Q10471 - INFO: Changes Introduced in DotImage 11.1

Releases

Release Notes

The full official release notes for DotImage can be found on our Release Notes Page

For a more search-friendly version please see:

INFO: 11.1 Full Release Notes

Individual releases

Initial Release (December 15, 2018) - v11.1.0.0 (Build 321)

Fix Pack 1 (January 23, 2019) - v11.1.0.1 (Build 340)

Fix Pack 2 (February 28, 2019) - v11.1.0.2 (Build 360)

Fix Pack 3 (April 16, 2019) - v11.1.0.3 (Build 377)

Fix Pack 4 (May 29, 2019) - v11.1.0.4 (Build 395)

Fix Pack 5 (July 16, 2019) - v11.1.0.5 (Build 405)

Fix Pack 6 (August 29, 2019) - v11.1.0.6 (Build 427)

Fix Pack 7 (October 16, 2019) - v11.1.0.7 (Build 438)

Please see the 11.1 Full release notes KB for a singe page (easily searchable with CTRL+F) summary.

Updates Introduced in Fixpacks

INFO: Changes Introduced in DotImage 11.1

11.1.0.4 - Fix Pack 4

PdfDecoder

Enable property RenderSettings.ColorSettings.BackColor

Addresses issue [PdfDecoder] Request to add support for transparent backgrounds

PdfDecoder now implements PdfDecoder.RenderSettings.ColorSettings.BackColor property which allows a developer to tell the PdfDecoder what color to use as the background. This is useful if you need to render an image with a transparent background

```
dfDecoder pdfDec = new PdfDecoder() { Resolution = 200, RenderSettings = new RenderSettings()
{ ColorSettings = { BackColor = Color.Transparent } } });
```

Now, add pdfDec to RegisteredDecoders.Decoders collection in a static constructor, so you can use it with a FileSystemImageSource:

```
sing (FileSystemImageSource fsis = new FileSystemImageSource("c:\\path\\to\\source.pdf",
true)) { using (FileStream outStream = new FileStream(c:\\Path\\to\\out.tif",
FileMode.Create)) { TiffEncoder enc = new TiffEncoder(); enc.Save(outStream, fsis, null); } }
```

or use it directly with

```
talaImage img = pdfDec.Read(streamContainingPdf, frameIndex, null);
```

The rendering is also available in Document class as well, but requires more work (example for splitting PDF into single page transparent pngs):

```
nt resolutionForDecoding = 200; Dpi resolutionDpi = new Dpi(resolutionForDecoding,
resolutionForDecoding, ResolutionUnit.DotsPerInch); RenderSettings renderSettings = new
RenderSettings() { ColorSettings = { BackColor = Color.Transparent } }; using (FileStream fs
= new FileStream(c:\\path\\to\\source.pdf, FileMode.Open, FileAccess.Read, FileShare.Read)) {
using (Document doc = new Document(fs)) { doc.Resolution = resolutionDpi; for (int i = 0; i <
doc.Pages.Count; i++) { var imgWidth = (int)(doc.Pages[i].Width / 72.0 *
```

```
resolutionForDecoding); var imgHeight = (int)(doc.Pages[i].Height / 72.0 *
resolutionForDecoding); using (AtalaImage img = new AtalaImage(imgWidth, imgHeight,
PixelFormat.Pixel32bppBgra)) { img.Resolution = resolutionDpi;
doc.Pages[i].Draw(img.GetGraphics(), renderSettings); img.Save("c:\\Path\\to\\out_"
+i.ToString() + ".png", new PngEncoder(), null); } } } }
```

11.1.0.4 - Fix Pack 4

Abbyy Engine

New property RetainLayout

Feature implements fix for [ABBYY] Formatting loss on RTF output files

AbbyyEngine has a new property: AbbyyEngine.RetainLayout which can be set to true to tell Abbyy to attempt to retail the original layout when converting to rtf, docx, odt, txt, csv, html, and pptx using the AbbyyEngine supported translators.

The default value is false so that we do not introduce breaking changes

Simply set your **AbbyyEngine.RetainLayout = true;** to enable this new feature

New Property AbbyyEngine.PredefinedProfile

This new property was added to address [ABBYY] OCR results on ID card are poor

It allows developers finer control over how AbbyyEngine handles certain situations

Member Name	Description
Default	Sets all the processing parameters to the default values.
DocumentConversion_Accuracy	Suitable for converting documents into an editable format (e.g. RTF, DOCX). The settings have been optimized for accuracy: Best quality. Enables font style

	detection and full synthesis of the logical structure of a document.
DocumentConversion_Speed	Suitable for converting documents into an editable format (e.g. RTF, DOCX). The settings have been optimized for processing speed: Best quality. Enables font style detection and full synthesis of the logical structure of a document. The processes of document analysis and recognition are faster.
DocumentArchiving_Accuracy	Suitable for creating an electronic archive (converting to PDF, PDF/A, PDF and PDF/A with MRC). The settings have been optimized for accuracy: Enables detection of maximum text on an image, including text embedded into the image. Skew correction is not performed. Fonts and styles are not detected. Full synthesis of the logical structure of a document is not performed. **IMPORTANT:** The profile is not intended for converting a document into RTF, DOCX, PDF text only. Use the document conversion profiles for such purpose.
DocumentArchiving_Speed	Suitable for creating an electronic archive (converting to PDF, PDF/A, PDF and PDF/A with MRC). The settings have been optimized for processing speed: Enables detection of maximum text on an image, including text embedded into the image. Skew correction is not performed. Fonts and styles are not detected. Full synthesis of the logical

	structure of a document is not performed. The processes of document analysis and recognition are faster. **IMPORTANT:** The profile is not intended for converting a document into RTF, DOCX, PDF text only. Use the document conversion profiles for such purpose.
BookArchiving_Accuracy	Suitable for creating an electronic library (converting to PDF, PDF/A, PDF and PDF/A with MRC). The settings have been optimized for accuracy: Best quality. Enables font style detection and full synthesis of the logical structure of a document.
BookArchiving_Speed	Suitable for creating an electronic library (converting to PDF, PDF/A, PDF and PDF/A with MRC). The settings have been optimized for processing speed: Best quality. Enables font style detection and full synthesis of the logical structure of a document. The processes of document analysis and recognition are faster.
TextExtraction_Accuracy	Suitable for extracting text from a document. The settings have been optimized for accuracy: Enables detection of all text on an image, including small text areas of low quality (pictures and tables are not detected). Fonts and styles are not detected. Full synthesis of the logical structure of a document is not performed.

	IMPORTANT: The profile is not intended for converting a document into RTF, DOCX, PDF text only. Use the document conversion profiles for such purpose.
TextExtraction_Speed	Suitable for extracting text from a document. The settings have been optimized for processing speed: Enables detection of all text on an image, including small text areas of low quality (pictures and tables are not detected). Fonts and styles are not detected. Full synthesis of the logical structure of a document is not performed. The processes of document analysis and recognition are faster. **IMPORTANT:** The profile is not intended for converting a document into RTF, DOCX, PDF text only. Use the document conversion profiles for such purpose.
FieldLevelRecognition	Suitable for recognizing short text fragments. Currently this profile has default settings.
HighCompressedImageOnlyPdf	Suitable for creating high-compressed PDF files which contain entire documents saved as pictures. The following settings are used: Document recognition and synthesis of the logical structure of a document are not performed. Skew correction is not performed. PDF export is optimized for the minimum size of the resulting file. The entire document is saved as a picture (PEM_ImageOnly mode).

EngineeringDrawingsProcessing	Suitable for recognizing technical drawings. It takes into account large size and complexity of engineering diagrams, as well as possibility of different text orientation within the image. The profile is intended for converting such images into searchable PDF format. The following settings are used: Enables detection of all text on an image, including text blocks of vertical orientation. Full synthesis of the logical structure of a document is not performed.
Version9Compatibility	Provided for compatibility, sets the processing parameters to the default values of ABBYY FineReader Engine 9.0.

Breaking Changes

These are changes that may directly require code changes and/or will change default behaviors. Atalasoft strives to avoid breaking changes to the greatest degree possible, however, these changes were deemed necessary.

WebDocumentViewer DocumentSave File Extension Preservation

In all versions before 11.0, the DocumentSave (as well as DocumentStreamWritten and AnnotationStreamWritten) save events have provided the e.FileName as the original file name but with the extension stripped off. This led to a good deal of extra code having to be written to determine the proper file type to add an extension back in

In 11.1, this incorrect behavior is corrected. This will present as a BREAKING CHANGE to users who have WDV save events handled with any custom code that needed to compensate for the missing file extension.

example: Original file opened in WDV was "GettysburgAddress.tif"

11.0 and older behavior: DocumentSave event e.FileName in the handler would receive "GettysburgAddress" for the filename.. same with e.Filename in the AnnotationStreamWritten and DocumentStreamWritten events

in 11.1 and newer the e.FileName will be "GettysburgAddress.tif" so that combining the e.SaveFolder and e.FileName will provide the web relative path to the file without having to guess or append .tif

WebDocumentViewer Resource Changes - v11.1.0.0 and newer

WebDocumentViewer resources

We needed to update to newer jQuery and jQueryUI versions for new feature support. The versions shipped with any DotImage represent the minimum that we support. You may use newer versions, but remember to test for compatibility. The version we ship with is fully tested. If you run into an issue with a newer version not working, please contact support. (Make sure you let us know specifically that you're using a newer version than shipped with DotImage).

Updates to jQuery and jQueryUI Versions	
Old File (11.0)	**New File (11.1)**
jquery-1.11.0.min.js	jquery-3.3.1.min.js
jquery-ui-1.10.4.min.css	jquery-ui-1.12.1.min.css
jquery-ui-1.10.4.min.js	jquery-ui-1.12.1.min.js
jquery.easing.1.3.js	removed
Updates to WebDocViewer\Images	
Old File (11.0)	**New File (11.1)**
images\atala-ui-icons-16.png	UPDATED
images\atala-ui-spinner.gif	UPDATED
NOT PRESENT	atala-ui-dragdropfile.png
NOT PRESENT	ui-icons_444444_256x240.png
NOT PRESENT	ui-icons_555555_256x240.png

INFO: Changes Introduced in DotImage 11.1

NOT PRESENT	ui-icons_777620_256x240.png
NOT PRESENT	ui-icons_777777_256x240.png
NOT PRESENT	ui-icons_cc0000_256x240.png
NOT PRESENT	ui-icons_ffffff_256x240.png
images\ui-bg_flat_0_aaaaaa_40x100.png	REMOVED
images\ui-bg_flat_75_ffffff_40x100.png	REMOVED
images\ui-bg_glass_55_fbf9ee_1x400.png	REMOVED
images\ui-bg_glass_65_ffffff_1x400.png	REMOVED
images\ui-bg_glass_75_dadada_1x400.png	REMOVED
images\ui-bg_glass_75_e6e6e6_1x400.png	REMOVED
images\ui-bg_glass_95_fef1ec_1x400.png	REMOVED
images\ui-bg_highlight-soft_75_cccccc_1x100.png	REMOVED
images\ui-icons_222222_256x240.png	REMOVED
images\ui-icons_2e83ff_256x240.png	REMOVED
images\ui-icons_454545_256x240.png	REMOVED
images\ui-icons_888888_256x240.png	REMOVED
images\uiicons_cd0a0a_256x240.png	REMOVED

This means you will need to update your script references from

```
!-- Script Includes for Web Viewing --> <script src="WebDocViewer/jquery-1.11.0.min.js"
type="text/javascript"></script> <script src="WebDocViewer/jquery.easing.1.3.js"
type="text/javascript"></script> <script src="WebDocViewer/jquery-ui-1.10.4.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> <!-- Style for Web Viewing --> <link
href="WebDocViewer/jquery-ui-1.10.4.min.css" rel="Stylesheet" type="text/css" /> <link
href="WebDocViewer/atalaWebDocumentViewer.css" rel="Stylesheet" type="text/css" />
```

to:

```
!-- Script Includes for Web Viewing --> <script src="WebDocViewer/jquery-3.3.1.min.js"
type="text/javascript"></script> <script src="WebDocViewer/jquery-ui-1.12.1.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> <!-- Style for Web Viewing --> <link
href="WebDocViewer/jquery-ui-1.12.1.min.css" rel="Stylesheet" type="text/css" /> <link
href="WebDocViewer/atalaWebDocumentViewer.css" rel="Stylesheet" type="text/css" />
```

NOTE:

We've had reports that after upgrading, some toolbar buttons/icons are not behaving as expected. Cached versions of the old images and such. The new WebCapture resources shipped with new icons, so existing demos/apps may need to have their IIS Express cache cleared

SYMPTOM

You've updated a solution that used WDV 11.0 or older to 11.1 and made certain that you removed the old WebDocViewer and WebDocViewer\images folders and replaced them with the new ones and have updated your scripts and css as above... but when you run the solution the toolbar icons are misaligned and/or showing button text incorrectly.

FIX:

- Shut down all copies of Visual Studio
- Run a Visual Studio command prompt
  - Start->All Programs->Visual Studio 2013->Developer Command Prompt for VS2013
  - cd "C:\Program Files (x86)\IIS Express\"
  - appcmd.exe list site /xml | appcmd delete site /in
- This will delete the cached sites
- Now, re-open the solution, rebuild and run
- You may still see the cached images.. hold SHIFT and hit F5 to force a refresh
- The stale icons should update

New Features

INFO: Changes Introduced in DotImage 11.1

These are features new in 11.1

New in WebDocumentViewer / WebDocumentThumbnailer

WebDocumentViewer File Upload

Starting in 11.1 we are proud to provide the much-requested ability to upload files from local machine to server through our WebDocumentViewer.

IMPORTANT:

If you are not planning on actively using this feature please take steps to fully disable it on the server side:

HOWTO: Completely Disable Upload Feature in WebDocumentViewer

The File Upload feature can use drag/drop or programmatic calls to initiate.

Enabling/Configuring Upload

You need to add the new upload configuration map to your viewer Config

example allowing uploads to the directory ./upload allowing only .jpg,.raw and tiff files, restricting size to 10MiB, allowing multiple files and enabling drag drop

```
pload: { uploadpath: 'upload', allowedfiletypes: '.jpg,.raw,image/tiff', allowedmaxfilesize: 10 * 1024 * 1024, // this works out to 10MB allowmultiplefiles: true, allowdragdrop: true }
```

Upload via Drag/Drop

If the **allowdragdrop** config is **true**, then users can just drag/drop appropriate files onto the viewer to initiate upload

Programmatic upload

INFO: Changes Introduced in DotImage 11.1

Whether allowdragdrop is true or not upload (if configured) can be initiated programmatically

Call file upload directly using new method :

```
viewer.uploadFiles(files, uploadpath, callback);
```

PLEASE SEE our Web Viewing Demo for a fully functional sample solution for the new upload feature

Also, please see HOWTO: Properly and Securely Enable File Upload Feature in WebDocumentViewer

Example:

```
form> <input type="file" name="fileUpload"/> <button type="button" onclick="uploadFile();
return false;">Upload!</button> </form> <script type="text/javascript" language="javascript">
function uploadFile() { var files = []; for (var i = 0; i <
document.getElementsByName('fileUpload')[0].files.length; i++) {
files.push(document.getElementsByName('fileUpload')[0].files[i]); }
_viewer.uploadFiles(files, guid()); return false; } </script>
```

File Upload Events

Several file-upload-related events have been provided

- **fileaddedtoupload** – *event*, is fired when file is added to the list of files to upload.
- **uploadstarted** – *event*, is fired when upload operation is started.
- **uploadfinished** – *event*, is fired when upload operation is finished.
- **fileuploadstarted** – *event*, is fired when file upload is started.
- **fileuploading** – *event*, is fired during file upload process. Can be used to track upload progress.
- **fileuploaderror** – *event*, is fired when file upload has failed/
- **Fileuploadfinished** – *event*, is fired when file upload is finished/

Example:

```
ploadfinished: function (eventObj) { if (confirm('Your files were uploaded. Do you want to view it?')) { _thumbs.OpenUrl(lastUploadedFile); } }
```

File Upload Error Handling

When a file upload is rejected, it will return FileUploadRejectReason

Atalasoft.Utils.FileUploadRejectReason	
ReasonCode	**Description**
None	file is not rejected from upload.
Size	file size is bigger than allowed.
type	file type or extension is not allowed for upload.
Name	file with the same name is already added to upload queue

WebDocumentViewer Document Saved Event Updates

documentsaved – save operation updated to pass back the file name of the document that was saved. Allows developer to have a reference to the file for subsequent operations

```
viewer.bind('documentsaved', function (eventObj) { if (eventObj.success) { lastSavedFile = eventObj.fileName; alert('The file ' + lastSavedFile + ' was saved'); } });
```

WebDocumentViewer Customize Server Responses

In WebDocumentRequest handler customize responses back to the client via the existing events

Clientside	**WebDocumentRequestHandler (server side)**
documentinfochanged	DocumentInfoRequestResponseSend
annotationloaded	AnnotationsDataResponseSend

formsloaded	FormsDataResponseSend
pagetextloaded	PageTextRequestResponseSend
documentsaved	DocumentSaveResponseSend

Example for documentsaved event

Server-Side

```
ublic WebDocViewerHandler() { RegisteredDecoders.Decoders.Add(new PdfDecoder() { Resolution = 200 }); this.DocumentSaveResponseSend += MyWdv_DocumentSaveResponseSending; } private void MyWdv_DocumentSaveResponseSending(object sender, ResponseSendEventArgs e) { e.CustomResponseData.Add("MyMessage", "Hello World!"); }
```

Client-side

```
viewer.bind('documentsaved', function (eventObj) { if (eventObj.success) { myCustomMessage = eventObj.customData.MyMessage; alert('My custom message to you is ' + myCustomMessage); } });
```

Improved Text Search - Search on Page

Newly added ability to limit text searches to specific pages. (Old behavior was to search all pages).

Old Code (pre-11.1)

```
viewer.search("text to Search", startPageIndex, callbackHandler);
```

New Method to search(11.1 and newer)

```
viewer.searchOnPages("text to Search", startPageIndex, endPageIndex, activePageIndex, callbackHandler); // set activePageIndex to the startPageIndex (it's there for backward compatibility) // note that empty search text clears results
```

Example:

```
form name="searchform" id="searchform"> Search String - <input type="text" name =
"searchstring" id="searchstring" /> Starting Page - <input type="text" size="4" name =
"startingpage" id="startingpage" value="0" /> Ending Page <input type="text" size="4" name =
"endingpage" id="endingpage" value="100" /> <button type="button" onclick="search(); return
false'">Search</button> </form> <script type="text/javascript" language="javascript">
function search() { var searchstring = document.getElementById('searchstring').value; var
startingpage = document.getElementById('startingpage').value; var endingpage =
document.getElementById('endingpage').value; _viewer.text.searchOnPages(searchstring,
startingpage,endingpage,startingpage, function (it, match) { if (it.isValid()) {
_viewer.text.selectPageText(match.page, match.region, match.line, match.word); } }); }
</script>
```

WebDocumentViewer Error Handling for Missing Dependencies

Console messages are now provided if there are missing dependencies or dependencies with incorrect versions

for example, commenting out this reference

```
!-- <script src="WebDocViewer/jquery-ui-1.12.1.min.js" type="text/javascript"></script> -->
```

Would show an error in the console of

> The dependency verification for jQuery UI has failed. Dependency is not loaded or not found. The minimal required version is 1.12.1

http://www.atalasupport.net/downloads/v11.1/AtalasoftDotImageReleaseNotes_11.1.0_EN.pdf

WebDocumentViewer New annotations.scrollTo

Allows user to quickly scroll through all the annotations in a document

function scrollTo(annotation)

where annotation is an annotation object from the document

INFO: Changes Introduced in DotImage 11.1

usage overview

1. Iterate over the pages of a document
2. Populate an array of annotations on a page
3. var annos = viewer.getAnnotationsFromPage(I);
4. Scroll to the annotations viewer.annotations.scrollTo(annos[j]);

WebDocumentViewer Preserve File Extension

See note in Breaking Changes above as well

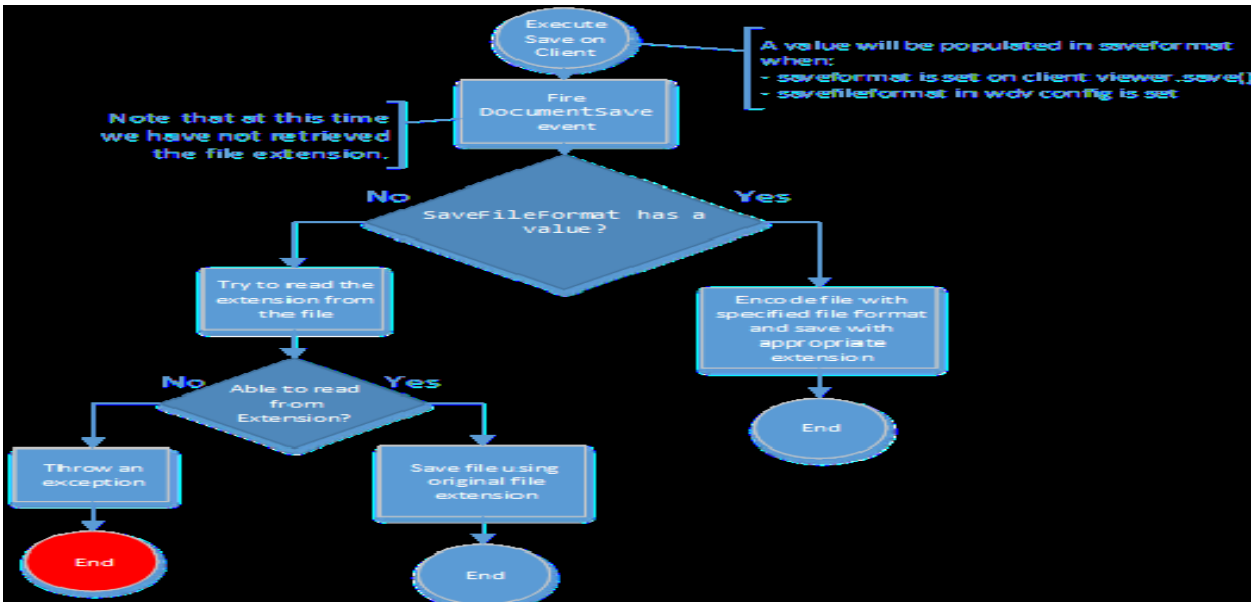
Prior to 11.1, the server-side DocumentSave event (DocumentSaveEventArgs) did not include the file extension

in 11.1, the file extension is included in the file name property

NOTE: if you have created your own workaround to persist or add file extension prior to 11.1, you may need to disable/back out of it for 11.1 and test thoroughly

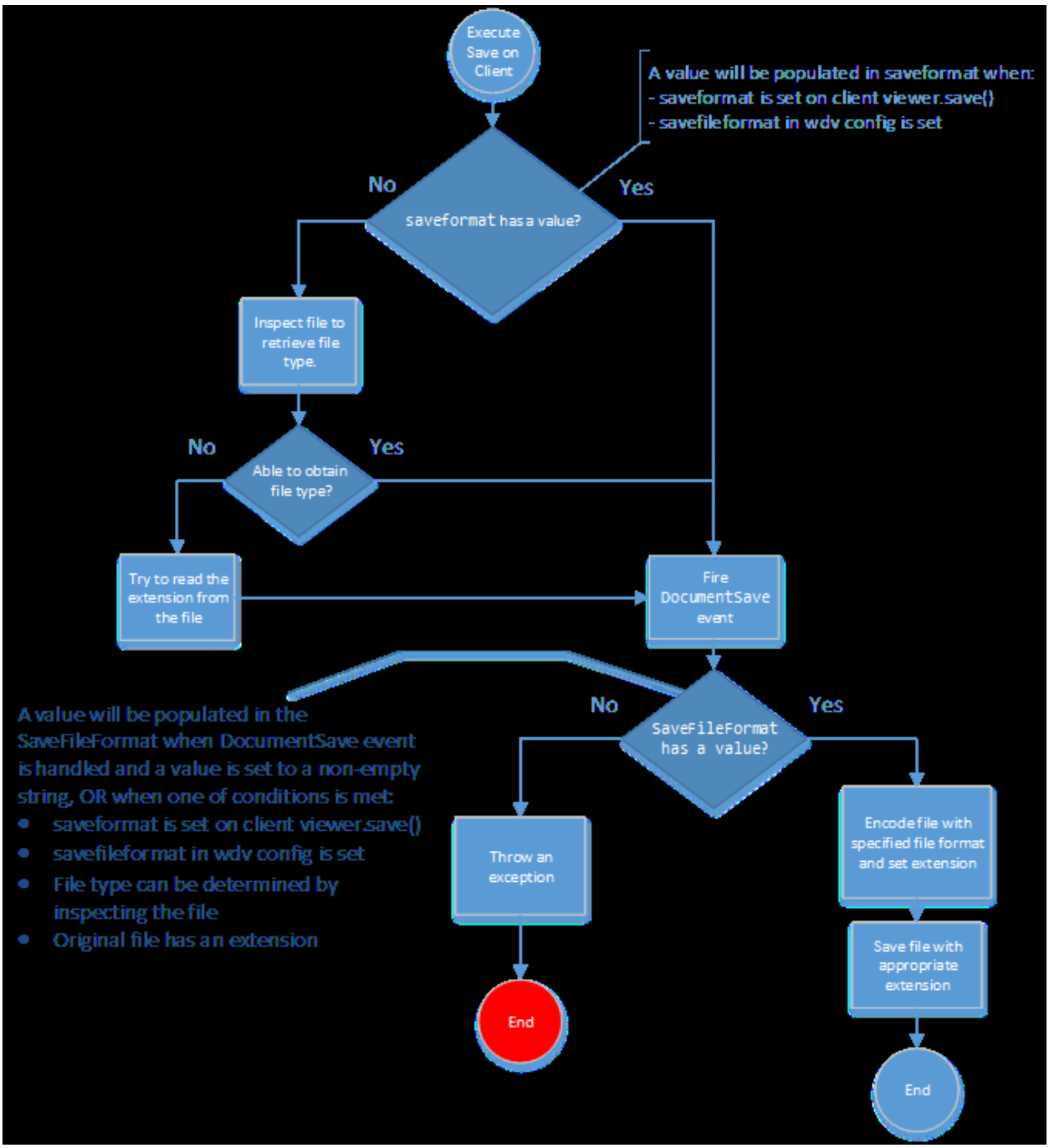
WebDocumentViewer Modifying Default Saving Behavior

Behavior in 11.0



INFO: Changes Introduced in DotImage 11.1

Behavior in 11.1+



11.1 has a new flag that can control the new behavior - ReplaceFileExtensionsOnSave

ReplaceFileExtensionsOnSave	
Setting	**Description**

ReplaceFileExtension.None	Preserves the original extension or lack thereof, in all cases. This applies even if there is a safefileformat or saveformat value
ReplaceFileExtension.FilesWithoutExtension	if no savefileformat / saveformat is set • files without extensions have extension added based on image type • all others preserve original extension if safefileformat / saveformat is set • Files with no extension are saved with extension matching the savefileformat value • other files preserve their original extension though they are still saved as the file type specified in savefileformat
ReplaceFileExtension.AllFiles	if no savefileformat / saveformat is set • saving a file that has no extension appends an extension based on the image type • saving a file with an extension preserves the original extension if a savefileformat / saveformat is set • file is saved with an extension matching the savefileformat value

WingScan

MacOS Scanning for WingScan

Instead of a local windows service, MacOS scanning will be installed via a .pkg

Kofax.WebCapture.macOS.pkg

INFO: Changes Introduced in DotImage 11.1

- Supported OS Versions
  - 10.12 (Sierra)
  - 10.13 (High Sierra)
  - 10.14 (Mojave)
- The MacOS module will be installed by Kofax.WebCapture.macOS.pkg
- One single license works for Windows and Mac (no clientside licensing needed)
- JavaScript API remains unchanged
- EVRS (with VRS add-on license for WingScan) available
- UI is similar.. a small Kofax icon will sit in the status menu (MacOS) when active instead of system tray (windows)

MacOS: Scanning

MacOS scanning will use Apple Capture instead of TWAIN

There are some limitations because of this:

- No Scanner UI will show (no Config dialog of progress indicator)
- maxPages will not stop the page feed
- Imprinters are not supported

List of supported Scanners:

NOTE: (update July 2020) when initially released, we supported only Apple Capture and this was the supported scanner list: https://support.apple.com/en-us/HT201465

However, Apple has since discontinued that and uses AirPrint / ImageCapture. So, for MacOS supported scanners, check your scanner manufacturer documentation to see if it says it supports AirPrint / ImageCapture technology for MacOS.

MacOS: Barcode Reading

We are using the AtalaBar engine instead of the Honeywell engine (used in the windows version)

This means the MacOS version has **no support** for the following in our MacOS scanning

- i2of5
- Aztec
- Micro Pdf417
- Micro QR

Logging and Troubleshooting

Use the built in console

/Applications/Utilities/Console.app

Log files will be written to

~/Library/Logs/WebCaptureService

Imprinting Support (windows WingScan)

WingScan in 11.1 adds support for Imprinters on scanners which support it.

Due to variations in TWAIN scanner driver support for Imprinting, there may be variations

Added imprinter settings under ScanningOptions ImprinterConfig

PDF/A support

DotImage 11.1 added a lot of additional PDF/A support options

PDF/A in PdfDocument

For PdfDocument class, we added support for handling PDF/A documents. NOTE that you can not use PdfDocument to convert non-PDF/A doucments to PDF/A.. but what this does for you is allows you to safely open/save PDF/A documents without stripping their PDF/A compliance

Added support for the Following PDF/A Specs

- PDF/A-1 (a, b)
- PDF/A-2 (a, b, u)
- PDF/A-3 (a, b, u) without portfolio

Criteria needed to get PDF/A documents:

- All source documents need to be PDF/A (this does NOT convert non-PDF/A to PDF/A
- Color Profiles should have the same color spaces for all documents

This means you can not combine non-PDF/A documents with PDF/A documents and get a PDF/A out.. and all source PDF/A documents being combined must be of matching color profile

PdfDocument Save Behavior

PdfASavigBehavior values:

- PreserveOriginalPdfType (default)
- SavePdfA
- SavePdf

Save() method behavior			
PdfASavingBehavior	**Regular PDFs**	**Mixed**	**PDF/A Documents**
PreserveOriginalPdfType	Regular PDF	Regular PDF	PDF/A or PdfAException
SavePdfA	PdfAException	PdfAException	PDF/A or PdfAException
SavePdf	Regular PDF	Regular PDF	Regular PDF

Combining Multiple Files

		Pdf/A-1		Pdf/A-2			Pdf/A-3		
		a	b	a	b	u	a	b	u
Pdf/A-1	a	Pdf/A-1a							
	b	Pdf/A-1b	Pdf/A-1b						
Pdf/A-2	a	exception	exception	Pdf/A-2a					
	b	exception	exception	Pdf/A-2b	Pdf/A-2b				
	u	exception	exception	Pdf/A-2b	Pdf/A-2b	Pdf/A-2u			
Pdf/A-3	a	exception	exception	exception	exception	exception	Pdf/A-3a		
	b	exception	exception	exception	exception	exception	Pdf/A-3b	Pdf/A-3b	
	u	exception	exception	exception	exception	exception	Pdf/A-3b	Pdf/A-3b	Pdf/A-3u

INFO: Changes Introduced in DotImage 11.1

PDF/A Compatibility Verification

IsPdfACompatible property

- verifies metadata
- verifies main color profiles
- Does not verify document for PDF/A compliance

```
ar firstDoc = new PdfDocument("first.pdf"); var secondDoc = new PdfDocument("second.pdf");
firstDoc.Pages.AddRange(secondDoc.Pages); PdfSaveOptions options = new PdfSaveOptions {
PdfASavingBehavior = firstDoc.IsPdfACompatible() ? PdfASavingBehavior.SavePdfA :
PdfASavingBehavior.SavePdf }; firstDoc.Save("output.pdf", options);
```

PDF/A in PdfGeneratedDocument

New class PdfARenderer is used to produce PDF/A.. please note it supports PDF/A-1b only

```
sing (var file = File.OpenRead("doc.pdf")) { using (var document = new
PdfGeneratedDocument(file)) { // note, you will need to download a CMYK Color profile online
// suggest https://www.adobe.com/support/downloads/iccprofiles/iccprofiles_win.html using
(var cmykProfile = new PdfIccColorSpaceResource(File.OpenRead("CMYK.icc"), true)) { using
(var result = File.Create("result.pdf")) { PdfARenderer renderer = new PdfARenderer(result) {
CmykColorSpace = cmykProfile, ImageExtractor = new AtalaImageExtractor(),
IgnoreUnsupportedAnnotsAndActions = true, ConvertIncompatiblePagesToImages = true };
renderer.StreamlessFontFound += (o, args) => arg.AlternativeFontPath = GetTTFont(args);
renderer.Render(document); } } } }
```

Note that GetTTFont is not part of our SDK - you would need to provide a method to pass in the font resources and use the info to return the path to the requested font

Here is a rough example (provided as-is) it is your responsibility to make your own GetTTFont that meets your needs

```
ublic string GetTTFont(FontEventArgs args) { const string localFontFolder = "%folder with
fonts%"; // means that internal mechanism found alternate font in system fonts. if
(args.AlternativeFontPath != null) return args.AlternativeFontPath; // replace specific font
if (args.FontResource.FontFamily == "MinionPro-Regular") return Path.Combine(localFontFolder,
"arial.ttf"); // try to find font in the local font folder var fontPath =
```

```
Path.Combine(localFontFolder, $"{args.FontResource.FontFamily}.ttf"); if
(File.Exists(fontPath)) return fontPath; // use default font return
Path.Combine(localFontFolder, "arial.ttf"); // or throw exception // throw new
Exception("font does not found"); }
```

PdfARenderer Configuration

PdfARenderer Properties

- ConvertIncompatiblePagesToImages to convert incompatible pages to images
- Color spaces related properties
  - RgbColorSpace
  - CmykColorSpace
- ImageExtractor for converting PDF pages to images
  - AtalaImageExtractor
  - Defined in Atalasoft.dotImage.PdfDoc.Bridge.dll
- IgnoreUnsupportedAnnotsAndActions
- StreamlessFontFound event

has ConvertIncompatiblePatesToImages as part of the PdfARenderer

```
dfARenderer renderer = new PdfARenderer(result) { RgbColorspage = rgbProfile, CmykCOlorSpace
= ckykProfile, ImageExtractor = new AtalaImageExtractor(), IgnoreUnsupportedAnnotsAndActions
= true, ConvertIncompatiblePaesToImages = true };
```

AtalaImageExtractor is in the Atalasoft.dotImage.PdfDocBridge.dll

if IgnoreUnsupportedAnnotsAndActions is set to false then it will throw exception on fail

StreamlessFontFound event

- all fonts used in PDF must be embedded in PDF/A
- if it is nable to find font it ires StreamlessFontFound event where you can provide

PDF/A in PdfEncoder

INFO: Changes Introduced in DotImage 11.1

Added PDF/a-2b support

DocumentType can now be set to PdfDocumentType.PdfA2b

PDFA/2b allows the user of UseAdvancteedImageCompression = true

this allows Jpeg2000 / Jbig2 compression to be used in PdfEncoder and produce PDF/A-2b compatible output

```
sing (FileSystemImageSource fs = new FileSystemImageSource(imagePath, true)) { PdfEncoder encoder = new PdfEncoder { DocumentType = PdfDocumentType.PdfA2b, UseAdvancedImageCompression = true }; using (var outfs = File.Open("output.pdf", FileMode.Create, FileAccess.ReadWrite)) { encoder.Save(outfs, fs, null); } }
```

PDF/A in PdfTranslator

Added PDF/A-2b support

Can be enabled with PdfDocumentType.PdfA2b

Compression selector can support Jpeg2k and Jbig2

this allows Jpeg2000 / Jbig2 compression to be used in PdfTranslator and produce PDF/A-2b compatible output

```
dfTranslator trans = new PdfTranslator(PdfDocumentType.PdfA2b); trans.CompressionSelector = new PdfCompressionSelector(format => PdfImageCompressionType.Jpeg2000); ocrEngine.Translators.Add(trans); ocrEngine.Initialize(); var src = new FileSystemImageSource(imagePath, true); using (var stream = File.Create("outputTransl.pdf")) { ocrEngine.Translate(src, "application/pdf", stream); }
```

XRef Streams Compression in PdfDocument and PdfGeneratedDocument

Applicable to following objects

- fonts
- images
- colorspaces
- UseCompressedObjectStreams

Set documentObject.UsecompressedObjectStreams = true;

NOTE: compression will make saves take longer

PdfEncoder example:

```
dfEncoder encoder = new PdfEncoder() { UseCompressedObjectStreams = true };
```

PdfTranslator Example:

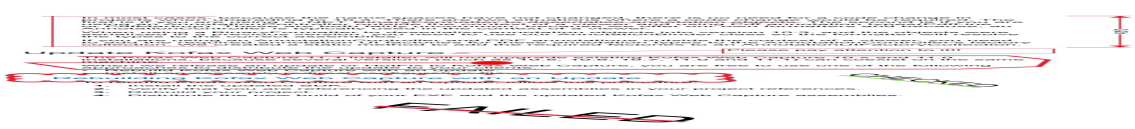
```
dfTranslator trans = new PdfTranslator() { UseCompressedObjectStreams = true };
```

PdfAnnotations Appearance in PdfDecoder

Our old PdfDecoder (pre-11.0) could not render several native type PDF annotations when PdfAnnotionRendering was enabled. Our 11.1 PdfIum engine can render these correctly (as part of the rasterized image.. this is NOT full support for these annotation types in our Annotation tools)

Applicable for PDF documents without appearance

- Polygons and polylines
- "Cloud" annotations
- Underline and Strikethrough
- Textbox
- Callout
- Measure lines



OCR Multilanguage recognition support

We have added support for multiple languages to our OCR engine classes for those engines which can support it

Abbyy and Tesseract3 are the only engines that currently support this new Multiple Languages setting.

The Atalasoft OcrEngine class added the SupportedMultiCultureResognition property. Read this to check if your specific engine supports multiple culture recognition

If it's supported in your engine, then instead of setting

```
ngine.RecognitionCulture = SomeRecognitionCulture;
```

you can pass in a List to

```
ngine.RecognitionCulturesList = listOfCultures;
```

This will instruct the engine to be able to handle documents with mixed multiple languages such as documents continaing both English and Hebrew or English and Chinese, etc...

External Libraries update

REMOVED

- Foxit (actually removed in 11.0.0.9)
- Tesseract2
- RecoStar

UPDATED

- PdfIum

- EVRS 3.3.0.265
- Tesseract (v3.05.02)
- Perceptive Document Filters 11.4.0.2822
- Luratech codecs (staring in 11.0.0.5)
  - Jbig2
  - Jpeg2000

SDK Updates

DEPRECATED

- Silverlight
- SharePoint

DROPPED SUPPORT

- IE < 11
- Visual Studio 2008

Original Article

Q10471 - INFO: Changes Introduced in DotImage 11.1

Atalasoft Knowledge Base
https://www.atalasoft.com/kb2/KB/50033/INFO-Changes-Introduced-in-DotImage-...