Changes Introduced != "release notes"

Please note, this document is intended to "go beyond" the basic release notes and give more details where needed. Such as Release notes may say "new feature - added single page view to WDV" where the Changes Introduced will tell you more useful information such as "to enable this feature you need to set the singlepage config option to true here..."

For basic Release notes, please see

INFO: 11.5 Full Release Notes

Issues / Fixes in FixPacks

Updates in 11.5.0.9

Fix for: 957298 [WDV] Opacity for Stamp annotations was not applied correctly when burning annotations while saving the file.

In order to fix this, you need to use the newly added opacity property in AnnotationTextConfig.

Don't forget, if you're updating the annotation programmatically, you will need to also call .update() on the annotation object to apply the change

Fix For955483 [WCS] Added an option that enables the WCS Import window to restrict importing to specific file types.

Added importFileTypes to scanningOptions.

955469 [WCS] Added the ability to import images as files.

Added onImageFileAcquired event for WebScanning namespace.

NON-BREAKING CHANGE - 11.5.0.5 removes REQUIREMENT For PdfMultiprocessing in WDV

Previous to 11.5.0.5, any time you referenced the WebControls components with WDV, we were labeling the Pdf Multiprocessing (and all of its myriad of dependencies as required) this was causing a lot of headaches for folks who were not intending to use that advanced feature.

11.5.0.5 disconnected that requirement from WDV / Atalasoft.dotImage.WebControls unless you explicitly enable the multiprocessing feature.

BREAKING CHANGE 11.5.0.5 Now Requires jQueryUI-1.14.0

11.5.0.5.230 WebDocViewer Resources now depend on JqueryUI-1.14.0

Updated JqueryUI version

11.5.0.5 and newer use JqueryUI-1.14.0

This means the proper includes for 11.5.0.5 and up are

```
!-- Js scripts in correct order -->

<script src="WebDocViewer/jquery-3.5.1.min.js" type="text/javascript"></script> <script
src="WebDocViewer/jquery-ui-1.14.0.min.js" type="text/javascript"></script> <script
src="WebDocViewer/raphael-min.js" type="text/javascript"></script> <script
src="WebDocViewer/clipboard.min.js" type="text/javascript"></script> <script
src="WebDocViewer/atalaWebDocumentViewer.js" type="text/javascript"></script>

<!-- CSS files in correct order --> <link href="WebDocViewer/jquery-ui-1.14.0.min.css"
rel="Stylesheet" type="text/css" /> <link href="WebDocViewer/atalaWebDocumentViewer.css"
rel="Stylesheet" type="text/css" />
```

KNOWN ISSUE: WDV Annotations from older versions loading as ReadOnly in 11.5

The readonly property of annotations was added to WebDocumentViewer in a previous

release, thus annotations saved to XMP before that feature do not have the setting present.

There is a known issue in DotImage 11.5 (but not in 11.4) where XMP read in from those old versions which is missing the readonly tag are defaulting to True instead of False.

FIX

This issue was addressed in 11.5.0.3 fixpack and beyond. We recommend that you upgrade to the latest version at your earliest convenience.

WORKAROUND

In the meantime, the workaround is that after you finish loading, you can iterate through all annotations on all pages and ensure the readonly flag is false.

Example Code:

```
/// The following code requires that you pass in the viewer object you're using //// example usage assumes your WebDocumentViewer is named "_viewer" //// Example // ToggleAllAnnosReadOnly(_viewer, false); // This code loops through all pages and calls the ToggleReadOnlyOnPage with the // passed in readOnlyValue (defaults to false if null) // the ToggleReadOnlyOnPage will do the actual hevy lifting for each page's annos function ToggleAllAnnosReadonly(wdv, readOnlyValue) { if (readOnlyValue == null) { readOnlyValue = false; } var pageCount = wdv.getDocumentInfo().count; if (pageCount != null && pageCount > 0) { for (var i = 0; i < pageCount; i++) { ToggleReadonlyOnPage(wdv, i, readOnlyValue); } } } // this is to set the readonly flag of all annos on a specific page to the // readOnlyValue provided function ToggleReadonlyOnPage(wdv, pageIndex, readOnlyValue) { var annos = wdv.getAnnotationsFromPage(pageIndex); if (annos != null) { for (var i = 0; i < annos.length; i++) { var anno = annos[i]; anno.readonly = readOnlyValue; anno.update(); } } }
```

The fix was provided in 11.5.0.3 in October 2024

Added LimitsPageCount to EmlDecoder, HtmlDecoder & TxtDecoder

In 11.5.0.2 and forward, there is now a AtalasoftConfigSection property called LimitsPageCount which you can use to increase the default page count limit.

NOTE: the page counts are limited for memory/performance reasons. Adjusting this value in production should only be done after testing thoroughly in your app/deployment to ensure it does not cause memory issues

These are not "Properties" of the decoders - so you won't find it under EmlDecoder.LimitsPageCount in the typical place. These need to be configured in a special config section in your app.config

Example AppConfig (showing the defaults which are 3000 pages)

```
?xml version="1.0" encoding="utf-8" ?>
<configuration>
   <configSections>
      <section name="Atalasoft"
                type="Atalasoft.Shared.AtalasoftConfigSection, Atalasoft.Shared"/>
   </configSections>
   <Atalasoft>
      <EmlDecoder LimitsPageCount="3000"/>
      <HtmlDecoder LimitsPageCount="3000"/>
      <TxtDecoder LimitsPageCount="3000"/>
    </Atalasoft>
</configuration>
```

Missing jquery-ui-1.13.1.min.css in NuGet distribution

The initial 11.5.0.156 release is missing a css file

The file in question is jquery-ui-1.13.1.min.css

It should have been installed as part of Atalasoft.Web.Document.Viewer package. but somehow got missed

We're working on ensuring this gets included in the 11.5.0.1 fixpack release

In the meantime, you can either get it from installing the SDK from the main download site, where it can be found in

C:\Program Files (x86)\Atalasoft\DotImage 11.5\bin\WebResources\WebDocViewer

or you can grab a copy here : Download
DotImage_Clientside_Web_Resources_v11.5.0.0.zip

NEW Features

PdfDecoder Multiprocessing

Overview

The PDF viewer has been upgraded with multiprocessing to support concurrent rendering of pages, aimed at reducing delays with complex and large pages. Performance impact may be affected as follows:

- Documents with vector content will experience performance improvements.
- High page-count documents with rasterized content may experience performance improvements.
- High page-count documents with vector content, especially if the vector quantity per page is small, may not show substantial performance improvement.
- Some PDF processing use case types may experience a degradation in performance as a result of parallel orchestration and system resource constraints. These use case types include the following:
    - Using the multiprocessing feature for single page PDF documents.
    - Using the multiprocessing feature for PDF documents without any graphics.
    - Using the multiprocessing feature in one-core CPU environment

How To: Out of the Box

If you're using a default WebDocumentRequestHandler (not custom handling), all you need to do is set the WebDocumentViewer (and WebDoucmentThumbnailer if you are using it) config multiprocessing setting to true)

multiprocessing: true

```
et _viewer = new Atalasoft.Controls.WebDocumentViewer({ parent: $('#_containerViewer'),
toolbarparent: $('#_toolbar1'), serverurl: _serverUrl, allowannotations: true,
multiprocessing: true, forcepagefit: true, fitting: Atalasoft.Utils.Fitting.Width, }); let
_thumbs = new Atalasoft.Controls.WebDocumentThumbnailer({ viewer: _viewer, parent:
$('#_containerThumbs'), serverurl: _serverUrl, documenturl: _docUrl, allowannotations: true,
multiprocessing: true });
```

For full information please see HOWTO: Enable PdfMultiprocessing In WDV (new in 11.5)

NON-BREAKING Changes

These are important changes but ones that don't directly require code changes or other large rewrites

New Decoders

We've added several new decoders in DotImage 11.5.

CommonDecoders

The new "common Decoders" (non MS_Office) do not require additional licensing outside of DotImage Document Imaging, they are

- **WebpDecoder**

  .webp files: Decoding for newer web image formats.
- **TxtDecoder**

  For text and other types of files

  Can be used for any plain text file including xml

  NOTE there is no "pretty printing" / formatting of code. If you wish to have source code or XML, YAML etc properly indented, you'll want to run such files through the relevant pretty printing/lint utilities before passing them to TxtDecoder
- **HtmlDecoder**

  .html and .txt files: Expanded web and text file decoding.

NOTE: HtmlDecoder has some nuances to it that require a deeper explanation. Please see: INFO: HtmlDecoder Deep Dive

- **EmlDecoder**

  .eml files: Enhanced email decoding capabilities.

In order to implement them you must follow these steps:

If you're using NuGet packages, everything you need should be in the

Atalasoft.dotImage.CommonDecoders.x64

or

Atalasoft.dotImage.CommonDecoders.x86

package depending on the "bitness" you need

If you're adding references manually from the installed SDK:

Add a reference to Atalasoft.dotImage.CommonDecoders.dll to your project (NOT available for .NET Framework 3.5)

Add the Perceptive Filters (64 or 32 bit depending on your platform target for your app) to your project as ALWAYS COPY or COPY IF NEWER or place them directly in the bin directory of the app

or if you installed the SDK the files are:

```
SYS11df.dl ISYSreaders.dll ISYSreadershd.dll Perceptive.DocumentFilters.dll
```

They can be found in

C:\Program Files (x86)\Atalasoft\DotImage 11.5\bin\PerceptiveDocumentFilters\intel-32\

or

C:\Program Files (x86)\Atalasoft\DotImage 11.5\bin\PerceptiveDocumentFilters\intel-64\

depending on which bitness you're targeting

Finally, Add the relevant decoder(s) to the RegisteredDecoders.Decoders collection in a STATIC CONSTRUCTOR for your class/program (do this once and only once)

```
egisteredDecoders.Decoders.Add(new TxtDecoder() { Resolution = 200 });
RegisteredDecoders.Decoders.Add(new HtmlDecoder(){ Resolution = 200 });
RegisteredDecoders.Decoders.Add(new EmlDecoder() { Resolution = 200 });
RegisteredDecoders.Decoders.Add(new WebpDecoder());
```

Documentation / Help Files

In previous versions, the ApiReference.chm and DevelopersGuide.pdf were installed to a folder named help to:

C:\Program Files (x86)\Atalasoft\DotImage 11.x\Help\

These files are now provided as download links in the DotImage 11.5 start menu folder

Developers Guide

START->All Programs->Atalasoft DotImage 11.5->DevelopersGuide

Direct Download

.NET API Reference

START->All Programs->Atalasoft DotImage 11.5->ApiReference

Direct Download

WebDocumentViewer (clientside) API Reference

https://atalasoft.github.io/web-document-viewer/

WebCapture (Clientside) API Reference

https://atalasoft.github.io/web-capture-service/

WDV Setting e.FilePath in DocumentInfoRequested now Persists to other events

Previously, there was a bug where setting e.FIlePath in the DocumentInfoRequested event did not persist the chagned file name to ImageRequested and other events

This has been fixed in 11.5.

This is very uesful for situations where you might for instance call openUrl from the client and the request fetches a file from another location and you want to update the viewer to point to a copy of the file you've grabbed (cached)... say the user did openurl(Document1234) where Document1234 was the ID of a document to be fetched from a remote server

You could write code to fetch Document1234 and place it in say

/images/Document1234_someRandomNumber

Then set e.FilePath to "/images/Document1234_someRandomNumber"

With no further handling, now the viewer will now look for Document1234_someRandomNumber at that path instead of for just Document1234 as the original open

This can be incredibly useful too for simple "cache breaking"

You request Document1234 and it's local but you copy it to Document1234_SomeGUID and set that file path to the new name - thus breaking any caching done by file name by server or browser

BREAKING Changes

DLL Restructuring

INFO: Changes Introduced in DotImage 11.5

In 11.5 you may notice some DLLs are "missing" versus previous versions. This is by design.

We distribute our SDK in sets based on Framework

our dlls for targeting .NET Framework 3.5 are in

C:\Program Files (x86)\Atalasoft\DotImage 11.5\bin\3.5

Our DLLs for targeting .NET Framework 4.6.2 through 4.8.x are in

C:\Program Files (x86)\Atalasoft\DotImage 11.5\bin\4.6.2

Our DLLs for .NET 6 and p

C:\Program Files (x86)\Atalasoft\DotImage 11.5\bin\6.0

In older dotImage we had 3.5, 4.5.2, and 5.0 respectively

.NET Framework Minimum

The DLL restructure has a specific impact on the .NET Framework minimum, Any project you have that targets .NET Framework < 4.6.2 must be updated to .NET Framework 4.6.2 or later (but NOT .NET 5 or higher - stay with the 4.6.2-4.8.x (4.9 maybe someday?) for .NET Framework. .NET 5 and up is a totally different beast - see below

.NET 6+ Minimum

For those using our .NET 5 dlls previously, we have dropped .NET 5 and you must target .NET 6 or up (we support .NET 6, .NET 7, and .NET 8 currently - we will drop 6 and 7 in 11.6)

Please see FAQ: Support for ASP.NET Core / .NET Core / .NET 5 / .NET 6 /.NET 7 / .NET 8

WebDocumentRequestHandler - Hard Dependencies

WebDocumentRequestHandler (in support of our new CommonDecoders and Multithreading) has 2 hard dependencies at runtime. (Your solution will build but you'll get a crash if the

following have not been installed:

Microsoft.CodeAnalysis.CSharp 4.5.0

Please note our dependency is on the very specific 4.5.0 version, and NOT on the latest - you must install the specific 4.5.0 version (for now, we're working on possibly being able to set it to 4.5.0 minimum in future)

NOTE That if you install the NuGet package Atalasoft.dotImage.PdfReader.Multithreading.x64 or Atalasoft.dotImage.PdfReader.Multithreading.x86, this should get installed for you automatically

We have also noticed that IISExpress (and possibly IIS) caches the install so once it's been installed once, you may not find that it complains)

The symptom of this dependency being missing will not show up at compile - but at runtime the WebDocumentRequestHandler will throw an exception and if you look at the network trace and error text is on it will mention that it's missing Microsoft.CodeAnalysis.CSharp 4.5.0 specifically

Perceptive Filter Dlls

These are supposed to be available in a NuGet package, but the PerceptiveFilter packages currently available are not correct. This is being investgated

For now, you'll need to grab them from the SDK (you would need to download and install it. The files are:

```
SYS11df.dl ISYSreaders.dll ISYSreadershd.dll Perceptive.DocumentFilters.dll
```

They can be found in

C:\Program Files (x86)\Atalasoft\DotImage 11.5\bin\PerceptiveDocumentFilters\intel-32\

or

C:\Program Files (x86)\Atalasoft\DotImage 11.5\bin\PerceptiveDocumentFilters\intel-64\

depending on which bitness you're targeting

Alternately, for your convenience you can download them here:

PerceptiveDocumentFilters_11.5.0.0.zip

Adding the Perceptive Filters

Once we get a NuGet package, you can just add that.

However, the NuGet package DLLs can not be referenced, so you can't just add them as references. Instead, you need to add the DLLs to your project then set them to "COPY ALWAYS" or "COPY IF NEWER"

Dropped Support

Windows Server 2012 R2 and older

Microsoft has end of lifed Windows server 2012 R2 and older - minimum server OS is now Windows Server 2016 and up

Windows 8.1 and Older

Microsoft has end of lifed Windows 8.1 and older - minimum desktop OS is now Windows 10 and up

MacOS Catalina and older dropped

This applies to the Web Capture client-side component. The installable MacOS package that clients using pages with WebCapture need to install. The package no longer supports MacOS Catalina and older.