

This is an excerpt from a document I wrote while the Evangelist for Acrobat at Adobe Systems, Inc. This document was written in 1994 and originally published as part of the Acrobat Developers Information Kit which was sent to software developers who were interested in integrating their products with Adobe Acrobat. The intended audience for this document as well as the rest of the Acrobat Developers Information Kit were software engineers and engineering management hence the technical nature of the information.

Acrobat Integration Opportunities

Adobe Acrobat 2.0 is more than just a document communication platform, it is also a development platform. The depth and richness of the Acrobat 2.0 development environment allows for innovative and diverse projects including:

- Enhancing existing applications by incorporating Acrobat document viewing.
- Building effective electronic document communication systems around the Acrobat products.
- Customizing or extending the functionality of the Acrobat viewers through the creation of plug-ins.
- Automating the creation of Acrobat navigation and readability features (bookmarks, links, notes and articles).

To better understand the variety of development opportunities with the Acrobat products, this document provides a detailed look at the Acrobat 2.0 development environment. In addition, a set of documents is available that provides suggestions on integrating Adobe Acrobat with specific types of applications (e.g. document management or electronic mail). These documents are part of the Acrobat Developers Information Kit which can be obtained through the Adobe Developers Association.

General Compatibility Testing

The first step to integrating existing applications with Adobe Acrobat software is to perform compatibility testing between the existing application and the Acrobat products.

If your application can generate printed output of any kind, it should be tested with both the Acrobat PDFWriter and the Acrobat Distiller. Test by "printing" through the PDFWriter or by printing to a PostScript language file and sending this file through the Acrobat Distiller, and then compare the resulting PDF file to printed output.

Acrobat 2.0 Exchange viewer for the Windows environment is an OLE 2.0 server application. Applications that are OLE containers should test for compatibility by embedding PDF files using OLE.

Further compatibility testing depends on the features of your specific application.

Integrating with Adobe Acrobat

The Acrobat products provide a broad range of integration opportunities that allow developers to do far more than just invoke Acrobat to view a file. There are Application Programming Interface (API) calls and or Interapplication Communication (IAC) calls for every part of the Acrobat environment. These allow integration with:

- The Acrobat viewers (Reader, Exchange and Exchange LE)
- The PDF creation tools (PDFWriter and Distiller)
- The Acrobat Search system (Acrobat search plug-in and Acrobat Catalog)

The following section detail the development opportunities available with each part of the Acrobat environment.

Integrating with the Acrobat Viewers

The Acrobat development environment provides multiple ways for developers to enhance and control the Acrobat viewers. These includes using IAC and writing plug-ins to the Acrobat Exchange viewers using the API provides (see figure 1).

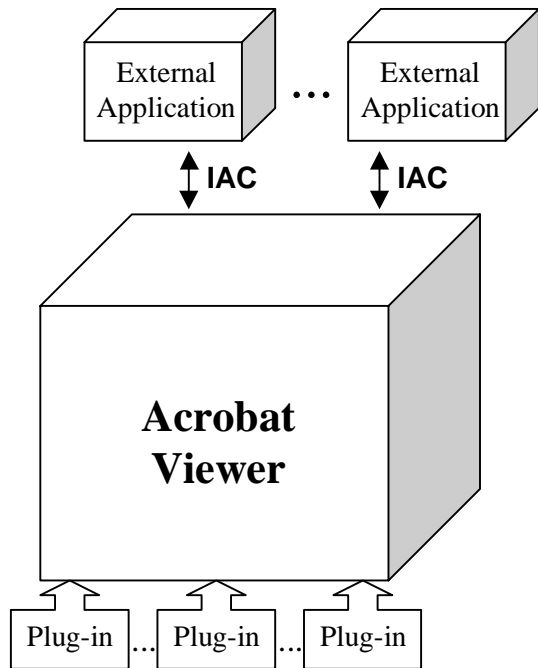
Plug-ins have access, through the extremely rich API, to the majority of PDF objects and Acrobat viewer features. Plug-ins can be written to provide stand alone functionality to the Acrobat Exchange viewers, or they can be used to integrate with an external application.

Object Oriented Interapplication Communication (OLE automation on Windows and Apple events on the Macintosh platform) gives the developer access to a substantial portion of what is provided in the API directly from an external application. In many cases, integration can be done solely through IAC without writing a plug-in. However, it is appropriate, and will sometimes be necessary, to combine the use of plug-ins with IAC. Additionally, IAC methods (including DDE) can be used for remote control of the Acrobat viewers.

The Acrobat Exchange viewers for Windows are OLE 2.0 server, providing a certain level of access for OLE container. Beyond just being OLE 2.0 servers, the Acrobat Exchange viewers also support the Lotus Notes/FX technology allowing Acrobat

viewing to become an integral part of a Notes application.

Figure 1: Two ways to integrate with the Acrobat Viewers



Acrobat Exchange Plug-In API

The Acrobat Exchange plug-in API allows the developer to enhance and control the Acrobat Exchange viewers in order to add functionality and integrate with existing environments. This API exposes the vast majority of Acrobat Exchange for use in development.

The API is available only from the Acrobat Exchange viewers; it is not supported by the Acrobat Reader. Acrobat Exchange supports the entire API, while Exchange LE supports all API calls except those that require a save action.

To use the API, you must write an Acrobat plug-in. Plug-ins are C or C++ programs implemented as DLLs on windows and code resources on the Macintosh. Acrobat Exchange looks in a special folder/directory for plug-ins when it launches. All plug-ins found will be loaded. The operation of plug-ins has three phases: initialization, operation and unloading.

Acrobat Exchange handshakes with each plug-in it locates, obtaining the plug-in's name and the address of its initialization and unload procedures. Plug-ins generally use initialization to add user interface items, define custom actions or custom annotations, or replace portions of the file system.

After initialization, a plug-in generally waits for the procedures it registered during initialization to be

called. Procedures will be called based on user actions or on the actions of other plug-ins.

Unloading currently occurs only when Acrobat Exchange quits. The plug-in unload procedure can clean up, close any files it might have opened, and free any memory it allocated.

Plug-ins themselves can enrich Acrobat exchange in a variety of ways. Examples of plug-in Functionality range from adding tools (e.g. highlighter pen or red-lining pen) to modifying the Acrobat file access procedures. Plug-ins can add private data (data which is not inherently part of the PDF structure) to a PDF file. Private data can be used to store a variety of information from new annotation types (e.g. highlight ink or red lines) to entire application files (e.g. the source document from which the PDF file was created). These are just a few examples of what Acrobat plug-ins can do. Plug-ins can also provide their own API which can be called from other plug-ins.

The API itself is organized into objects, although it is implemented using a plain C interface. There are approximately 50 object types organized into the following four groups (see figure 2):

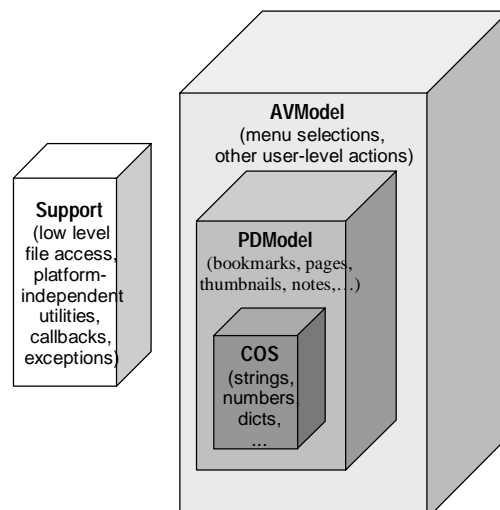
- *AVModel* —access to user level actions.
- *PDModel* —access to document level actions.
- *Cos* —access to low level data.
- *Support* —utilities and file access.

Object types represent items in Acrobat Exchange (e.g. menus and toolbar buttons), items in PDF files (e.g. fonts and notes), and files system objects (e.g. files). Each object type has methods that allow the creation, destruction, examination and modification of objects of that type. The API contains more than 500 methods.

Figure 2: Exchange Viewer API Organization

AVModel

The AVModel group contains objects that represent parts of the Acrobat Exchange application, including menus, menu items,



document windows, toolbar buttons, and the application itself. Some of the methods in this group allow you to open a window containing a PDF document, add or remove menus, menu items and toolbar buttons, and modify settings in the application's preferences file.

PDModel

The PDModel group contains objects that represent components in a PDF file, including pages, fonts bookmarks, notes and actions. The structure of a PDF file is described in The Portable Document Format Reference Manual published by Addison-Wesley and available in both of the Acrobat Software Development Kits and from the Adobe Developers Association. PDModel methods are used to extract text from a PDF file (see the Text Extraction Tools section later in this document), to add notes or actions, determine the fonts a document uses, etc.

Cos

The low-level data in a PDF file is represented by objects in the Cos group. Cos contains the seven data types of a PDF file: booleans, numbers, strings, names, arrays, dictionaries and streams. Cos methods are used to add private data to a PDF file, or to read or modify parts of a PDF file for which PDModel methods are not provided.

Support

The Support method group provides a collection of utility and file access methods. Included are utilities (fixed point math, memory allocation), file methods and methods that allow a plug-in to export its own callable functions to other plug-ins. Methods in this group are used to convert pathname specifications between platform-independent and native formats, or allow files to be read via modem or from a database by replacing the file system.

...