Aquaforest OCR
SDK for .Net

Reference Guide Version 1.30

August 2011

http://www.aquaforest.com/

CONTENTS

1 INTRODUCTION

1.1 SDK Overview

The Aquaforest OCR SDK for .Net incorporates the same high performance OCR engine that is included in the Aquaforest TIFF Junction and Autobahn DX products.

The SDK API allows developers full control over OCR processing to enable customized integration of OCR within .Net applications.

- OCR Bitmap or multi-page TIFF and PDF Files.
- Create Searchable PDF, RTF or Text output files.
- Control pre-processing options such as despeckle, deskew, line removal and autorotate.
- Specify one of the following supported document languages:

  English
  German
  French
  Russian
  Swedish
  Spanish
  Italian
  Russian English
  Ukrainian
  Serbian
  Croatian
  Polish
  Danish
  Portuguese
  Dutch
  Czech
  Roman
  Hungar
  Bulgar
  Slovenian
  Latvian
  Lithuanian
  Estonian
  Turkish

- Enumerate the OCR results, examining the words and characters recognized along with their co-ordinates.
- Process multi-page TIFF and PDF files one page at a time or all in one operation

1.2 Technical Support

Please contact Aquaforest Technical Support with any queries by email support@aquaforest.com. If required, telephone support is also available; please contact Aquaforest using the telephone contact details provided on the company website contact page.

2 SDK OVERVIEW

The SDK is provided as a set of .Net Assemblies, Native DLLs and configuration files designed to allow for straightforward integration into .Net applications.

2.1 System Requirements

2.1.1 Supported Environments

Windows 2003, 2008, Vista, Windows XP.

2.1.2 .Net Framework

.Net Version 3.5

2.1.3 Visual C++ Runtime

The Visual C++ 2008 Redistributable package is required for deployment as well as development.

2.2 Folders

The SDK contains the following folders :

Bin – This contains all the assemblies, DLLs and configurations files.
Docs – SDK Documentation
Samples – Sample C#, VB.Net and ASP.Net samples

2.3 A simple example

The full API reference is in section 5 of this guide, but as a starting point a simple example of a C# and VB.Net console application that creates a searchable PDF from a source TIFF file is shown below.

2.3.1 References

A reference to the Aquaforest.OCR.Api DLL should be added in your application.
If you wish to access the results of the OCR on a word by word basis, for example to obtain word and character results including positional information then you will also need to reference Aquaforest.OCR.Definitions DLL.

2.3.2 Classes

There are two classes used for the OCR :

PreProcessor – This class configures and performs image pre-processing (such as deskewing images) to ensure optimal OCR performance.

Ocr – This is the class that configures and performs the Optical Character Recognition.

Additionally, for accessing the OCR results at an individual word level the following classes are used:

Words – This class contains a collection of words in which is contained all the data available for the words and characters for any given page.

WordData – This class contains a collection of characters that make up the word along with the positional information for each character and the whole word.

StatusUpdateEventArgs – This class is available for each page processed when subscribing to the StatusUpdate event and provides information relating to the processing outcome for the page.

2.3.3 Processing Steps

The following steps are involved in this example

1. Create the Ocr and PreProcessor objects
2. Specify the location of the OCR bin folder
3. Specify Pre-Processor options
4. Specify OCR Options
5. Read the source file
6. Perform the recognition
7. Save the searchable PDF
8. Delete temporary files (these are by default stored in %TEMP% but the location can be specified using ocr.TempFolder)

2.3.4 C# Example

```
using System;
using Aquaforest.OCR.Api;

namespace ocr
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                // 1. Create Ocr and Preprocessor Objects
                // and enable console output

                Ocr _ocr = new Ocr();
                PreProcessor _preProcessor = new PreProcessor();
                _ocr.EnableConsoleOutput = true;

                // 2. OCR bin folder Location
                // The bin files can be copied to the application bin
                // folder.  Alternatively the System Path and ocr
                // Resource folder can be set as shown below and
                // then just the files in the bin_add folder added
                // to the application bin folder.

                string OCRFiles = @"C:\Aquaforest\OCRSDK\bin";
                System.Environment.SetEnvironmentVariable("PATH",
System.Environment.GetEnvironmentVariable("PATH") + ";" + OCRFiles);

                _ocr.ResourceFolder = OCRFiles;


                // 3. Set PreProcessor Options
                _preProcessor.Deskew = true;
                _preProcessor.Autorotate = false;


                // 4. Set OCR Options
                _ocr.Language = SupportedLanguages.English;
                _ocr.EnablePdfOutput = true;


                // 5. Read Source TIFF File
                 _ocr.ReadTIFFSource(
@"C:\Aquaforest\OCRSDK\docs\tiffs\sample.tif");

                // 6. Perform OCR Recognition
                if (_ocr.Recognize(_preProcessor))
                {
                // 7. Save Output as Searchable PDF
                    _ocr.SavePDFOutput(
@"C:\Aquaforest\OCRSDK\docs\tiffs\sample.pdf", true);
                }
                // 8. Clean Up Temporary Files
                _ocr.DeleteTemporaryFiles();
            }

            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

2.3.5 VB.Net Example

```
Module Module1

    Sub Main()
        ' 1. Create Ocr and Preprocessor Objects

        Dim _ocr As New Aquaforest.OCR.Api.Ocr()

        Dim _preProcessor As New Aquaforest.OCR.Api.PreProcessor()

        _ocr.EnableConsoleOutput = True

        ' 2. OCR bin folder Location
        ' The bin files can be copied to the application bin folder.
        ' Alternatively the System Path and ocr Resource folder
        ' can be set as shown below.

        Dim OCRFiles As String
        ' 2. OCR bin folder Location
        ' The bin files can be copied to the application bin
        ' folder.  Alternatively the System Path and ocr
        ' Resource folder can be set as shown below and
        ' then just the files in the bin_add folder added
        'to the application bin folder.

        OCRFiles = "C:\\Aquaforest\\OCRSDK\\bin"
        System.Environment.SetEnvironmentVariable("PATH",
System.Environment.GetEnvironmentVariable("PATH") + ";" + OCRFiles)

        _ocr.ResourceFolder = OCRFiles


        ' 3. Set PreProcessor Options
        _preProcessor.Deskew = True
        _preProcessor.Autorotate = False


        ' 4. Set OCR Options
        _ocr.Language = Aquaforest.OCR.Api.SupportedLanguages.English
        _ocr.EnablePdfOutput = True


        ' 5. Read Source TIFF File

_ocr.ReadTIFFSource("C:\\Aquaforest\\OCRSDK\\docs\\tiffs\\sample.tif")

        ' 6. Perform OCR Recognition
        If _ocr.Recognize(_preProcessor) Then

        ' 7. Save Output as Searchable PDF
_ocr.SavePDFOutput("C:\\Aquaforest\\OCRSDK\\docs\\tiffs\\sample.pdf", True)

        End If

        '8. Clean Up Temporary Files
        _ocr.DeleteTemporaryFiles()


    End Sub

End Module
```

3 APPLICATION DEVELOPMENT AND DEPLOYMENT

3.1 References

To use the API a reference to Aquaforest.Ocr.Api must be included in your application. If you wish to enumerate the OCR results rather than simply generate PDF, RTF or TXT outputs then you will also need to add a reference to Aquaforest.Ocr.Definitions.

3.2 Deploying C# and VB.Net Applications

Any deployment method should ensure that the target system meets the requirements (see section 2.1) and install the Visual C++ 2008 Redistributable package and Net Version 3.5 framework if necessary in addition to the the full contents of the SDK ***bin*** folder.

There are two approaches to handling the contents of the SDK ***bin*** folder that can be used when building and deploying C# and VB.Net applications.

Approach 1

The entire contents of the SDK ***bin*** folder can be copied to the application bin folder.

Approach 2

The contents of the SDK ***bin_add*** folder can be copied to the application bin folder and the ocr resource folder and path of the full bin folder must be specified as shown in the sample code in section 2 above. In certain cases, e.g. deploying as a service, this approach is not suitable.

3.3 Deploying ASP.Net Applications

The same two approaches that work for C# and VB.Net can also be employed for ASP.Net applications. Note that with trial licenses a pop-up dialog box appears on the server.

3.4 Licensing

Production system deployment requires that a license string is defined in the code. The license string defines the number of concurrent OCR processes that can be run.

For example :

```
ocr.License =
"MT0xMjM0NTY7BLk4uT3RoZXOzM9NDs0PVRydWEYzMDRFOEQxMzg0QkQ5ODREQTk3RQ";
```

If the string is not specified the SDK will run in evaluation mode. In evaluation mode :

- *A trial "pop-up" will appear for each document processed*
- *Generated searchable PDFs will include indelible watermarks*
- *Only 3 pages are generated for text or RTF files.*

3.5 64 bit support

Whilst applications built using the SDK will run on 64 bit systems, they must currently be built as 32 bit.

4 SAMPLE APPLICATIONS

The samples folder includes a number of sample applications in C#, VB.Net and ASP.Net. The project files are for Visual Studio 2008. The sample applications are as described below.

4.1 Simple

This includes VB and C# version of the simple application listed in section 2.

4.2 ConvertFolderOfFiles

This forms-based application demonstrates converting a folder of TIFF or Bitmap files. Both VB.Net and C# versions are included.

4.3 ZonalOCR

This forms-based application demonstrates converting a folder of TIFF or Bitmap files. Both VB.Net and C# versions are included.

4.4 ASP.Net

This demonstrates a simple web-based application that allows uploading an image file for OCR and conversion to text, RTF or searchable PDF.

5 API REFERENCE

To use the API a reference to Aquaforest.Ocr.Api must be included in your application. If you wish to enumerate the OCR results rather than simply generate PDF, RTF or TXT outputs then you will also need to add a reference to Aquaforest.Ocr.Definitions.

5.1 PreProcessor Class

A PreProcessor object, which must be created and passed to the Ocr object, controls all of the pre-processing that can be performed on the input image in order to improve the quality of the output. Instantiation of the PreProcessor object will initialise a default set of pre-processing options which result in minimal image manipulation. For a full description of the pre-processing options available and appropriate values see section 5.1.2 Properties below.

5.1.1 Constructor

```
PreProcessor preProcessor = new PreProcessor();
```

5.1.2 Properties

Property	Description
bool Autorotate	Auto-rotate the image – this will ensure all text oriented normally. The default value is false (disabled). Note: When using a PDF source Autorotation will be disabled on any pages already containing text.
int Binarize	This value should generally only be used under guidance from technical support. It can control the way that color images are processed and force binarization with a particular threshold. A value of 200 has been shown to generally give good results in testing, but this should be confirmed with “typical” customer documents. By setting this to -1 an alternative method is used which will attempt to separate the text from any background images or colors. This can give improved OCR results for certain documents such as newspaper and magazine pages.
int BlankPageThreshold	Use this to set the minimum number of "On Pixels" that must be present in the image for a page not to be considered blank. A value of -1 will turn off blank page detection. A value of 100 produced reasonable blank page detection in testing, but the validity of this should be confirmed using “typical” source documents.
int BoxSize	This option is ideal for forms where sometimes boxes around text can cause an area to be identified as graphics. This option removes boxes from the temporary copy of the imaged used by the OCR engine. It does not remove boxes from the final image. Technically, this option removes connected elements with a minimum area (in pixels and defined by this property). This option is currently only applied for bitonal images.

PreProcessor Class Properties - Continued

Property	Description
int Despeckle	Despeckle the image – The method removes all disconnected elements within the image that have height or width in pixels less than the specified figure. The maximum value is 9 and the default value is 0.

bool Deskew	Deskew (straighten) the image. The default value is false (disabled).
string Morph	Image Morphology. This option should generally only be used under guidance from technical support.
bool MRC	This enables Mixed Raster Compression which can dramatically reduce the output size of PDFs comprising color scans. Note that this option is only suitable when the source is not a PDF.
int MRCBackground	Sampling size for the background portion of the image. The higher the number, the larger the size of the image blocks used for averaging which will result in a reduction in size but also quality. Default value is 3
int MRCForeground	Sampling size for the foreground portion of the image. The higher the number, the larger the size of the image blocks used for averaging which will result in a reduction in size but also quality. Default value is 3
int MRCQuality	JPEG quality setting (percentage value 1 - 100) for use in saving the background and foreground images. Default value is 75
bool NoPictures	By default, if an area of the document is indentified as a graphic area then no OCR processing is run on that area. However, certain documents may include areas or boxes that are identified as “graphic” or “picture” areas but that actually do contain useful text. Setting NoPictures to true will cause it to ignore areas identified as pictures whilst setting it to false will force OCR of areas identified as pictures.
bool RemoveLines	When set to true this will enable the removal of lines. This feature is particularly useful where pages contain tables and underlining which can prevent the OCR engine from recognising characters. The lines are removed only from the image used in OCR and not from the image used in the final PDF if PDF creation is enabled.

5.1.3 Methods

Method	Description
ConfigurePDFStamp(string prefix, string suffix, Nullable<int> start, Nullable<int> digits, PagePositionEnum position, StampType stampType)	Using this method stamps can be configured to be added to each page of the PDF output. The stamps contain one or more of the following: • Prefix – a string to be added to the beginning of the stamp, before the number section. • Start – the value that the number portion of the stamp should start at. The number portion will be incremented by 1 each page. • Digits – a value indicating the minimum length that the number portion of the stamp should be displayed as. Preceding 0's will be used to pad any numbers less than this whilst numbers greater than this will be displayed in full. • Suffix - a string to be added to the end of the stamp, after the number section. Thus a stamp with Prefix = "Beginning", Start = "1", Digits = "4" and Suffix = "End" would produce the text "Beginning0001End" on the first page. Any one of these can be set to null resulting in the exclusion of that part from the final text. Additionally the stamp can be added either as visible searchable text or as an image and can be positioned in one of the following: • Top Left • Top Centre • Top Right • Centre Left • Centre • Centre Right • Bottom Left • Bottom Centre • Bottom Right

5.2 OCR Class

The OCR object is used to control OCR processing, obtain status updates during processing and retrieve the resulting output from this processing upon completion.

5.2.1 Constructor

```
Ocr ocr = new Ocr();
```

5.2.2 Properties

Property	Description
String ResourceFolder	This property can optionally be used to set the location of the resources folder when the resources are not located in the same folder as the assembly using the API.
SupportedLanguages Language	Sets the language to be used for the OCR processing. This takes a value from the enumeration SupportedLanguages which is defined in the API. Default language is English.
bool EnablePDFOutput	Enables or disables the production of Portable Document Format output. Default value is false (disabled).
bool EnableTextOutput	Enables or disables the production of simple text final output. Default value is true (enabled).
bool EnableRTFOutput	Enables or disables the production of Rich Text Format output. Default value is false (disabled).
int StartPage	Sets the first page of the source file that the OCR process will be begin from (for a multipage source). Throws an ArgumentOutOfRangeException if a source file has not been set already (by using the ReadBMPSource or ReadTIFFSource method prior to setting this property) or if the page is greater than the number of pages in the source. By default the whole of the document will be processed.
int EndPage	Sets the last page of the source file that the OCR process will be run to (for a multipage source). Throws an ArgumentOutOfRangeException if a source file has not been set already (by using the ReadBMPSource or ReadTIFFSource method prior to setting this property) or if the page is greater than the number of pages in the source. By default the whole of the document will be processed.
int CurrentPage	Returns the current page for which the OCR has been performed. This is useful only when using Recognize() in another thread.
bool HandleExceptionsInternally	When set to true the Ocr object will catch any exceptions for method calls and simply return false from the method. The exceptions caught are stored in the LastException property overwriting any previous value.

OCR Class Properties - Continued

Property	Description
Exception LastException	Stores last exception caught by the Ocr object.
bool EnableConsoleOutput	If enabled then progress messages will be sent to the console. Default is false.
string TempFolder	Specifies a temporary folder for storing bitmap images and intermediate output during OCR processing. If this is not specified, the first of the following environment variables that is defined will be used : TMP, TMPDIR, TEMP.
String License	Specifies the license key
bool EnableConsoleOutput	If set to True, progress messages will be written to the console output. Default false.
int EnableDebugOutput	If set to a value greater than 0 (default value) debug messages will be written to the console output. Please contact Aquaforest for guidance on suitable values if you need to generate debug output.
bool RemoveExistingPDFText	RemoveExistingPDFText if set to true will result in the removal of any existing text from the output PDF*. *Note: when PDF output is generated from a PDF source it is a copy of the PDF that is manipulated rather than generating a new one. This approach offers several advantages such as potential size savings and performance enhancements.
bool DeleteTemporaryFilesOnPageCompletion	When set to true the temporary files generated for each page during OCR processing will be removed as soon as the OCR engine has finished with them*. *Note: the OCR engine is finished with the temporary files for a page as soon as the output for that page is added to the overall output. If you wish to use functionality such as ReadPageWords, GetPageImage, etc then this will require that the temporary files are available for the page requested and so will fail if DeleteTemporaryFilesOnPageCompletion is true.
bool Dotmatrix	Set this to true to improve recognition of dot-matrix fonts. Default value is false. If set to true for non dot-matrix fonts then the recognition can be poor.
bool OneColumn	The default value for this is true which improves the handling of single column text. Better handling of multi-column text such as magazine or news print can be achieved.
bool OptimiseOcr	When set to true this will enable the Optimised OCR functionality. See section 5.8 for details.

5.2.3 Methods

Method	Description
void Abort()	Stops processing of an ongoing call to Recognize. Processing will stop on completion of any ongoing page.
void DeleteTemporaryFiles()	Removes temporary files created during the OCR processing from the system. Note, do not call this before you have completely finished processing a file.
Image GetPageImage(int pageNumber)	Returns a System.Drawing.Image containing the processed image.
void Recognize(PreProcessor preProcessor)	Performs any pre-processing defined in the PreProcessor object and then carries out OCR processing on the pre-processed image.
bool ReadBMPSource(string fileName)	Checks for the existence of the source file and sets up the OCR engine for handling the bitmap image.
bool ReadImageSource(Image image)	Reads an Image object checking the number of frames (pages).
bool ReadPDFSource(string fileName)	Checks for the existence of the source file and sets up the OCR engine for handling the PDF.
bool ReadPDFSource(string filename, string password)	Checks for the existence of the source file and sets up the OCR engine for handling the secure PDF for which the password is provided. If PDF output is generated from this the output will have no security settings defined.
string ReadPageString(int pagenumber)	Returns a string containing the words from the specified page.
string ReadPageString(int pagenumber, Rectangle region)	Returns a string containing the words for the specified page where the words are fully enclosed in the bounds of the region specified.
Words ReadPageWords(int pagenumber)	Returns an instance of the Words class for the specified page.
Words ReadPageWords(int pagenumber, Rectangle region)	Returns an instance of the Words class for the specified page where the words are fully enclosed in the bounds of the region specified.
void ReadTIFFSource(string fileName)	Checks for the existence of the source file and sets up the OCR engine for handling the TIFF image.

OCR Class Methods - Continued

Method	Description
bool SavePDFOutput(string fileName, bool	Saves the output to a PDF file with the name

overwriteExisting)	specified. If any text was extracted then this will be searchable in the PDF.
bool SaveRTFOutput(string fileName, bool overwriteExisting)	Saves the output to a RTF file with the name provided.
bool SaveTextOutput(string fileName, bool overwriteExisting)	Saves the text extracted to a simple text file with the name provided.

5.2.4 Events

Event	Description
void PageCompleted(int pageNumber, bool ocrSuccess, bool imageProcessingSuccess, bool blankPage)	This event is raised when processing of a page is complete. The variable pageNumber indicates which page has just completed, the variable ocrSuccess indicates whether the OCR for that page was successful or not (note, a successful OCR does not necessarily indicate that text was found on a page), imageProcessingSuccess indicates whether the pre-process was successful for that page, blankPage indicates whether the page detected as blank.
void StatusUpdate (object sender, StatusUpdateEventArgs statusUpdateEventArgs)	This event is raised when processing of a page is complete. The StatusUpdateEventArgs object provides access to information relating to the status of the page processed.

5.2.5 Subscribing to StatusUpdate using C#

Include a reference to Aquaforest.OCR.Definitions.dll in the solution and define a method to match the event signature, see below.

```
private void OcrStatusUpdate(object sender, StatusUpdateEventArgs
statusUpdateEventArgs)
{
     double confidenceScore = statusUpdateEventArgs.ConfidenceScore;
     // anything confidenceScore below 1 might be worth investigation
     int pageNumber = statusUpdateEventArgs.PageNumber;
     int rotation = statusUpdateEventArgs.Rotation;
     // rotation used in 90° steps from beginning
     // orientation (0), i.e. 1 = 90, 2 = 180, 3 = 270
     bool textAvailable = statusUpdateEventArgs.TextAvailable;
     bool imageAvailable = statusUpdateEventArgs.ImageAvailable;
     bool blankPage = statusUpdateEventArgs.BlankPage;
}
```

Finally add a new reference to the event on the OCR object:

```
_ocr.StatusUpdate += OcrStatusUpdate;
```

5.2.6 Subscribing to StatusUpdate using VB.NET

Include a reference to Aquaforest.OCR.Definitions.dll in the solution and define a method to match the event signature, see below.

```
Private Sub OcrPageCompleted(ByVal sender As Object, ByVal
statusUpdateEventArgs As StatusUpdateEventArgs) Handles
_ocr.StatusUpdate

    Dim confidenceScore As Double
    Dim pageNumber As Integer
    Dim rotation As Integer
    Dim textAvailable As Integer
    Dim imageAvailable As Integer
    Dim blankPage As Boolean

    confidenceScore = statusUpdateEventArgs.ConfidenceScore
    ' anything confidenceScore below 1 might be worth
    ' investigation
    pageNumber = statusUpdateEventArgs.PageNumber
    rotation = statusUpdateEventArgs.Rotation
    ' rotation used in 90° steps from beginning orientation (0),
    ' i.e. 1 = 90, 2 = 180, 3 = 270
    textAvailable = statusUpdateEventArgs.TextAvailable
    imageAvailable = statusUpdateEventArgs.ImageAvailable
    blankPage = statusUpdateEventArgs.BlankPage

End Sub
```

Declare the OCR object using “WithEvents”:

```
Private WithEvents _ocr As New Ocr
```

5.2.7 Enumerations

Enumeration	Description
SupportedLanguages	This enumeration includes all of the languages currently supported by the API.

5.3 StatusUpdateEventArgs Class

This class contains information relating to the conversion status of a page.

5.3.1 Constructor

An instance of this class is obtained for each page processed when subscribing to the event StatusUpdate.

5.3.2 Properties

Property	Description
int PageNumber	This property returns page for which the object relates to.
int Rotation	A value from 0 to 3 which indicates the rotation used for the output in terms of the number of 90° steps away from the orientation in which the input page was provided. If AutoRotation is set to false this will always be 0.
double ConfidenceScore	Generally a value of 1 or greater would indicate that reasonable OCR of a page, but this should be confirmed using “typical” source files.
bool TextAvailable	This property indicates whether text was extracted for the page.
bool ImageAvailable	This property indicates whether an image (after all appropriate pre-processing) was successfully extracted.
bool BlankPage	This property indicates whether the page was detected as blank.

5.3.3 Words Class

This class contains a collection of WordData objects which are available on a page by page basis.

5.3.4 Constructor

An instance of this class is obtained by calling the ReadPageWords method on the Ocr object, passing the page for which the words are required.

5.3.5 Properties

Property	Description
int Count	This property returns the number of WordData objects in the collection.
int Height	This property returns the height of the current word.
int Width	This property returns the width of the current word.

5.3.6 Methods

Method	Description
WordData GetFirst()	Returns the first WordData object in the collection and sets the index to this item.
WordData GetNext()	Returns the next WordData object in the collection and sets the index to this item.
int GetHeight(int index)	Returns the word height from the WordData object stored at the specified index in the collection.
int GetWidth(int index)	Returns the word width from the WordData object stored at the specified index in the collection.

5.3.7 WordData Class

This class contains the individual characters along with the positional information relating to each character in the word and to the word as a whole.

5.3.8 Properties

Property	Description
float AverageCharacterHeight	This property returns the average height of all the characters in the word.
float AverageCharacterWidth	This property returns the average width of all the characters in the word.
int Bottom	This property returns the bottom of the word.
int CharacterList	This property returns a list of CharacterData objects for the word.
int Height	This property returns the height of the word.
int Left	This property returns the left edge of the word.
int Top	This property returns the Top of the word.
int Width	This property returns the width of the word.
string Word	This property returns the word as a string.

5.4 PdfMerger Class

This class can be used to merge two PDFs

5.4.1 Constructor

```
PdfMerger pdfMerger = new PdfMerger("C:\\out\\Merged.pdf");
```

5.4.2 Methods

Method	Description
void Append(string pdfFileToAdd)	Appends the document specified to the in memory PDF document.
void Close()	Writes the output to the file specified in the constructor.
void Dispose()	Clears any resources not yet released. This is useful if Close (which will automatically free such resources) is not called, for example if as a result of an error you do not wish to write the merged output.

5.5 Error Handling

There are two options regarding error handling using the API.

1. Using the default settings various exceptions can be thrown by the Ocr object so these should be trapped within the calling code.
2. Alternatively HandleExceptionsInternally can be set to true with the result that method calls will return false on error but throw no exceptions. The calling code can obtain the last exception from the LastException property if details of the failure are required.

5.6 Disposal and Temporary Files folders

During the OCR processing various temporary files are generated and used at different stages. These temporary files can be removed by calling DeleteTemporaryFiles. However, such a call should not be made until all processing (both within the Ocr object and calling code) on a file is complete as these files are required when calling SaveRTFOutput, SavePDFOutput, SaveTextOutput, GetPageImage and ReadPageWords. When the Ocr object is disposed of the temporary files are automatically removed.

5.7 Multi-threaded applications

Temporary files created and used throughout the OCR processing are named according to the page number, therefore if Ocr objects are instantiated in multiple threads then a different temporary folder must be set for each folder. If this is not done then un-expected behaviour will result.

5.8 Optimised OCR

When the OptimiseOcr property on the OCR object is set to false the OCR and image processing engines will use the settings in the ImagePreProcessingDefaults section of the file Properties.xml modified by any properties set on the OCR and PreProcessing objects.

Setting OptimiseOcr true will enable the use of these default settings first (without modification by the properties set on the OCR and PreProcessing objects) followed by the same defaults modified by the values in the ImagePreProcessing sections from ID="1" to ID="n" where n is the last consecutive set defined in Properties.xml.

Using heuristics and dictionary lookup the quality of the OCR output is then compared in order to determine the optimum set to output. In this way it is possible to define different sets of OCR and pre-processing conditions that are suited to different types of source documents. This approach can also improve the handling of documents that contain different types of pages, e.g. scanned at different qualities, containing different languages, containing standard and dot matrix prints, etc.

5.9 Properties File

The following are descriptions of those properties in the file Properties.xml that are most likely to be changed to improve engine performance. If you require further information regarding any properties in the file then please contact Aquaforest via support@aquaforest.com for assistance.

Binarize – This setting determines how the image will be converted into a bitonal one for OCR. The following are valid options:

> -1 – This utilizes a technique whereby those parts of the image that have certain characteristics indicative of characters are extracted from the underlying image. This approach can give the best results on pages such as magazine images, news print, etc and will handle light text on darker backgrounds. This approach can cause an increase in processing time with certain images.
> 0 – This utilizes the binarisation capabilities built into the OCR engine and whilst it can give good results in limited situations it is not generally recommended.
> >0 – A value greater than 0 (the recommended default is 200) will use a simple threshold technique comparing the intensity of the pixel to the threshold value to determine whether it should be set to black or white. This simple approach is the fastest option.

BoxSize – Setting a value above 0 will cause the removal of enclosing boxes from the image used for the OCR processing. The default recommended is 100, i.e. where the box edges are 100 pixels or greater.

BackgroundFactor - Sampling size for the background portion of the image. The higher the number, the larger the size of the image blocks used for averaging which will result in a reduction in size but also quality. Default value is 3

DotMatrix - Set this to True to improve recognition of dot-matrix fonts. Default value is False. If set to true for non dot-matrix fonts then the recognition can be poor

ForegroundFactor - Sampling size for the foreground portion of the image. The higher the number, the larger the size of the image blocks used for averaging which will result in a reduction in size but also quality. Default value is 3

Jbig2EncFlags – These are the flags that will be passed to the application used to generate JBIG2 versions of images used in PDF generation (assuming this compression is enabled). Options are as follows:

> -b <basename>: output file root name when using symbol coding
> -d --duplicate-line-removal: use TPGD in generic region coder
> -p --pdf: produce PDF ready data
> -s --symbol-mode: use text region, not generic coder
> -t <threshold>: set classification threshold for symbol coder (def: 0.85)
> -T <bw threshold>: set 1 bpp threshold (def: 188)
> -r --refine: use refinement (requires -s: lossless)
> -O <outfile>: dump thresholded image as PNG
> -2: upsample 2x before thresholding
> -4: upsample 4x before thresholding
> -S: remove images from mixed input and save separately
> -j --jpeg-output: write images from mixed input as JPEG
> -v: be verbose

Language – The acceptable vales are as follows:

0 - English
1 - German
2 - French
3 - Russian
4 - Swedish
5 - Spanish
6 - Italian
7 - Russian English
8 - Ukrainian
9 - Serbian
10 - Croatian
11 - Polish
12 - Danish
13 - Portuguese
14 - Dutch
19 - Czech
20 - Roman
21 - Hungar
22 - Bulgar
23 - Slovenian
24 - Latvian
25 - Lithuanian
26 - Estonian
27 - Turkish

MaxDeskew – Maximum angle by which a page will be deskewed.

Morph – Morphological options that will be applied to the binarized image before OCR. If left blank none is applied. Common options include those listed below but for more options please contact support@aquaforest.com:

d2.2 – 2x2 dilation applied to all black pixel areas, useful for faint prints.
e2.2 – 2x2 erosion applied to all black pixel areas, useful for heavy prints.
c2.2 – closing process that performs a 2x2 dilation followed by a 2x2 erosion with the result that holes and gaps in the characters are filled.

NoPictures - By default, if an area of the document is indentified as a graphic area then no OCR processing is run on that area. However, certain documents may include areas or boxes that are identified as “graphic” or “picture” areas but that actually do contain useful text. Setting NoPictures to True will cause it to ignore areas identified as pictures whilst setting it to False will force OCR of areas identified as pictures.

OneColumn - The default value for this is true which improves the handling of single column text. Better handling of multi-column text such as magazine or news print can be achieved.

PdfToImage – The SDK ships with two engines for the conversion of PDF pages to images for OCR. The default engine is used when this is set to 0 but if certain PDF source documents are proving problematic then the alternate engine can be used by changing this value to 1.

PdfToImageIncludeText – When set to False this will prevent the conversion of real text (i.e. electronically generated as opposed to text that is part of a scanned image) from being rendered in the page images extracted from the PDF. This is because the text is already searchable and so generally does not require OCR. The value can be set to True however if the OCR is required on this real text.

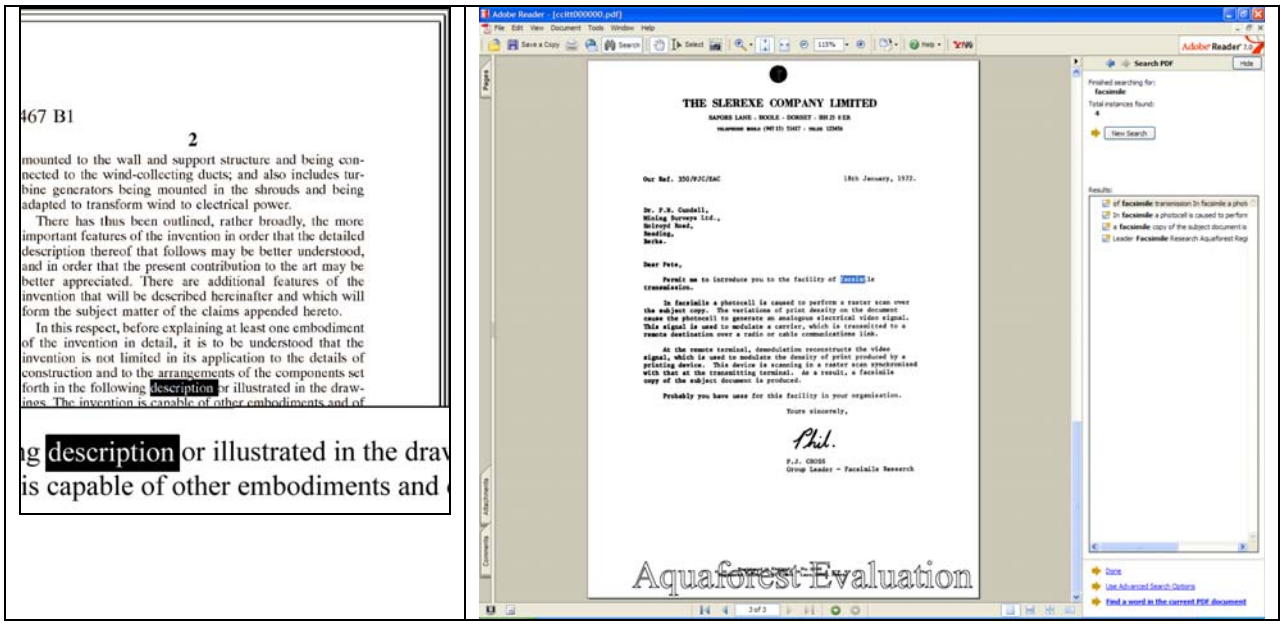
Quality - JPEG quality setting (percentage value 1 - 100) for use in saving the background and foreground images. Default value is 75

RemoveLines – The value used in Line removal. If blank no line removal will occur. The normal value to use to enable line removal is 100.5 but it you are experience difficulties with this value or have any questions then please contact support@aquaforest.com.

6 BACKGROUND - SEARCHABLE PDFS

6.1 What is a Searchable PDF?

A searchable PDF file is a PDF file that includes text that can be searched upon using the standard Adobe Reader "search" functionality. In addition, the text can be selected and copied from the PDF. Generally, PDF files created from Microsoft Office Word and other documents are by their nature searchable as the source document contains text which is replicated in the PDF, but when creating a PDF from a scanned document and an OCR process needs to be applied to recognize the characters within the image.



6.2 Inside a Searchable PDF

In the context of Document Imaging, a searchable PDF will typically contain both the original scanned image plus a separate text layer produced from an OCR process. The text layer is defined in the PDF file as invisible, but can still be selected and searched upon. PDF files are able to store images using most of the native compression schemes used in TIFF files, so for example Group 4 TIFF files do not usually require any format conversion.

6.3 OCR Accuracy

A number of factors affect the accuracy of the text produced by the OCR process – 100% accuracy is certainly possible under good conditions but each of the following issues, and OCR processing options will have an impact.

6.3.1 Original Image Quality

Although some pre-processing options such as despeckle and deskew can help in some cases, the visual quality of the original scan is of paramount importance.

6.3.2 Image DPI and Format

The image resolution should be at least 150 DPI for OCR processing, and preferably 300 DPI for optimal results, although for good quality scans 200 DPI is often sufficient. Non-lossy formats (TIFF Group 4, LZW etc) are preferred over lossy formats such as JPEG.

6.3.3 Despeckle

This pre-processing option removes isolated "dots" within the image which can cause recognition problems, and makes the result image "cleaner".

6.3.4 Deskew

This option can improve OCR results by straightening crooked pages.

6.3.5 Auto-Rotate

OCR processing usually recognizes text written top-to-bottom, left-to-right, so pages that are orientated any other way (usually landscape pages) need to be re-oriented to enable recognition.

6.3.6 Graphics Areas

There are two options that can be used to control how the OCR engine processes parts of the document image that appear to be graphics areas.

To ensure that the OCR engine can be forced to process such areas there are two options :

"Treat all Graphics Areas as Text". This option will ensure the entire document is processed as text.

"Remove Box Lines in OCR Processing". This option is ideal for forms where sometimes boxes around text can cause an area to be identified as graphics. This option removes boxes from the temporary copy of the imaged used by the OCR engine. It does not remove boxes from the final image. Technically, this option removes connected elements with a minimum area (by default 100 pixels).

6.3.7 Language Settings

The language setting determines the set of characters that will be recognized, and the dictionary that will be used as a guide.

6.4 Hardware and Performance

6.4.1 CPU Power

The OCR process is highly CPU intensive and will benefit from being given as much CPU power as possible. As a guide about 2,000 pages per hour can be processed on a 3.0 GHz processor core, although this will vary according to the source document and OCR options chosen.

6.4.2 Exploiting Multiple CPUs

To take advantage of multiple cores, multiple OCR instances should be run in parallel.

6.4.3 Memory

Memory can be a limiting factor when creating the final PDF, in the case of very large documents. A rule of thumb would be to have 1GB – 1.5 GB of memory per processor core.

7 ACKNOWLEDGEMENTS

This product makes use of a number of Open Source components which are included in binary form. The appropriate acknowledgements and copyright notices are given below.

LEPTONICA
Copyright (C) 2001 Leptonica. All rights reserved.

LIBJPEG
This software is based in part on the work of the Independent JPEG Group.

ZLIB
(C) 1995-2004 Jean-loup Gailly and Mark Adler.

ITEXT 4.1.6
Copyright (C) 1999-2009 by Bruno Lowagie and Paulo Soares et all. All Rights Reserved. Binaries distributed under the Mozilla Public License.

CUNEIFORM
Copyright (c) 1993-2008, Cognitive Technologies. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of the Cognitive Technologies nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

LIBTIFF
Copyright (c) 1988-1997 Sam Leffler. Copyright (c) 1991-1997 Silicon Graphics, Inc.

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that (i) the above copyright notices and this permission notice appear in all copies of the software and related documentation, and (ii) the names of Sam Leffler and Silicon Graphics may not be used in any advertising or publicity relating to the software without the specific, prior written permission of Sam Leffler and Silicon Graphics.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL SAM LEFFLER OR SILICON GRAPHICS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

FREEIMAGE
This software uses the FreeImage open source image library. See http://freeimage.sourceforge.net for details.
FreeImage is used under the (FIPL), version 1.0.