.. go ahead and run the solution. It should look like this:



- If you start with a blank viewer, then both buttons will simply scan and open what was scanned. However, if you then go to scan another document.. using the Scan New Document will REPLACE the existing one where Scan and Append will append to the existing document.

SAMPLE APP DOWNLOADS

It is our hope that you've been following along with the tutorial and have successfully built your solution. However, if you've run into issues or if you want a working reference app, we've implemented BasicWebCapture_withViewer (staring app) and WingScan_withViewer_append (final result) for you to download and run if needed. They also make great test harnesses for any experimental WingScan code you want to try out... letting you start from a known-working application.

- BasicWebCapture_withViewer.zip (11.5.0.10)

  The original sample app we will be modifying in today's lesson.
- WingScan_withViewer_append.zip (11.5.0.10)

  The final result after following this tutorial (minor adjustments made to project name and non-functional wording)

CONCLUSION

It is possible to seriously customize the viewer and the saving, and it is possible to perform many other useful tasks with WingScan and with WDV/WDT. This paper focused specifically on how to enable appending to the existing current document in the WebDocumentViewer / WebDocumentThumbnailer while scanning. Please see our growing list of whitepapers for more use cases / examples / tutorials.

Original Article

Q10470 - INFO: WingScan Whitepaper - Appending Additional Scans

Last Update

2026-11-11 - TD - (Moved Images)

Atalasoft Knowledge Base
https://www.atalasoft.com/kb2/KB/50034/INFO-WingScan-Whitepaper-Appending-A...