Publishing and Distribution

AFP User Guide

February 1, 2008

CONTENTS

SECTION 1. INTRODUCTION

Advanced Function Presentation (AFP) is an architecture whose purpose is to consistently deliver information to a wide variety of output devices through the utilization of resources located in central libraries. Print Services Facility (PSF) is the core of our AFP implementation. PSF receives the output data set, retrieves the appropriate resources as defined in JCL, then produces a print data stream for a specific printer. The resources used in AFP consist of Page Segments (Psegs), Fonts, Page Definitions (Page Def), Form Definitions (Form Def), and Overlays. These resources may be created using a variety of tools.

Information may consist of 'composed text' developed using a batch composition system such as IBM's Document Composition Facility (DCF) used for producing large book like documents; or, information may consist of data being generated by an application executing on a PC, in a client/server environment, or on a mainframe. Currently at the State of Wisconsin data generation for print to mail is how AFP is utilized.

The use of AFP enables the separation of content from presentation. What this means: one, there is no need for print specific code to be imbedded into the application; second, data is placed into an output data set without any formatting. The benefit of the separation of content from presentation is quicker application development, simplified application maintenance, independent development of presentation resources, easier adoption of other presentation technologies in the future.

SECTION 2. AFP RESOURCES

There are several AFP resources used when generating output, although it is not necessary that all of the resource types be used for every job.

Page Segments

Page segments, or Psegs, are bit maps. More specifically, they are scanned in objects, such as logos, signatures, even pictures and maps. These objects resources can only be called by another AFP resources type such as an Overlay or Page Definition Line command.

Fonts

AFP fonts are made up of three parts all stored in a central library. The three parts that make up a font are the Coded Font, The Character Set and the Code Page.

The Coded Font is the part that is actually called in an AFP resource. The Coded Font is a pointer to the Character Set and a specific Code Page. There are often multiple Coded Fonts which point to the same character set; however, they point to different Code Pages.

The Character Set is the actual bit map of all the characters available in a specific Type Face.

For each Type Face, Point or Pitch size, Character Weight, and tilt there is a unique Character Set.

Some Terms Explained

Type face—A type face or type family is the basic look of the characters. Some examples are Times New Roman, Helvetica, and Gothic Text on the mainframe; and Arial on the PC.

Character weight—Whether or not the characters are bold, light or medium is stroke width.

Point or pitch size—There are two factors here: a fixed pitch, or typewriter font, is where every character takes the same amount of page space regardless of the actual size. A lower case 'i' requires the same amount of space to print as does the upper case 'W'.

These fonts are generally measured by the number of characters that can be printed in a linear inch in the direction of the print line. A 10 pitch font would be able to print 10 characters in an inch. Proportionally spaced fonts, on the other hand, are measured in points and character cell height. There are 72 points in an inch; therefore, the tallest character in a 12 point font is close to 1/6" tall. The character width, on the other hand, is completely variable and based on the actual width of the character to be printed.

Character lean—This is just another way of saying whether or not the characters are italicized.

Code page—The code page maps the entered code point to a specific character in the character set up to 255 characters for a single by font. For example, a hexadecimal 'C8' in most code pages maps an upper case H. There are code pages which map almost all 255 code points, while there are others which map only a select few code points. This may all sound very complex and unnecessary until you refer back to the coded font; we can completely change the characters mapped by the code page used where other technologies require a separate font for every code page represented.

Overlays—An overlay consists of the 'static' information on a page. A couple of examples would be lines, boxes, logos and box headers found on a form or the 'boilerplate' instructions or contract 'terms and conditions' found on the back. It is called static because, for the most part, it does not change from customer to customer within an application. Overlays are normally called by the Formdef; however, overlays can also be called in via a PrintLine parameter within the Pagedef.

Formdef—The form definition (Formdef) is a called resource; that is, it is called out through the OUTPUT statement in the JCL. The Formdef defines how the printer will handle the physical pieces of paper. In the Formdef parameters such as Duplexing, Print Quality, Constant Forms, Copies and others are defined within the Formdef.

To control the environment even further, copy groups override or modify the initial environment defined by the Formdef. Copy Groups are mini-formdefs within the Formdef. A Formdef can contain several copy groups. The environment a copy group sets up remains until it is modified by another copy group; this can happen only through a Conditional Processing statement within a Page definition.

Pagedef—The Page Definition (Pagedef) is also a called resource that is called through the OUTPUT statement in the JCL. The Pagedef is a series of substructures and commands or parameters that process a 'subpage' of data. A subpage of data is defined as the data records starting at a Carriage Control 1 page break and ending at the record before the next Carriage Control 1 page break. It can also be defined as all the data in a single print record; or, it can be defined as a fixed constant number of print records. A Pagedef defines the size of a 'logical page', the general direction of print, fonts, Psegs, and overlays.

The substructure of a Pagedef is the PageFormat. The Pageformat contains parameters which define a subpage, uses conditional processing in order to modify the processing environment, call in and place overlays and Psegs, define what fonts will be used in printing specific data, and define where and how often a specific field of data will be printed.

SECTION 3. CREATING AFP RESOURCES

There are a number of tools available to design AFP resources. The tools used in the State of Wisconsin include:

Page Print formatting Aid (PPFA)—A batch compiler that runs on the MVS mainframe. The compiler reads the contents of a mainframe file (source code) and produces Formdef and PageDef code used by PSF and VPS to drive either the production laser printers or the remote PCL laser printers.

Overlay Generation Language (OGL)—A batch compiler that runs on the MVS mainframe. The compiler reads the contents of a mainframe file (source code) and produces the static information printed on a page by PSF.

Elixir Technologies Corp.—Elixir has developed several products that run in the Windows/NT environment. Elixir AppBuilder is a product used to develop Overlays, Pagedefs and Formdefs in a reasonably WYSIWYG environment. The files actually produced by the AppBuilder are proprietary Elixir files.

The Elixir Desktop Suite contains a number of converters whereby the Elixir proprietary files may be converted to a number of source and object formats including:

- Source OGL
- Object OGL
- Source Page / Form defs
- Object Page / Form defs
- Various Image formats to Elixir
- Elixir Image format to various other formats

SECTION 4. NAMING CONVENTION

AFP page definitions, form definitions, overlays, and page segments need to follow the naming conventions specified by the AFP standard and the agency prefix table. The resource names are restricted to 6 characters by the MVS file system with two additional characters generated by the OGL and PPFA compilers. When Elixir Desktop converts an Elixir Image to an AFP Page Segment, it will append an S1 to the file much like the OGL and PPAF compilers. AFP requires Page Definitions to begin with P1, Form Definitions with F1, overlays with O1, and Page Segments with S1. The P1, F1, and O1 prefixes are added by the PPFA and OGL compilers.

It is recommended that AFP resource names be consistent with the four-character form name used with the EOS electronic report distribution system. The following link shows how resources should be named:

http://operations.state.wi.us/asx/CustomerManual/docs/IT-306-8.pdf

The first, and in some cases the second, character are used to identify the agency who created the resource or for whom the resource was created. The remaining characters are at the agencies' discretion.

In the case of page segments (images) the last character should specify the orientation, using N for north (0 degree rotation), E for East (90 degrees rotation), S for South (180 degrees rotation), and W for West (270 degrees rotation). All other characters are at the discretion of the agency, but it is recommended that the naming be consistent for a print stream and 6 characters in length for the form definition and page definition to make it easy to give the overlays and page segments consistent names.

For example, the following would be a consistent resource-naming scheme:

```
F1YM01
Form Definition -Gaming
Commission Inventory
Questionnaire
P1YM01
Page Definition -Gaming
Commission Inventory
Questionnaire
O1YM01F
Overlay Front -Gaming
Commission Inventory
Questionnaire
O1YM01B
Overlay Back -Gaming
Commission Inventory
```

Questionnaire
S1YM01SN
Page Segment -Slot
Machine -Normal
Orientation
S1YM01RN
Page Segment -Roulette
Wheel -Normal Orientation
S1YM01DS
Page Segment -Dice Inverted

SECTION 5. PRINTERS

Any laser printer accepting the PCL/5 print stream identified to VPS can be used for printing the AFP print stream. The printers usually have an identifier beginning with U followed by 4 digits.

The high speed printers at 202 South Thornton Ave. (EDS) have a generic designation of U6 with the output being routed to an appropriate printer by the operator.

SECTION 6. JCL

AFP printing requires an OUTPUT statement to specify AFP resources. The following JCL statements will show how to write the OUTPUT statement so that the printed output will either be returned to the user or be sent on to the INSERTING & MAILING area for further processing. In this sequence DO NOT confuse the term *form* for the data layout or electronic form. The term *form* here refers to the physical paper ONLY.

The following JCL will enable you to have your output transferred from the Print section to the Inserting & Mailing section. One thing to notice is that the parameters NAME, ROOM, BUILDING, DEPT, ADDRESS, TITLE and DEST must all be in UPPER CASE because they are JCL parameters. The information following them, within the single quotes and up to 60 characters, may be in mixed case provided CAPS ON is turned off in your TSO profile. In the case of the ADDRESS parameter, you may enter up to four lines of 60 characters each, enclosed by single quotes and ended by a comma.

In the first JCL statement, replace the four question marks (????) after the Pagedef= and the Formdef= with the name, up to 6 characters, of the AFP resource you plan to use for printing your application output. You may either use one of the many 'generic' resources, or use a custom resource built for you application.

In the following situation there are two MAIL DD statements. They are mutually exclusive and should not be coded together. They are shown here for illustration purposes only.

If you want to use a paper other than 20# white no-hole paper, you must replace the ???? in the SYSOUT statement with the appropriate form number, for example, STD3 for 3-hole prepunched, or ZY58 for a 60# Tabloid-size paper. If you are using standard 20# no-hole paper, use the second //MAIL DD statement.

```
//MAIL OUTPUT NOTIFY=(userid),PAGEDEF=????,FORMDEF=????,
// NAME='Customer Name & Phone Number Up to 60 characters',
// ROOM='Print to Mail -Post Processing ',
// BUILDING='Thornton Ave.',
// DEPT='Dept. of Administration',
// ADDRESS=('Inserting & Labeling',
// '202 S. Thornton Ave.',
// 'Madison WI 53702',
// TITLE='FORM=???? Other Customer Information as Desired Up to 60
characters',
// DEST=U6
```

The DD statement refers back to the OUTPUT statement as follows:

```
//MAIL DD SYSOUT=(O,,????),OUTPUT=*.MAIL
```

or

```
//MAIL DD SYSOUT=P,OUTPUT=*.MAIL
```

The following JCL will have your output returned to you. The same rules apply here as in the previous illustration.

```
//OUT OUTPUT NOTIFY=(userid),PAGEDEF=????,FORMDEF=????,
// NAME='Output recipients name and phone number up to 60 characters',
// ROOM='What room or floor the recipient is located',
// BUILDING='What building i.e., GEF III or Wisconsin Administration
Building',
// DEPT='Department Name',
// ADDRESS=('Division name if available',
// 'Building Street Address',
// 'City State and Zip',
// 'An Additional Address Line is Available'),
// TITLE='Customer free form upto 60 characters long',
// DEST=U6
```

The DD statement refers back to OUTPUT statement as follows

```
//RETURN DD SYSOUT=P,OUTPUT=*.OUT
//FORM DD SYSOUT=(O,,FORM),OUTPUT=*.OUT
```

OUTPUT CLASS

Class 'P' is a 'No Operator Intervention Required' class. When the operators see a job show in Class P, it will be routed to a cutsheet printer that is using 20# White No-Hole paper. You will code this in your JCL as SYSOUT=P.

Class 'O' is an 'Operator Intervention Required' class. When an operator sees a job in Class 'O', they also look for a form (paper) name to mount on the printer. You will code this in you JCL as SYSOUT=(O,,STD3), STD3 being 20# white with 3 holes.

The OUTPUT statement may go before the job steps or be included within a job step.

MARK IV

For those using MARK IV, the following is some tested code.

```
//DOTXXX JOB ('TRXXXXXXXX,POSITION',XXXB),'USER,X',
// TIME=(,09),MSGLEVEL=(1,1),
// NOTIFY=DOTXXX,MSGCLASS=Y,
// CLASS=H
//JOBLIB DD DSN=SYS1.DMT.EXLIB,DISP=SHR
//*********************************************************************
//*********************************************************************
//*
//J0101 EXEC PROC=M4PROD
//OUT1 OUTPUT PAGEDEF=##2UPB,FORMDEF=##2UPB,CLASS=P,DEST=UXXXX,
// NAME='Output recipients name and phone number up to 60 characters',
// ROOM='What room or floor the recipient is located',
// BUILDING='What building i.e., GEF III or Wisconsin Administration
Building',
// DEPT='Department Name',
// ADDRESS=('Division name if available',
// 'Building Street Address',
// 'City State and Zip',
// 'An Additional Address Line is Available'),

// TITLE='Customer free form upto 60 characters long',
//MARK4.M4LIST DD SYSOUT=(,),OUTPUT=(*.OUT1) /* STANDARD NO HOLE PAPER
*/

or /* Remember one of these must be commented out */
//MARK4.M4LIST DD SYSOUT=(O,SDT3),OUTPUT=(*.OUT1) /* 3 HOLE PUNCH
PAPER */
//MARK4.M4OLD DD DSN=DOTXXXX.YR1999.HELP(0),DISP=(SHR,KEEP),
// UNIT=(,,DEFER)
//MARK4.M4INPUT DD *
```

Note: Remember that in Mark IV as in other TSO applications that the physical PAPER type is called in the SYSOUT statement using a Class of 'O' with a recognized form type.

SECTION 7. APPLICATION DEVELOPMENT

The design behind AFP is, again, to separate content from presentation. There are a number of ways that an output data set could be written in order to provide the data the customer wants the way the customer wants to see it. It is imperative that the Application Developer, the Customer and the AFP developer meet early in the process to determine the best way to develop the output data set. By using the capabilities of AFP, the customer's needs will be met and application development and maintenance will be simpler.

Three Ways to Develop Output

There are basically three methods to develop an output data set, and there are variations of those as well.

Application Formatted—When the Application Programmer does all of the formatting of the output within the application

Here the application programmer has to move a sequence of data and empty spaces from his data set to the output file in order to match a form. This was done in the days of line printing and paper tape-controlled carriage movement. Items such as Carriage Controls which advance to the top of a new page and advance the print line in order to skip over boiler plate are still employed and/or maintaining a consistent number of lines per page.

This method is very time-consuming, adds complexity to the application, and makes it increasingly more difficult to maintain. Minor changes requesting modification to the form and/or data placement become major projects. A very minor example follows.

In example A, the application programmer is formatting the output for an addressing block for a windowed envelope. The lower case sp indicates a Hex '40' i.e., a space the programmer must move in order to get the data to line up.

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp Sp sp sp sp sp sp sp
sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp Sp sp sp sp sp sp sp
sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp Sp sp sp sp sp sp sp
sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp Sp sp sp sp sp sp sp
sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp Sp sp sp sp sp sp sp
sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp Sp sp sp sp sp sp sp
sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp Sp sp sp sp sp sp sp
sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp Sp sp sp sp sp sp sp
sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp Sp sp sp sp sp sp sp
sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp Sp sp sp sp sp sp sp
sp sp sp sp sp sp sp sp J a m e s sp J . sp R e I s e n b U e c h l e r
sp sp sp sp sp sp sp sp 6 1 2 0 sp S . sp 6 0 t h sp S t . Sp sp sp sp sp sp sp
sp sp sp sp sp sp sp sp G r e e n d a l e sp W I sp 5 3 1 2 9 sp sp sp sp sp
sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp Sp sp sp sp sp sp sp
sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp sp Sp sp sp sp sp sp sp
```

The application programmer would have to do this for every line and every field that was planned to be printed. The AFP data layout (Pagedef) would be really simple for this.

One Field per Record—Another method to move data to an output data set is to move one printable field per data record

With this method, as in the above method, the application programmer would have to code a Carriage Control byte which would signal the printer and printer software when to skip to the top of a new page. This method is simpler for the application programmer to create and maintain as well as the AFP developer.

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
J a m e s sp R e i s e n b u e C h l e r
6 1 2 0 sp S . sp 6 0 t h sp S t . sp sp sp sp
G e e n d a l e sp W I sp 5 3 1 2 9
```

One Record per Page with Fixed Fields

The last method to cover is where all the data for an entire page is contained in a single output record, in fixed fields. What fixed fields means is that for a particular piece of data a certain amount of space will be allocated. Whether or not the data fills that space, that amount of data padded with spaces will be written to the output data set. With this method carriage control bytes may or may not be necessary, depending upon what the customer is looking for. I will use the same examples above; however, I will have to compress it due to space concerns.

```
Field 1 Start Byte 1
Length 40
Field 2 Start Byte 41 Length 35 Field 3 Start Byte 71 Length
40
James Reisenbuechler
(add 20 spaces)
6120 S. 60th St. (add 19 spaces) Greendale WI 53129 (add 23
spaces)
Jane Doe (add 32 spaces) N67 W14534 Appleton Ave. Suite #3
(add 2 spaces)
Menomonee Falls WI 53244 (add
16 spaces)
```

Application Development Conclusion

It cannot be stressed enough for two things to happen in the Application Development process.

First, the application developer, the AFP developer and the customer should maintain constant communication throughout the development cycle. This will ensure the most efficient development that will also be the easiest to maintain.

Second, with the IT environment in a constant state of flux, it is also important to eliminate all presentation specific code from the application and output data set. In other words, the content should never be tightly coupled with the presentation. This will ensure information portability.