

# PDF Direct™

Version 2.0



*Results Through Focused Technology...*

# PDF Direct™

Version 2.0



©1998-2003 Soft Solutions, Inc. All Rights Reserved.  
Published World-Wide by Soft Solutions, Inc.

Soft Solutions, Inc.  
2900 Chamblee Tucker Road  
Building 12, Suite 200  
Atlanta, GA 30341 USA



*Results Through Focused Technology...*

**IMPORTANT—READ CAREFULLY BEFORE OPENING.** By installing this software program, you indicate your acceptance of the following Soft Solutions, Inc. License Agreement.

**Software License and Limited Warranty Agreement:**

This is a legal agreement between you, the end user, and Soft Solutions, Inc. By opening this package, you are agreeing to be bound to the terms of this agreement. If you do not agree to the terms of this agreement, promptly return the unopened disk package and the accompanying items (including written materials and binders or other containers) to the place you obtained them for a full refund.

Soft Solutions, Inc. Software License

1. **GRANT OF LICENSE.** Soft Solutions grants to you the right to use one development copy of the enclosed Soft Solutions software program (the "SOFTWARE") on a single terminal connected to a single computer (i.e., with a single CPU), or on a LICENSED COMPUTER NETWORK. Each concurrent user of the SOFTWARE must have exclusive access to a Soft Solutions SOFTWARE manual during his/her use. Soft Solutions, Inc. as Licensor, grants to you, the LICENSEE, a non-exclusive, non-transferable right to use this Software subject to the terms of the license as described in the following sections:

A. You may make backup copies of the Software for your use provided they bear the Soft Solutions, Inc. copyright notice.

B. You may use this software in an unlimited number of custom or 4D- compiled commercial database applications created by the original licensee. No additional product license or royalty is required.

2. **COPYRIGHT,** THE SOFTWARE is owned by Soft Solutions or its suppliers and is protected by United States copyright laws and international treaty provisions. Therefore, you must treat the SOFTWARE like any other copyrighted material (e.g., a book or musical recording) *except that you may either (a) make one copy of the SOFTWARE solely for backup or archival purposes, or (b) transfer the SOFTWARE to a single hard disk provided you keep the original solely for backup or archival purposes. You may not copy the written materials accompanying the SOFTWARE.*

3. **OTHER RESTRICTIONS.** You may not rent or lease the SOFTWARE, but you may transfer the SOFTWARE and accompanying written materials on a permanent basis provided you retain no copies and the recipient agrees to the terms of this Agreement. You may not reverse engineer, decimalize, or decompile the SOFTWARE. If SOFTWARE is an update, any transfer must include the update and all prior versions.

**LIMITED WARRANTY.** Soft Solutions, Inc. warrants that (a) the SOFTWARE will perform substantially in accordance with the accompanying written materials for a period of ninety (90) days from the date of receipt. Any implied warranties on the SOFTWARE are limited to ninety (90) days and one (1) year, respectively. Some states do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

**CUSTOMER REMEDIES.** Soft Solutions, Inc. entire liability and your exclusive remedy shall be, at Soft Solutions option, either (a) return of the price paid or (b) repair or replacement of the SOFTWARE that does not meet Soft Solutions Limited Warranty and which is returned to Soft Solutions with a copy of your receipt. This Limited Warranty is void if failure of the SOFTWARE has resulted from accident, abuse, or misapplication. Any replacement SOFTWARE will be warranted for the remainder of the original warranty period of thirty (30) days, whichever is longer.

**NO OTHER WARRANTIES.** Soft Solutions disclaims all other warranties, either express or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to the SOFTWARE, the accompanying written materials, and any accompanying hardware. This limited warranty gives you specific legal rights. You may have others, which vary from state to state.

**NO LIABILITY FOR CONSEQUENTIAL DAMAGES.** In no event shall Soft Solutions or its suppliers be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use of or inability to use this Soft Solutions product, even if Soft Solutions has been advised of the possibility of such damages. Because some states do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

**Governing Law.** This entire agreement shall be governed by the laws of the State of Georgia.

PDF Direct™ is a trademark of Soft Solutions, Inc. Apple is a registered trademark of Apple Computer, Inc. 4th Dimension, 4D Compiler, 4D External Mover and 4D are registered trademarks of 4D, Inc. PDFWriter, Acrobat Exchange and Adobe Acrobat are registered trademarks of Adobe Corporation.



# Table of Contents

<b>Table of Contents</b> .....	<b>v</b>
<b>List of Figures</b> .....	<b>vii</b>
<b>Introduction</b> .....	<b>1</b>
Command Syntax Used in this Document.....	1
<b>Installation</b> .....	<b>2</b>
Which Plug-in Do I Use?.....	2
<b>System Requirements</b> .....	<b>3</b>
<b>Overview</b> .....	<b>3</b>
<b>Platform Specific Print Issues</b> .....	<b>4</b>
Mac OS X .....	4
4D for Windows.....	5
4D Write 6.5 (and later) for Windows .....	5
<b>Command Descriptions</b> .....	<b>6</b>
PD_Register .....	6
PD_IsAvailable .....	6
PD_GetPrinter.....	7
PD_SetPrinter .....	8
PD_GetPath.....	9
PD_Define.....	10
PD_GetStatus.....	11
PD_Cleanup .....	11
<b>Sample Application</b> .....	<b>12</b>
Setup .....	12
4D Write PDF Printing .....	16
4D Print Selection Printing .....	19
4D Print Form Printing .....	20

**PDF Direct Command Summary ..... 23**  
**Index ..... 25**

# List of Figures

- Fig.1: Print Driver Information 12
- Fig.2: Choosing the PDFWriter 13
- Fig.3: Settings Menu (Windows) 14
- Fig.4: Windows Printer Directory 14
- Fig.5: Set as Default Command 15
- Fig.6: Saving the Printer Settings 15
- Fig.7: New Print Driver Information (Mac) 15
- Fig.8: New Print Driver Information (Windows) 16
- Fig.9: PDF Merge Setup Dialog 17
- Fig.10: Select a Path Dialog (Mac) 18



# Introduction

Portable Document Format (PDF) is the most commonly used document format in the world and the most widely used format for viewing on-line documentation. PDF Direct™ is a 4th Dimension plug-in that provides the ability to print a PDF document directly from within a 4th Dimension database. Using PDF Direct, 4D Developers can now programmatically name and deliver any 4D Print job, with appropriate specifications, into an Adobe Acrobat “PDF” format file.

The PDF format has great appeal when paper or a “hardcopy” is not required for the intended readers. With the predominance of email and Adobe’s free Acrobat Reader, attaching a PDF document within an email transmission is universally appealing because of its speed and thrift. Similarly, in comparison to a faxed document, PDF has no loss of resolution as it delivers your documents with the exact same definition in which they were created.

PDF supports color, grayscale and, in general, preserves the creator’s document with no loss of definition regardless of the reader’s computer platform. Adobe Corporation, the creator of the PDF format, provides “Acrobat PDFWriter,” a PDF Printer Driver for both the Windows and Mac OS operating systems. Like any print driver, PDFWriter can be manually chosen through the operating system tools available (Windows Printer Directory and Mac OS Chooser). Once chosen, any user can directly print to a PDF output format. Through PDF Direct, programmers and end-users can enjoy full PDF creation capabilities seamlessly integrated within any 4D database.

While PDF Direct gives 4D Developers access to PDF document printing capabilities directly from within 4D, these print capabilities should not be confused with the direct construction of a PDF document using PDF language and commands. This type of ground-up, piecemeal building and construction of a PDF document is not presently a feature of PDF Direct.

## Command Syntax Used in this Document

**Command**(Parameter(Field Type))->\$X(l)

Where \$X receives the status code as specified in the Command Set documentation.

Field Type Legend
B = Boolean
S(n) = String and n is a number specifying the string length.
T = Text
I = Integer
L = Long integer

# Installation

PDF Direct is delivered in a compacted format, specific to the Windows or Mac OS platform, ready for extraction to your hard drive. Regardless of the platform, each platform version of PDF Direct contains the following items:

- 1). Product source code demo in 4th Dimension version 6.x.
- 2). The PDF Direct Plug-in
- 3). PDF Direct Program Documentation

**The Acrobat™ PDFWriter is required, but not included with PDF Direct.**

Acrobat™ PDFWriter can be purchased separately from Soft Solutions, Adobe Corporation or through any other number of authorized Adobe distributors and resellers. As of Release 2.0 of PDF Direct, to obtain Acrobat™ PDFWriter, you need to acquire, at minimum, Adobe Acrobat™ or another similar Adobe product where Acrobat™ PDFWriter is included. PDFWriter is not required under Macintosh OS X.

Under Macintosh OS X, PDF Direct will be installed in the "{BootDrive}/Library/Printers" directory as a "PDFDirect" folder. The "PDF Direct" printer must be added through the "Print Center" application.

## Which Plug-in Do I Use?

**Table 1:**

	<b>Mac OS 9.2 or earlier</b>	<b>Mac OS X or later</b>	<b>Windows - All</b>
<b>4D 6.8 or later</b>	Mac4DX or Mac4CX	Mac4CX	Win4DX
<b>4D 6.7 or earlier</b>	Mac4DX	N/A	Win4DX

# System Requirements

On Mac OS X, MacOS X, version 10.2 or later is required.

Under earlier versions of Macintosh, Acrobat PDFWriter is required. Under all versions of Windows, Acrobat PDFWriter is required.

## Overview

Before reviewing the individual PDF Direct commands in detail, it is helpful to have an overview of the basic steps to follow when coding your PDF Direct printing solutions.

PDF Direct requires the developer/user to own version 4 (or later) of the Acrobat™ PDFWriter. PDF Direct also expects the PDFWriter to be correctly installed and located in the correct location for printer drivers (these locations are platform specific). Failure to do so will prevent your 4D application from creating PDF documents. The **PD\_IsAvailable** command lets the 4D Developer know whether the PDFWriter is available before attempting to print to a PDF document.

Prior to launching a 4th Dimension (4D) application, there is normally a physical default printer established. Because computer users typically establish a standard or default printer driver for each and every machine, it is likely you (or your users) will want to ensure the default printer is re-established after creating PDF documents within 4D. Because PDF Direct provides the ability to programmatically switch printer drivers, you have the necessary tools to ensure this happens.

Printing to the Acrobat™ PDFWriter within 4D requires the print driver be switched to the Acrobat™ PDFWriter (printer driver (Windows) or extension (Mac OS)). This requires the 4D application to record the Acrobat PDFWriter name and location specifics. Once this information is established on a machine, printing to Acrobat PDF files and then resetting the Printer driver to the originally set default printer is straightforward. These tasks are accomplished with the **PD\_GetPrinter** command which retrieves the current Printer name along with its type, zone and driver.

Switching to the PDFWriter is accomplished by issuing the **PD\_SetPrinter** command. Used in conjunction with the **PD\_GetPrinter** command, the **PD\_SetPrinter** command specifies the Printer name along with its type, zone and driver. To reset a printer, the **PD\_SetPrinter** command is issued once the developer has secured the default printer specifications using the **PD\_GetPrinter** command.

With PDF Direct, it is possible to 1) programmatically name a PDF document 2) allow the User to name the 4D generated PDF document or 3) provide a programmable name to the user allowing them to make modifications as required. The **PD\_GetPath** command provides complete developer control for both the naming and placement of new PDF documents. This control comes in the form of a customizable file “Save As” mes-

sage dialog which prompts the User with either a specified filename (which can be altered) or a null filename allowing the User to completely type the desired filename. In both instances, the User can always change the full path by navigating the standard file “Save As” dialog.

Lastly, once a filename has been determined, the developer needs to communicate this filename to the PDFWriter printer driver. This is accomplished with the **PD\_Define** command. The **PD\_Define** command explicitly tells the PDFWriter the full pathname of the PDF document to be created. Additionally, the **PD\_Define** command allows the 4D Developer to specify whether the Acrobat Reader program be launched once the PDF document is created. This is extremely useful for quickly preparing and proofing PDF documents.

When specifying a null path name in the **PD\_Define** command, the Developer forces the PDFWriter printer dialog to appear. This is valuable when the developer or end-user wants to specify particular characteristics of the PDF document before creation.

Printing is handled differently on MacOS X. The Printing System does all of the upfront work. Macintosh OS X displays the printing dialogs; excepts printing output and turns it into PDF data; queues this as a print job; and eventually the print driver receives the PDF data.

With this in mind a new command, **PD\_GetStatus**, has been added to retrieve the status of the destination PDF file. Also, an additional parameter has been added to **PD\_Define** to return a reference which then can be used in **PD\_GetStatus**.

A useful command after generating PDF documents from 4D, **PD\_Cleanup** was created to ensure the PDFWriter is returned to a cleansed state after its use. Because it is possible within a 4D environment to invoke the PDFWriter printer dialog, it is possible that print settings from a previous print job are not desirable for subsequent print jobs. The **PD\_Cleanup** command allows the current settings (carried over from the previous print job) to be reset (initialized) to the default PDFWriter settings.

## Platform Specific Print Issues

### Mac OS X

Under 4D 6.8 and Macintosh OS X, 4D establishes a default printer at application start-up time and disregards programmatic changes to this default printer. To circumvent this issue, the 4D “Print Settings” command must be issued to establish PDF Direct as the new default print driver after launching 4D. For this reason, it will be necessary for 4D developers to present the 4D “Print Settings” dialog to allow the User to select the PDF Direct printer driver before attempting to create a PDF file. It is recommended the call to 4D Print Settings be made after calling the PDF Direct **PD\_SetPrinter** command.

Under OS X, the print settings command first brings up a page setup dialog followed by a print settings dialog. In the “Print settings” dialog, the User must select the PDF Direct print driver in order for 4D to respect the change in printer drivers.

In summary, the PDF Direct command `PD_SetPrinter` correctly switches the Print Driver to PDF Direct from the OS X Print Center, however, 4D does not honor this change. Therefore, it is still necessary for the user to change to the PDF Direct print driver manually before attempting to create a PDF file. 4D 2003 overcomes this limitation with a more robust printer command set.

## **4D for Windows**

When using the 4D Print Form command, 4D fails to properly toggle to the new printer until after a first Print job has been executed. For this reason, the 4D PDF Direct Demo for Windows calls 4D method to generate a dummy print job. This method permits the subsequent “real” PDF job to process correctly. Please include a comparable method in your 4D application under Windows to ensure this desired behavior.

## **4D Write 6.5 (and later) for Windows**

4D Write behaves similar to a stand-alone application. For this reason, 4D Developers should follow the programming technique provided within the PDF Direct Demo for 4D 6.5/6.8 for Windows when attempting to create a PDF file from a 4D Write area. In this PDF Direct Demo, the Letters output form method runs 4D code to switch the default printer to the PDF Direct printer before loading the 4D Write area. When the 4D Write area is subsequently loaded in the Letters input form, this process signals to 4D that PDF Direct is the default printer. We’ve found this programming methodology to be reliable when creating a PDF file from 4D Write under 4D for Windows, version 6.5 and later.

# Command Descriptions

---

## PD\_Register

### Syntax

```
PD_Register(LicenseeName(S);LicenseCode Macintosh(S);LicenseCode Windows(S)): ->$X(L)
```

### Description

This function allows the 4D developer to register the PDF Direct program. PD\_Register can be called in the start-up procedure of any 4D application using PDF Direct. A valid registration will return the long integer value of 1 and fully enable the plug-in. An invalid, empty or missing registration from the requesting machine's platform will return the long integer value of 0 and disable the plug-in after 30 minutes.

### Parameters

*Licensee Name* ..... A string expression providing the name of the PDF Direct licensee.

*LicenseCode Macintosh* ..... A string expression providing the matching Mac OS developer license code.

*LicenseCode Windows*..... A string expression providing the matching Windows developer license code.

### Example

```
$ReturnCode:=PD_Register("Name";"Mac Code";"Windows Code") 'Registrations are issued by Soft Solutions, Inc.
```

---

## PD\_IsAvailable

### Syntax

```
PD_IsAvailable->$X(l)
```

### Description

The PD\_IsAvailable function provides a return code indicating whether the Acrobat PDFWriter is currently available. Issue this command before attempting to print to a PDF document. This command is useful for determining whether the Acrobat PDF-Writer is currently in a busy state and unavailable to the requesting process within your 4D Application.

**Return Value:**

*ResultCode* ..... If the operation is successful, an integer value of zero is returned. If an error occurs, a non-zero error code value is returned.

**Example**

```
$X:=PD_IsAvailable
```

**PD\_GetPrinter****Syntax:**

```
PD_GetPrinter(Name(S);Type(S);Zone(S);Driver(S))->$X(l)
```

**Description:**

The PD\_GetPrinter command is used to obtain the current default printer address information. Both Windows and Mac OS versions use the same parameters. Pass process variables of Alpha 80 in each of the four (4) parameters. A Return code of 0 indicates successful execution. Upon successful execution, each of the parameters will return the correct value for the printer.

**Parameters:**

*Name* ..... A string expression providing the name of the current printer. If the current printer is local (directly attached to the requesting machine), the value returned in this parameter will be “Modem” or “Printer”.

*Type* ..... A string expression providing the printer type. For example, all Apple LaserWriters are given a value of “LaserWriter”.

*Driver* ..... A string expression which contains the name of the current printer driver, such as: “Appletalk ImageWriter”.

*Zone* ..... A string expression providing the current printer zone. If the currently chosen printer is within the same zone as the requesting machine, an “\*” will be returned. Otherwise the actual zone name is returned.

**Return Value:**

*ResultCode* ..... If the operation is successful, an integer value of zero is returned. If an error occurs, a non-zero error code value is returned.

**Example:**

```
$x:=PD_GetPrinter(Name;Type;Zone;Driver)
```

---

## PD\_SetPrinter

### Syntax:

```
PD_SetPrinter(Name(S);Type(S);Zone(S);Driver(S))->$X(l)
```

### Description:

The PD\_SetPrinter command is used to reset the current default printer address information. Both Windows and Mac OS versions use the same parameters. Use process variables of Alpha 80 in each of the four (4) parameters. A Return code of 0 indicates successful execution. Upon successful execution, the current printer driver will be changed to the one specified in these four (4) parameters.

### Parameters:

*Name* ..... A string expression providing the name of the desired printer. If the current printer is local (directly attached to the requesting machine), the value returned in this parameter will be “Modem” or “Printer”.

*Type* ..... A string expression providing the printer type. For example, all Apple LaserWriters are given a value of “LaserWriter”.

*Zone* ..... A string expression providing the current printer zone. If the currently chosen printer is within the same zone as the requesting machine, an “\*” will be returned. Otherwise the actual zone name is returned.

*Driver* ..... A string expression which contains the name of the printer driver the user or developer wants to switch to, such as: “Acrobat™ PDF-Writer”.

### Return Value:

*ResultCode* ..... If the operation is successful, an integer value of zero is returned. If a Return Code of -43 is returned, this indicates the name specified in the *Name* parameter could not be found.

### Example:

```
$x:=PD_SetPrinter(Name;Type;Zone;Driver)
```

---

## PD\_GetPath

### Syntax:

```
PD_GetPath(Pathname(S);Prompt(S);DefaultFileName(S)->$(I)
```

### Description:

The PD\_GetPath command is used to issue a File “Save As’ Dialog with 4D Developer defined defaults. Both Windows and Mac OS versions use the same parameters. Use process variables of Alpha 80 in each of the first two (2) parameters and Alpha 31 for the 3rd parameter. A Return code of 0 indicates successful execution. Upon successful execution, the *FilePath* parameter will contain either the filename originally specified in the *SFDefault* parameter or the path and name of the file specified by the User.

### Parameters:

*FilePath*.....A string variable specifying the full path to the filename you wish to the create. If you wish to allow the user to specify the location of the new file, assign a null string to this variable before calling the routine; in this case, the routine will display a standard file package “Save As” dialog to allow the user to enter a file name and select the folder in which to save the PDF file. When the routine has completed, the full path to the file specified by the user will be assigned to this variable.

*SFPrompt* ..... A string expression which specifies a display prompt for the user on the standard file package dialog. If a path value is assigned to FILEPATH before the routine is called, this parameter is ignored.

*SFDefault* ..... A string expression which specifies a default folder and/or filename for the user when the standard file package dialog is displayed. If a path value is assigned to FILEPATH before the routine is called, this parameter is ignored. If you pass the full path to an existing folder here, the standard file dialog will pre-select that folder for the user and display its contents in the catalog list area, thereby “suggesting” that folder as the location for storing the new file in its filename edit field. If you pass the full path to a file in an existing folder, the dialog will both pre-select the folder and enter the filename as the default name.

### Return Value:

*ResultCode* ..... If the operation is successful, an integer value of zero is returned. If an error occurs, a non-zero error code value is returned.

### Example:

```
$(X:=PD_GetPath(Pathname;Prompt;DefaultFileName)
```

---

## PD\_Define

### Syntax:

```
PD_Define(Pathname(S);LaunchReader(I))->$X(I;Capture Reference(S))
```

### Description:

The PD\_Define command is used to name the PDF document originating from the current print job and specify whether the Adobe Acrobat Reader is launched. A Return code of 0 indicates successful execution. Upon successful execution, the subsequent 4D print job will print to the filename specified in the *Pathname* parameter.

*Important note:* Regarding 4D Print jobs, when printing multi-page forms or multi-page print selections, it may be necessary to specify “Print One Job” within 4D Customizer to ensure all “printed” 4D forms remain within a single PDF output file. The 4D PRINT FORM command builds each printed page in memory. Each page is printed when the page in memory is full or when you call PAGE BREAK. In addition, when working with PRINT FORM, the 4D “PAGE BREAK” command provides an optional (>) parameter which allows subsequent form pages to remain part of the same print job. Adding this parameter ensures the print job will be spooled to one file.

The Capture Reference parameter returns a reference which then can be used in "PD\_GetStatus".

To ensure the printing of the last page after any use of PRINT FORM, you must conclude with the PAGE BREAK command. Otherwise, if the last page is not full, it stays in memory and is not printed (this excerpt taken from the 4th Dimension v6 manual).

### Parameters:

*Pathname* ..... A string expression which specifies the entire full pathname of the document to be saved. Entering a null string here invokes the PDFWriter dialog. The PDFWriter Dialog is a File “Save-As” dialog requesting the user to name the PDF document to be created and specify the desired directory or path where to save the document. Additionally, the PDFWriter dialog provides various other print options including resolution, compression, page size, orientation, scaling factors and more.

*LaunchReader* .... An integer expression which specifies whether or not the Adobe Acrobat Reader will be launched to review the created PDF Document. Pass a value of 0 (zero) to avoid launching the Acrobat reader or pass a 1 (one) to launch the reader to immediately review the PDF document.

*Capture Reference* A string expression which can be subsequently used within the PD\_GetStatus comand.

### Return Value:

*ResultCode* ..... If the operation is successful, an integer value of zero is returned. If an error occurs, a non-zero error code value is returned.

**Example:**

```
$X:=PD_Define(Pathname;LaunchReader;Reference)
```

---

## PD\_GetStatus

**Syntax:**

```
PD_GetStatus(Capture Reference(S);Status(I))->$X)
```

**Description:**

The PD\_GetStatus command is used to retrieve the status of the destination PDF file.

Please note: When using 4D Write to print on MacOS X, 4D Write does not suppress the printing dialogs. Also, 4D Write appears to save the printer along with the document so it ignores the default printer. When the printing dialog is displayed, the "PDF Direct" may have to be chosen as the "Printer:".

**Parameters:**

*Capture Refer.* .... A string expression which specifies the PDF file in question. Must be used after, yet in conjunction with the PD\_Defin command.

*Status* ..... An integer expression where 1 = ready for print driver, 2 = print driver processing, 3 = PDF complete.

**Return Value:**

*ResultCode* ..... If the operation is successful, an integer value of zero is returned. If an error occurs, a non-zero error code value is returned.

**Example:**

```
$X:=PD_GetStatus(vi_Reference;vs_Status)
```

---

## PD\_Cleanup

**Syntax**

```
PD_Cleanup->$X(I)
```

**Description**

PD\_Cleanup is a function call which returns an integer value of 0 after successful completion or a non 0 code if the function failed. Because of the variety of memory issues involved with managing and changing Print driver selections, it is recommended the PD\_Cleanup command be issued after the necessary PDF Direct commands have performed the intended work.

**Example:**

```
$X:=PD_Cleanup
```

## Sample Application

The PDF Demo application provides a working source code example which uses all of the PDF Direct plug-in commands.

## Setup

Before attempting to print any 4D Document to Adobe's Acrobat (PDF) format, the Developer/User must first establish the print driver information for the Adobe PDF Writer. When the User first selects the demo application menu item to display the current default printer, the following dialog will appear.



**Fig.1: Print Driver Information**

**Note**

Initially, this dialog will likely show a physical printer as the current default printer. Do NOT attempt to save the printer driver for the physical printer as the Acrobat PDFWriter. Doing so will direct your 4D print job to the default printer.

Establishing the Adobe PDFWriter as the default printer will be an essential first step to capture these PDFWriter print driver specifics within your database. Once the Adobe PDFWriter name and location is captured - it can be saved in a convenient place for future Acrobat printing.

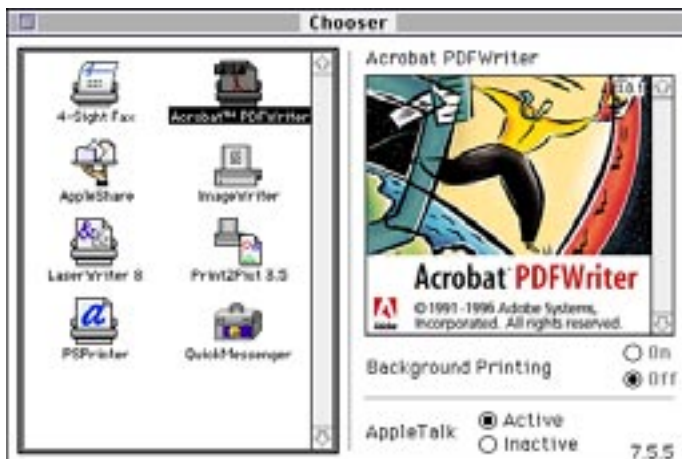
Establishing a default printer driver is easily accomplished by using the appropriate tools within the current development platform. Although the PDF Direct 4D demo application is set up as a single-user application, it can easily be modified for 4D Server applications with many client machines.

Depending on your database application, it is questionable whether you should associate the Acrobat Print driver location to a User. Because Users may occasionally change machines rendering a default Adobe PDFWriter location obsolete and incorrect, this is likely unwise. Ideally, it is recommended the local Adobe PDFWriter location be written to the local hard drive into a preferences file. This would be a good strategy on each machine intending to use the Adobe PDFWriter print driver. This ensures the PDF-Writer driver information will remain with the machine. While it is still possible the PDFWriter could be renamed or replaced, this strategy will likely be the safest and most durable.

Conversely, “hard-coding” the Adobe PDFWriter printer driver settings is certainly not recommended because different platforms, operating systems and releases of Adobe PDFWriter may be changed over time or user-modified—rendering static defaults as problematic and error-prone.

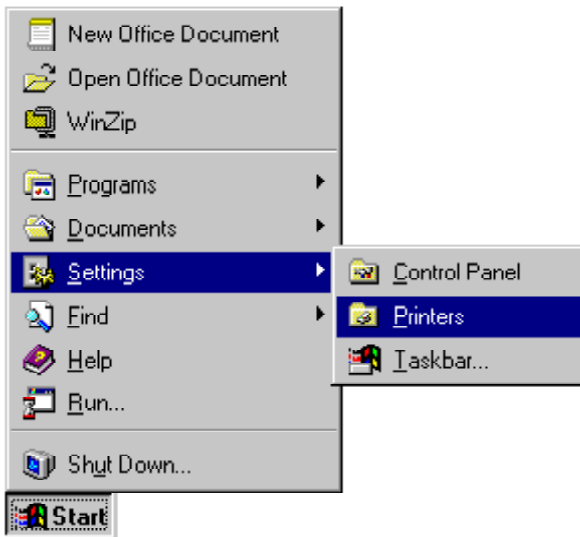
On the following pages are the steps for both Mac OS and Windows Users to establish the Acrobat PDFWriter as the default printer.

On the Mac OS, the User should go to the Print “Chooser” and directly select the Adobe PDFWriter.



**Fig.2: Choosing the PDFWriter**

On Windows, the User should first go to the Printers directory accessible from the Settings menu.



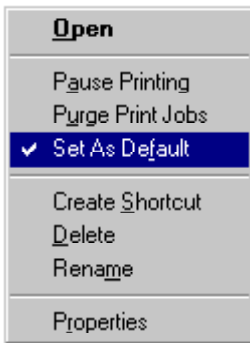
**Fig.3: Settings Menu (Windows)**

Once within the Windows “Printers” Directory, the User should select the Acrobat PDFWriter printer driver by right-clicking on the Acrobat PDFWriter to bring up the printer options.



**Fig.4: Windows Printer Directory**

The User should then select “Set as Default” to establish the Acrobat PDFWriter as the current default printer.



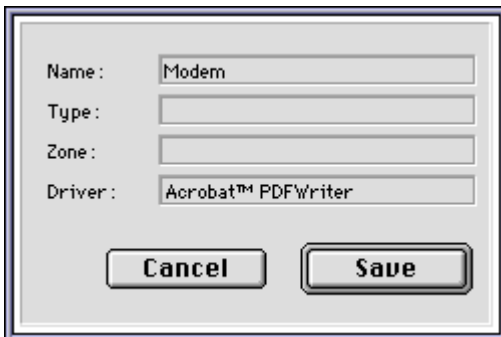
**Fig.5: Set as Default Command**

Once the Acrobat PDFWriter has been established as the default printer (driver), the Developer/User should return to the 4th Dimension application and save the settings to the database by selecting the appropriate Menu option (see Assign PDFWriter below).



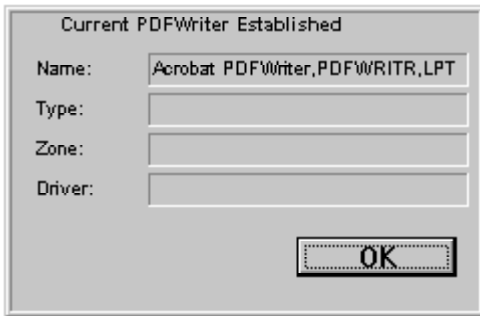
**Fig.6: Saving the Printer Settings**

On the Mac OS, the **Name** will normally be assigned to “modem” or “printer” while the **Driver** will be assigned to the Acrobat™ PDFWriter.



**Fig.7: New Print Driver Information (Mac)**

On Windows, the **Name** will normally be assigned to “Acrobat PDFWriter” with the port information following. The **Driver** information will be blank (null).



**Fig.8: New Print Driver Information (Windows)**

Once the Acrobat PDFWriter settings have been saved for each machine (planning to print Acrobat (PDF) documents), the User can return the default printer to the original printer chosen using the Operating System tools provided within each platform.

Following are some 4D coding examples used for the different types of 4D printing tasks.

## 4D Write PDF Printing

Performing a “Mail Merge” within 4D Write can be used effectively to programmatically create and name PDF documents. There are, however, some important rules which can cause your 4D application users problems if they not followed. Because 4th Dimension is very particular in how it handles different 4D print jobs (and how it deals with print drivers in general), these rules are definitely worth reading.

### Mac OS:

Issuing the WR PRINT command within a 4D “For Loop” causes the print driver to automatically append a continually increasing sequence number. For example, in using the WR PRINT command with the requested names “John Smith.PDF”, “Bill Jones.PDF” and “Jack Smith.PDF”, the first document would correctly be named “John Smith.PDF”. However, the 2nd and 3rd documents would be named “Bill Jones.PDF 2” and “Jack Smith.PDF 3”, respectively. Consequently, the 4D Write “WR DO” command is recommended to perform the mail merge within a 4D “For Loop”. Additionally, it is essential the end-users select the “Show Values” versus the “Show References” menu item from the 4D Write Database menu. Showing references will not correctly perform the variable merging, but “Showing Values” will.

```

$Err:=PD_GetPrinter (vname;vptype;vzone;vdriver)
  $Err:=PD_SetPrinter (<>PDF_Name;<>PDF_Type;<>PDF_Zone;<>PDF_Driver)

```

or `gp\_OnMenuPDF uses PDF Direct Mac/Win version 1.0

```
C_LONGINT($1;$2;$3;$Err;$R;$i)
```

```
C_TEXT($Name;$Path;$DocName)
```

Case of

```

: ($2=112) `Print Merge Menu Item
  USE NAMED SELECTION("<>Merge")

```

```
vPrompt_Option:=False `Disable PDF Print Dialog for Mail merge
```

```
vViewer_Option:=False `Don't allow reader to load in a mail merge
```

```
bLaunchViewer:=0
```

```
gpCenterWindow (228;265;1;"" )
```

```
vMerge_Message:=String(Records in selection([Contacts]))+" Contact records will be merged with the letter named "+[Letters]LetterName+"."
```

```
vTitle:="PDF Merge Setup"
```

```
DIALOG([Constants];"PDF_Options")
```

```
CLOSE WINDOW
```

The code (above) displays the following dialog:



**Fig.9: PDF Merge Setup Dialog**

If (bOk=1) `User selected the PDF Button

`Developer establishes default naming convention for end-User filename prompt

```
MESSAGES OFF `turn off messages
```

```
FIRST RECORD([Contacts])
```

```
$Name:=([Contacts]First_Name+" "+[Contacts]Last_Name
```

'Developer obtains current default printer information, stores this in database variables to use after Acrobat PDF print job is complete.

```
vname:= "" `Port name ("name" carryover from old Mac days), Used for both WIndows & Mac
vptype:= "" `What type of printer; serial = ""; Printer might be "LaserWriter"; Mac use only
vzone:= "" `Default Zone is "" for single zone or local zone; Mac use only
vdriver:= "" `Name of Mac OS Extension
$Err:=PD_GetPrinter (vname;vptype;vzone;vdriver)
```

'Developer sets the printer to Acrobat PDF print driver.

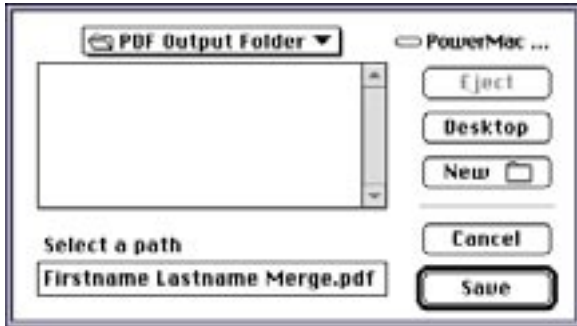
```
$Err:=PD_SetPrinter (◇PDF_Name;◇PDF_Type;◇PDF_Zone;◇PDF_Driver)
```

```
If (bPromptDefault=1)
```

```
    $Err:=PD_GetPath ($Path;"Select a path";$Name+" Merge.pdf")
```

```
End if
```

The "PD\_GetPath" command shown above provides the following dialog and User prompt:



**Fig.10: Select a Path Dialog (Mac)**

```
$WhereDelete:=Position($Name+" Merge.pdf";$Path) `Strip out the document name
$ShortPath:=Delete string($Path;$WhereDelete;250) `so we can obtain just the path name
```

```
$R:=Records in selection([Contacts])
```

```
For ($i;1;$R)
```

```
    Case of
```

```
        : (bAutoName=1)
```

```
            $Name:=[Contacts]First_Name+" "+[Contacts]Last_Name
```

```
            $DocName:=$Name+" Merge.pdf"
```

```
        : (bPromptDefault=1)
```

```
            $Name:=[Contacts]First_Name+" "+[Contacts]Last_Name
```

```
            $DocName:=$ShortPath+$Name+" Merge.pdf"
```

```
    End case
```

```

$Err:=PD_Define ($DocName;bLaunchViewer) `Avoid launching Acrobat within Mailmerge
  If ((z_n_Platform=Windows)
    WR PRINT (Letter;0) `Specify Merge Document area; show values; not references
    Else `Make sure user selects show Values so merge works!!!
    WR DO COMMAND (Letter;111) `4D Write WR Do command used to prevent filename
      # suffix
    End if

  If (bCreateLog=1)
    gpBuild_PDF_Log ($Name+" Merge.pdf")
  End if

  NEXT RECORD([Contacts])
End for ` ($i;1;$R)

$Err:=PD_SetPrinter (vname;vptype;vzone;vdriver) `Return Printer to it's default driver
$Err:=PD_CleanUp `Reset Acrobat defaults & clean up memory

MESSAGES ON `turn on searching and sorting messages
Else `User selected Cancel Button

End if ` If(bOK=1) `User selected Proceed button

Else `User selected Menu item other than Print Merge
  WR DO COMMAND ($1;$2;$3) `Do the command through 4D Write
End case

```

## 4D Print Selection Printing

Performing a “Print Selection” within 4D Write can be used effectively to programmatically create and name PDF documents.

```

`gpPrintSelection
C_BOOLEAN($Continue)
$Continue:=False

If ((z_n_Platform=Windows) `This example is only set up for single User
  If ((PDF_Name # "")) `Ensure Windows PDF Driver has been obtained
    $Continue:=True
  Else
    gpCenterWindow (352;131;1;""))
    vMessage:="The PDFWriter printer must first be made the default printer and then saved"
    vMessage:=vMessage+" into the database by selecting menu item Assign PDFWriter."
    DIALOG([Constants];"Show Status")
  End if

```

```

Else
  If ((<<PDF_Name # "")) & (<<PDF_Driver # ""))
    $Continue:=True
  Else
    gpCenterWindow (352;131;1;""))
    vMessage:="The PDFWriter printer Driver must first be made the Chooser default and
    then"
    vMessage:=vMessage+"saved into the database by selecting menu item Assign PDF
    Writer."
    DIALOG([Constants];"Show Status")
  End if
End if

If ($Continue)
  gpCenterWindow (260;130;4;"Output Options")
  DIALOG([Constants];"Print_or_PDF")
  CLOSE WINDOW

  If (bOK=1) `User selected Proceed button
    If (bPrint=1) `Print Selection to current Print driver

      PRINT SELECTION(vFilePtr->)

    Else `Print Selection to PDFWriter
      $Err:=PD_GetPrinter (vname;vptype;vzone;vdriver)

      $Err:=PD_SetPrinter (<<PDF_Name;<<PDF_Type;<<PDF_Zone;<<PDF_Driver)

      PRINT SELECTION(vFilePtr->)

      $Err:=PD_SetPrinter (vname;vptype;vzone;vdriver) `Return Printer to it's default driver
      $Err:=PD_CleanUp `Reset Acrobat defaults
    End if
  End if
End if

```

## 4D Print Form Printing

Performing a “Print Form” within 4D requires extra coding for Windows. 4th Dimension on Windows encounters a problem when first attempting to print after a print driver has been changed. This is very important when dealing with the 4D “Print Form” command. The following rule is critical on Windows:

On Windows machines, the 4D Developer must issue a 4D “Print Selection” command to a null set of records immediately after changing Printer drivers (using the PDF Direct “PD\_SetPrinter” command).

```

If (<>z_n_Platform=Windows) `This example is only set up for single User
    Query([PDF_Log];[PDF_Log]Unused1="***??**") `Ensure 0 records are found and printed
    Print Selection([PDF_Log];*) `Specify asterisk to avoid print dialog
End If

```

```

`gpOrder; Assist user in ordering PDF Direct

```

```

C_BOOLEAN($Continue)

```

```

$Continue:=False

```

```

If (<>z_n_Platform=Windows) `This example is only set up for single User
    If (<>PDF_Name # "") `Ensure Windows PDF Driver has been obtained
        $Continue:=True
    Else
        gpCenterWindow (352;131;1;")
        vMessage:="The PDFWriter printer must first be made the default printer and then saved"
        vMessage:=vMessage+" into the database by selecting menu item Assign PDFWriter."
        DIALOG([PDF_Log];"Show Status")
    End if

```

```

Else

```

```

    If ((<>PDF_Name # "") & (<>PDF_Driver # ""))
        $Continue:=True
    Else
        gpCenterWindow (352;131;1;")
        vMessage:="The PDFWriter printer Driver must be made the Chooser default and then"
        vMessage:=vMessage+"saved into the database by selecting menu item Assign PDF
            Writer."
        DIALOG([PDF_Log];"Show Status")
    End if

```

```

End if

```

```

If ($Continue)

```

```

    gpCenterWindow (540;290;4;"PDF Direct Order Form") `Set up for 13" Screen
    DIALOG([Contacts];"OrderEntry")
    CLOSE WINDOW

```

```

If (bOk=1) `The User clicked the PDF button

```

```

    vPrompt_Option:=True
    vViewer_Option:=True `Load Acrobat Reader option enabled - in case user wants to see
    what they printed

```

```

    vMerge_Message:="Print Current Order Form"
    gpCenterWindow (228;265;1;")
    vTitle:="PDF Print Setup"
    DIALOG([Constants];"PDF_Options")
    CLOSE WINDOW

```

C\_TEXT(\$Path)

\$Name:=vCreator

vname:="" `Port name (from old Mac days), Used for both WIndows & Mac OS

vptype:="" `What type of printer; serial = ""; Print might be "LaserWriter"; Mac only

vzone:="" `Default Zone is "" for single zone or local zone

vdriver:="" `Name of Mac OS Extension

\$Err:=PD\_GetPrinter (vname;vptype;vzone;vdriver) `Get the current default printer

\$Err:=PD\_SetPrinter (PDF\_Name;PDF\_Type;PDF\_Zone;PDF\_Driver) `Now set to  
Adobe's PDFWriter

Case of

: (bAutoName=1)

\$Name:=vCreator+" Order.pdf"

\$Path:=\$Name

\$DocName:=\$Path

: (bPromptName=1)

\$Path:=""

: (bPromptDefault=1)

\$Name:=vCreator+" Order.pdf" `Get the name entered on order form

\$Err:=PD\_GetPath (\$Path;"Select a path";\$Name) `Prompt User with file name for editing/  
selecting

\$WhereDelete:=Position(\$Name,\$Path)

\$ShortPath:=Delete string(\$Path;\$WhereDelete;250)

End case

If (bPromptName=0)

\$Err:=PD\_Define (\$Path;0) `Pass either a 1 for True or 0 for False

Else

`User wants the Arcrobat dialog

End if

If (z\_n\_Platform=Windows) `This example is only set up for single User

Query([PDF\_Log];[PDF\_Log]Unused1="\*\*?\*?") `Ensure 0 records are found and printed

Print Selection([PDF\_Log];\*)

End If

For (\$y;1;vCopies) `Use 4D Customizer to set "Print One Job" ON

PRINT FORM([Contacts];"OrderForm")

End for

PAGE BREAK

```
If (bPromptName=0) `We only have access to PDF docs we've named outside of Acrobat
dialog
  ARRAY STRING(31;aDocList;0)
  DOCUMENT LIST($ShortPath;aDocList) `Get list of current directory documents
  $x:=Find in array(aDocList;$Name)
  If ($x=-1) `Document not found
    ALERT("Doc name may be altered (appended with numeric suffix) or a memory problem
    may e")
  End if
End if ` If (bPromptName=0)

$Err:=PD_SetPrinter (vname;vptype;vzone;vdriver) `Return Printer to it's default driver

End if ` If (bOk=1) `The User clicked the PDF Print button
End if
```

## PDF Direct Command Summary

<i>Routine</i>	<i>Page #</i>
<b>PD_Register</b> (LicenseeName(S);LicenseCode Macintosh(S); LicenseCode Windows(S)):L.....	6
<b>PD_IsAvailable</b> ->(I) .....	6
<b>PD_GetPrinter</b> (Name(S);Type(S);Driver(S);Zone(S)):I .....	7
<b>PD_SetPrinter</b> (Name(S);Type(S);Driver(S);Zone(S)):I .....	8
<b>PD_GetPath</b> (Pathname(S);Prompt(S);DefaultFileName(S)):I.....	9
<b>PD_Define</b> (Pathname(S);LaunchReader(I);Capture Reference(S)):I .....	10
<b>PD_GetStatus</b> (Capture Reference(S);Status(I)):I .....	11
<b>PD_CleanUp</b> :I .....	12



# Index

## **Numerics**

- 4D Print Form Printing 20
- 4D Print Selection Printing 19
- 4D Write PDF Printing 16

## **C**

- Command Descriptions 6
- Command Summary 23
- Command Syntax 1

## **I**

- Installation details 2
- Introduction 1

## **P**

- PD\_Cleanup 11, 23
  - description of 11
  - Example 12
- PD\_Define 10, 23
  - description of 10, 11
  - Example 11
  - Parameters 10, 11
  - Return Value 10, 11
- PD\_GetPath 9
  - description of 9
  - Example 9
  - Parameters 9
- PD\_GetPrinter 7, 23

- description of 7
- Example 7
- Parameters 7
- Return Value 7

- PD\_IsAvailable 6, 23
  - description of 6
  - Return Value 7

- PD\_Register 6, 23
  - description of 6
  - example 6
  - parameters 6

- PD\_SetPrinter 8, 23
  - description of 8
  - Example 8
  - Parameters 8
  - Return Value 8

- PDF Direct Overview 3
- PDF Direct Setup 12

## **S**

- sample application 12
- setting up PDF Direct 12