PDF Optimizer

Documentation

FyTek, Inc.

Web site: http://www.fytek.com

FyTek's PDF Optimizer

Trademarks

FyTek Disclaimer

Acknowledgments

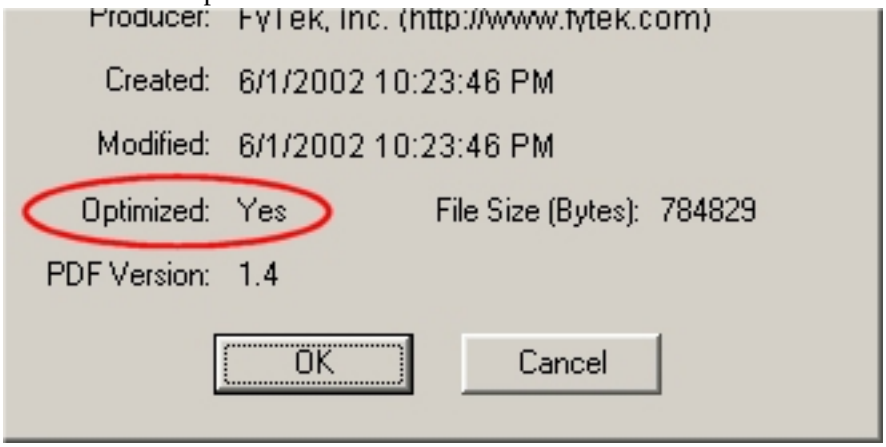
Software Development: Mike Bernardo
Writing: Mike Bernardo

FyTek, Inc.
2335 Pontiac Lake Road
Waterford, MI 48328

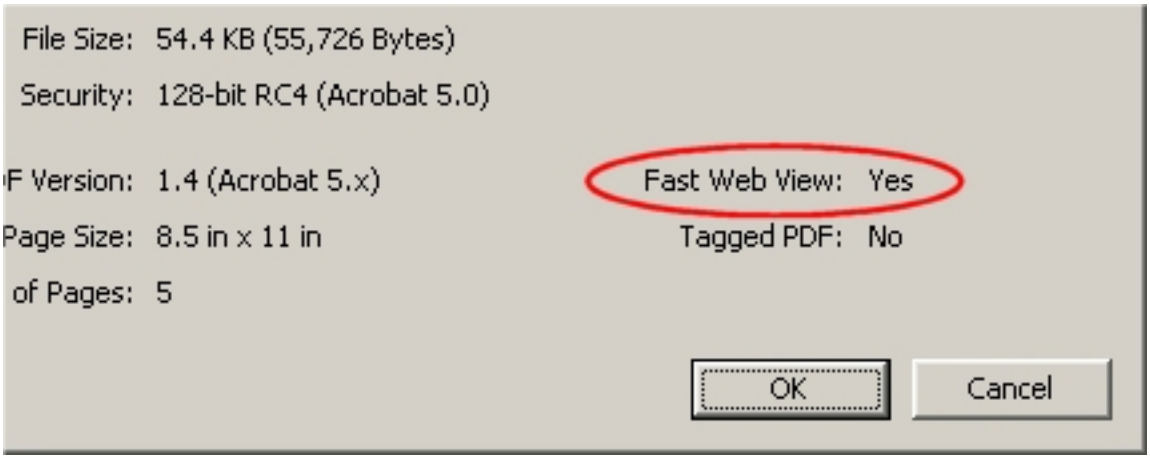
Introduction

PDF Optimizer is a program to convert standard PDFs into optimized PDFs. Optimized, or linearized, PDFs display much quicker when viewed via the web. An optimized PDF is one that has been restructured, along with some additional information, that allows the web server to transmit only the portion necessary for the first page to display in the user's browser. Keep in mind you must use a web server capable of byteserving. Adobe has information at their web site about servers and versions that support byteserving at http://www.adobe.com/support/techguides/acrobat/byteserve/byteservmain.html.

The document properties in Acrobat or Reader show if a PDF has been optimized. In Acrobat 4, the term optimized is used. In Acrobat 5, it's called fast web view but they are both the same thing. Here's what you'll see in each version for optimized PDFs:



Acrobat 4



Acrobat 5

Introduction

PDF Optimizer can encrypt the resulting PDF as well as optimize it. The source PDF must be an unencrypted PDF or a secured PDF where you know the owner password of the secured PDF. Acrobat Reader will show a key in the bottom status bar if the PDF is encrypted (like this document is). Note that an encrypted PDF does not mean it is password protected. However, having password protection on a PDF does imply that the PDF is encrypted.

Encrypting or password protecting a PDF usually means you want to limit the ability of the end user to interact with the PDF. There are several areas you can prevent access to. These are:

- Printing the document
- Print quality allowed
- Copying of text and/or images from the document
- Changes to the document (via Adobe® Acrobat® for example)
- Changes to the form fields and annotations

There two passwords you can apply to a PDF. The first is the owner password. Opening a PDF with this password will allow you full access to the PDF when a user password has been assigned as well. This means that even if printing was disabled you will still be able to print the PDF when using this password. No password will be prompted for when opening the PDF if you secure it with only an owner password. The second is the user password. Opening a PDF with this password will restrict you based on how the PDF was password protected (printing may be disabled, for instance).

Note that the owner password defaults to the user password if only the user password is specified when securing the PDF. This means that it's possible to create a PDF where printing isn't allowed, even for the owner, since the user permissions are used.

Secured By	Result
Owner password only	No password is prompted for when opening and restrictions apply to everyone
User password only	The user password is required to open and restrictions apply to everyone
Owner and User password	A password is required to open and restrictions apply when opened with the user password

The next sections describe the options for the executable version of the program and the methods of the DLL version.

Options

Using the Executable

The program pdfoptim.exe is the Windows executable program. The syntax is:
`pdfoptim.exe` *`filein.pdf fileout.pdf`* `[options]`
or, to optimize all the PDFs in a directory:
`pdfoptim.exe` *`in-directory out-directory`* `[options]`

Note that files are not overwritten when optimizing directories of PDFs unless -force is used. A file to be optimized is skipped if it already exists in the output directory. You may also leave the output file name off and simply optimize the PDF while keeping the name the same. Only use this method when you can easily recreate the original PDF if necessary. For example, run the following to optimize a PDF called myfile.pdf, keeping the output file name the same as the input:
`pdfoptim.exe myfile.pdf`

-o *password*	Sets the owner password for the PDF. If not specified but the user password is, this is set to the user password. Also, when not specified, the owner has only the rights granted when the document was created. So for example, if -noprint was specified, then it is impossible for the owner to print the document.
-u *password*	Sets the user password for the PDF. No password is prompted for when opening the PDF if only an owner password was specified. This will allow you to restrict users from printing, for example, without requiring a password to open the document.
-inowner *password*	Use this option if the input PDF is encrypted. Pass in the owner password for the input PDF.
-keepsec	Use this option if the input PDF is encrypted and you want to keep the same security settings, including owner/user passwords.

Options

-noprint	Disables printing of the document. To create a PDF with both printing and copying disabled for the user you would run something similar to: `pdfoptim.exe filein.pdf fileout.pdf -o abc123 -u xyz -noprint -nocopy` The file could only be opened by someone who knows one of the two passwords (abc123 or xyz). Using a password of abc123 gives full access while using the password of xyz does not allow printing or copying of text.
-nochange	Disables changes to the document.
-nocopy	*40-bit*: Disables copying of text and/or graphics from the document. *128-bit*: Disables copying of text and/or graphics from the document other than in support of accessibility to disabled users or for other purposes.
-noannote	Disables add/change of form fields or annotations.
-nofillin	*(128-bit only)* Disables fill in interactive fields when -noannote is used.
-noextract	*(128-bit only)* Disables extraction of information in support of accessibility to disabled users or for other purposes.
-noassemble	*(128-bit only)* Disables assembly (insert, rotate, delete pages or create bookmarks) when -nochange is used.
-nodigital	*(128-bit only)* Disables printing at digital quality - can only print low resolution. The -noprint option overrides this option so you'll want to use -noprint or -nodigital but not both.

Options

-e128	Sets 128-bit encryption method. Files encrypted with 128-bit encryption can only be opened with Acrobat or Acrobat Reader 5.0 or above. The default encrytion is 40-bit which works with Acrobat and Acrobat Reader 4.0 and above.
-norights	Turns off all rights (default is all are granted). Setting of options such as -noprint or -nocopy turns those rights on rather than off. Use this if you typically are turning off most or all of the rights.
-force	Turns off the prompt asking if it's OK to overwrite the output file if it already exists. Also used to force overwrite when converting a directory of PDFs.
-ro	Sets any interactive fields in the PDF to read-only.
-s	Include subdirectories when input/output are directories rather than individual files.
-title *Title*	Sets the document title.
-subject *Subject*	Sets the document subject.
-author *Author*	Sets the document author.
-keywords *Keywords*	Sets the document keywords.
-creator *Creator*	Sets the document creator.
-producer *Producer*	Sets the document producer.
-creationdate *YYYYMMDDHHmmSS* or *today*	Sets the document creation date. Provide a date in the format shown or use the word today to use the current system date/time. For example, to set to January 15th, 2002 at 4:15:30 PM, enter 20020115161530.
-moddate *YYYYMMDDHHmmSS* or *today*	Sets the document modification date. Provide a date in the format shown or use the word today to use the current system date/time.
-e *path-file*	Sets the error/result file. Use this option to check if any errors were encountered. This file will contain the word OK if the new PDF was created.

Options

-deleteinput	Deletes the input file when done creating the optimized PDF. Use with caution. Only use if you have a backup of the original or you can easily create it again if necessary.
-np	Turn off the box that shows how far along the program is in building the pdf.
-pbt *title*	Sets the title of the progress box.
-pbm *message*	Sets the message of the progress box.
-btn *text*	Sets the text for the button (default is "Cancel").
-nospin	Turns off the spinner in the dialog box.

Using the DLL (Dynamic Link Library)

The file pdfoptim.dll is the dynamic link library. This file should reside in your Windows or Winnt directory under the system32 sub-directory. You first must register the DLL on your system (note this step happens automatically when you run the setup program). Do this by running

regsvr32 pdfoptim.dll

You should see a message box that reads:

DllRegisterServer in pdfoptim.dll succeeded.

Click OK to continue. You are now ready to use the DLL. To use with Visual Basic, go to the Project|References dialog from the Visual Basic menu and add the reference to pdfoptim.dll.

Note that files are not overwritten when optimizing directories of PDFs unless setForce is used. A file to be optimized is skipped if it already exists in the output directory. A file will be overwritten if it exists, however, if you are optimizing a single PDF.

The methods of pdf.Optimize are:

setInFile(path-file)	Full path and name of the input file to optimize or a directory. This must be an existing unencrypted PDF or a directory containing unencrypted PDFs. You'll need to specify a different directory for the output using the setOutFile method if this is set to a directory.
setInStream(path-file)	Pass a PDF as a string, all at once or in chunks, for processing rather than an existing PDF. Use this method instead of setInFile if you built the PDF and have the stream in memory and just want to pass it to the program.
setOutFile(path-file)	Full path and name of the output file or the directory to store optimized PDFs in if setInFile is a directory. This is the optimized version of the input PDF.

DLL Methods

pdfOptimize	Call this method to build the optimized PDF. The return values are: 0 = Optimized PDF was built successfully -1 = Can't open input file -2 = Can't open output file
setOwner(password)	Sets the owner password for the PDF. If not specified but the user password is, this is set to the user password. Also, when not specified, the owner has only the rights granted when the document was created. So for example, if setNoPrint was specified, then it is impossible for the owner to print the document.
setUser(password)	Sets the user password for the PDF. No password is prompted for when opening the PDF if only an owner password was specified. This will allow you to restrict users from printing, for example, without requiring a password to open the document.
setInOwnerPass(password)	Use this method if the input PDF is encrypted. Pass in the owner password for the input PDF.
setKeepSec	Use this method if the input PDF is encrypted and you want to keep the same security settings, including owner/user passwords.
setNoPrint	Disables printing of the document.
setNoChange	Disables changes to the document.
setNoCopy	*40-bit*: Disables copying of text and/or graphics from the document. *128-bit*: Disables copying of text and/or graphics from the document other than in support of accessibility to disabled users or for other purposes.
setNoAnnote	Disables add/change of form fields or annotations.
setNoFillIn	*(128-bit only)* Disables fill in interactive fields when setNoAnnote is used.

DLL Methods

setNoExtract	*(128-bit only)* Disables extraction of information in support of accessibility to disabled users or for other purposes.
setNoAssemble	*(128-bit only)* Disables assembly (insert, rotate, delete pages or create bookmarks) when setNoChange is used.
setNoDigital	*(128-bit only)* Disables printing at digital quality - can only print low resolution. The setNoPrint method overrides this option so you'll want to use setNoPrint or setNoDigital but not both.
setNoRights	Turns off all rights (default is all are granted). Calling of methods such as setNoPrint or setNoCopy turns those rights on rather than off. Use this if you typically are turning off most or all of the rights.
setEncrypt128	Sets 128-bit encryption method. Files encrypted with 128-bit encryption can only be opened with Acrobat or Acrobat Reader 5.0 or above. The default encrytion is 40-bit which works with Acrobat and Acrobat Reader 4.0 and above.
setForce	Used to force overwrite when converting a directory of PDFs.
setReadOnly	Sets any interactive fields in the PDF to read-only.
setSubDir	Include subdirectories when input/output are directories rather than individual files.
setTitle(title)	Sets the document title.
setSubject(subject)	Sets the document subject.
setAuthor(author)	Sets the document author.
setKeywords(keywords)	Sets the document keywords.
setCreator(creator)	Sets the document creator.
setProducer(producer)	Sets the document producer.

DLL Methods

setCreationDate Year, Month, Day, Hour, Minute, Second	Sets the document creation date. Provide the values or just pass in 0 to set to current system date/time. The only required value is Year. So, for instance, you may specify Year, Month, Day and leave the rest off.
setModDate Year, Month, Day, Hour, Minute, Second	Sets the document modification date. Provide the values or just pass in 0 to set to current system date/time. The only required value is Year. So, for instance, you may specify Year, Month, Day and leave the rest off.
setErrFile(path-file)	Sets the error/result file. Use this option to check if any errors were encountered. This file will contain the word OK if the new PDF was created.
setDeleteInput	Deletes the input file when done creating the optimized PDF. Use with caution. Only use if you have a backup of the original or you can easily create it again if necessary.

DLL Examples

Here is an example of calling the DLL using Visual Basic.

```
Set PDF = CreateObject("pdf.Optimize")
PDF.setInFile "c:\temp\hello.pdf"
PDF.setOutFile "c:\temp\hello2.pdf"
PDF.setOwner ("abc")
PDF.setUser ("123")
PDF.setEncrypt128
PDF.setNoPrint
PDF.setCreationDate 2002, 2, 15
PDF.setModDate 0
rslt = PDF.pdfOptimize
If rslt <> 0 Then
  MsgBox ("Error " & rslt)
End If
Set PDF = Nothing
```

Here is an example of calling the DLL using PowerBuilder.

```
OLEObject PDF
PDF = CREATE OLEObject
li_rc = PDF.ConnectToNewObject("pdf.Optimize")
PDF.setInFile "c:\temp\hello.pdf"
PDF.setOutFile "c:\temp\hello2.pdf"
PDF.setOwner ("abc")
PDF.setUser ("123")
PDF.setEncrypt128
PDF.setNoPrint
PDF.pdfOptimize
```

Here is an ASP example creating the PDF and redirecting the browser.

```
<%
Dim PDF
Set PDF = Server.CreateObject("pdf.Optimize")
PDF.setInFile ("c:\temp\hello.pdf")
PDF.setOutFile ("c:\inetpub\webpub\output\hello2.pdf")
PDF.setOwner ("abc")
PDF.setUser ("123")
PDF.setNoPrint
PDF.pdfOptimize
Response.redirect("output/hello2.pdf")
set PDF = nothing
%>
```

DLL Examples

Here is an example using C.

```
#include <iostream.h>

// The import directive reads the typelib information from the DLL
// and creates pdfoptim.tlh and pdfoptim.tli, which are included.
// These define wrappers for each of the pdfoptim object methods.

#import <pdfoptim.dll>

// Using VC++ 5.0 Smart Pointers makes this much easier.
// The parameter string for a method is converted to Unicode, allocated
// and passed as a variant. The wrappers call IDispatch::Invoke
// This is all compatible with MFC (use AfxOleInit insead of CoInitialize, etc.).
int main(int argc, char* argv[])
{
  HRESULT      hr;

using namespace pdfOptimize_TypeLib;

  hr = CoInitialize (NULL);    // Initialize COM
  if (SUCCEEDED(hr))
  {
    try    // Each of the following lines can throw exceptions
    {
      // Create the instance and get a pointer to the interface
      IpdfOptimizePtr pPDF(__uuidof(pdfOptimize));
      pPDF->setInFile (_bstr_t(L"c:\\temp\\hello.pdf"));
      pPDF->setOutFile (_bstr_t(L"c:\\temp\\hello2.pdf"));
      pPDF->setOwner (_bstr_t(L"abc"));
      pPDF->setUser (_bstr_t(L"123"));
      pPDF->setEncrypt128 ();
      pPDF->setNoPrint ();

      _variant_t outval = pPDF->pdfOptimize (); // Build the optimized PDF file
    }
    catch (_com_error e)
    {
      cout << e.ErrorMessage() << endl;
    }
  }
  else
    cout << "CoInitialize Failed" << endl;

  CoUninitialize(); // Uninitialize COM

  return 0;
}
```