AFP Application Components and Processing

The successful processing of AFP files for the extraction of content and/or the conversion to PDF requires the deployment of the AFP Base Modules and a set of customizations. The base modules contain function to perform a generic set of conversions. Customizations can be performed to improve the efficiency, performance and fidelity of the resulting PDF.

Base Module Components

The base AFP module components consist of the following:

- AFP PageMapper
- PageMap XML to PDF Converter
- AFP to PDF Converter
- XML Browser

- AFP PageMapper

The AFP PageMapper is a utility that converts AFP files and resources into one of several predefined XML formats. The PageMapper uses several configuration files to tailor the conversion of AFP code page character data to Unicode in the XML. All AFP resources can be converted to XML. The configuration of the AFP character data to Unicode conversion is the most time consuming and difficult customization activity. The XML files produced by the PageMapper can be used to extract information, index or manipulate the AFP source or can be converted to PDF.

- PageMap XML to PDF Converter

The PageMap XML to PDF Converter utility converts the PageMapper MAP XML format to PDF. The configuration can be modified to include Type 3 raster fonts created from the raster AFP fonts, Type 1 outline fonts created from AFP outline fonts, and/or TrueType fonts. These options can be configured on a font by font basis. This is typically the main customization effort required when tailoring the PDF conversion.

- AFP to PDF Converter

The AFP to PDF conversion utility combines the processes of the PageMapper and PDF Converter into a single multi-threaded process. The recommended approach to implementing a new AFP application conversion is to first configure the AFP to XML PageMapper conversion then configure the XML to PDF conversion. When these both are correct then combine the configuration using the AFP to PDF converter for production. If content is to be added or manipulated in the PDF then the XML can be processed before it is converted to PDF.

- XML Browser

The XML Browser is a tool that understands the supported tag syntax of the XML produced by the PageMapper and the XML based control files used by the TallPine products. It is a useful tool for reviewing and modifying the XML files produced and used by the Base Components. The XML browser can open a very large XML MAP or LIST file quickly and is the recommended tool to use when doing the AFP application analysis and customization work.

Base Module AFP Support

AFP is a document architecture that supports a rich set of graphic, image and text objects.
The TallPine conversion tools support all common IBM image data formats and many uncommon ones.
These include:

1. IOCA (Image Object Content Architecture) FS45 RGB, CMYK banded and JPEG. Support has been recently added for Image transparency masks. Multiple image tiles in IOCA are not supported. These formats are becoming more common as the use of color increases.
2. IOCA 1-bit images in all but ABIC compressed format are supported. These are the most commonly seen image formats and are still considered strategic.
3. IMB IM1 celled and un-celled image is supported. These are AFP legacy formats and are still very prevalent in a typical AFP application.

The AFP vector graphic object format is GOCA (Graphics Object Content Architecture). The TallPine tool supports a significant subset of the drawing orders and commands found in the GOCA architecture. It is difficult to quantify the limits of the support. We have not encountered an AFP file with a GOCA object that we do not support. It would be very easy to build a pathological GOCA object that would destroy most AFP programs out there!

AFP text is traditionally encoded in single byte format using a separate FOCA (Font Object Content Architecture) code page object and FOCA character set object. Legacy AFP applications use 240 PEL/inch or 300 PEL/inch raster (bitmap) fonts. FOCA uses code pages that define a mapping of a single byte code point to an 8 byte IBM glyph name. That glyph name maps to a character bitmap in the character set object. Many AFP legacy applications uses raster fonts that were derived from Adobe type one fonts. The AFP core fonts are Time New Roman, Helvetica and Courier. AFP code pages are typically based on EBCDIC as opposed to ASCII, but they can use any encoding. FOCA also includes Type 1 vector font objects. The most current AFP architecture exploits TrueType and OpenType fonts. AFP now supports Unicode UTF 16 encoded data. To date this is seldom encountered.

The base support (not requiring any customization) will handle image and graphics as well as character data that uses AFP core raster fonts and code pages. There are no customization options for image and graphics other than specifying image compression algorithms.

For the purposes of text conversion, the Base Module includes an extensive list of IBM AFP glyph name to Unicode mappings. This mapping is necessary to create the XML from AFP. The base module also supports converting AFP raster fonts to type 3 bitmap PDF fonts. The most complex part of configuring an AFP conversion is handling the issues related to the use of non-standard fonts. There is some support for IBM double byte (Chinese, Japanese, etc). The UTF Unicode text format is supported as well but has not been extensively tested in the real world.

Standard AFP to PDF Conversion Process

When tackling the configuration of a new AFP conversion application it is important to understand the objectives. If the objective includes the ability to access the text content of the document in the XML or PDF it is essential to be able to correctly map all the character data to Unicode. If the only objective is to create PDF then it is possible to create PDF that uses custom encodings and exploit inline PDF fonts that are Type 1, Type 3 or TrueType. When ever possible you want to correctly map the character data to Unicode.

If the only objective is to access the document text content, the processing of graphics and image can be skipped. This greatly improves performance.

Since the main focus of this document is the conversion of AFP to PDF that objective will be assumed from here forward.

Before you begin conversion

- Determine the origin of the AFP. Was it produced from line data using a PageDef? Was it produced by Exstream? The customer might not know but it doesn't hurt to ask. Determine if the customer has the ability to customize the AFP application. The answer to this is almost always NO-NO-NO, even if they can.
- Request that the customer include all required resources. Preferably inline in a standard AFP resource group. If not inline then as separate resource objects that you can place in a resource library (folder).

Run the AFP file through the PageMapper Utility

The PageMapper can run as both a batch command line program or interactively with a GUI interface. During this phase it is recommended that you use the GUI. You can easily modify and save the configuration this way and then use it with the command line interface.

- Place all separate resource objects in the resource libraries pointed to by the AFPResourceLib tag in the AFPResourceLib.xml file.
- Select the debug = yes option. Run the AFP file with both the list and the map formats. The will create XML files that will allow you to determine the type of objects that are used in the AFP.
- Check the PageMapper log file for errors. The error log will indicate if resources are missing and if there are font code page issues.
- Resolve any missing resources by getting them from the customer, creating them or by other means.
- Resolve font code page and character set issues. Most of the code page errors will need to be resolved before you can proceed to PDF conversion. If you are missing character sets the font character widths will be indeterminate which can lead to incorrect character placement.
- Make custom font configuration changes to map non-standard code page glyph names. If this is problematic you may be able to simply turn on Type 3 fonts for that problem

font which will then work for PDF but the text will be garbled (unreadable) in the XML and PDF source.

Run the PageMapper MAP XML file through the PDF Converter Utility

- Select Type 3 fonts = ALL and Type 1 fonts = all in the properties panel. Run the MAP XML file and create a PDF. This will give you a quick and dirty PDF file that in many cases has sufficient fidelity with regard to printing. The result is a large PDF file that does not view well. This is because all the fonts are bitmap format.
- This above step will produce a baseline PDF conversion that will normally have fairly good page fidelity but will be inefficient.
- Decide if this is good enough. If print is the only concern this may be all you need.

Evaluate the PDF output and the LOG files to determine what customization is required.

1. Edit the AFPResourceLib.xml file to add custom font resources and to select specific fonts for Type 3 and Type 1 inclusion or exclusion in the PDF.
2. Add entries to the CustomGlyphNameMap.xml file to define custom glyph names used. Or define custom glyph encoding files for specific fonts

Rerun the PageMapper and PDF Converter

- Make configuration changes and rerun both applications until the results are acceptable.

Run the AFPToPDF Converter

- Specify the PageMapper and PDF Converter config file created in the previous steps and run the AFP to PDF conversion. The results should be the same as running the AFP to XML and XML to PDF conversions separately.

Deploy the Batch Conversion