

# ADA news

## In This Issue

Volume 3, Number 11

p . 1

Integrating with the Acrobat  
File-Creation Products

p . 2

How to Reach Us

p . 4

Questions and Answers

p . 6

Developing with Illustrator

## Integrating with the Acrobat File-Creation Products

The previous two articles in this series described Adobe™ Acrobat™ Exchange version 2.0's plug-in API and interapplication communication (IAC) support. In this concluding article of the series we'll describe the ways in which you can control the other Acrobat version 2.0 products and integrate them into existing systems. We'll describe the interfaces for the PDF Writer, the Distiller™ program, and the Acrobat Catalog™ program. Figure 1 shows an overview of the Acrobat products, with rectangles representing programs, bent-corner pages representing files, and semicircles representing plug-ins.

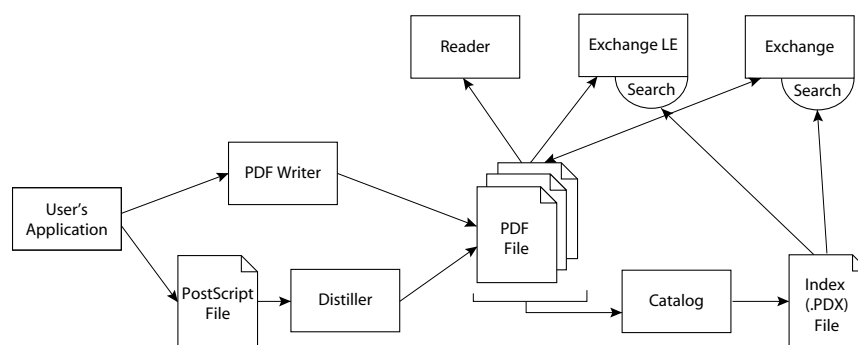


Figure 1 Overview of the Acrobat products

As in Acrobat Exchange, the IAC interfaces we'll describe are implemented using each platform's native interprocess communication mechanisms: Apple® events and AppleScript™ on the Apple Macintosh® computer, and DDE or Windows broadcast messages in the Microsoft® Windows™ environment.

### PDF Writer

The PDF Writer is a system-level printer driver that uses the platform's native imaging model (QuickDraw™ on the Macintosh, GDI under Windows). It allows you to "print to PDF" from almost any existing application. The PDF Writer does not have a true API, but is controlled by storing values in a PREC resource on the Macintosh, and in the WIN.INI file under Windows. The control interface is simple, but powerful. It allows your application to:

- Control whether or not the "Save File As" dialog box is displayed.
- Programmatically set the PDF file's name.
- Control whether or not the Acrobat viewer is automatically launched to display the PDF file after it has been created.

continued on page 2

## How To Reach Us

### DEVELOPERS ASSOCIATION HOTLINE:

U.S. and Canada:

(415) 961-4111

M-F, 8 a.m.-5 p.m., PDT.

If all engineers are unavailable, please leave a detailed message with your developer number, name, and telephone number, and we will get back to you within 24 hours.

Europe:

+31-20-6511-355

### FAX:

U.S. and Canada:

(415) 967-9231

Attention:

Adobe Developers Association

Europe:

+31-20-6511-313

Attention:

Adobe Developers Association

### EMAIL:

U.S.

devsup-person@mv.us.adobe.com

Europe:

eurosupport@adobe.com

### MAIL:

U.S. and Canada:

Adobe Developers Association

Adobe Systems Incorporated

1585 Charleston Road

P.O. Box 7900

Mt. View, CA 94039-7900

Europe:

Adobe Developers Association

Adobe Systems Europe B.V.

Europalaza

Hoogoorddreef 54a

1101 BE Amsterdam Z.O.

The Netherlands

*Send all inquiries, letters and address changes to the appropriate address above.*

## Acrobat File-Creation Products

The Windows PDF Writer also provides several additional capabilities:

- It sends Windows broadcast messages when it starts and when it finishes producing a PDF file.
- Additional WIN.INI settings allow you to control text and image compression and set the driver's resolution in dots per inch.

Using these features, you can for example build a system that programmatically creates PDF versions of a collection of documents created using different applications, then automatically copies the resulting PDF files to a server or into a document management system.

### Distiller

The Distiller program converts any PostScript language page description into a PDF file. The Distiller program for the Macintosh supports Apple events, also available using AppleScript, that let your application:

- Launch the Distiller program.
- Quit the Distiller program.
- Distill a file, optionally specifying the output path as well as the input path. You can also optionally specify that the Apple event should not complete until the file is done distilling.

The Distiller program for Windows supports both command line options and Windows messages. Using command line options, your application can:

- Specify that the Distiller program should automatically quit when it has distilled all the files in its queue and all watched directories are empty.
- Specify a list of files to distill and specify a directory into which the resulting PDF files are placed.

Using Windows messages, your application can:

- Specify a list of files to distill, a directory into which the resulting PDF files are placed, and an optional window handle. A message is sent to the specified window handle when each file has completed distilling. The message includes the input and output pathnames.

Using these features in the Distiller program for Macintosh or Windows platforms, you can for example build a system that has multiple queues for distilling files, with each queue having a different priority, or running at a different time of day. The system can also automatically copy the PDF files to a server or check them into a document management system.

## Attention: Developers!

*Each month, we reserve space in our newsletter for input from our developers. Our Developers Column is a place for members of the ADA to share information with other developers.*

*If you would like to contribute to our Developers Column, please contact us with your ideas. We look forward to hearing from you.*

## Acrobat File-Creation Products

### Acrobat Catalog

Acrobat Catalog is a newly-introduced program that builds the full-text search indices used by the Acrobat Search™ plug-in. It is currently available only in a Windows version. It scans specified directories or volumes and indexes any PDF file it finds.

Acrobat Catalog supports DDE messages that allow your application to:

- Open an index file and display the “Edit index definition” dialog.
- Open an index file and build the index.
- Quit the application.


In addition, Acrobat Catalog uses Windows broadcast messages to:

- Allow other applications to query whether Acrobat Catalog is running and, if it is, whether or not it is currently building an index.
- Notify other applications when it finishes building an index, is stopped while building an index, or fails while building an index.

### Conclusion

All of the Acrobat products now provide interfaces that allow you to customize them and integrate them into existing systems. The uses to which they can be put are in many ways limited only by your imagination. To help your imagination along, we’ve created a Developers Information Kit that contains much of the information that has been presented in this series of three articles, plus information on licensing, marketing programs, and suggestions on the types of integration that can be done for various types of software such as authoring applications, e-mail applications, document management systems, and several others.

To integrate the Acrobat products into your application or system, you’ll need one of the Acrobat Software Development Kits (SDKs). If you’re developing plug-ins as well as using the other integration hooks into the Acrobat products, you’ll need the Acrobat Plug-Ins SDK. If you’re not developing plug-ins, but are using the other interfaces, you’ll need the Acrobat SDK. Both SDKs contain detailed technical information on the interfaces described in this series of three articles, as well as sample code to help you get started, and limited technical support.

Contact the Adobe Developers Association to obtain the Acrobat Developers Information Kit, and for information on the availability and pricing of the Acrobat SDKs. 

## Questions Answers

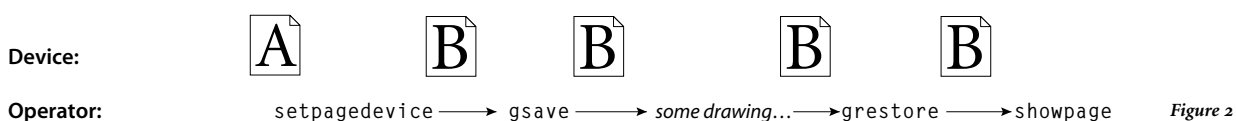
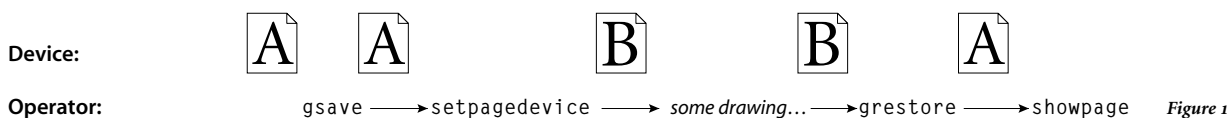
**Q** I want to turn my PostScript language file into an EPS file. As one step in this conversion, I wrap a **gsave**/**grestore** pair around the file. The original file prints out fine. However, when I add a **gsave** to the beginning of the file and a **grestore** right before the **showpage** call, I get a blank page as my output. The PostScript language code is as follows:

```
%!PS-Adobe-3.0
gsave
%%BeginProlog
.
.
/m /moveto load def
/l /lineto load def
.
.
%%EndProlog
%%BeginSetup
%%BeginFeature: *ManualFeed False
1 dict dup /ManualFeed false put setpagedevice
%%EndFeature
.
.
%%EndSetup
%%Page: 1 1
360 410 m
360 530 l
stroke
.
.
grestore
showpage
%%EOF
```

**A** The problem is caused by the **setpagedevice** call. **setpagedevice** installs a new raster output device in the graphics state. Since the device is part of the graphics state, it is affected by **gsave** and **grestore**. Tables 4.1 and 4.2 of the *PostScript Language Reference Manual, Second Edition* list the parameters that are part of the graphics state.

Among other things, the **gsave** operator saves a copy of the current device; and **grestore** restores the device to the one in use at the time of the last **gsave**. For the sake of discussion, suppose device A is the device into which you are rendering at the start of the job. By calling **setpagedevice**, you install a new device which we will call device B. You render your page on device B and are ready to print. At this point, you call **grestore**, which restores the graphics state that was present at the time of the last **gsave**; this call to **grestore** switches you back to device A. Figure 1 shows the device after each operation. Since you have not rendered any graphics into device A, a **showpage** results in a blank page printing. One solution is to move the **setpagedevice** call to a position in the code before the call to **gsave** as shown in Figure 2.

**save** and **restore** will cause the same problem when they encompass a call to **setpagedevice**. This is one reason why **setpagedevice** is not allowed in EPS files, since the importing application should always surround an EPS file by a **save/restore** pair. For a list of other operators that are illegal in EPS files, see section H.2.4 of the *PostScript Language Reference Manual, Second Edition*.



**Q** My application creates both PDF and PostScript language files directly from our driver. I have a problem with a particular PDF file that our application creates. When I use Acrobat Exchange or Acrobat Reader version 1.0 to view the file, I receive the following error message:

An error occurred while reading a note or link.  
A page object is missing or invalid.

**What is wrong with this code:**

```
%PDF-1.0
.
.
5 0 obj
<<
/Length 457
>>
stream
.
.
endstream
endobj
.
.
7 0 obj
<<
/Type /Page
/Parent 8 0 R
/Resources <<
/Font << /F2 3 0 R >>
/ProcSet 2 0 R >>
/Contents 5 0 R
>>
endobj
.
.
8 0 obj
<<
/Type /Pages
/Kids [ 5 0 R ]
/Count 1
/MediaBox [ 0 0 792 612 ]
>>
endobj
.
%%EOF
```

**A** The problem with the code is that a Pages object is incorrectly referencing a stream object, instead of referencing Page objects or other Pages objects. In the PDF format, the pages of a document are accessible through a tree of nodes

known as the “Pages tree”. Figure 1 is an example of a Pages tree. From the code we know that object 5 is a stream object, object 7 is a Page object, and object 8 is a Pages object.

The Kids array in a Pages object specifies the children of the Pages object. Figure 2 is the Pages tree of the program. The leaves of a Pages object can only be Page objects or Pages objects, but object 5 is neither of those. This is the cause of the error message.

To fix this code, you need to change the Kids array to [7 0 R] since object 7 is the Page object with object 5 as its contents. Figure 3 shows this correction. For more information on object parameters, see the *Portable Document Format Reference Manual*.

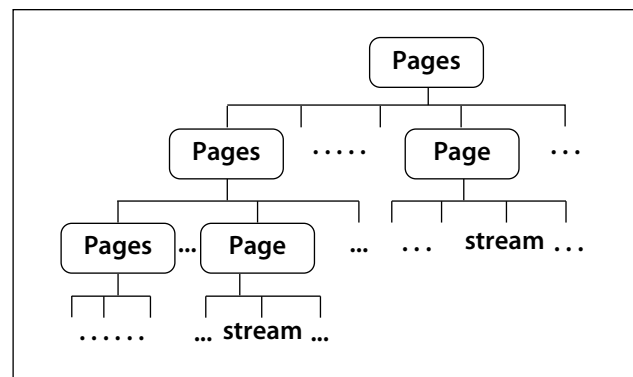


Figure 1 An example of a valid Pages tree

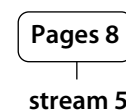


Figure 2 The Pages tree of this program

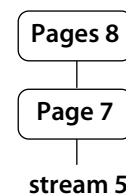


Figure 3 The corrected Pages tree

## Developing With Adobe Illustrator

**Q** I am having trouble implementing the `SetPlacedFile()` function of the Adobe Illustrator™ API. I am calling it with a volume reference number and a Pascal string with the file name. With Adobe Illustrator version 5.5 this even causes a crash. What am I doing wrong?

**A** Some clarification is in order regarding the Adobe Illustrator API function:

```
SetPlacedFile( AIArtHandle art, short volumeNumber, char *fileName).
```

In the Adobe Illustrator 5.0 API implementation, a working directory reference number should be passed in for `volumeNumber`. Check out *Inside Macintosh* for more information on working directories. The filename for this call is a C style string. If you get a file name with a Macintosh Standard File dialog you will need to convert it, since Standard File returns a Pascal string.

In the Adobe Illustrator 5.5 API, the interface has changed. The new function is:

```
SetPlacedFSSpec( AIArtHandle art, FSSpec *file)
```

and it replaces the 5.0 API call. All future API releases will use this call. To ensure that a plug-in created with the 5.0 SDK headers works with Adobe Illustrator 5.5 or later, you need to coerce the `SetPlacedFile()` call so that the correct number and types of arguments are passed. Fold the `SetPlacedFSSpec()` declaration into a `typedef` and use it when the placed art call is made with Adobe Illustrator 5.5. Check the version of Adobe Illustrator being used and make the call accordingly.

The discussion above and the sample code below also apply to the API call `GetPlacedFile()`, where a similar change was made. Like `SetPlacedFSSpec()`, `GetPlacedFSSpec()` will be used in all future versions of Adobe Illustrator, so you will need to do similar function coercion to be compatible.

The following code shows how to place an EPS file under either version 5.0 or 5.5 of Adobe Illustrator. *It assumes you are using the 5.0 header files.*

```
// Wrapper to call Get/SetPlacedFile() via Illustrator 5.0 header files.
typedef pascal FXErr (*SetPlacedFSSpecProc) ( AIArtHandle art, FSSpec *file );
typedef pascal FXErr (*GetPlacedFSSpecProc) ( AIArtHandle art, FSSpec *file );

...
// At this point, assume pathName is a string with the full path to the
// desired file and fileName is a string with just the desired file's name.
// The variable k is a pointer to the list of API functions.
// error is of type FXErr.
if (pb->interfaceVersion == 0x05000001) // The original API is running
{
    // so do normal 5.0 API call

    short      rc;
    WDPBRec    wdpb;

    wdpb.ioNamePtr=(StringPtr)pathName; // The full path name
    wdpb.ioWDDirID=0;                   // so using 0 for
    wdpb.ioVRefNum=0;                   // rest of record
    wdpb.ioWDProcID=0;

    rc = PBOpenWD( &wdpb, FALSE );
    if ( rc ) {
        error = rc;
        goto handleError;
    }

    PtoCStr((StringPtr)fileName); // Make sure the file name is
                                // a C string!

    error = k->SetPlacedFile( image, wdpb.ioVRefNum, (char*)fileName );
}
else // The version of Illustrator
{
    // is 5.5 or later, so use the
    FSSpec myFSSpec; // new call with an FSSpec

    pStrCopy(fileName, myFSSpec.name); // Note: this call uses
                                    // a Pascal string!

    myFSSpec.vRefNum = 0; // Example uses default volume
    myFSSpec.parID = 2; // and the root directory
    error = ((SetPlacedFSSpecProc)k->SetPlacedFile)( image, &myFSSpec );
}
if (error) goto handleError;

...
```

The 5.5 API headers use the FSSpec calls; so to be compatible with 5.0 you would coerce them to the 5.0 API functions.

The code would be similar to the above, but the function coercion would be as follows. *Note: This example assumes you are using the 5.5 API headers.*

```
// Wrapper to call Get/SetPlacedFSSpec() via Illustrator 5.5 header files.

typedef    pascal FXErr (*GetPlacedFileProc) ( AIArtHandle art,
                                                short *volumeNumber, char *fileName );

typedef    pascal FXErr (*SetPlacedFileProc) ( AIArtHandle art,
                                                short volumeNumber, char *fileName );

if (pb->interfaceVersion == 0x05000001) // The original API is running
{
    // so coerce to 5.0 API call
    ...
    error = ((*SetPlacedFileProc)k->SetPlacedFile)( image,
                                                    wdpb.ioVRefNum, (char*)fileName );
}
else
    // The version of Illustrator
    {
        // is 5.5 or later, so use the
        ...
        // normal 5.5 API call
        error = k->SetPlacedFile( image, &myFSSpec );
    }
if (error) goto handleError;
...
```

## C o l o p h o n

This newsletter was produced entirely with Adobe PostScript software on Macintosh and IBM® PC compatible computers. Typefaces used are from the Minion™ and Myriad™ families, all of which are from the Adobe Type Library.

Managing Editors:

**Nicole Frees, Debi Hamrick**

Technical Editor:

**Jim DeLaHunt**

Art Director:

**Gail Blumberg**

Designer:

**Karla Wong**

Contributors:

**Tim Bienz, Matt Foster,  
Sun-Inn Shih**

Adobe, the Adobe Logo, Acrobat, Acrobat Catalog, Acrobat Search, Distiller, Illustrator, Minion, Myriad, PostScript and the PostScript logo are trademarks of Adobe Systems Incorporated or their subsidiaries which may be registered in certain jurisdictions. Apple and Macintosh are registered trademarks and AppleScript and QuickDraw are trademarks of Apple Computer, Inc. Microsoft is a registered trademark and Windows is trademark of Microsoft Corporation. IBM is a registered trademark of International Business Machines Corporation. All other brand and product names are trademarks or registered trademarks of their respective holders.

©1994 Adobe Systems Incorporated.  
All rights reserved.

Part Number ADA0052 10/94



**Adobe PostScript**