Initial Exploration of the Acrobat 5 SDK Software Development Kit

Don Lancaster
Synergetics, Box 809, Thatcher, AZ 85552
copyright c2002 as GuruGram #11
http://www.tinaja.com
don@tinaja.com
(928) 428-4073

The latest version of **Acrobat 5** from **Adobe Systems** provides some major new features and upgrades that include transparency, improved file import/export, and internal text capture. To fully explore this release, a new and free **Acrobat SDK Software Development Kit** is available.

Central to the kit is the ability to create your own **plugins** that let you interact both with **Acrobat** internals and your operating system to provide new and improved capabilities. I was (and remain) particularly interested in any ability to let raw **PostScript** code directly generate **transparency options**.

Because it took me much longer and cost a lot more than expected to get started, I thought I'd share some of my startup discoveries with you here. Especially the more frustrating and non-obvious gotchas.

On bootup, Acrobat looks into its **Plug_ins** folder (and all of its subdirectories) to find any code with a **.api** plugin trailer. Under Windows, properly written plugins are a variant of a **.dll** dynamic linking library module. These plugins are first installed and then initialized by Acrobat.

> **Note that all subdirectories are searched within the Acrobat plugin folder. If you want to hide or disable a plugin, you'll have to move it elsewhere!**

The more plugins you have, the greater the performance and flexibility. But the longer your startup time.

Plugins obviously have to interact with their operating system, so they will wildly differ between Windows, Mac, and Unix systems. Adobe chose to provide platform independent sourcecode written in **c++** for plugin support. This sourcecode is then compiled and linked by suitable software on the chosen system. After learning and using their example plugins, they can be modified or improved for your custom uses.

With one exception, ready-to-run plugins are absent from the SDK. You have to compile and link your own to proceed!

Not being either a Microsoft or a c-language enthusiast, I first tried installing the fine and free **Command Line c++ Compiler** from **Borland** and later on their **c++ Builder Trial Edition CD**. Only to discover I was in way over my head on linking problems. Some newsgroup inquiry at **comp.text.pdf** and **comp.lang.postscript** pointed out that the Adobe SDK is specifically intended for use with the **Microsoft Visual c++** program.

Compiling and linking Win32 Acrobat plugins with anything but Microsoft Visual c++ appears to be enormously difficult!

Which promptly led to another rude surprise. I first bought the Learning Edition of Visual c++, only to find that the words "full featured" clearly labeled on the product really mean "hopelessly crippled". You are allowed to compile code but not use it. Getting a blocking error message instead.

Thus…

The "learning editions" of Visual c++ are utterly and totally useless for generating exportable code!

Nowhere, anywhere was I able to find any mention of this. I then grudgingly found a standard edition of **Visual c++ 6.0** on **eBay** for around half list price. You have to uninstall the learning edition before you can install the standard edition.

Once installed, things finally started to work...

To win32 compile a SDK plugin, go to the win32 folder inside your chosen SDK plugin folder. Click on the file that ends in .dsw. This should autoboot Visual c++. Go to the Build menu. Click on the line that starts with "Build" and ends with ".api"

Which should auto compile and link everything you need, create the plugin, and install that plugin into a new or an already existing **AcrobatSDK** subdirectory you'll find in the main Acrobat plugin folder. Any modified or custom plugins should, of course, go in your own new folder.

A quick check can be run for successful installation by viewing the plugin name under **Help -> About Acrobat SDK Plugins** in Acrobat.

Naturally, once you start modifying your code, you'll get all sorts of arcane "**fails to compile**" error messages. Which leads you to another can of worms entirely.

Some Ready-to-Use SDK Plugins

I've gone ahead and compiled several of the SDK plugins for you. By using a Windows ME system. These plugins are quite useful in their own light and can be studied ahead of actually getting and using Visual c++. Note that these were all compiled **from** Adobe sourcecode but are **not** an Adobe product. The only warranty here is "approximate quantity — one".

Note further that **Adobe** places use restrictions on such compiled code. Details can be found on their SDK or on their main **website**.

Here are a few comments on some of the plugins available…

AboutAcrobatSDK.api

This is the only ready-to-use plugin on the SDK. It adds a new **About Acrobat SDK Plugins** menu item to the Acrobat **Help** menu. It both summarizes and fully describes all other available SDK plugins. It also shows whether any particular plugin is installed and ready to use.

While you can manually install this plugin ahead of time, it will automatically get added the first time you successfully compile any other SDK plugin.

AddImage.api

This plugin lets you add **.jpg** images to any pdf page. At present, the image has to be named **sample.jpg** and has to be in the same folder as your PDF file. If you want to change the image name, size, resolution, position, or transparency, you'll have to recode and recompile from the SDK. Be sure to save separate instances to your own plugin folder.

You add your image by clicking on **Add Image** in the SDK pulldown menu. It is fun to repeatedly place transparent images on top of each other to watch them get darker and contrastier. But don't do this in your final code as your file sizes will get outrageous.

Transparency.api

Gives you some really neat sliders that let you adjust the transparency of any already transparent object on a PDF page. You can use your own PDF files or this Adobe **threerects.pdf** demo.

You gain transparency control by clicking on **Transparency** in the SDK pulldown menu. Fill and stroke transparency is independently controllable. More on transparency in **GuruGram** #9 as **PSTRANS.PDF** .

VerifyURLs.api

This plugin is just plain wonderful. It will go through a PDF file and quickly and automatically check each and every URL by an actual online pinging. You access it from a new **Locate Web Addresses** in the **Tools** pulldown. You do, of course, have to be online for this feature to work.

There is also an option to convert any single line PDF string that starts with **http://** into an automatic url link. But I very much prefer my sourcecode-based **Gonzo** Utilities instead since they let you link any text to any url.

UncompressPDF.api

This plugin lets you remove most compression from a PDF document. It appears as a **Uncompressed PDF Files (*.pdf)** inside your **File/Save As** menu option.

Which is most handy for PDF code analysis. And considerably simpler than the methods we looked at back in **GuruGram** #8 as **FLATEVUE.PDF** Note that font compression is **not** removed by this plugin.

Watermark.api

This plugin lets you place watermarks (such as "copyright 2002" or "confidential", or "file copy" or whatever) onto any and all pages of your .PDF document. At present, a one line, fixed font message is centered on the page. The font size and angle of rotation is adjustable. There is also a choice of four fixed colors.

You add a watermark by clicking on **Watermark** in the SDK pulldown menu.

Consulting assistance on any and all of these topics can be found at **http://www.tinaja.com/info01.asp**.

Additional **GuruGrams** wait your support as a **Synergetics Partner**.