IDistiller() class: PDFLib replacement

- Introduction
- MakoDistillerCmd sample
  - Matching the results of PDFLib
- API
  - Public Types
  - Public Member Functions
  - Static Public Member Functions

Introduction

Jaws PDF Library, usually abbreviated to *PDFLib*, was created to provide easy access to the Jaws RIP technology for the purpose of converting PostScript to PDF and vice versa. This role has been superseded by Mako Core, the digital document SDK from Global Graphics.

For most PDFLib use cases, the introduction of PostScript input in Mako 5.0 (the `IPSInput()` class) allowed a PDFLib user to switch to Mako, opening up many opportunities for document processing beyond the simple conversion task that PDFLib is able to handle.

However, the two stage conversion (using `IPSInput()` immediately followed by `IPDFOutput()`, for example) was slower than the direct conversion offered by PDFLib. Additionally, `IPSInput()` was less flexible in its handling of a PostScript *prolog* to modify the conversion process.

Mako 5.2.0 introduces the `IDistiller()` class to overcome these limitations. Designed as a replacement for PDFLib, testing has shown up a significant speed improvement over the previous conversion route, matching or in many cases improving on PDFLib performance.

MakoDistillerCmd sample

A command-line application is provided that offers access to the `IDistiller` class without the need to write code. It is modeled after the equivalent sample application, *testpdflibcmd.exe*, that is included with Jaws PDF Library. Running MakoDistllerCmd without arguments displays usage information:

Makodistillercmd Usage

```
================================================================
(C) Copyright 2020 Global Graphics Software Ltd.
All Rights Reserved.
================================================================

Usage: makodistillercmd <arg_file>

Options
-------

Mode switches:
-d : distiller mode, convert PS files to PDF (default)

Distill (PDF output) mode options:
-dc... : Colour image options (see below)
-dd<dpi> : resolution of the whole file (default 72)
-dfe : embed fonts
-dfs : subset fonts
-dg... : Greyscale image options (see below)
-dm... : Monochrome image options (see below)
-dP<format> : PDF file format (for example: -dP1.3 or -dP1.4)
-dta : Apply transfer functions
-dtp : Preserve transfer functions
-dtr : Remove transfer functions
-dz : Flate/Zip compress text
-dZ<option> : PDF1.5 object compression; option can be None, Tags or All
-dZ is equivalent to -dZAll; no -dZ means -dZNone

Distill (PDF output) mode image options:
-dc* : colour image options
-dg* : greyscale image options
-dm* : monochrome image options
```

```
where * is one or more of the following:
A = Auto compression
(NB: for colour/greyscale only, set JPEG options
with l/m/h/q switches)
f = Flate/Zip compression
p = Flate/Zip with Predictor compression
l = JPEG low compression (QFactor 0.1)
m = JPEG medium compression (QFactor 0.5)
h = JPEG high compression (QFactor 1.3)
q<QFactor> = JPEG compression using specified QFactor
(NB: JPEG is for colour/greyscale only)
c = CCITT compression (monochrome only)

Font-related options:
-fp<fontpath>: specifies the font directory (must occur BEFORE
-fa or -fr if they're used). If you use this,
remember to put the Font\*.* files in the specified
directory! (same as the -Pf option)
-fa<filename>: adds the font filename so that Mako can use it,
the filename can contain wildcards
-ff<filename>: lists the names of the fonts that are available
in the specified font file
-fr<fontname>: removes the named font from Mako (this switch
may be repeated if necessary)

Path options:
-Pf<fontpath> : sets the path where font files can be found (same as -fp).
-Pr<respath> : sets the path where the external resource files are found.

PostScript injection:
-J<when><type><source><data> :
where <when> is p for prolog or e for epilog
and <type> is d for PDF output
and <source> is f for a filename (data is a filename)
or c for command line (data is PostScript code)
and <data> specifies PS code (literal or a filename) to be injected
into the stream fed to the interpeter at the specified point for
the specified type of output. Implicit newline characters are
added to the start and end of the data if <source> is c.
Example: '-Jpdf prolog.ps' would prefix all jobs producing PDF output
with the PostScript code contained within the file 'prolog.ps'
Example: '-Jedcshowpage' would perform an extra 'showpage' after all jobs
producing PDF output.
Note that at most only one of each of the four -J<when><type> options
should be used.

Miscellaneous options:
-o<filename> : overrides the default output file name
-i<options> : passes the <options> string verbatim as extra options.
<options> is a semicolon-separated list of key=value pairs.
Note: only ONE -i argument can be supplied and it should be
the first argument on the command line.
Valid options are:
defaultpanosestyle
panosedb

-h or -? : this usage information


Example:
--------
makodistillercmd "dist.args"

where dist.args is the configuration file for this particular
instance of distiller.

Sample dist.args
----------------
-d
test1.ps
test2.ps
```

In many cases it will be possible to swap a `testpdflibcmd.exe` command for `makodistillercmd.exe` command in a script or batch file, when the distill function of PDFLib (PostScript to PDF) is specified.

Matching the results of PDFLib

Internally PDFLib executes a PostScript prolog that prepares for processing on Windows. As `IDstiller()` is a Mako class and therefore inherently cross-platform, this prolog has not been built in.

However it is recommended for use with MakoDistillerCmd on Windows to ensure its behavior matches that of PDFLib. The prolog file can be downloaded from the link to the right.

To use it with MakoDistillerCmd, add this switch to your args file:

`-Jpdfprologue-pdflib+enabletransparency+pjl.ps`

So a complete args file might look like this:

```
-d
-dd300
-Jpdfprologue-pdflib+enabletransparency+pjl.ps
MyPostScriptFile.ps
```

API

An instance of the JawsMako Distiller class.

`#include <distiller.h>`

Public Types	

enum	ePDFVersion {
	ePDF1_3 = DIST_MAKE_PDF_VERSION(1, 3),
	ePDF1_4 = DIST_MAKE_PDF_VERSION(1, 4),
	ePDF1_5 = DIST_MAKE_PDF_VERSION(1, 5),
	ePDF1_6 = DIST_MAKE_PDF_VERSION(1, 6),
	ePDF1_7 = DIST_MAKE_PDF_VERSION(1, 7)
	}

	Supported PDF versions.
enum	eTransferFunctionMethod { eTRApply, eTRRemove, eTRPreserve }
	Enumeration for transfer function methods.
enum	eImageCompression { , eICDCT }
	Enumeration for image compression formats.
enum	eJpegQuality
	Enumeration for JPEG quality.

Public Member Functions	
virtual void	setParameter (const U8String &param, const U8String &value)=0
	Apply a key value pair output parameter with a string value. The parameter name is case insensitive. Please refer to the supplied documentation for the details of the available parameters and their ranges.

virtual void	setPdfVersion (ePDFVersion version)=0
	Set the PDF version to generate.
virtual void	setCompressPages (bool compressPages)=0
	Set whether or not compression should be applied to page content. The default is true.
virtual void	setSubsetFonts (bool subset)=0
	Set whether embedded fonts should be subset or not. The default is true.
virtual void	setEmbedFonts (bool embed)=0
	Set whether fonts should be forcibly embedded. The standard 14 fonts are not affected by this. The default is true.
virtual void	setResolution (float resolution)=0
	Set the target resolution for the output, in dots per inch. The default is 288. Equivalent to calling setParameter with the parameter name "Resolution".
virtual void	setTransfers (eTransferFunctionMethod transfers)=0
	Set the desired method to for controlling transfer functions.
virtual void	setColorImageCompression (eImageCompression compression)=0
	Set the image compression for color images. The default is eICFlate.
virtual void	setGrayImageCompression (eImageCompression compression)=0
	Set the image compression for gray images. The default is eICFlate.
virtual void	setMonoImageCompression (eImageCompression compression)=0
	Set the image compression for monochrome images. The default is eICCCITT.
virtual void	setColorJPEGQuality (eJpegQuality quality)=0
	Set the JPEG quality to use when compressing color images in DCT format. Equivalent to calling setParameter() with the parameter name "ColorJPEGQuality" enumeration value as a string ("Low", "Medium", "High" or "User"). The default is eJQLow.

virtual void	setColorQFactor (float qfactor)=0
	Set the QFactor to use when compressing color images in DCT format. Applies only when setting the JPEG quality to eJQUser with setColorJPEGQuality().
virtual void	setGrayJPEGQuality (eJpegQuality quality)=0
	Set the JPEG quality to use when compressing gray images in DCT format. Equivalent to calling setParameter() with the parameter name "GrayJPEGQuality" enumeration value as a string ("Low", "Medium", "High" or "User"). The default is eJQLow.
virtual void	setGrayQFactor (float qfactor)=0
	Set the QFactor to use when compressing gray images in DCT format. Applies only when setting the JPEG quality to eJQUser with setColorJPEGQuality().
virtual void	setProlog (const IInputStreamPtr &prolog)=0
	Set a prolog stream to be consumed by the PostScript interpreter before the input stream is processed, or NULL to clear.
virtual void	setEpilog (const IInputStreamPtr &epilog)=0
	Set a epilog stream to be consumed by the PostScript interpreter after the input stream is processed, or NULL to clear.
virtual void	setDefaultPanoseStyle (const U8String &defaultPanoseStyle)=0
	Sets the default Panose style to be used when emitting a CID font that is not in the Panose list, or NULL to clear. The Panose style must be a 12-byte hexadecimal string as described in the PDF reference, for example "010502020300000000000000".
virtual void	setPanose (const IInputStreamPtr &panose)=0
	Sets a stream that refers to a list of CID font names and their associated Panose styles.
virtual void	getFontNames (const U8String &font, CU8StringVect &names)=0
	Gets a list of font names from a given file.
virtual void	addFont (const U8String &font)=0
	Adds a font.

virtual void	addFonts (const CU8StringVect &fonts)=0
	Adds fonts. More...
virtual void	removeFont (const U8String &fontName)=0
	Removes a named font. More...
virtual void	setFontDevicePath (const U8String &path)=0
	Sets the font device path. More...
virtual void	setResourceDevicePath (const U8String &path)=0
	Sets the resource device path. More...
virtual void	distill (const IInputStreamPtr &inputStream, const IOutputStreamPtr &outputStream, const IProgressMonitorPtr &progressMonitor=IProgressMonitorPtr())=0
	Convert (distill) an input PostScript stream to PDF. More...
Public Member Functions inherited from IRCObject	
virtual void	addRef () const =0
	Increases the reference count of the actual object pointed to. This would take place during an assignment or copying.
virtual bool	decRef () const =0
	Decreases the reference count of the actual object pointed to. When the reference count falls to Zero, it deletes the actual object pointed to. More...
virtual int32	getRefCount () const =0
	Retrieve the current reference count of the actual object pointed to.

Static Public Member Functions

static JAWSMAKO_API IDistillerPtr	create (const IJawsMakoPtr &jawsMako)
	Create a IDistiller interface.