

A - Shell

PDFX

Reference

This edition published February 2007

Copyright © 2007 Jack McGregor

Copyright © 1995 - 2007 Jack McGregor
All rights reserved.

Publisher

MicroSabio
22704 Ventura Blvd., #476
Woodland Hills, CA 91364 USA
Web site: www.microsabio.com
Email: info@microsabio.com
voice: 818-710-8437
fax: 818-704-4882

Authors

Jack McGregor has been developing and documenting A-Shell since 1994, and his authorship continues through the date of this publication. Ty Griffin was responsible for editing, formatting, spell-checking and publishing.

Documentation

The [MicroSabio web site documents page](#) contains current documentation on A-Shell and related software products from MicroSabio. The documents in the A-Shell collection are available in multiple formats: HTML, Windows Help (CHM) and PDF, to name the most common. If you do not find the document format you prefer, [contact MicroSabio](#) and inquire.

The primary A-Shell documentation set consists of the following:

- Set-Up Guide
- Command Reference
- Development Guide
- XCALL Reference

See "Help Menu Links" in the *A-Shell Setup Guide* for information on accessing A-Shell documentation from within A-Shell.

Support

MicroSabio provides extraordinarily good support for A-Shell and its other software products.

The first place to go with a question is the [A-Shell bulletin board](#) ("BBS"). You can search the BBS by keyword, quite possibly finding the answer(s) you need. If not, then post your question there; you have to create an account to post a question, but doing so is painless and free. All of the forums are moderated by MicroSabio, which means that if another user or developer does not answer your inquiry, MicroSabio staff will.

The main [MicroSabio web site](#) contains a variety of documentation, utility, and other support pages and downloads.

The [A-Shell Advanced Support Network](#) contains updates, utilities, technical notes, and various other A-Shell resources, mostly of a somewhat technical nature. It also requires that you create an account, but, like the BBS, doing so is painless and free.

If you are not getting satisfaction via these support avenues, [send MicroSabio an email](#) with your question and it will be answered promptly.

Legal Notice

This documentation and the software it describes contain proprietary information belonging to MicroSabio, a California proprietorship owned entirely by Jack McGregor. The software and this related information is provided under a license agreement containing restrictions on use and disclosure, and is protected by copyright law. This information is confidential between MicroSabio and the client, and remains the exclusive property of MicroSabio.

Due to continued product development, the information contained herein may change without notice. It is believed to be accurate and reliable, at least at the time of its writing. However, no responsibility for the accuracy, completeness or use of this information is assumed by MicroSabio. If you find any problems in the documentation, please report them to the publisher.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the expressed or licensed permission of MicroSabio.

Trademarks

MicroSabio, A-Shell, INMEMO, INFLD and EZ-SPOOL are trademarks of Jack McGregor.

AIX and Informix are registered trademarks of [International Business Machines Corporation](#).

Alpha Micro, AMOS, and AlphaBASIC are registered trademarks of [Alpha Micro Products, Inc.](#)

HP-UX is a trademark of the [Hewlett-Packard Company](#).

Linux is a registered trademark of Linus Torvalds.

Microsoft and Windows are registered trademarks of [Microsoft Corporation](#).

Red Hat is a trademark of [Red Hat, Inc.](#)

UNIX is a registered trademark of [The Open Group](#).

ZTERM is a trademark of Rod Hewitt at [Cool.stf](#).

All other products, services, companies, events and publications are trademarks, registered trademarks or service marks of their respective owners.

Contents

Introduction	7
Overview.....	7
PDF Benefits.....	8
PDFX Options.....	9
PDFX FAQ.....	10
Operational Description.....	11
Related Information.....	12
Operations	13
Setup.....	13
Install PDFX Printer Driver.....	14
Create A-Shell PDF Printer.....	17
Installing and Configuring ATE.....	18
Installing and Configuring AshLPD.....	19
Test.....	20
Other Issues.....	21
Unload Driver.....	21
Hiding Progress Window.....	21
Using PDF-XChange Directly.....	21
GDI Print Commands	23
Bookmarks.....	25
Compression.....	26
Convert.....	27
Downsample.....	28
Email.....	29
Fonts.....	32
General.....	33
Info.....	34
Links.....	35
Optimization.....	36
Overlay.....	37
Paper.....	39
Save.....	40
Security.....	42
Security.Digisig.....	43
Watermarks.....	44

Introduction

Overview

"PDFX" is the name given to a PDF-generation process that has been built into A-Shell. In simplest terms, the PDFX process of generating and processing a PDF document works like this:

- A report is generated by A-Shell, and any desired PDF control variables ("Save file as...", "Email to...", "Require password," etc.) are embedded into the report as GDI print commands. For example, the command to set the document author to "Bob Jones" is //PDFX,Info.Author,"Bob Jones."
- The report is sent to a PC and "printed" to the PDFX driver.
- The PDFX driver interprets the embedded commands and produces, saves and post-processes the PDF document.

This process is platform-independent in that (a) the commands controlling the PDF process are generated within A-Shell, which may be running on any platform, and (b) the execution of those commands takes place on a PC workstation. How the report moves from its point of origin in A-Shell to the PC where it will be executed depends on the platform(s) involved; going from Linux or AIX to a PC is different than going from PC to PC, or from a PC to itself.

PDFX is the implementation of a third-party software product called "PDF-XChange" which MicroSabio has licensed and adapted for use with A-Shell.

PDF Benefits

PDF documents have become a de-facto standard in the world's electronic community. Virtually all business and home computer users have the free Adobe Acrobat reader installed and running on their computers, and can easily get it if they don't. This means that PDF documents are readable by anybody, anywhere, and can be used more or less universally. So instead of worrying about whether intended users would be able to read documents produced in formats a through z, organizations can now produce documents using the PDF standard with the near-certainty that they will be readable by the intended audience. Next week or next year or ten years from now. On PCs and Macintoshes and Linux machines and whatever else comes along.

The generation of PDF documents can be extremely helpful to an organization, and can significantly improve operations and reduce costs. As one example, consider invoices; using PDFX, you may easily specify that instead of sending invoices to a printer where they begin a long and expensive journey involving paper and the postal service, instead they get "printed" to PDF documents and automatically emailed to the intended recipient. This is really quite easy to do and would likely involve very minor changes to an application. The commands—ALL of the commands—that would have to be added to the invoice/report in order to have it produced as a PDF document and emailed are:

```
//PDFX,Email.Type,2
//PDFX,Email.Content,"Here is your invoice from ABC Manufacturing."
//PDFX,Email.To,"jill@microsabio.com, josie@microsabio.com"
```

The first command tells PDFX to use the PC's default email client program and send the email without user intervention. The second command specifies some text for the body of the email, and the third line gives the "send to" address. That's all there is to it. Really!

The arguments for using the PDFX method of generating PDF output, versus using some other package (such as PDF995, Adobe, etc.), are:

- It works like a printer driver so no coding changes to existing reports are required
- It offers an interface, accessed via the //PDFX command, which allows you much more control and flexibility than most printer-driver implementations of PDF writers.

PDFX Options

PDFX provides a convenient and familiar method for communicating from a report program to the PDF-generation process. Among other things, PDFX provides the following options, tools, methods and capabilities:

- **Image conversion** (from color to gray scale or black and white) enhances compression and reduces file size.
- **Image resolution (dpi) adjustment** is helpful in reducing file sizes.
- **Fonts:** choose to embed or not embed the fonts used. Embedding fonts increases file size, but ensures viewing consistency. Not embedding fonts reduces file size.
- **Non ASCII characters:** PDFX fully supports Chinese, Japanese and Korean fonts and characters.
- **Watermarks:** You can add watermarks to your documents, using both text and images.
- **Overlays** allow you to utilise a PDF page as a "merge document." This is most useful when printing simple reports onto the PDF equivalent of pre-printed forms, thereby ending up with a merged document that shows the form and all the data.
- **Security:** choose whether to allow users to view, print, extract or add content to your files and password protect access to the functions allowed. Also allows you to use a Digital Signature on your PDF documents and to use the new Acrobat Reader 7 Page Markup/Commenting feature.
- **Links** processing allows you to tell the PDF-creation process to look for, recognize, mark and "make live" text phrases that look like "www.abc.com," "http://www.abc.com," "info@abc.com," etc. When this feature is enabled, all such links in the document become "hot."
- **Bookmarks:** by turning on this option and telling PDFX what to look for (font size, face, etc.), PDFX will analyze the document and turn all recognized headings into bookmarks--i.e., a hierarchical table of contents.
- **Save options** allow you to specify where to save files, what and how to name them, etc.
- **Email:** these options allow you to send out your file after creation via your default MAPI software or choose to setup a simple SMTP server.(more than 1 allowed) and send you email out using this method with the PDF generated file attached with a message. PDF-XChange also includes a means to enclose your PDF in an industry standard ZIP file to attached to the email.
- **Paper Properties:** A wide range of standard paper sizes are supported, and you can create and save your own custom paper sizes for future use. You can set the resolution from 50 to 2400 DPI.
- **Scaling** allows you reduce or enlarge output to better fit the desired page size.
- **"Pages per page:"**set up to 16 pages per sheet so that many pages from your original file can fit on to a single page within your PDF file.
- **Compression and optimization** settings and standards are available for for both images and text, helping to reduce file size.

PDFX FAQ

Here are some common questions and answers on PDFX and its capabilities.

Will current GDI formats convert to PDF with this utility?

All of A-Shell's GDI print commands work as they did before; there is no change to that whole set of operations. PDFX introduces an additional batch of GDI print commands that are specifically for handling PDF-related functions. Just as the font commands begin with //SETFONT, so do all of the PDFX command begin with //PDFX,.

In case that didn't answer your question: yes, everything that currently prints through the A-Shell GDI printing system (i.e. anything with PASSTHROUGH=OFF, whether or not there are actually explicit GDI commands in it) should come out identically via PDFX.

The //PDFX commands are just for extra functions that only apply to PDFX. They will will be ignored if you send the report to a non-PDFX printer.

I want to suppress the A-Shell printer dialog. What is NOABORTDLG and how does it work?

From the *A-Shell Setup Guide* comes the following

```
NOABORTDLG = <boolean>
```

By default, when A-Shell is printing, it displays a typical Windows printing abort dialog, which allows the user to abort the printing operation before it completes. Although this is typical Windows practice, the abort dialog could be inappropriate for a couple of reasons. One is that all but the largest printouts are generated so quickly that the abort dialog is an annoying flash that appears and disappears before the user has a chance to even see what it said. The other is that you might not want to make it so easy for the user to accidentally abort a printout. Whatever the motivation, you can eliminate the dialog by specifying the NOABORTDLG = TRUE option in the printer initialization file.

Operational Description

The PDFX document generation process is almost ridiculously easy to envision and understand. Because A-Shell is used in different environments (Linux, AIX, Windows, etc.), there are different ways that A-Shell and therefore PDFX are configured. But the block diagram is the same in all cases:

- A virtual printer is set up in A-Shell. This consists of building a simple .INI or PQI printer initialization file, just like one would do for any other printer.
- A report is generated in A-Shell, and appropriate PDFX commands are programmatically embedded in the report. Examples of GDI commands, which are formatted and handled like other GDI commands, are functions like "Overwrite file if it already exists," "Email finished document to specified address," "Analyze for links and make the links live," "Embed fonts," etc. A GDI command looks like this: //PDFX,Save.WhenExists,1.
- The report is sent to a PC where the PDFX driver is installed, and the driver interprets and executes embedded PDF commands.

That's basically all there is to it. There are other details to attend to in the set up, of course, such as installing the PDFX drivers on the applicable PC(s), and making sure that the report is delivered to the PC(s) where the PDFX drivers are installed.

In the simplest A-Shell environment, a stand-alone Windows PC running A-Shell, enabling PDF generation and processing is a matter of perhaps five minutes; define an A-Shell virtual printer, install the PC drivers, and print. Really. That's all there is to it.

In more complicated A-Shell environments, delivering the report from A-Shell running on, say, AIX or Linux, to one or more PCs on a connected LAN, is a little more involved; but A-Shell has the tools and functionality to manage delivery of the reports in a straightforward matter. In such circumstances, one would use either ATE (A-Shell Terminal Emulator) or AshLPD (A-Shell's network print server) as the "transport" mechanism for delivering the report.

Related Information

PDFX is not included in the standard A-Shell release; it is an optional product that is licensed and purchased separately from A-Shell. Both AshLPD and ATE (A-Shell Terminal Emulator) are likewise not included with A-Shell, but rather are sold and licensed separately.

While PDFX commands originate in A-Shell, they are executed by a special printer driver than runs on a PC. The drivers must be installed on each PC where the PDF generation process takes place. These drivers, along with an installation program that may be run in either normal Windows fashion or via a command line, are provided by MicroSabio when the PDFX package is purchased.

PDFX was integrated into A-Shell in build 960 of July 2006. If you are using or plan to use PDFX, make sure your version of A-Shell is 960 or later.

Note that all of the options provided by the //PDFX interface are native characteristics of the PDF specification and generation process, which means that they are all documented and explained in Adobe's documentation for Acrobat. If the information on the //PDFX commands does not provide all of the depth or explanation that you need, consult the Acrobat help file and/or other Adobe Acrobat information resources.

There are various ways to speed up the loading of Acrobat. Here's one: <http://dwtips.com/2006/06/17/how-to-speed-up-pdf-loading-with-adobe-acrobat/>

GDI Commands versus Default Settings

The driver provides a default value for every possible PDFX setting. This means two things:

- If you don't issue a GDI command for a particular setting, the system will use its default value.
- If you wish to always use a particular value for one of the settings, you can change the default and thereby save yourself the trouble of having to specify the setting via the GDI process.

To access the native or default settings for the driver, see [Using PDF-XChange Directly](#).

Operations

Setup

Regardless of platform, environment or configuration, the following set-up steps are required. They may be performed in any order.

Update A-Shell

The PDFX environment was added to A-Shell in version 4.9.960, so make sure you are running that version or later.

Create A-Shell PDF Printer

See the topic [Create A-Shell PDF Printer](#).

Install PDFX Printer Driver

Any PC that is going to act as a "PDF Generator" must have the PDFX drivers installed. See the following topic [Install PDFX Printer Driver](#).

Windows Platform

If you are operating in a stand-alone PC environment, that's all there is to it; go to Test and you're done. The same applies if you are running A-Shell in a "standard" Windows peer-to-peer LAN situation, which is essentially a series of connected stand-alone PCs. For other PC configurations (terminal servers, remote PCs, etc.), contact MicroSabio for custom installation guidelines.

Other Platforms

If A-Shell runs on a non-Windows host (AIX, Linux, UNIX, etc.), you will be using either AshLPD or ATE as your "transport" or "delivery" method to get reports from A-Shell to the PC running the PDFX driver. Unfortunately, this will likely be the most complicated part of setting up the PDFX environment. But it's not too bad. Depending on your needs, see [Installing and Configuring ATE](#) or [Installing and Configuring AshLPD](#). Then go to Test and you're done.

Install PDFX Printer Driver

When you purchase (or evaluate) the PDFX package from MicroSabio, you will receive a standard Windows installation file containing the third-party software from PDF-XChange. This needs to be installed on the PC(s) that you'll be using as your "PDF generator(s)." Run the installation program as you would any other Windows install program. You can select whatever folders and options you wish; the defaults are recommended. Note (and don't change) the exact name of the installed printer, as you will need it later. Although the installation program says a reboot of Windows is required, this does not seem to be true.

Installation of the PDF-XChange drivers must be performed for each PC that you intend to have act as a PDF generator.

For "silent" or other under-developer-control installations, it is possible to run the installation via a command-line. See Command Line Installation for more information.

Notes

You do **not** have to read the PDF-XChange License Agreement, as it does not apply to you. Your license is granted by MicroSabio, and therefore had different terms and conditions than the native PDF-XChange license. That license is intended for single-PC-users of PDF-XChange.

You do **not** have to install the native Help, and in fact should not do so; uncheck the option on the "Select Components" dialog. The reason not to install this help file, and especially not on end-users' machines, is that it refers to some capabilities and options that are not included in the version of PDF-XChange that you are using. All of the Help information that is applicable to PDFX has been extracted and put into a Help file of that name.

Command Line Installation

This functionality is provided for Software Developers to distribute the end user of PDF-XChange to their software clients either as a standalone installation – or perhaps more commonly wrapped within their own installation executable.

The install may be run from the command line by the user or silently and as part of a developers own installation, called with certain switches to tailor the installer behavior. The Switches are listed and described in the following table.

The string below maybe called from the developers installer – or by a batch file (.bat) created by the developer – an example of which is provided with the full SDK installation supplied to developers.

The installation executable you should distribute is available directly from the web site: http://www.docu-track.com/PDFX3SA_sm.zip.

The example batch file supplied with the complete SDK installation for Developers is called SDK_API_RunInstall.bat and is installed in the main help folder.

The End User installer can install PDF-XChange to many workstations - it may be useful to be able to silently install without the need for any user interaction. This allows you to execute the install either from the command line with certain parameters and switches applied - or create a batch file (.bat file) and distribute this with the main installation executable for your users to run directly - alleviating the need for the System Administrator to visit each user and install.

Example

Here is an example of a valid command line, this one using all available switches. Note that most of the items on this command line are optional; see Parameters below for more information. Also note that this sample line has been artificially broken into separate lines for purposes of readability; in reality, this is just one long line of text.

```
C:\vm\pdf\PDFX3SA_sm.exe
/VERYSILENT
/NORESTART
/COMPONENTS="pdfSaver,PDF-XChange driver, Help,Languages"
/DIR="C:\Program Files\PDF-XChange 3 Pro\"
"/UserName:I am"
"/Organization:Favorite Customer"
"/UserEmail:my@email.com"
"/InstallProfile:C:\Documents and Settings\All Users\Templates\my Profile1.pxp"
"/AutoUnload:nn"
```

Switches

Note that all switches are optional except for DIR=.

Also note that failure to ensure that special switch characters such as quote marks (") and "/" are included in your command line or batch file, and are in the correct positions, will result in the failure of the command to behave as expected.

Switch	Description
VERYSILENT	Designates that the installation once started should be silent and require no user interaction.
SILENT	Designates that the install should only request essential information from the user once started.
NORESTART	Specify that no reboot occur when installation is complete. PDF-XChange does require a reboot before updates will take effect.
DIR=	Required. The full path to install all required files to, created if not already available, i.e.: /DIR="C:\Program Files\PDF-XChange 3 Pro" Note that the path details must begin and end with double quotation marks (").
GROUP=	the Windows 'Start Menu' folder in which to locate the Menu options provided when installing PDF-XChange - this will always be a 'Child' menu option of the default 'Programs Files' menu and this cannot be changed, i.e.: /GROUP="Tracker Software\PDF-XChange 3 Pro" Note that the path details must begin and end with double quotation marks (").
COMPONENTS =	If this switch is specified only those components actually listed will be installed, otherwise all components are installed. The available components are: pdfSaver (Always Required) PDF-XChange driver (Always Required) Help Languages For example, this will install all components except the User Interface local language files. /COMPONENTS="pdfSaver,PDF-XChange driver, Help"
NOICONS	If this option is specified, no shortcuts to PDF-XChange resources such as help files etc will be added to the Windows 'Start' Button Programs listing fro the end user client installation.
PNAME	Sets the PDF-XChange printer name as required - please note that the first 12 characters are reserved - including the space after "PDF-XChange " the default option without this entry is currently "PDF-XChange 3.0', i.e. : /PName:"PDF-XChange for my application" Also The revised Printer name must not end with a space - must be an Alpha/Numeric character.
PDEFAULT=	Sets PDF-XChange to be the system default Printer - the default option without this entry is that PDF-XChange will not be the default printer.
AutoUnload:nn	If this option is specified, after the specified timeout (nn, in minutes) the pdfSaver3 module of the PDF-XChange driver will unload. This ensures that the minimum necessary memory is in use at any time - this is only necessary where multiple users may be using PDF-XChange on a single PC, Server or Citirx/WTS system at a given time as the resources used are minimal - but could be collectively quite large in such a system as described above.
UserName:	Registration: Allows the user's <data> details to be registered during silent installation, i.e.: "/UserEmail:my@email.com"
Organization:	
UserEmail:	

Create A-Shell PDF Printer

On the host computer where A-Shell is running, define a logical printer as you would for any other logical printer—i.e., create ASHCFG:<name>.PQI or SYS:<name>.INI—as follows:

```
DEVICE=PDF-XChange
PASSTHROUGH=OFF
PITCH=AUTO
CPP=82,*
LPP=60
```

The first two commands are critical, the others recommended. If you create multiple copies of the driver in order to have multiple preset configurations, then you would add suffixes to the end of the DEVICE name—e.g., DEVICE=PDF-XChange Landscape.

The first twelve characters of the DEVICE name PDF-XChange, including the trailing space, are reserved and cannot be changed. Make sure any changes you make to the name preserve those first twelve characters.

The other commands are optional but appropriate for printing of standard text without any // GDI commands in it. (For example, you could convert a stock text file like SYS:ERRMSG.USA to PDF via PRINT <name>=SYS:ERRMSG.USA.

Installing and Configuring ATE

ATE, the A-Shell Terminal Emulator, was created to allow an A-Shell host to create and control a graphical user interface (GUI) on a connected PC. A-Shell on the host "talks" to ATE on the PC, thereby making all the GUI capabilities of Windows available for A-Shell's use. The communications channel between the Linux/AIX host and the PC "terminal" may be used to relay reports—in this case a PDFX document—regardless of whether ATE's display capabilities are used for their original purpose.

Creating PDF documents via ATE is essentially the same as any other kind of ATE printing. The basic principle is that on the application server (i.e. the machine that ATE is being used to telnet/ssh to), you define a logical printer special pseudo device AUXLOC:, e.g.:

```
DEVICE=AUXLOC:
```

When a report is sent a logical printer defined using the AUXLOC: driver, A-Shell forwards it to the telnet client via the emulation of the auxiliary port terminal interface support by ATE, ZTERM and other terminal emulators. Typically, such terminal emulators then either prompt the user to select a Windows printer, or use a previously selected printer, to forward the output to. Any such terminal emulator could thereby generate PDF documents by routing the output to a PDF-generating printer driver. However among the available terminal emulators, only ATE would support A-Shell's GDI printing commands, and only ATE will be able to pass the necessary license code to the PDF-XChange driver to enable it. So if you want to use the //PDFX commands, and/or use A-Shell's license for the PDF-XChange driver, you will have to use ATE rather than another terminal emulator.

There are two ways to pre-configure the printing options in ATE:

- Use the printer configuration dialog (Settings..Connection Properties). Using this method for PDF generation, you can either set the printer choice to <prompt>, in which case it will prompt you each time for a printer, at which time you could choose the PDF-XChange driver when you wanted to create a PDF, or you could just set the printer choice to "PDF-XChange", in which case reports printed through the auxiliary interface would go directly to that printer. In either case, make sure to select the GDI printing radio button.
- Create one or more printer init (PQI) files (logical printer definitions) and store them on the ATE client (in the 001007 directory). (The application on the server might do this under program control and arrange to have them sent automatically to the %MIAME%\DSK0\001007 directory on the ATE client in advance of printing, via ATSYNC.LIT, direct FTP, or other methods.) Once the printer definitions have been stored on the ATE client, the host application can select them by appending the logical printer name to the DEVICE=AUXLOC: statement, e.g.:

```
DEVICE=AUXLOC:PDF1
```

In the second case above, you would have multiple logical printer definitions on the application server, each corresponding to a printer definition on the ATE client (which, in the case of PDF printing, would look something like the example shown in the topic

Create A-Shell PDF Printer. While this method is a bit more complex to set up, it allows the host application the same power and flexibility in selecting printers that are otherwise under ATE's control, as it can for printers that are directly accessible from the application server.

Note that when the PDF option is licensed on the application server, the license will automatically be distributed to ATE clients.

Installing and Configuring AshLPD

AshLPD is an A-Shell/Windows application that runs on a PC and accepts print requests from other computers across the network. It may be useful in the context of PDF generation for one or more of the following reasons:

- Your A-Shell application is running on a UNIX server and is being accessed by telnet clients other than ATE. In this case, you need AshLPD (or something like it) in order to provide an A-Shell/Windows presence on the network so that you can access the PDFX functionality.
- You have a heterogeneous network in which not all of the workstations are running a form of A-Shell/Windows (such as ATE or a peer-to-peer A-Shell/Windows client) capable of accessing the PDFX driver. In this case, you might choose to install one (or more) AshLPD servers on the network for the PDF-generation benefit of those workstations.
- You choose to route (or a certain class) of your PDF output through AshLPD in order to centralize your PDF operations on one PC, or perhaps to take advantage of one of the ability of AshLPD to add more intelligence or processing power to your PDF/printing operations than might otherwise be practical with the standard methods of printing.

Whatever the motivation, each AshLPD server would be installed on a PC as follows:

- Download the AshLPD setup program from our website (ashlpd-x.x.xxx.exe) and execute it to install the program. It installs both A-Shell/Windows and the application AshLPD, and configures it to launch (from the Startup folder) as a system tray application.

Important: If you are installing "over the top" of your existing copy of A-Shell windows, be sure to change the install folder to whatever you have used before and do **not** use the default installation path.

- Contact MicroSabio for an AshLPD license, and enter it by selecting the Register option on the AshLPD menu bar. (You can also run it in demo mode, but the pop-up messages are particularly annoying since AshLPD is really meant to work as an unattended server. We can issue you an eval license if you want to test it thoroughly before purchasing a copy. Note that without such a license, all PDFs will be generated with a demo watermark.)
- Consult the AshLPD documentation for general information on configuring the network interface, printers, etc.
- Since AshLPD is intended to run without user intervention most of the time, it is important to configure appropriate PDFX options so that it will not present user dialogs (i.e. for saving the PDF file, launching the Acrobat reader, email client, etc.) You can do this via the "normal" method (of embedding them in every print file), but it will probably be more sensible to create a single set of default //PDFX command that is always invoked by AshLPD, by putting them into a file that can be specified as a prefix (via the printer init PREFIX command). The sample printer definition file, PDFLPD.PQI uses this technique, referencing PREFIX=PDFLPD.PFX. You can edit the PDFLPD.PFX file by whatever means is convenient, including this one: Use File..Exit from the AshLPD window, and select the "No" option to exit to the dot prompt. Then use VUE ASHCFG:PDFLPD.PFX to edit the file. Restart the service by entering the command ASHLPD.

Test

After completing the earlier steps of the setup, you should be ready to generate a PDF document. If you defined your new logical printer as "PDF1," then you should be able to produce a PDF document when issuing the command from the A-Shell prompt:

```
PRINT PDF1=SYS:ERRMSG.USA
```

Note that you do not have to have any GDI commands included in the document being printed in order for it to generate a PDF. All GDI commands are optional, and default values for all driver settings will be used in the absence of GDI commands.

In the case of AshLPD, there should be a pre-defined printer named PDFLPD installed on both the application server and the AshLPD server, so you should be able to test it via:

```
PRINT PDFLPD=SYS:ERRMSG.USA
```

If the connection is not working, the pop-up window will display appropriate messages on the screen of the user requesting the printout. Otherwise, the pop-up window will quickly disappear, and you can then look at the status window of the AshLPD service to verify that the file was received and "printed". (If the AshLPD window is not visible, double-click on the A-Shell icon in the system tray to display it.)

Other Issues

Unload Driver

To force the driver to be unloaded, which may be the only way to remove a progress dialog which won't otherwise go away (or to install an update of the driver), execute Start...Run... and type in:

```
"C:\Program Files\Tracker Software\PDF-XChange 3\pdfSaver\pdfSaver3.exe" /quit
```

We don't particularly like this command, but it is useful for the problem mentioned above regarding the progress bar. We hope to offer a better solution, but for the moment this is it.

Hiding Progress Window

At this time, there is not a //PDFX option to control the display of the PDFX print progress window. However, should you prefer your application not to display the print progress Window (the default behavior) there is a way for this to be switched off. The switch which controls this behavior is located in the registry under the following path:

```
HKEY_CURRENT_USER\Software\Tracker Software\PDF-XChange 3.0\Drivers\pdfSaver
```

Within this path, you can set the value of the **HideProgressWindow** field to 1 to hide the progress dialog for all PDFX printouts generated by the current user. To do this for all users on the current machine, use the equivalent path in the HKEY_LOCAL_MACHINE hive, i.e.:

```
HKEY_LOCAL_MACHINE\Software\Tracker Software\PDF-XChange 3.0\Drivers\pdfSaver
```

Note that the PDFX print progress dialog may be appear redundant or confusing when A-Shell is also displaying its own printer abort dialog. (See the NOABORTDLG printer init command in the Setup Guide for information on disabling A-Shell's printer abort dialog.)

Using PDF-XChange Directly

The PDF-XChange driver that enables PDFX has its own user interface, which you may want to access directly for the purpose of familiarizing yourself with it and just to see what it looks like. Also, two functions (Bookmarks and Watermarks) have parameters that are both too complex to pass via regular arguments, and are normally set once and then left alone. If you wish to use these functions, you will have to access the PDF-XChange driver directly.

To do so, Go to the Windows *Start* menu, then *Printers and Faxes*, then *PDF-XChange* (or whatever name you're using), then *Printing Preferences*.

Licensing

The software license granted to you by MicroSabio with your purchase of PDFX allows the use of the PDF-XChange system only with A-Shell. If you try to print to the PDF-XChange device from any application other than A-Shell, you will get conspicuous, annoying and un-defeatable PDF-XChange watermarks printed on all pages.

In order to use PDF-XChange outside the confines of A-Shell, you must purchase per-PC licenses for that purpose. These licenses are available to A-Shell users and resellers from either Tracker Software (the developer/vendor of PDF-XChange) or from MicroSabio. We would, of course, prefer you to purchase those licenses from MicroSabio.

GDI Print Commands

PDFX is the name for A-Shell's PDF-generation process. Using //PDFX commands in a properly configured and licensed PDFX environment, programmers can automatically produce PDF documents that have certain characteristics and which cause certain things to happen (email, save with a particular filename, show or don't show finished document to user, etc.). All these types of controls are managed through the GDI commands, which are documented here. For information on the larger PDFX environment—installation and set-up, licensing, operational concerns, trouble-shooting, etc.—see the section entitled "Generating PDFs."

When using the PDFX GDI commands, note the following:

- As with all GDI commands, the lead-in for PDFX is double slash (//). A double slash followed by a semi-colon indicates a comment, and the line is ignored by A-Shell—although it may be useful in understanding what you were trying to accomplish (e.g. //;the following command is a mystery to me).
- The PDFX lead-in cannot contain any spaces; the command "//PDFX..." will result in the desired command NOT being executed and a line of text saying "//PDFX..." appearing in the print file.
- Hex values in the form &h#### are supported, although this only applies to the color command.
- A-Shell does not perform any validation on the PDFX commands; the commands are simply passed to the driver if the driver is PDFX. In the absence of the PDFX (i.e., the report is directed to some other printer), PDFX commands are still legal but will be ignored.
- Also in keeping with GDI conventions is the handling of strings. When entering strings, you MAY quote the entire string (i.e., enclose it with double quote marks), but you only MUST quote the string if it contains commas. Always enclosing strings in quotes eliminates the possibility of the stray or unusual comma causing the string to be truncated at the point of the comma.
- The order in which commands are entered is entirely optional.
- In the following tables of commands, note that many of the values are shown in quotes. This means that the quoted values are valid entries for that command. The value entry 0 ("*AllOpened*"), for example, means that when specifying this value you can use either "0" or "*AllOpened*".

The PDFX variables fall into several categories, as listed below. The category keyword, with which all commands in that category begin, gives an indication of the type of control offered. Click on these links for more information on the category (what is "Downsample"?), as well as syntax, values and descriptions for the individual commands.

Bookmarks	Compression	Convert	Downsample
Email	Fonts	General	Info
Links	Optimization	Overlay	Paper
Save	Security	Security.Digisig	Watermarks

Examples

Here are a few examples showing PDFX commands from an actual print file.

Command	Result
//PDFX,Info.Title,"Test PDFX document"	Specify doc title
//PDFX,Save.ShowSaveDialog,0	Do not display Save dialog on completion
//PDFX,Save.App.Run,0	Do not run Acrobat on completion
//PDFX,Save.WhenExists,1	Automatically overwrite doc if it already exists
//; this is a comment	//; indicates comment; rest of line ignored
//PDFX,Security.Use,1	Enable security
//PDFX,Security.Level,40	Set encryption bytes
//PDFX,Security.UserPwd,jack	Set user password (for opening the document)
//PDFX,Email.Type,1	Invoke PC's default mail client
//PDFX,Email.Subject,Testing PDF	Insert "Testing PDF" as subject in email message
//PDFX,Links.Analyze,1	Convert text that looks like links into actual links

Bookmarks

Bookmarks are Adobe-speak for subsection headings, also known as table of contents entries, which are "active" in that they allow the user to navigate the PDF document by clicking on the bookmarks. When documents are large, well-designed, outline-oriented and organized well, bookmarks are quite valuable. They are not particularly valuable in small documents.

The following commands provide the means to have headings recognized as bookmarks, and to control the behavior of the bookmarks pane in Acrobat when the document is opened.

Like Watermarks, bookmarks have some attributes that can only be set in the PDF-XChange driver itself. You must call up the driver using normal Windows controls (see Using PDF-XChange Directly) and tell the driver how to recognize the text that you wish to have interpreted as bookmarks. You specify characteristics such as font, size, color and other text attributes, and the driver will mark any matching text as a bookmark. You should be able to easily figure out how to add multiple bookmark definitions and equate them to different bookmark levels.

Color matching may be difficult, because the dialog allows you to pick from color samples, whereas A-Shell would specify color printing using RGB values. The font, style and size options are fairly simple to use. Most practical would be to use a variation of a standard font that you wouldn't otherwise be tempted to use but which wouldn't look too weird in the printed document.

If you were to tell the driver that you wanted to interpret 14 point bold Arial Black as bookmarks level 2, for example, you could create a bookmark in the document by including the following:

```
//SETFONT,140,Arial Black
This is a bookmark
//TEXTOUT,100,200,This is another bookmark
```

Key	Possible Values	Description
Bookmarks.Print	0, 1	Enables or disables analyzing of bookmarks for the document during printing.
Bookmarks.Multiline	0, 1	Enables or disables multi-line bookmarks recognition.
Bookmarks.MatchLevel	0, 1	Enables or disables Matchlevel in bookmarks recognition.
Bookmarks.DisplayMode	0 ("AllOpened") 1 ("AllClosed") 2 ("ByItem") 3 ("UpToNodeLevel")	Defines how the bookmarks tree will be displayed when document will be opened.
Bookmarks.UpToLevel	1 - 99	Defines to which node level the bookmarks tree will be opened and has meaning only when Bookmarks.DisplayMode has a value. i.e. a value of '2' would open the top level and all immediate child bookmarks only. UpToNodeLevel.

Note that *Possible Values* enclosed in quotes are valid values.

Compression

Because file size is a major concern when dealing with PDF documents, Acrobat provides a variety of ways to save space inside files. Following is a list of the compression methods available for different types of content. See Downsample for additional comments and options on image compression and reduction, and Fonts for information on the file space usage aspects embedding versus not embedding fonts.

Key	Possible Values	Description
Compression.Graphics	0, 1	Enables or disables the compression of images within the PDF document. If enabled, compression methods may be specified for each type of image format. In addition, compression may be set off for each image format.
Compression.Text	0, 1	Enables or disables compression of text within the PDF document.
Compression.ASCII	0, 1	Enables or disables using ASCII mode. Creates larger PDF files when on!
Compression.Color.Method	0 ("None"); 1 ("Auto"); 2 ("JPEG"); 3 ("ZIP"); 4 ("JPEG/ZIP"); 5 ("JPEG 2000"); 6 ("JPEG 2000/ZIP")	Set the compression method for True Color images within the PDF document.
Compression.Color.JPEGQuality	1 – 100	Set JPEG compression quality for True Color images. Only valid when JPEG or JPEG 2000 compression is set through the Compression.Color.Method.
Compression.Indexed.Method	0 ("None"); 1 ("Auto"); 2 ("RLE"); 3 ("ZIP"); 4 ("LZW")	Set the compression method for Indexed images within the PDF document.
Compression.Mono.Method	0 ("None"); 1 ("Auto"); 2 ("ZIP"); 3 ("CCITT3"); 4 ("CCITT4"); 5 ("RLE"); 6 ("JBIG2")	Set the compression method for b&w images within the PDF document.
Compression.Mono.JBIG2Method	0 ("Standard"); 1 ("Crop"); 2 ("Symbols")	Set the JBIG2 compression method for compression b&w images. All of these compression methods are lossless.

Note that *Possible Values* enclosed in quotes are valid values.

Convert

The variables provide another way to help reduce the size of the PDF files being produced. They apply only to images.

Key	Possible Values	Description
Convert.Color.To Convert.Indexed.To Convert.Drawing.To	0 ("None"); 1 ("Gray"); 2 ("Mono"); 3 ("B&W")	Specifies the output format for conversion of True Color images i.e. to decrease size of the output PDF document.
Convert.Color.Dither Convert.Indexed.Dither	0, 1	Sets the dithering usage for converting True Color and Indexed images.
Convert.Drawing.Threshold	1 - 255	When converting color images to b&w, every color is converted to a graduation of gray (from 1 to 255); if the graduated level is lower than the threshold value set, the color will be converted to black otherwise the conversion will be to white. Zero sets the threshold level to the default value for the driver, which is 128

Note that *Possible Values* enclosed in quotes are valid values.

Downsample

Downsampling refers to the process of taking a relatively high resolution file and reducing the amount of pixels used in that image. This works fine for a file that will be only viewed on a computer monitor or posted on the Web, but will significantly reduce the image quality of a file that will be digitally printed. For more precise control of file-space-saving techniques, see Compression.

Key	Possible Values	Description
Downsample.Color.Use Downsample.Indexed.Use Downsample.Mono.Use	0, 1	Enables or disables downsampling corresponding image types.
Downsample.Color.DPI Downsample.Indexed.DPI Downsample.Mono.DPI	50 - 2400	Sets the DPI value from which images (True Color, indexed, or b&w) will be downsampled (if enabled).
Downsample.Color.Method Downsample.Indexed.Method Downsample.Mono.Method	0 ("Linear"); 1 ("Bilinear"); 2 ("BiCubic")	Specifies the method of downsampling to be used to downsample True Color, indexed, or b&w images (if enabled).

Note that *Possible Values* enclosed in quotes are valid values.

Email

The purpose of generating a report is normally to communicate information to a third party. The combination of creating a PDF file and automatically sending it to the desired party via email can be extremely useful. Note the following:

- If you are going to send the PDF file via email, using Email.Type 1, 2, or 3, you should normally set the *Save.App.Run* option to no ("0") so that running Acrobat to display the just-generated PDF file doesn't visually interfere with the email process.
- If you specify Email.Type 2 but do not provide an *Email.To* address, the PDF file will be generated per normal but nothing else will happen—i.e., no email message is created or sent.
- The SMTP options provide the means to send an email using only an internal email process—i.e., NOT using the PC's normal email client.
- If *Email.Type=1* (run email client) and the other email variables are provided via the GDI commands (*Email.To*, etc.), the email client will pop up and be available for user input or override, but values provided will be inserted into the email message.
- Unless you have some specific reason for wanting to use direct SMTP mailing (*Email.Type 3*), it is recommended that you do not do so. Using the PC's default email client (*Email.Type 1* or *2*) provides benefits like putting failed messages in the Outbox, sent messages in the Sent folder, etc.
- The PDFX process does not support the sending of send multiple PDF documents in a single email message.

Key	Possible Values	Description
Email.Type	0 - no email; 1 - run with client; 2 - send with default client; 3 - send with SMTP.	Specifies how to send email.
Email.From	Quoted string	Specifies the <data> field for emailing. Only the "To" information is required for a successful email send.
Email.To		
Email.CC		
Email.Subject		
Email.Content		See following discussion.
Email.ZIP	0, 1	Specifies whether to send the generated document as a PDF file or to create and then archive as a ZIP file containing one PDF file. No 3rd party ZIP library required
The following apply only if Email.Type = 3		
Email.SMTP.Address	Quoted string	Specifies the name/IP address of the SMTP server through which to send email. Used only when Email.Type is 3.
Email.SMTP.UserName		Specifies user name/account from which to send the email via the SMTP server may or may not be required.
Email.SMTP.Password		Specifies the account password for SMTP server, if required.
Email.SMTP.Port	1 - 65535	Specifies the port to be used for SMTP server (usually 25).
Email.SMTP.UseSSL	0, 1	Specifies whether to use a Secure Connection for the SMTP server.

Examples

The following command causes the PC's default mail client (typically Outlook Express) to be displayed, and the PDF file to be attached. The email has no address, no cc, no subject; those values must be filled in by the user, who must also manually send the message.

```
//PDFX,Email.Type,1
```

These commands cause the PC's default mail client to build and send a message with no user interaction. The message is sent to Ty and Jack with courtesy copies to various other addresses, has the subject of "PDF Testing," and has a message content (body) of "This is the text..."

```
//PDFX,Email.Type,2
//PDFX,Email.To,"ty@microsabio.com,jack@microsabio.com"
//PDFX,Email.CC,"other1@microsabio.com,other2@microsabio.com,other3@microsabio.com"
//PDFX,Email.Content,"This is the text that will appear in the body of the email message."
//PDFX,Email.Subject,"PDF testing"
```

The following is a hybrid of the two examples above: it runs the user interface for the PC's default email client, preloads "PDF testing" as the subject, and then waits for the user to provide an address and manually send the message.

```
//PDFX,Email.Type,1
//PDFX,Email.Subject,"PDF testing"
```

Email.Content

This variable gets special mention because it is handled and used differently than most other variables. Note the following:

- Unlike other //PDFX commands, multiple occurrences of Email.Content are supported.
- Each occurrence of the variable starts a new line in the PDF output file unless the previous line is terminated with a backslash (\). Use the backslash line to suppress the normal line termination that otherwise occurs at the end of each line.
- The last Email.Content line must be followed by an Email.... line that is not Email.Content. In the example below, note that the final Email.Content line is followed by the Email.Subject line; that is done for the purpose of communicating "end of email content" to the driver.
- The maximum number of characters specified in a single Email.Content line is 1024. (This is why multiple Email.Content lines are supported.) The combined content of the Email.Content is limited to 32K.
- Blank Email.Content lines may be used to enter blank lines into the email text.
- This command does not offer control of text characteristics, such as font, color, etc.

Examples

The following few commands give an example of how to create a multi-paragraph message. Note how "Email.Subject" follows the "Email.Content" lines, to indicate that the "Email.Content" is finished.

```
//PDFX,Email.Type,1
//PDFX,Email.To,"jill@microsabio.com, josie@microsabio.com"
//PDFX,Email.Content,"This is the text that will appear in the body of the \"
//PDFX,Email.Content,"email message. This is more text that will appear in \"
//PDFX,Email.Content,"the same paragraph of the body of the email message. \"
//PDFX,Email.Content,"It can go on and on and on for as long as you wish. When \"
//PDFX,Email.Content,"you wish to stop and create a new paragraph, then end \"
//PDFX,Email.Content,"the Email.Content line without a slash, which will end the paragraph."
//PDFX,Email.Content,
//PDFX,Email.Content,"Then create a new Email.Content line with no text on it, then create
    another Email.Content line and resume your message. You may either break up the lines using
    the backslash as shown above, or write one or more long lines that wrap in this viewer but do
    in fact simply constitute one long time of no more than 1024 characters."
//PDFX,Email.Content,
//PDFX,Email.Content,"Yours truly,"
//PDFX,Email.Content,
//PDFX,Email.Content,"MicroSabio"
//PDFX,Email.Subject,"PDF testing"
```

Fonts

Font embedding is a fairly complicated topic that has a major impact on the size of a PDF file. By electing to embed fonts, you guarantee that the document will be reproduced exactly on other computers as it appears on this computer—but the price you pay for that precision is a larger file, sometimes very much larger. Where absolute font fidelity is not required, NOT embedding fonts is recommended; in this case, the computer on which the document is displayed does a "best guess" substitution of available fonts for the one(s) you specified.

Key	Possible Values	Description
Fonts.ExtraInfo	0, 1	Enables or disables storing additional information within a PDF for UNICODE fonts to enable support for search/edit/extract text functionality for non-Roman character sets.
Fonts.Embedd.All	0, 1	Enables or disables fonts embedding for the document. Font embedding generates larger files but ensures fonts are not substituted in the document when viewed.
Fonts.Embedd.Protected	0, 1	Enables or disables force embedding protected fonts within the document. Note that users and developers are responsible for all font licensing issues.
Fonts.Embedd.Always		List of fonts to embed within the document. Font names must be separated by the ';' symbol.
Fonts.Embedd.Never		List of fonts never to embed into the within the document. Font names must be separated by the ';' symbol.

General

Key	Possible Values	Description
General.PDFSpec	0 (Auto), 3, 4, 5, 6	Set the desired PDF format to be used (1.3, 1.4, 1.5 or 1.6). If value is 0 PDFX will automatically use acceptable PDF format.
General.PageLayout	-1 - Viewer Default; 0 - Single Page; 1 - One Column; 2 - Two Columns Left; 3 - Two Columns Right	Specifies the page layout to be used when the printed document is opened in a suitable viewer.
General.PageMode	-1 - Viewer Default; 0 - Show None; 1 - Show Bookmarks; 2 - Show Thumbnails; 3 - Full Screen	Specifies how the document should be displayed when opened.
General.VP.HideToolbars	0, 1	
General.VP.HideMenubar		
General.VP.HideUI		
General.VP.Fit		
General.VP.Center		
General.VP.ShowTitle		
General.VP.FSPageMode	0 - Show None; 1 - Show Bookmarks; 2 - Show Thumbnails; 3 - Show Optional Content Panel	

Info

These variables all relate to the properties that one may—but is not required to—associate with a document: title, subject, author, creator and a variable number of keywords. Although the variables and values in this category appear straightforward, they are actually a little tricky because of a non-obvious hierarchy of precedence. If you set *Info.Ask* to yes (1), then a dialog box will dutifully pop up and ask the user for title, subject, etc. However, the values entered by the user will be overridden by the values entered via the GDI commands for *Info.Title*, *Info.Subject*, etc, if they are present. In other words, the values entered by the user will only be used in the PDF document if there is no corresponding GDI command.

Key	Possible Values	Description
Info.Ask	0, 1	Specifies whether to show on printing the prompt dialog with the information (properties) fields for a PDF document, and allow the user to input this data prior to the file being written to disk. The five info fields, shown below, are displayed in one dialog box, and you cannot tailor that dialog to include or exclude only the fields you want; it's either all or nothing.
Info.Title	Quoted string	Specifies the <data> field in the .pdf file information structure--i.e., document properties.
Info.Subject		
Info.Author		
Info.Keywords		
Info.Creator		
Info.Use	0, 1	Specifies whether to include additional information within the PDF document such as Title, Subject, Author, etc. This variable does not appear to actually do anything.

Examples

The following tells Acrobat to not ask the user to provide properties info, to set the author to "George Author," and to ignore the rest of the properties options (because of the semicolon after the double slash).

```
//PDFX,Info.Ask,0
//PDFX,Info.Author,"George Author"
//;PDFX,Info.Creator,"Henry Creator"
//;PDFX,Info.Keywords,"Robert,George,Henry,Peter"
//;PDFX,Info.Subject,"Robert Subject"
//;PDFX,Info.Title,"Peter Title"
```

Links

The options in this category provide tools one may use to identify HTML links contained within the document, and to format those links so as to be easily recognizable to the reader of the document. If you include in your print file the text "www.microsabis.com" and do not turn on the *Links.Analyze* option below, then the phrase "www.microsabis.com" will appear in the PDF document as regular text. If you turn on the *Links.Analyze* switch, however, www.microsabis.com will (a) appear in the document as highlighted text, and (b) will function as a live link—i.e., the user may click on the link and will be directed to the appropriate website.

Key	Possible Values	Description
Links.Analyze	0, 1	Enables/disables analyzing of URLs within the document.
Links.Border.Type	0 ("None"); 1 ("Underline"); 2 ("Rectangle")	Sets border type for URLs within the document.
Links.Border.Style	0 ("Solid"); 1 ("Dashed"); 2 ("Dotted")	Sets border style for URLs within the document.
Links.Border.Width	5 - 50	Sets border width (in tens of points) for URLs within the document.
Links.Border.Color	0x000000 - 0xFFFFFFFF	Sets border color for URLs within the document. Value must be an RGB value for the color required.
Links.Files.Analyze	0, 1	Enables/disables analyzing of local filenames within the document, thus allowing a user to embed a link to a local (disk) file within the PDF document.
Links.Files.Check	0, 1	Check for the existence of the local file before link creation and link embedding within the generated PDF.

Note that *Possible Values* enclosed in quotes are valid values.

Examples

The following commands turn on PDFX's "recognize links and make them active" option, and then specifies that any such links will be marked with a ten-point underline.

```
//PDFX,Links.Analyze,1  
//PDFX,Links.Border.Type,1  
//PDFX,Links.Border.Width,10
```

Optimization

Key	Possible Values	Description
Optimization.BlendMode	0 ("Normal"); 1 ("Multiply"); 2 ("Screen"); 3 ("Overlay"); 4 ("Darken"); 5 ("Lighten"); 6 ("ColorDodge"); 7 ("ColorBurn"); 8 ("HardLight"); 9 ("SoftLight"); 10 ("Difference"); 11 ("Exclusion");	Specifies the required blending operation to be used for merging operations when drawing (for example, when the "Lines Merge" option in AutoCAD is enabled).
Optimization.DeTessellation	0 ("None"); 1 ("Simple");	Specifies the de-tessellation mode used during optimization of drawings.
Optimization.NoTransparency	0, 1	When a value of 1 is set, prevents the use of transparency drawing, currently Adobe Acrobat viewer/reader (last tested V6.1) has issues displaying/printing transparent objects.
Optimization.NoMaskedImages	0, 1	When a value of 1 is set, prevents the use of masked images during optimization, currently Adobe Acrobat viewer/reader (last tested V6.1) has issues displaying such images.
Optimization.OptimizeImages	0, 1	When a value of 1 is set, optimization of images' sequences will be turned on; otherwise, it will be turned off.
Optimization.SwapToDisk	0, 1 - 8	When this value is 0, swapping of large images to disk will be turned off; otherwise this value specifies the size (in megabytes) after which images will be swapped to the disk. This value has meaning only when images' sequences optimization is enabled.

Note that *Possible Values* enclosed in quotes are valid values.

Overlay

Key	Possible Values	Description
Overlay.Enable	0, 1	Enable overlaying for the existing PDF document. This allows the pages of a PDF file to be used as foreground or background for other pages to be overlaid on or behind.
Overlay.File		Specifies the full path and name of the existing PDF file. The filespec may be in native or AMOS format, and may include %env% (environment) variables.
Overlay.Password		Specifies the password required to access the PDF file to be used as an overlay file, if encrypted.
Overlay.Foreground	0, 1	Specifies how to place the overlay PDF file: as background (0) or as foreground (1).
Overlay Repeat	0 ("No"); 1 ("LastPage"); 2 ("Continuous")	Specifies the repeat option for overlaying. See the extended descriptions in the linked table.
Overlay.HAlign	0 ("Left"); 1 ("Center"); 2 ("Right")	Specifies the horizontal alignment for the overlay PDF pages relative to the new PDF Document pages.
Overlay.VAlign	0 ("Top"); 1 ("Middle"); 2 ("Bottom")	Specifies the vertical alignment of the overlay PDF pages relative to the new PDF Document pages.
Overlay.Fit	0, 1	Enables or disables the 'Fit' property for the overlay PDF pages to the newly created PDF document pages.
Overlay.KeepAspect	0, 1	Enables or disables aspect ratio status during 'Fit'ting, only used when Overlay.Fit has a value of 1.

Note that *Possible Values* enclosed in quotes are valid values.

Note that in order for the fit and alignment options to be effective, the overlay has to be generated on a page size smaller than the target page.

On the other hand, if, for example, one was going to use an overlay consisting of just a logo, on different page sizes, it might be possible/practical assuming that you always wanted it in one of the four corners (or center) of the target page, regardless of the size differences. If you really want to have precise positional control of overlays, especially for logos, then what you want is not really this kind of overlay, but the kind referenced by the OVERLAY= printer init command, in which you essentially specify the GDI commands, including //IMAGE, to effectively generate the overlay anew for each document.

Overlay Repeat

Value	Meaning	Description
0	No	Begin at the start of the overlay file and apply each corresponding page from this file to the corresponding page number in the newly generated PDF file. Should the newly created file have a greater number of pages than available in the overlay file, no overlay is applied to the remaining pages created in the newly generated PDF document.
1	Last Page	Apply the PDF overlay file pages to the corresponding pages on the newly generated PDF file. Should the newly created file have a greater number of pages, apply the last page of the overlay file to all remaining pages. If the overlay file is a single page, this will be placed on all of the generated files pages by default.
2	Continuous	Begin at the start of the overlay file and apply each corresponding page from this file to the corresponding page number in the generated PDF file. Should the newly created file have a greater number of pages than available in the overlay file, start at the beginning of the Overlay file and begin the process again, applying the first page in the Overlay PDF file to the next page in the generated output PDF document in the sequence, then page 2 of the overlay file to the next generated PDF document page and so on, until the generated document is completed. If the overlay file is a single page, this will be placed on all of the generated files pages by default.

Paper

The two paper options have to do with placement of "pages" on another "page." This is similar to mounting individual snapshots on the page of a photo album, or perhaps mounting multiple thumbnail images on one web page. A modest amount of flexibility is provided, in that the individual pages must be laid out in a regular pattern.

Key	Possible Values	Description
Paper.Nup	1, 2, 4, 6, 8, 9, 16	Set the quantity of pages to be 'mounted' on a single page of the PDF document.
Paper.NupOrder	0 - Across From Left (Left to Right); 1 - Down From Left Or Right to Left, depending on value; 2 - Across from Right to Left; 3 - Down From Right	Set the placement order of pages. Ignored if Paper.Nup = 1. If Paper.Nup = 2, value may be only 0 or 1.

Example

The following will produce a page that contains, in each quadrant, one page of the document being printed. If the original document has 20 pages, the PDF output will have five pages.

```
//PDFX, Paper.Nup, 4  
//PDFX, Paper.NupOrder, 0
```

Save

Key	Possible Values	Description
Save.Type	0 - Use pdfSaver Job Management; 1 - Append To existing Job; 2 - Save Created PDF File. 3 - Generate PDF, email, then Delete.	Sets the type of processing for the printed document, whether to use 'PDFsaver' for Job Management, append to a 'JOB' file, or create a PDF immediately.
Save.JobsGroupName		Name of the Jobs' group where the new job must be added. If the specified group does not exist, it will be created. If this field is empty or not specified and Save.Type is "1", the spooled job will be added to the last available jobs group. When this field is not empty, and the Save.Type is equal to "2", then the spooled job will be added to a corresponding Jobs Group, and then this Group will be saved.
Save.LeaveGroup	0, 1	If the value of this field is equal to 1, then the Jobs Group after saving will not be destroyed otherwise the group will not be saved.
Save.ShowSaveDialog	0, 1	Enables/disables the display of the Save dialog for file naming. If you disable this option, and do not specify a file name using Save.FullFileName, then the name of the PDF file will be the same as the print file name.
Save.FullFileName		See following discussion.
Save.StripPathFromDoc Name	0, 1	If document name contains path, it will be stripped, when this parameter has value 1. This parameter have no effect when Save.FullFileName specified.
Save.WhenExists	0 ("Warning"); 1 ("Overwrite"); 2 ("Auto number"); 3 ("Append"); 4 ("Insert")	Specifies what should happen when the specified (PDF) file already exists.
Save.AutoNumber.Start	0 - 999999	Specifies the start number when Save.WhenExists is equal to "Autonumber" (or 2).
Save.AutoNumber.Digits	1 - 6	Specifies the number of digits for auto numbering when generating file names.
Save.App.Run	0 or "0" or "None"; 1 or "1" or "default"; or application path	Specifies the application to start (RUN) immediately after print completion. after printing. If the value of this key is equal to "0" (or is "None"), no application will be executed after file creation. If the value of this key is equal to "1" (or string "Default"), the default application for this workstation for PDF files will start as set by the file type association within Windows.
Save.App.Params		Specifies application parameters to execute after printing is done. %f represents the PDF file name.

Note that *Possible Values* enclosed in quotes are valid values.

Save.FullFileName

This parameter is used to control the output file name. There are three possible conditions:

- If this parameter is not specified, the file is saved as the original filespec with a .PDF extension. This is the default condition.
- If Save.FullFileName contains a dot (i.e., the indicator of any extension), then it is interpreted as a <filespec> and causes the output to have that filespec—i.e., full path and full file name.
- If Save.FullFileName does not contain a dot, the argument is interpreted as the directory where you want the output to go. The file will be placed into that directory and the output filename will be xxxxx.pdf, where xxxxx is the original file name.

Note that the <filespec> or path may be in native or AMOS format and contain %env% variables. Also note that the directory will be created if necessary, provided that its parent exists and allows write privileges.

Examples

Assuming that the file being printed is c:\vm\miame\dsk0\007006\test.txt:

```
///  
//PDFX,Save.FullFileName,c:\temp  
  
///  
//PDFX,Save.FullFileName,c:\temp\x.pdf  
  
///  
//PDFX,Save.FullFileName,%miame%\pdfdir  
  
//PDFX,Save.FullFileName,PDFDIR:
```

Security

Key	Possible Values	Description
Security.Use	0, 1	Enables or disables security for the printed document.
Security.Level	0 or 40; 1 or 128	Specifies the encryption level for the document.
Security.UserPwd		Specifies the User password for the document.
Security.OwnerPwd		Specifies the Owner (master) password for the document.
Security.40.AllowPrinting Security.40.AllowChanging Security.40.AllowCopying Security.40.AllowComments	0, 1	Specifies the user's permissions for the encrypted document (when the encryption level is set to 40).
Security.128.ContentAccess Security.128.AllowCopying	0, 1	Specifies the user's permissions for the encrypted document (when the encryption level is set to 128).
Security.128.Changes	0 (none), 1, 2, 3, 4	
Security.128.Printing	0 - None; 1 - Low Resolution; 2 - Fully Allowed	

Security.Digisig

Key	Possible Values	Description
Security.Digisig.Enable	0, 1	Enable digital signatures for the document.
Security.Digisig.PFXFile		Specifies the full path and name of the PFX file where the certificate is stored.
Security.Digisig.PFXPassword		Specifies the string password used for the PFX file.
Security.Digisig.Reason		Specifies the reason for signing, such as (I agree...).
Security.Digisig.Location		Specifies the CPU host name or physical location of the signatory.
Security.Digisig.ContactInfo		Specifies the information provided by the signatory to enable a recipient to contact the signer and verify the signature; for example, a phone number.
Security.Digisig.Image		Specifies the full path and file name of the image which may be displayed within the signature field.
Security.Digisig.Flags		Combination of flags determining how the signature field should appear on the page. See PDF-XChange documentation for possible values.
Security.Digisig.Page	-1, 0...N	Specifies the page number for which the signature field should be located. If this parameter is -1, the last page of the document will be used.
Security.Digisig.Left,		These values specify the location on the page and the dimensions of the signature field. Values should be specified in tenths of millimeters.
Security.Digisig.Top,		
Security.Digisig.Width,		
Security.Digisig.Height		

Watermarks

Watermarks are words, symbols or images that are impressed into paper when the paper is made. Later, after the paper has been printed on, the watermark is still visible in the background and thereby serves as a mark of the authenticity of the paper. Watermarks were first used in Italy in the thirteenth century, and are still used in various types of high-quality paper. A digital watermark is simply the translation of this ancient practice—a "background" image that shows through whatever is printed on the paper—into the computer age.

The PDFX option for watermarks is a bit unusual, in that it provides only partially control of watermarks. You can use the following GDI command to turn watermark printing on or off, but you cannot (with GDI commands) control anything else about the watermark. For all other watermark-related controls (what it looks like, where it is placed, color, font, etc.), you must go directly to the PDFX printer driver. Go to Start...Printers and Faxes....PDF-XChange (or whatever)...Printing Preferences...Watermarks, and use the available tools to controls the characteristics of the watermark.

One great way to use watermarks is to place images—company logos, interesting artwork, baby photos of yourself—in the background of invoices and/or other reports. There may or may not be valid security or branding reasons to do this, but it certainly makes for very esthetic reports.

Note that you may create as many watermarks as you wish, and you can "turn on" as many of those as you wish; they are cumulative, and all the ones checked will print per the specs you have provided. Again, however, your only control from within A-Shell is to print watermarks or not print them.

Key	Possible Values	Description
Watermarks.Print	0, 1	Enables or disables printing of watermarks for the PDF document.