Adobe Developers Association

ADA news

In This Issue

Volume 4, Number 1

The Year in Review

This has been a year of change for the Adobe Developer Relations Group and the Adobe Developers Association. The group has grown significantly during the year, and the Association now serves developers in a much broader range of fields than it did twelve months ago.

We would like to give you a recap of our year and discuss where we think we are headed. As a result, we hope you can better understand our support strategies and help us improve our services to you in 1995.

First of all, many thanks to the Developer Relations team for the hard work they continually put into supporting the Adobe Developers Association members. We would like to recognize the whole group here. You hopefully will recognize many of the names listed below:

IN THE U.S.:

Tim Bienz, Developer Support Engineer for Acrobat

John Ciccarelli, Developer Support Engineer for Acrobat

Jennifer Cohan, Manager, Adobe Developers Association

Debbie Dahl, Administrative Assistant

Jim DeLaHunt, Manager, Developer Technical Support

Mark Donohoe, Computer Scientist, Developer Tools

Matt Foster, Developer Support Engineer for Adobe Illustrator

Nicole Frees, Developer Support Engineer for PostScript

Dave Hackel, Developer Support Engineer for Acrobat

Debi Hamrick, Developer Programs Specialist

Cynthia Johnston, SDK Project Manager

Terrie Kerth, Director, Acrobat Developer Support

Jeff Matulich, Evangelist

Mike Mitchell, Developer Support Engineer for PageMaker Additions

Kathe Morris, Developer Support Engineer for Font OEMs

Terry O'Donnell, Developer Support Engineer for Fonts

continued on page 2

Adobe

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
Europlaza
Hoogoorddreef 54a
1101 BE Amsterdam Z.O.
The Netherlands

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

The Year In Review

Carrie Requist, Evangelist

Michelle Sellars, Developer Support Engineer for Fetch

Sun-Inn Shih, Developer Support Engineer for ADA Hotline

Tracey Stewart, Manager, Developer Support Programs

Ed Svoboda, Developer Support Engineer for Acrobat Software

Robert Teng, Developer Support Engineer for Adobe PostScript Printer Drivers

Elisa Thompson, Manager, Developer Support Systems

Dave Wise, Developer Support Engineer for Photoshop and Premiere

IN EUROPE:

Mirjam van Avezaath, Manager, Adobe Developers Association

Mike Clarke, Director, Engineering Support

Trefor Jones, Developer Support Engineer for ADA Hotline

Roberto Loria, Developer Support Engineer for Applications Products

Ivan Markovic, Developer Support Engineer for PostScript Software

It was a busy year. We upgraded Lasertalk™ for Windows™ to version 1.3 and started distributing it in May. Thank you to those of you who participated in the beta test and helped to make it a better product.

We decided the best strategy for our UNIX® developers was for them to purchase Display PostScript™ software via the Visual Action Toolset, created by Visual Edge Software. Display PostScript software and support can be purchased separately, as part of the Visual Action Toolset. Bluestone is the U.S. distributor for Visual Edge. Bluestone may be reached by phone at (609) 727-4600, or by e-mail at blustone!info@uunet.uu.net. The European distributor is Protek, which may be reached by phone at (44) 01628-75959, or by e-mail at Info@protek.co.uk. We sincerely hope that this new distribution method will work well for you.

We redesigned the *ADA News* "look" and started columns specific to the different Adobe technologies, such as the Acrobat™ column and the Developing with Adobe Illustrator™ column. You will see more of these technology-specific columns this new year in our effort to provide more focused and pertinent help regarding the technologies you are most interested in.

The Year In Review

A major step forward in 1994 was getting funding for and expanding our team's support for Adobe Photoshop™, Adobe Premiere™ and Adobe Illustrator. We have been delighted to be able to provide this support to you, and it has been a real pleasure to begin to get to know our plug-in developer community. We look forward to building on those relationships this year.

Another major event in 1994 was the merger with Aldus Corporation. As a result, we added developer support programs for PageMaker™ developers and Fetch™ content publishers and developers. And we have developer support engineers in Seattle near the engineering teams to make sure we can provide the best support and tools possible. We have been working with David Appel, Director of the Aldus Developers Cooperative (soon to be renamed) and spent time understanding how the co-op can help our plug-in developers. We hope you will take advantage of the co-op's services.

Adobe Acrobat 2.0 was launched in September along with a brand new developer support program strategy offering subscription services. The subscription service programs offered across all Adobe technologies allow us to send interested members automatic SDK updates as soon as they are available and so keep you updated with the latest information. We are very excited about these new services and hope to continue to provide you with quality technical information and sample code on a timely basis, so that you can continue to support Adobe technology in the finest way possible.

With Acrobat 2.0 we have added system integrators and VARS to our list of types of developers that we support. We now have developers of all types in the ADA:

1. **Independent Software Vendors**—Shrink wrap retail products
2. **Corporate Developers**—in-house corporate developers
3. **Plug-in Developers**—building "add on" products for our applications
4. **System Integrators and VARS**
5. **Direct Independent Software Vendors**—sell direct to customers and work with system integrators and VARS, example: document management companies
6. **Content Publishers**—building CD products with Adobe Acrobat and Fetch

The ADA is now a very diversified group of people. It's not just about PostScript™ technology anymore. We welcome the change here in Adobe Developer Relations, and we hope you will help us to provide the services and support you need. Please feel free to contact us and suggest ways in which we can improve our service to you in the future.

All our best in the New Year. §

Developing With Adobe Fetch

Adobe Fetch is a multimedia database product for the Macintosh® that allows customers to catalog, browse, search, retrieve, and reuse graphic, movie, and sound files.

Tips for Fetch Content Publishers: Getting the Best Thumbnails for Your Money

If you are using Adobe Fetch to catalog images on your CD, you want your thumbnails to represent your images in the best possible light. The following tips will help you determine which settings in Fetch will result in the best looking thumbnails for your images.

All of the following tips involve changing the settings in Fetch's Add/Update dialog box.

Tip #1

Set the Modification Method to "Add/Update" and save that as your default. This will ensure that all of your items are added to the catalog.

Tip #2

Which thumbnail type(s) you should select in the Add/Update dialog box should be based on the type of CPU (not monitor type) your end user is likely to own.

1-bit (black and white) thumbnails are for users who can't run color QuickDraw™ (Mac® Portable, SE, MacPlus). If 8- or 32-bit thumbnails are used in this situation, your user will see an error thumbnail, letting the user know that the thumbnail could not be drawn.

8-bit (256 colors) are for machines that don't have the QuickTime® extension installed.

32-bit (millions of colors) can be viewed on any Mac that has QuickTime installed.

We recommend that you only create 32-bit thumbnails (even for black and white source files). Because Fetch has built-in QuickTime thumbnail compression algorithms, 32-bit thumbnails consume less space in the catalog than 8-bit thumbnails. Since 32-bit thumbnails require QuickTime, you might consider licensing the QuickTime extension from Apple and including it on your CD.

Here is a matrix to show you the trade-offs in size for various thumbnail and image types:

	Thumbnail Types		
Image Type	**1 Bit**	**8 Bit**	**32 Bit**
Simple B&W	2126 Bytes	4096 Bytes	3113 Bytes
Complex B&W	2294 Bytes	6472 Bytes	5816 Bytes
Simple Color	2212 Bytes	6308 Bytes	5571 Bytes
Complex Color	2458 Bytes	9994 Bytes	7782 Bytes

Tip #3

Set the Compressor Criteria to "Any." This setting will tell Adobe Fetch to choose the best compressor for your file types, if multiple compressors are available.

Tip #4

Set the Compressor Type to "Best Overall" when cataloging digitized photographs. This setting will be fast while maintaining image fidelity. For any other image type use the Graphics compressor. (You must select "Other…" from the Compressor Type pop-up menu to display the Compressor Setting dialog.)

Note: Compressor Criteria and Compressor Type only apply to 32-bit thumbnails.

Tip #5

Enable the "Extract Keywords" checkbox if you have attached keywords to your source files using the "pnot" Editor utility that is included with the Fetch Content Publisher's Toolkit. If you do not have this option checked, Fetch will ignore the keywords stored in the files.

Tip #6
Enable the "Extract Thumbnail" checkbox if you have attached custom thumbnails to your source files or if your source files were created by pnot-aware application, such as Adobe Photoshop or PageMaker. If you do not have this option checked, Fetch will ignore the thumbnail stored in the files.

Tip #7
Enable the "Extract Description" checkbox if you have attached descriptions to your source files using the `pnot` Editor utility. If you do not have this option checked, Fetch will ignore the descriptions stored in the files.

Tip #8
Enable the "Include File Name As Keyword" checkbox if you want to include each item's file name as a keyword.

Tip #9
Enable the "Include Folder Name As Keyword" checkbox if you want to include each item's parent folder name (where the item is located) as a keyword.

Tip #10
Disable the "Mount Remote Volumes". You should not need to mount remote volumes that are referenced by aliases on your hard drive.

Tip #11
Disable the file types you do not want to catalog by clicking on the check mark next to the file type. This will cause certain dialog boxes to come up faster for your end users.

Tip #12
QuickTime movies and Adobe Photoshop 2.5.1 files store smaller thumbnails than Fetch. If you are cataloging these file types, follow these instructions: update the items with the small thumbnails in Fetch using the Modification Method of "Update Unconditionally" with the "Extract Thumbnail" option disabled.

Tip #13
Inspect the quality of every thumbnail. Look for banding in areas of smooth color transitions. Digitized photos of skies are strong candidates for banding.

Use these steps to eliminate the banding:

1. Select Add/Update Items from the Admin menu.
2. Set High Quality (big/slow) as the default Compressor Type. This setting will tell Fetch to be more concerned about the quality of the thumbnail versus the speed of generating the thumbnail.
3. Click Done to dismiss the Add/Update dialog.
4. Copy the thumbnails into the Pasteboard.
5. Select all the thumbnails in the Pasteboard.
6. Hold down the Option key and Select Update Now in the Admin menu.

As you can see, simple settings can make all the difference in the world. The tips we have provided are just recommendations. Experiment with the settings to determine which ones work best for you.

If you have more questions about Fetch thumbnail generation and the options available to you, refer to the *Fetch User's Guide*, pages 54–56. You may also refer to Chapter 4 of the CPT Part 2 document located on the Fetch Content Publisher's Toolkit CD-ROM. If you do not have the Fetch Content Publisher's Toolkit and you would like to order one, please contact the Adobe Developers Association.

Developing With Adobe PageMaker

Q What are PageMaker additions? What can additions do?

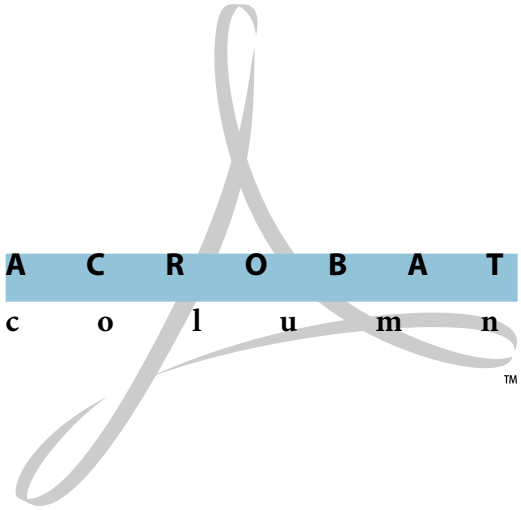
A PageMaker Additions can take many forms, from simple scripts written in Adobe PageMaker, to dynamically-loaded modules that appear on the PageMaker Additions submenu, to an application capable of supporting application-to-application communications (System 7.0 Apple events on the Macintosh or Dynamic Data Exchange, or DDE in Windows applications).

The additions mechanism provides an interface to the PageMaker application's commands and publication data. It extends control of PageMaker from direct mouse and keyboard action to include commands and queries generated from other sources, namely additions. Additions offer a new method to access and automate the PageMaker program's existing capabilities and provide greater functionality.

The range of possible additions is vast. We encourage additions to be written that can:

- Automate common tasks —allow users to organize, browse, and navigate publications, reformat pages, or compose pages from a boilerplate.
- Improve the transfer of information to and from PageMaker publications—automated layout from an external specification, check publications in and out of a server, export layout information to other systems, import and layout information from a database or spreadsheet.
- Supplement existing menu commands—add special print capabilities, for example, signatures or personal print settings, perform workgroup tasks prior to closing a document.
- Provide special functions—create a fill-in-the-blanks script generator, develop utilities for special markets, for example, listing the downloadable fonts and linked files used in a publication for service bureaus, automate style and outline generation, create scripts from existing publications.

This brief list should give you just an idea of what PageMaker additions are and how powerful they can be.



The Microsoft® Windows version of Adobe Acrobat Exchange 2.0 supports OLE 2.0. This provides new possibilities for developers to integrate Acrobat with other Windows applications.

Acrobat Exchange allows users to embed Portable Document Format (PDF) files into compound documents created in OLE container applications. Compound documents are single documents that contain data from different sources. For example, a PDF file can be embedded into a Lotus Notes® document.

Moreover, developers can incorporate additional functionality into their applications through OLE automation. An OLE client can send OLE automation messages to control Acrobat Exchange. Twelve OLE objects and over 120 methods are exposed in version 2.0.

One use of OLE automation is to display PDF files in your application's window. This is useful in a variety of situations including document management, communications, and CD-ROM information retrieval.

Before your application can open a PDF file using OLE automation, it must first create a new `CAcroAVDoc` object, which represents a single visible document within the Acrobat Exchange program. You must then invoke the `OpenInWindow()` method, providing a handle to the window into which Acrobat Exchange should draw. The following is a simplified code segment written for Microsoft Visual C++™ demonstrating this in action:

```
COleException e;

// Create new CAcroAVDoc object
m_pAcroAVDoc = new CAcroAVDoc;
m_pAcroAVDoc->CreateDispatch("AcroExch.AVDoc".&e)

// Open specified PDF file in window
m_pAcroAVDoc->OpenInWindow(pszPathName, (short) hWnd);
```

The Adobe Acrobat Software Development Kit (SDK) includes the documentation and code needed to develop software using OLE, DDE, and Apple events. This SDK contains several OLE automation example applications written using Microsoft Visual Basic™ and Visual C++. The Adobe Acrobat Plug-ins SDK contains everything in the Adobe Acrobat SDK, plus specifications and sample code for creating Acrobat viewer plug-ins using the Acrobat Exchange API.

If you would like to receive an Acrobat Developer Information Kit, with detailed information about the SDKs, contact the Adobe Developers Association.

Developers Column

The developers column is an avenue for members of the ADA to share information with the rest of the development community. The following was submitted by David Gelphman, an Adobe Developer Support alumnus.

In the Q&A section of a recent *ADA News,* you printed the following code to check the language level of a given PostScript output device:

```
/L2? {
   /languagelevel where {
      pop languagelevel 2 eq {
            true
        }{
            false
        }ifelse
      }{
           false
   } ifelse
}def
```

I would suggest using the following code in lieu of the above:

```
/My_L2?
   /languagelevel where {
         pop languagelevel 2 ge
   }{
         false
   } ifelse
def
```

There are actually three distinct differences between my suggested code and that used in the Q&A:

1) The first difference is that the definition of My_L2? is a boolean whereas the definition of L2? is a procedure which, when executed returns a boolean. Since the evaluation of the L2? procedure would be a constant for a given PostScript language implementation, there is no need to repeatedly execute it. This is a very common mistake that people make in their PostScript language code.

2) In My_L2? the return value from **languagelevel** is evaluated as to whether it is greater than or equal to 2. If the **languagelevel** operator ever returns a value > 2, My_L2? will work right. On the other hand, L2? will incorrectly indicate a Level 1 device.

3) The last point I'd like to make is about the portion of the code in L2? that looks like:

```
languagelevel 2 eq {
     true
}{
     false
}ifelse
```

This is equivalent to the expression:

```
boolean {true} {false} ifelse
```

If you think about it, this evaluates exactly to the expression: boolean. In other words, there is no need to do an **ifelse** on a boolean, the boolean itself is the desired result.

It's always amazing how a small bit of PostScript language code can actually have so many issues associated with it. I hope other developers in the ADA benefit from these comments.

Questions Answers

Q My application, which runs under Microsoft Windows, uses the Adobe Type Manager™ API to display arbitrarily transformed text. There is one problem that I have been unable to solve. I use the GDI calls CreateFont( ) **and** SelectObject( ) **to select a Type 1 font for screen display. When I have a Type 1 font and a TrueType™ font with the same name, such as Courier, a call to** ATMFontSelected( ) **returns** FALSE**. I think what has happened is that the TrueType Courier is selected by default instead of the Type 1 Courier. How can I ensure that the Type 1 Courier is used instead of the TrueType Courier?**

A When your application uses SelectObject( ), ATM™ will not have the opportunity to realize a Type 1 font if a TrueType font of the same name exists. Because GDI can realize a TrueType font, it just selects the TrueType font and rasterizes the text without ATM. To ensure a Type 1 font is used even when a TrueType font is available, use ATMFontSelected( ) to select the Type 1 font.

If you don't care who does the rasterization when outputting text, the code looks like this:

```
HFONT newfont, oldfont;
HDC hdc;
          .
          .
          .
newfont = CreateFont(...);
oldfont = SelectObject(hdc, newfont);
          .
          .
          .
SelectObject (hdc, oldfont);
DeleteObject(newfont);
```

To force ATM to do the rasterization, the code looks like this:

```
HFONT newfont, oldfont;
HDC hdc;
          .
          .
          .
newfont = CreateFont(...);
ATMSelectObject(hdc,newfont, ATM_SELECT, oldfont);
          .
          .
          .
SelectObject(hdc, oldfont);
DelectObject(newfont);
```

For details on the functionality of ATMSelectObject( ), please refer to technical note #5073, "Adobe Type Manager Software API: Windows" for more information.

continued on page 10

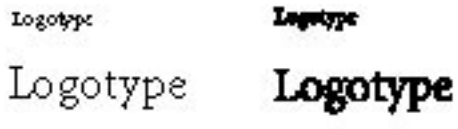
Questions & Answers

Q My company uses its logo in EPS format for a variety of company documents. The text part of the logo has been saved as graphic outlines. When I create a pdf file for electronic distribution, and view it in Acrobat, the quality of the text in the logo is relatively poor. Any suggestions about how this can be improved?

A When text is converted to a "graphic" in an EPS file, it loses the hint information in the Type 1 font that helps it scale accurately for small point sizes. The solution is to have the logo (symbol and text) made into a Type 1 font, using a commercial font utility (see below). When the PostScript language file is distilled and the logo font is embedded, the text will be rendered as a font, rather than as a graphic or scaled screen-preview image. This results in significantly better quality than the case where the logo is represented as a vector-based or scanned image graphic.

However, even if your logo consists of only a symbol, there are other advantages to converting it to a Type 1 font. The symbol will be rasterized more cleanly because it will be handled by the ATM rasterizer rather than the general graphics fill algorithm. Also, having the logo as a font is a great way to organize many variations and elements used for a logo. If your company has a different address for each division, or different configurations of the text and symbol are used in different contexts, it can be a great convenience to have all of the variations in a single Type 1 font, rather than having to manage a number of separate files. In addition, each application may have a slightly different method for importing and scaling EPS files, which can sometimes be confusing and tedious. With a Type 1 font, the logo is accessed by selecting it from the font menu, and scaling is simply a matter of changing the point size.

The figure below illustrates an example of text, as it would appear on a 72-dpi screen (the illustration was taken from a screen shot). The example (in 10- and 22-point) on the left shows the text represented as a font, and illustrates the benefits of the hint information. While such low resolution rendering is never ideal, the Type 1 font version is more clear and legible than the EPS version.



Screen renditions of text as a Type 1 font (left), and as an EPS file (right)

The characters of the Type 1 version are adjusted to fit the grid of the target resolution, and uniform stem weights are maintained. The text from the EPS file is rasterized with the graphics fill routine, which tends to "overfill" the shapes to make sure the thin strokes are connected. The resulting characters are much heavier in appearance; stem weights may not be equal, and curve shapes will be handled less aesthetically.

Note that when viewing and zooming in an application file (other than Acrobat), it is the screen preview image of the text that is usually being scaled, not the actual font or EPS graphic. The example on the left, above, consists of a screen shot of an Acrobat document, where the characters were rendered by ATM and Acrobat software.

There are several ways to convert a logo into a Type 1 font. You can try it yourself, or enlist the help of a designer. Two popular software products that allow the creation of a logo as a Type 1 font are Fontographer®, from Altsys Corporation, and FontMonger™, from Ares Software (see the listing of font utilities in the August 93 edition of the *ADA News*). Both of these programs allow you to directly draw a logo on the screen, and Fontographer allows you to either draw over, or even autotrace an imported bitmap image. Both programs

Questions & Answers

can also import EPS files created by either drawing or autotrace software such as Adobe Illustrator or Adobe Streamline™.

Once the Bézier representation of the logo has been created, these programs can "autohint" the font and create a Type 1 font program. Should you run into problems with the screen rendering, Fontographer allows you to adjust the hint information to obtain better results. This is only likely to be necessary with slightly more complex logos.

Type 1 fonts do have some restrictions regarding maximum range of coordinates and the complexity of the path. If a logo is extremely complex, it may need to be represented as a Type 3 font. If a logo is particularly wide, it might help to divide it into two characters, so that the user types two successive characters to display the logo. Also, in some cases of more complex logos, the hints may need to be edited by a knowledgeable font developer to achieve the desired quality.

The end result will depend on the complexities of the logo and the skill of the creator, but representing your logo as a font is important if you distribute electronic documents for on-screen viewing. §

Colophon

All proofs and final output for this newsletter were produced using Adobe PostScript software. The document review process was accomplished via electronic distribution using Adobe Acrobat software. Typefaces used are from the Minion™ and Myriad™ families from the Adobe Type Library.

Managing Editors:
Nicole Frees, Debi Hamrick

Technical Editor:
Jim DeLaHunt

Art Director:
Karla Wong

Designer:
Lorsen Koo

Contributors:
Nicole Frees, Mike Mitchell, Terry O'Donnell, Michelle Sellars, Sun-Inn Shih, Tracey Stewart, Ed Svoboda

Part Number ADA0054 1/95

Adobe PostScript