MakeAFP

MakeAFP Weaver for Windows User’s Guide

Version 3.7

Contents

Chapter 1. Getting Started

MakeAFP Weaver provides enhancements with the highest performance and many unique functions to your existing non-indexed or indexed AFP files, it lets you:

- Add AFP page group indexes to non-indexed AFP file by the encoding of ASCII/EBCDIC/ DBCS-PC (BIG5, GBK, KSC, SJSIS)/DBCS-HOST/UTF-8/UTF-16, so the AFP file can be archived and retrieved by an AFP content management system, or used to speed up AFP sorting/ reprinting.
- Add over 50 types of popular linear barcodes and 2D barcodes by both BCOCA and AFP drawing.
- Add ASCII/EBCDIC/DBCS-PC(BIG5/GB18030/KSC/SJIS)/UTF-8/UTF-16 texts by using either OpenType/TrueType fonts or legacy old FOCA outline/raster fonts.
- Add overlays, page segments, data-object images (TIFF/JPEG/GIF) to replace pre-printed forms with electronic forms and images.
- Add OMR, lines, boxes.
- Hide/mask areas.
- Capture ASCII/EBCDIC/DBCS-PC/DBCS-HOST text fields by their locations or index values from your existing AFP file.
- Define a location and a string to uniquely identify the beginning of a page group or a page, or identify a data field string to be captured from an AFP page.
- Remove current copy-groups from the input AFP.
- Insert copy-groups (also called medium-maps) to control the physical attributes of the IPDS print output, like stapling, N-UP printing, choose input and output trays, etc.

Applications that use MakeAFP Weaver functions must obey the hierarchy and structural rules of the MakeAFP Weaver language when doing enhancements to your existing AFP documents which are very easy to understand. Writing applications according to these restrictions is quite straightforward.

This chapter describes the step-by-step processes on how to use MakeAFP Weaver with C to enhance your AFP document.

MakeAFP Weaver samples in C/C++, C#, VB.NET, and Java are shipped along with the MakeAFP Weaver installation package, to provide you with realistic practice scenarios. MakeAFP understands that the sample code is one of the most important aspects of a toolkit for quick understanding and learning, and as such, we will try our best to include more samples within our installation package.

Starting a MakeAFP Weaver Session

“Start” function starts and establishes the initiation of a MakeAFP Weaver session, allocates the memory required, opens an AFP input file, AFP output AFP file and MakeAFP definition file, parses the parameters defined in MakeAFP definition file, retrieves all of the AFP resources or OpenType/TrueType fonts from the input AFP file or resources libraries, required by your program by either generating an external resource file or by putting resources inline within your AFP document files, and also retrieves AFP font or OpenType/TrueType fonts information required by MakeAFP Weaver for the text formatting alignments.

C Sample:

```
void main( )
{
  Start();                 // Starts initiation, opens default input AFP,
                           // output AFP and definition files, retrieves
                           // AFP resources and information, as well
                           // as OpenType/TrueType font, allocates
                           // memory required
      :
      :
      :

}
```

Opening and Closing an AFP Document

You must call the “Open Document” function to initialize an AFP output document before you open an AFP page, and you must close this AFP output document by the “Close Document” function before ending your program.

MakeAFP Weaver transfers AFP resources into the AFP output document file if the AFP resource inline parameter is specified in the MakeAFP definition file.

C Sample 1: single AFP output file

```
void main( )
{
  Start();               // Starts initiation, opens default input AFP,
                         // output AFP and definition files, retrieves
                         // AFP resources

  OpenDoc();             // Opens AFP output document

          :
          :

  CloseDoc();            // Closes AFP output document

}
```

Setting Default Units

You can set default units by calling the “Set Unit” function before calling the “Open Document” function.

MakeAFP Weaver default is INCH if the “Set Unit” function is not called.

C Sample:

```
void main( )
{
  Start();

  SetUnit(MM_U600);      // Set default units to MM, 600 pels

  OpenDoc();

  OpenPage(210,297);     // A4 paper size, 210 x 297 mm

       :
       :

  ClosePage();

  CloseDoc();

}
```

Getting/Opening and Closing an AFP Page

You must get/open and close every page within an AFP document. MakeAFP Weaver does not automatically close pages, page closing has to be done by your application control.

With MakeAFP Weaver, you can get/open multiple AFP pages with “Get Page” or “Open Page” functions by either getting AFP pages from an existing AFP input file or opening new AFP pages, and then process different pages in arbitrary order once each page is initialized, all of the MO:DCA data stream will be kept in memory buffers in page-level, and only to be written to the AFP output document file until the page is closed with the “Close Page” function.

C Sample:

```
void main( )
{
  Start();

  SetUnit(IN_U600);

  OpenDoc();

  GetPage();                    // Getting an AFP page from AFP input file

     :                          // Then you can do some enhancements
     :                          // to this page

  ClosePage();                  // Close AFP page, write to AFP output
                                // file
     :
     :

  OpenPage(8.5,11);             // Open an AFP new page
                                // LETTER paper size, 8.5" x 11"

     :                          // Then you can compose this new AFP page
     :

  ClosePage();                  // Close AFP page, write to AFP output
                                // file

  CloseDoc();

}
```

Text Positioning

MakeAFP Weaver provides powerful text positioning functions and variables to help you add and position your new texts. After you called the "Get Page" or "Open Page" function, you can use these positioning functions to position your texts quickly. The following table shows the functions and variables.

MakeAFP Weaver functions	Description
Margin (float value)	Sets the inline left margin
LineSp (float value)	Sets baseline spacing
LPI (float value)	Sets baseline spacing by LPI (lines per inch)
NextLine ()	Skips to the next begin line position defined by Margin() and LineSp() or LPI()
Skip (float value)	Skips lines then start from margin
Xpos (float value)	Sets absolute horizontal position
Ypos (float value)	Sets the absolute vertical position
Pos (float x, float y)	Sets absolute horizontal and vertical position
Xmove (float value)	Sets horizontal position relative to the current horizontal position, you can specify a negative value
Ymove (float value)	Sets vertical position relative to the current vertical position, you can specify a negative value
cm (float value)	Specifies a value in centimeters
mm (float value)	Specifies a value in millimeters
inch (float value)	Specifies a value in inches
GetXpos()	Gets current AFP text X position
GetYpos()	Gets current AFP text Y position

C Sample 1:

```
OpenPage(8.5,11);             // Open an new AFP page to compose,
                              // LETTER paper size, 8.5" x 11"

Pos(1,1);                     // Position at (1",1")
Ltxt("Testing sample 1");     // Left put text at (1", 1")
Ymove(0.5);                   // Move Y baseline down 0.5"
Ltxt("Testing sample 2");     // Left put text at (1", 1.5")
Xpos( mm(15) );               // Set X position at 15 mm
YPos( cm(2.5) );              // Set Y position at 2.5 cm
Ltxt("Testing sample 3");     // Left put text at (15mm, 2.5cm)

ClosePage();                  // Close AFP page and write to AFP file
```

C Sample 2:

```
GetPage();                    // Get a page from AFP input file

Ypos(1);                      // Set Y position to 1"
Margin(1.5)                   // Set left margin to 1.5"
LineSp(0.25);                 // Set baseline increment to 0.25"
Skip(25.4);                   // Skip 25.4 lines
Ltxt("Testing sample 4");     // Left put text at X = 1.5",
                              // y = 1" + (0.25" x 25) = 7.25"

ClosePage();                  // Close AFP page and write to AFP file
```

Putting Color Text on the Page

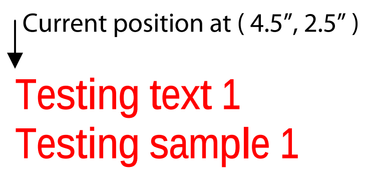
After you opened a page, you must specify the position and font before you can add color text to the page. You can specify IBM OCA standard color, RGB color, or CMYK color for your text.

The text string will be presented exactly as entered except alignment, if you want to do some manipulation on your text string, you can use string functions provided by C and C++ or MakeAFP Weaver before you put the text on the page.

MakeAFP Weaver provides the “Left Text”, “Right Text”, “Center Text” functions for putting your text on a page by the left, right, and center alignments relative to the current position, and it also provides alignment functions for DBCS-PC (GBK, GB18030, BIG5, SJIS, KSC), DBCS-HOST, as well as the new generation Unicode data string UTF-8 and UTF-16 by using OpenType/TrueType fonts. Refer to *MakeAFP Weaver Reference* for more details.

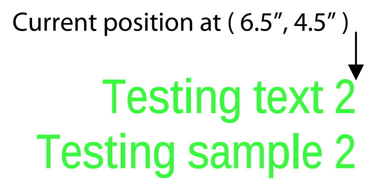
Left Alignment:

```
Pos( in(4.5), in(2.5) );

/* Use font 1 defined by MakeAFP */
/* definition file               */
Font(1);

/* OCA RED color */
Color(RED);

Ltxt("Testing text 1");
Ltxt("Testing sample 1");
```



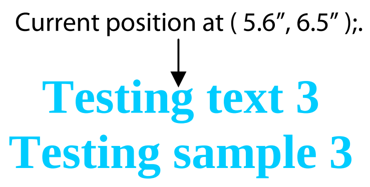
Right Alignment:

```
Pos( in(6.5), in(4.5) );

/* Use font 2 defined by MakeAFP */
/* definition file               */
Font(2);

/* RGB color */
ColorRGB(76, 230, 76);

Rtxt("Testing text 2");
Rtxt("Testing sample 2");
```



Center Alignment:

```
Pos( in(5.6), in(6.5) );

/* Use font 3 defined by MakeAFP */
/* definition file               */
Font(3);

/* CMYK color */
ColorCMYK(230,76,76,0);

Ctxt("Testing text 3");
Ctxt("Testing sample 3");
```



* OCA color data stream is supported by most of the IPDS printer controllers, RGB and CMYK color data streams are only supported by the new IPDS printer controllers.

Formatting Paragraphs

Paragraphs are blocks (boxes) of texts. With MakeAFP Weaver, you can specify the width of your paragraph, colors and underline your text, and also control text alignment. MakeAFP Weaver provides paragraph functions to handle ASCII or EBCDIC text, as well as DBCS-PC (GB18030, GBK, BIG5, SJIS, KSC), DBCS-HOST, UTF-8, and UTF-16.

Sample 1: fixed paragraph

```
const helv14 = 1;     // For your own convenience, you can define a constant
                      // variable as your local font alias name

  char *text = "AFP is an integrated hardware and software architecture";

  SetUnit(MM_U600);
  OpenDoc();
  OpenPage(210,297);

  LPI(5);                      // Set line spacing to 5 LPI
  Pos(20,10);
  Font(helv14);                // use local alias font-name helv14
  Color(CYAN);
  ParTxt(text, 45, LEFT);      //  Left align paragraph, width = 45 mm

  Pos(20,30);
  Color(BLUE);
  ParTxt(text, 45, RIGHT);     //  Right align paragraph

  Pos(80,10);
  Color(BLACK);
  BgnUscore();                 //  Begin underscore
  ParTxt(text, 45, CENTER);    //  Center align paragraph
  EndUscore();                 //  End underscore

  Pos(80,30);
  Color(RED);
  ParTxt(text, 45, JUSTIFY);   //  Justify align paragraph
  ClosePage();
  CloseDoc();
```

Output:

AFP is an integrated
hardware and software
architecture

<u>AFP is an integrated</u>
<u>hardware and software</u>
<u>architecture</u>

AFP is an integrated
hardware and software
architecture

AFP is an integrated
hardware and software
architecture

Sample 2: variable paragraph

```
char *name = "David B. Lee";

char *msg1 = "CONGRATULATIONS, ";
char *msg2 = ", because of your excellent credit rating, ";
char *msg3 = "you are now eligible for free credit insurance.";

SetUnit(MM_U600);
OpenDoc();
OpenPage(210,297);

Pos(20, 50);
LPI(4);                          // Set line spacing to 4 LPI

BgnParTxt(110, LEFT);            // Begin a variable paragraph, 110 mm
                                 // width, left aligned
Font(4);                         // Use font 4 defined in MakeAFP definition
PutParTxt(msg1, PINK);           // Put 1st text in pink color

Font(3);                         // Use font 3 defined in MakeAFP definition
PutParTxt(name, BLUE);           // Put client name in blue color

PutParTxt(msg2, BLACK);          // Put 3rd text in black color

PutParTxt(msg3, GREEN, ON);      // Put 4th text in green color, and
                                 // turn on underscore

EndParTxt();                     // End variable paragraph

ClosePage();
CloseDoc();

#ifdef _DEBUG                    // Only view AFP in debug mode

  ViewAFP();                     // View AFP file generated

#endif
```

Output:

CONGRATULATIONS, David B. Lee, because of your excellent credit rating, you are now eligible for free credit insurance.

Drawing Lines and Boxes

With MakeAFP Weaver, you can add lines and boxes.

C Sample:

```
SetUnit(IN_U600);

           :
           :

OpenPage(8.5,11);

Hline(0.5, 0.5, 7.5, 0.02);            // Draw a horizontal line from
                                       // (0.5",0.5"), 7.5" length,
                                       // 0.02" width, red color

Vline(0.5, 1.5, 5.5, 0.01);            // Draw a vertical line from
                                       // (0.5",1.5"), 5.5" length,
                                       // 0.01" width, BLUE color

Box(0.5, 4, 7.5, 0.4, 0.02);           // Draw box from (0.5",4"),
                                       // box width = 7.5", height = 0.4"
                                       // line thick = 0.02"

ClosePage();
```

Including AFP Object and Data-Object

AFP resources are the objects that you have previously created with MakeAFP or other tools and are stored in the AFP resource library directories. With MakeAFP Weaver, you can include an AFP overlay or AFP page segments at a fixed or dynamic position.

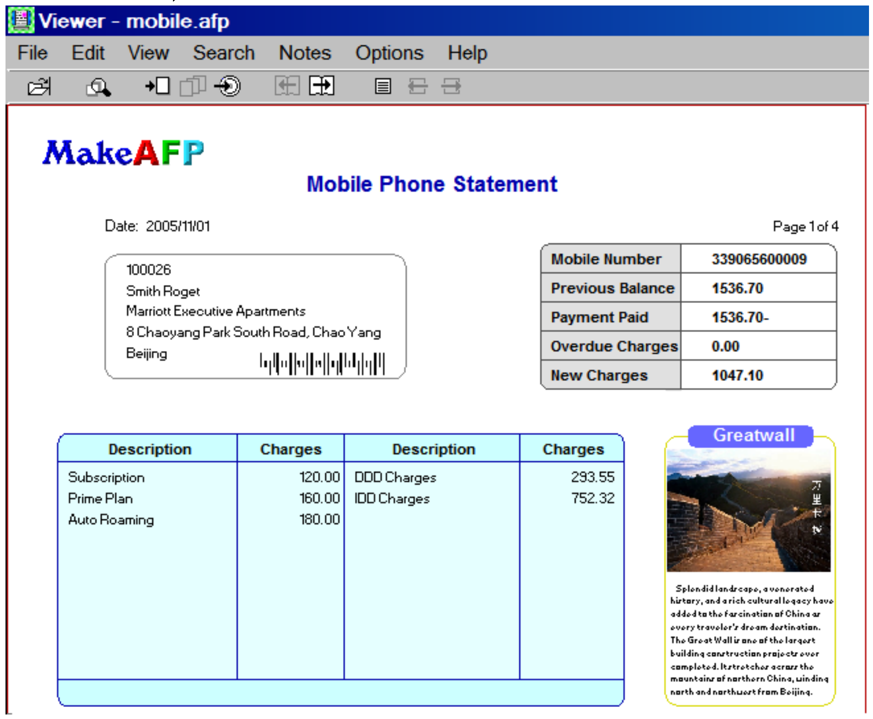
With the latest AFP Systems, you can Include a reference directly to an AFP object(image, graphic, barcode), or a non-AFP data-object(TIFF, JPEG, GIF, etc) at the specified position or current position, and specify the area size, rotation, mapping option for the object to be presented in high performance with a CMR (Color Management Resource) and specific color rendering intent.

Using an object as a resource is more efficient when that object appears more than once in a print job; resources are downloaded to the printer just once and referenced as needed.

C Sample 1: Including an AFP page segment

```
SetUnit(IN_U600);

OpenPage(8.5,11);
          :

InclPseg("S1MKAFP",0.2,0.2);          // Include AFP page segment at
                                      // (0.2", 0.2")

          :

ClosePage();
```

Output:



Mobile Phone Statement

Date: 2005/11/01

Page 1 of 4

100026
Smith Roget
Marriott Executive Apartments
8 Chaoyang Park South Road, Chao Yang
Beijing

Mobile Number	339065600009
Previous Balance	1536.70
Payment Paid	1536.70-
Overdue Charges	0.00
New Charges	1047.10

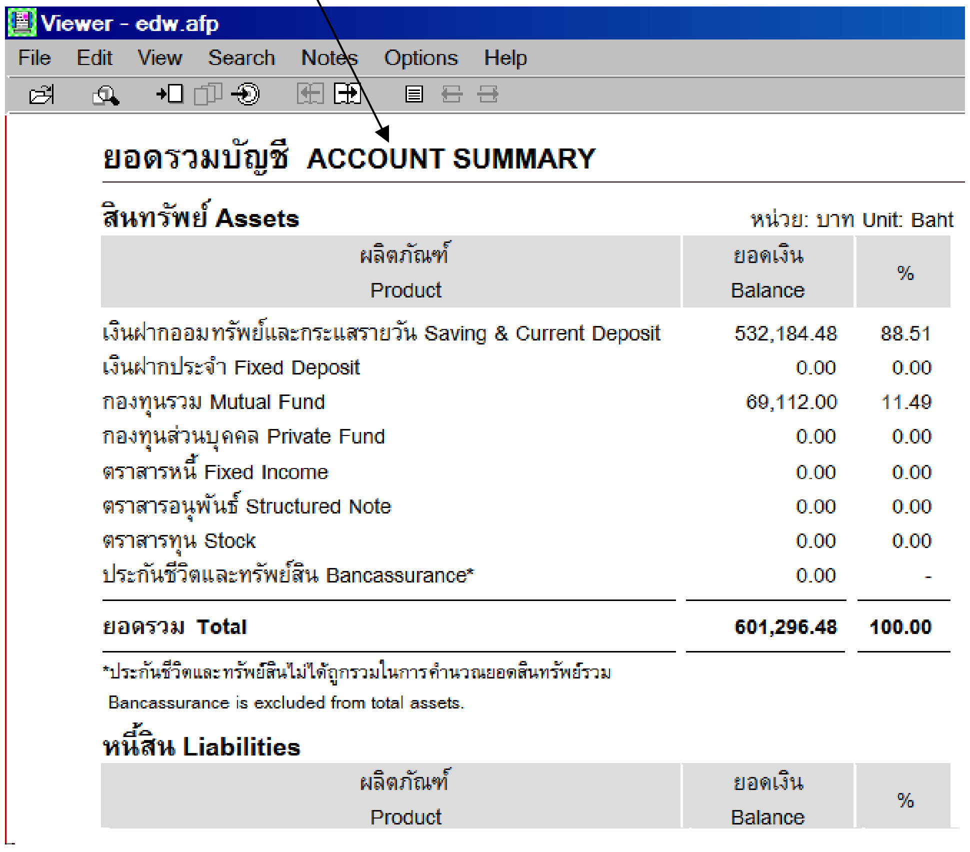
Description	Charges	Description	Charges
Subscription	120.00	DDD Charges	293.55
Prime Plan	160.00	IDD Charges	752.32
Auto Roaming	180.00		

Splendid landscape, a venerated history, and a rich cultural legacy have added to the fascination of China as every traveler's dream destination. The Great Wall is one of the largest building construction projects ever completed. It stretches across the mountains of northern China, winding north and northwest from Beijing.

C Sample 2: Including an AFP overlay

```
SetUnit(IN_U600);

OpenPage(8.5,11);

          :
          :

InclOvly("01SUMARY",0.2,0.3);          // Include AFP overlay at

          :                            // (0.2", 0.3")
          :

ClosePage();
```

Output:



ยอดรวมบัญชี ACCOUNT SUMMARY

สินทรัพย์ Assets

หน่วย: บาท Unit: Baht

ผลิตภัณฑ์ Product	ยอดเงิน Balance	%
เงินฝากออมทรัพย์และกระแสรายวัน Saving & Current Deposit	532,184.48	88.51
เงินฝากประจำ Fixed Deposit	0.00	0.00
กองทุนรวม Mutual Fund	69,112.00	11.49
กองทุนส่วนบุคคล Private Fund	0.00	0.00
ตราสารหนี้ Fixed Income	0.00	0.00
ตราสารอนุพันธ์ Structured Note	0.00	0.00
ตราสารทุน Stock	0.00	0.00
ประกันชีวิตและทรัพย์สิน Bancassurance*	0.00	-
ยอดรวม Total	**601,296.48**	**100.00**

*ประกันชีวิตและทรัพย์สินไม่ได้ถูกรวมในการคำนวณยอดสินทรัพย์รวม

Bancassurance is excluded from total assets.

หนี้สิน Liabilities

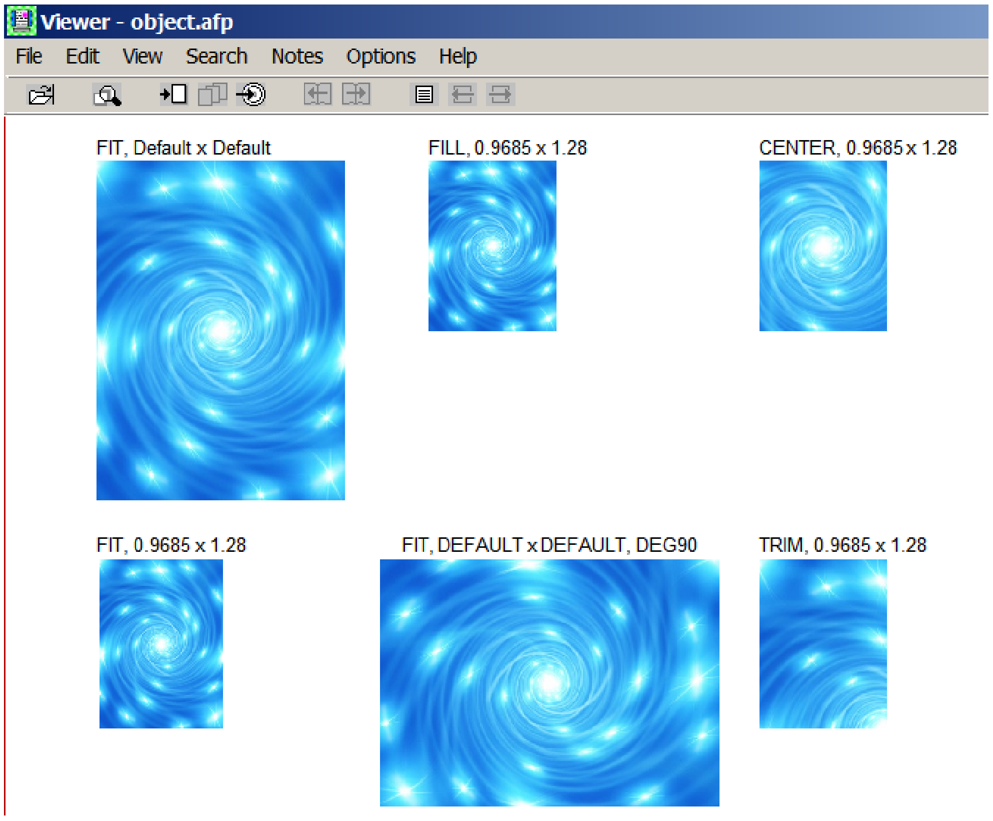
ผลิตภัณฑ์ Product	ยอดเงิน Balance	%

C Sample 3: Including non-AFP images

```
SetUnit(IN_U600);              // Set default units to inch, 600 dpi

OpenPage(8.27, 11.67);         // A4 Paper size

Font(1);

Pos(0.7,0.95);
Ltxt( "FIT, Default x Default");
InclObjt("SAMPLE",0.7,1, DEFAULT, DEFAULT);    // Included an JPEG image,
                                               // image type is defined
Pos(3.2,0.95);                                 // in MakeAFP definition
Ltxt( "FILL, 0.9685 x 1.28");                  // file
```

```
InclObjt("SAMPLE",3.2,1, 0.9685, 1.28, FILL);

Pos(5.7,0.95);
Ltxt( "CENTER, 0.9685 x 1.28");
InclObjt("SAMPLE",5.7,1, 0.9685, 1.28, CENTER);

Pos(0.7,3.95);
Ltxt( "FIT, 0.9685 x 1.28");
InclObjt("SAMPLE",0.7,4, 0.9685, 1.28, FIT);

              :
              :

ClosePage();
```

Output:



Adding Bar codes

MakeAFP Weaver supports all of the linear and 2D barcodes defined in IBM's latest BCOCA standard.

MakeAFP Weaver supports barcodes not only by IBM BCOCA object but also over 50 types of popular linear and 2D barcodes by MakeAFP barcode drawing with a small AFP data stream size.

Linear and 2D barcodes generated by MakeAFP Weaver drawing can be displayed and printed on any type of printer or presentation device with full fidelity and high print/display quality.

C Sample 1: Code 128 bar code by AFP BCOCA

```
char *data = "1234567890";

SetUnit(IN_U600);

OpenDoc();

OpenPage(8.5,11);

BBarCode(CODE128,         // BCOCA Bar code type is Code 128
         data,            // Bar code data variable
         1,               // Bar code x position at 1"
         1,               // Bar code Y position at 1"
         20,              // Bar code module width in mils (thousandths
                          // of an inch)
         0.5);            // Bar code element height 0.5"
                          // Other parameters use defaults

ClosePage();

CloseDoc();
```

Print output:

1234567890

* BCOCA object is supported by the new IPDS printer controllers, please check with your printer vendor whether its printer controller microcode level supports the bar code type you defined.

* Same BCOCA objects may be printed in different dimensions on different vendor's IPDS printers.

* Not every AFP viewer can view BCOCA objects, please check with the vendor of your AFP viewer.

C Sample 2: Royal 4-state postal barcode by MakeAFP barcode drawing

```
char *data = "123456";
SetUnit(IN_U600);
OpenDoc();
OpenPage(8.5,11);
Barcode(RM4SCC,             // Royal 4-state postal barcode
        data,               // Bar code data variable
        1,                  // Bar code x position at 1"
        1,                  // Bar code Y position at 1"
        1.3,                // Bar code dimension width 1.3"
        0.35);               // Bar code dimension height 0.35"
ClosePage();
CloseDoc();
```

Print / display output:

C Sample 3: DataMatrix 2D barcode by MakeAFP barcode drawing

```
char *data = "1234567890 this is testing of DataMatrix";
SetUnit(IN_U600);
OpenDoc();
  OpenPage(8.5,11);
    DataMatrix(data,1.2,1.5,1,1);       // position at (1.2",1.5"), size is
                                        // 1" x 1"
  ClosePage();
CloseDoc();
```

Print / display:

Showing AFP Indexing Information

You may want to use some of the index tag values to generate barcodes, like account numbers. SHOWIDX utility is specially developed to dump the AFP indexes information from your existing AFP file so that you can quickly find out the index names and their values in an input AFP file.

SHOWIDX utility reads the compressed AFP files zipped in ZIP compression, or encrypted by standard ZIP encryption, WinZip 9.0 or PKZIP 8.0 compatible 128-bit, 192-bit, and 256-bit AES strong encryption.

SHOWIDX command syntax

```
SHOWIDX    [ afpFile | zipFile[;afpFileInZip] ] [ -p password ]
           [ -f fromCode -t toCode ]  [ -r range ]
```

Parameters

afpFile
The input AFP filename.

If the input file is not specified, then standard input (STDIN) will be used. With STDIN you can read the AFP data stream piped from an application's standard output (STDOUT), like the AFP data stream decompressed from the UNZIP, RAR, or 7-ZIP console utilities.

zipFile
Specifies a zip filename if the AFP files were compressed by ZIP compression. MakeAFP supports standard ZIP encryption, WinZip 9.0 or PKZIP 8.0 compatible 128-bit, 192-bit and 256-bit AES strong encryption.

afpFileInZip
Optional, specifies a selected AFP filename if your ZIP file contains multiple AFP files, it must be separated by a semicolon (;) or comma (,) without any blanks from the ZIP filename.

Dumped AFP indexing information is output to standard output (STDOUT).

-f *fromCode*
Optional, specifies the original AFP encoding if it was encoded in EBCDIC, mixed SBCS/DBCS, UTF-16 or UTF8, refer to *MakeAFP Encoding Name* document for details.

-t toCode
Optional, specifies the destination text in PC native encoding, refer to *MakeAFP Encoding Name* document for details.

-p *password*
Optional, the password of the encrypted AFP file if it was encrypted. Alternatively, for enhanced security reasons you can key in the password by DOS prompt in interactive mode.

-r *range*
Optional, the number of pages to be dumped, default is up to about 1000 pages if it is not pecified.

Example 1:

Assuming the indexed AFP file stmt1.afp is encoded in USA English with EBCDIC codepage 037, you can issue the following command to dump its indexing information to the text file dump.txt:

```
showidx  stmt1.afp -f ibm-037 -t ibm-437 > dump.txt
```

Output Example:

```
Begin Index Group
  Index Tag Name:  Insured
           Value:  Geoffrey R Stephens
  Index Tag Name:  Policy
           Value:  324-1443255-11
    *** 2 pages ***
End Index Group

Begin Index Group
  Index Tag Name:  Insured
           Value:  Beth N McShine
  Index Tag Name:  Policy
           Value:  434-5676889-01
    *** 5 pages ***
End Index Group
```

Example 2:

Assuming the indexed AFP file mobile_ch.afp is encoded in Simplified Chinese SBCS-HOST/DBCS-HOST with mixed codepage 1388, you can issue the following command to dump its indexing information to the text file dump.txt:

```
showidx  mobile_ch.afp -f ibm-1388 -t ibm-1386 > dump.txt
```

Output Example:

```
Begin Index Group
   Index Name : Mobile Number
         Value: 139165601013
   Index Name : Customer Name
         Value: 徐柏治
      *** 3 pages ***
 End Index Group

 Begin Index Group
   Index Name : Mobile Number
         Value: 139065602210
   Index Name : Customer Name
         Value: 萧仰定
      *** 3 pages ***
 End Index Group
```

Showing AFP Presentation Information

SHOWPTX utility is specially developed to quickly dump the information of AFP text contents and their positions, and whether AFP overlays, page segments, data objects images, copy-group are used so that you can uniquely identify the beginning of a page group or a page, or quickly capture the text fields for new purposes, like adding of barcode, pagination or page serial number, AFP page group indexes, etc.

SHOWPTX utility reads the compressed AFP files zipped in ZIP compression, or encrypted by standard ZIP encryption, WinZip 9.0 or PKZIP 8.0 compatible 128-bit, 192-bit, and 256-bit AES strong encryption.

SHOWPTX command syntax

SHOWPTX [*afpFile* | *zipFile* [;*afpFileInZip*]] [-p *password*]
[-f *fromCode* -t *toCode*] [-r *range*]

Parameters

afpFile
The input AFP filename.

If the input file is not specified, then standard input (STDIN) will be used. By STDIN you can read the AFP data stream piped from an application's standard output (STDOUT), like the AFP data stream decompressed from the UNZIP, RAR, or 7-ZIP console utilities.

zipFile
Specifies a zip filename if the AFP files were compressed by ZIP compression. MakeAFP supports the standard ZIP encryption, WinZip 9.0 or PKZIP 8.0 compatible 128-bit, 192-bit, and 256-bit AES strong encryption.

afpFileInZip
Optional, specifies a selected AFP filename if your ZIP file contains multiple AFP files, it must be separated by a semicolon (;) or comma (,) without any blanks from the ZIP filename.

Dumped information is output to standard output (STDOUT).

-f *fromCode*
Optional, specifies the original AFP encoding if it was encoded in EBCDIC, mixed SBCS-HOST/DBCS-HOST, UTF-16, or UTF8, refer to *MakeAFP Encoding Name* document for details.

-t *toCode*
Optional, specifies the destination text in PC native ASCII, mixed ASCII/DBCS-PC (BIG5, GB, KSC, SJIS) encoding, refer to *MakeAFP Encoding Name* document for details.

-p *password*
Optional, the password of the zipped AFP file if it was encrypted. Alternatively, for enhanced security reasons you can key in the password by DOS prompt in interactive mode.

-r *range*
Optional, the number of pages to be dumped, default is up to about 1000 pages if it is not specified.

Example 1:

Assuming the AFP file stmt2.afp is encoded in the native ASCII, you can issue the following command to dump its presentation information to the text file dump.txt:

```
showptx  stmt2.afp > dump.txt
```

Output Example:

```
Page 1     (x, y) position in PELS

  ( 8496, 862)   'Insurance'
  ( 8496, 1120)  'Specialists'
  (  792, 2308)  'Disability Income Policy'
  (  792, 2638)  'The Preferred Professional'
  ( 7330, 2209)  'Insurance Specialists Company'
  ( 7330, 2418)  '100 Main Street'
  ( 7330, 2627)  'Anycity, Anystate 99999-9999'
  (  792, 5353)  'Insured'
  ( 2448, 5353)  'Geoffrey R Stephens'
  (  792, 6097)  'Policy Number'
  ( 2448, 6097)  '324-1443255-11' ←— May to be used as barcode value
  ( 7114, 6097)  '03-25-53'
  ( 9048, 6097)  'Date of Issue'
```

Example 2:

Assuming the AFP file mobile_ch.afp is encoded in SBCS-HOST/DBCS-HOST with mixed codepage 1388, you can issue the following command to dump its presentation information to the text file dump.txt:

```
showptx  -f ibm-1388 –t ibm-1386 mobile_ch.afp > dump.txt
```

Output Example:

```
Page 1

  Page units per unit base for the X axis:  6000
  Page units per unit base for the Y axis:  6000
  Page extent for the X axis (Page Width):  4962
  Page extent for the Y axis (Page Length): 7002

  Overlay Included: O1OVL1
  Overlay Included: O1PIC1

          (x, y) position in PELS
  ( 1200, 720)  '2005'
  ( 1485, 720)  '11'
  ( 1680, 720)  '01'
  (  660, 960)  '212914'
  (  660, 1110)  '徐柏治'
  (  660, 1230)  '上海延安东路２２２号外滩中心'
  (  660, 1350)  '２２楼２２号'
  (  660, 1470)  '上海市'
  ( 3930, 900)  '139165601013'   ←— may used as barcode value
  ( 3930, 1068)  '534.56'
  ( 3930, 1236)  '534.56-'
  ( 3930, 1404)  '0.00'
  ( 3930, 1572)  '1437.90'
  (  330, 2160)  '基本月租费'
  ( 1125, 2160)  '120.00'
  (  330, 2280)  '本地基本费'
  ( 1125, 2280)  '160.00'
  (  330, 2400)  '国内漫游费'
  ( 1125, 2400)  '180.00'

                 :
```

Defining Encodes of AFP and PC Native

MakeAFP Weaver provides a special “Encoding” function to allow you to define the default encoding names of your AFP document and your PC native, to let MakeAFP Weaver convert your nonnative encoded AFP index values or text fields to native encoding automatically.

This function must be called before processing your non-ASCII or non-ASCII/DBCS-PC encoded AFP or can be recalled again for any default encoding changes if needed.

C Sample:

```
/***********************************************************************/
/* AFP was encoded in CP-037, USA EBCDIC, Encoding function must be    */
/* called, so that Weaver does able to do conversion from EBCDIC to    */
/* ASCII internally                                                    */
/*                                                                     */
/* This sample also shows how to mask an area, capture an index value  */
/* as the part of string for add a barcode, and add a page segment     */
/***********************************************************************/

int main( )
{
  unsigned int i, grpPages, pageSN = 0;
  char tmp[80], policyNo[20];

  $MaxPaging = 50;            // Maximum paging is up to 50 pages

  SetUnit(IN_U600);           // Set default unit to inch

  Start();                    // Start initiation, open default input,
                              // output and definition files, retrieves
                              // AFP resources, allocate memory

  Encoding("ibm-037","ibm-437");   // AFP – CP037, PC - CP437

  OpenDoc();                  // Open AFP document

  while ($Edt == 0)           // Until end of AFP document
  {
    $Page = 0;                // Reset AFP page buffer number

    do {

      $Page++;                // Point to next AFP page buffer

      GetPage();              // Get a page from existing AFP file

    } while ($Eng == 0);      // Until end of each indexed AFP page group


    // Now got all pages of a page group, it is
    // ready to compose the new AFP output

    grpPages = $Page;         // keep total number of pages per group

    for (i = 0; i < grpPages; i++)
    {
      $Page = i + 1;          // Point to page buffer number to be opened again

      InclPseg("S1OWL", 0.3, 0.25);    // Add a new page segment image
```

```
        MaskArea(5, 0.4, 2, 0.75);       // Mask an area on every page

        sprintf(tmp, "Page %d of %d", $Page, grpPages); // Generate pagination
        Font(1);  Pos(8, 0.45);  Rtxt(tmp);

        sprintf(tmp, "%06d", ++pageSN);  // generate page serial number

        Font(2);  Pos(0.2, 10.8);  Ltxt(tmp);  // With MakeAFP Weaver, you can
                                               // use a AFP font encoded in ASCII
                                               // directly

        GetIdx("Policy", policyNo);            // MakeAFP Weaver does auto-
                                               // conversion with the
                                               // encoding names defined with
                                               // Encoding() function

        sprintf(tmp, "%d %d %s", pageSN, $Page, policyNo);

        BarCode(CODE128, tmp, 0.3, 2, 2, 0.2, DEG90);  // Add 1D barcode 128
        DataMatrix(tmp, 5.4, 0.8, 0.4, 0.4);           // Add 2D DataMatrix

        ClosePage();              // Close AFP page, write each page to AFP file
      }
    }

    CloseDoc();                   // Close AFP document, close AFP output file

    #ifdef      _DEBUG
      ViewAFP();                  // Only view AFP output in debug mode
    #endif

    return 0;
  }
```

Getting Index Values

MakeAFP Weaver provides the “Get Index” function that allows you to retrieve the index values of page group level or page level index from an indexed AFP.

“Get Index” function can be called to retrieve the text strings of an index value, after an indexed page group, the first page of page group, or a page is read-in by the “Open Page” function.

MakeAFP Weaver converts the index value of non-ASCII or non-ASCII/DBCS-PC encoded AFP to ASCII or ASCII/DBCS-PC native encoding if AFP and PC encoding are specified by the “Encoding” function. Make sure the “Encoding” function is called before the “Get Index” function is called.

MakeAFP Weaver provides two special variables, $Bng indicates whether the “Begin of Name Group” of AFP index boundary has been detected, $Eng indicates whether the “End of Name Group” of AFP index boundary has been detected.

With the MakeAFP ShowIDX utility, you can quickly dump the index names and their values.

C Sample:

See the previous sample under the “Defining Encodes of AFP and PC Native” section.

Getting Text Data Fields

MakeAFP Weaver provides some special "Trigger" functions, with which you can define a location or a location area with a string or mask pattern, an object name (such as overlay, page segment, data-object image, copy-group) as the trigger to uniquely identify the beginning of a page group or a page.

MakeAFP Weaver also provides some special "Get Field" functions, with which you can capture a text field string by its coordinate location, or a location area.

The trigger defines the indication information that indicates which AFP page containing the data fields we need. The trigger is consistent as a milepost throughout the AFP document.

The data fields are associated with triggers and contain the information that will be used for the AFP indexing or repurposes. Fields are defined by location or location range relative to the location of the trigger.

These functions can be called to detect a trigger or capture the text strings of a data field once the AFP page is read-in by the "Get Page" function.

MakeAFP Weaver converts the data field of nonnative encoded AFP to the ASCII or ASCII/DBCS-PC native encoding if AFP and PC encoding are specified by the "Encoding" function. Make sure the "Encoding" function is called before this function is called properly.

With the MakeAFP ShowPTX utility, you can quickly dump the data fields and their coordinate locations in PELS, as well as the name of overlay, page segment, data-object image, and copy-group.

MakeAFP Weaver provides a special variable $Edt, indicating whether the "End of AFP Document" is detected.

C Sample:

```
/***************************************************************************/
/* This sample shows how to capture a trigger and field from first page of */
/* each page-group and adding barcode to existing AFP.                    */
/*                                                                         */
/* AFP was encoded in native CP-437, ASCII                                 */
/***************************************************************************/

int main( )
{
  unsigned int i, grpPages, pageSN = 0;
  char tmp[80], savingsNo[20];
  bool bog = 0;

  $MaxPaging = 50;        // Maximum paging is up to 50 pages

  SetUnit(IN_U600);          // Set default unit to inch

  Start();                // Start initiation, open default input,
                          // output and definition files, retrieves
                          // AFP resources, allocate memory

  OpenDoc();              // Open an AFP document

  $Page = 1;              // Set AFP page buffer number to 1 for the first page
                          // of AFP file

  GetPage();              // Get first page of AFP file
```

```
    while ($Edt == 0)       // Until end of AFP document
    {
                            // Get Savings A/C number, like 123-4758-9586, from
                            // first page of each group by a match pattern
      GetField2(180, 180, 1080, 1120, savingsNo, "###-####-####");

      do {

        $Page++;              // Point to next AFP page buffer

        GetPage();            // Get next AFP page

        // detecting if it is the first page of a group,
        // "Page 1 of" text string only appears at
        // first page of each page group
        bog = Trigger2(2187, 2187, 1120, 1200, "Page 1 of");

     } while (!bog && !$Edt);  // Until beginning of next page group or end of
                               // AFP file

     bog = 0;                  // Reset it for next group

     // Now got all pages of a page group and first page of next group, now it is
     // ready to compose new AFP output

     if (!$Edt)
       grpPages = $Page -1 ;   // keep total number of pages per group, need to
                               // minus 1 page of the first page of next group

     for (i = 0; i < grpPages; i++)
     {
       $Page = i + 1;          // Point to page buffer number to be opened again

       sprintf(tmp, "%d %d %s", ++pageSN, $Page, savingsNo);
       DataMatrix(tmp, 0.2, 2.2, 0.4, 0.4);

       ClosePage();              // Close AFP page, write to AFP file
     }

     MovePage(1, grpPages + 1);  // As we got first page of next group previously,
                                 // now need to move its contents to page buffer 1
                                 // for next page group
     $Page = 1;                  // Reset page buffer to page 1 of next page group
   }

   CloseDoc();                   // Close AFP document, close AFP output file

   return 0;
 }
```

Working with Pagination and Grouping

MakeAFP Weaver offers a powerful capability that allows you to add OMR lines, dynamic barcodes, pagination, such as "Page x of y", on every page of your statements. With MakeAFP Weaver, you can open multiple pages by the "Get Page" or "Open Page" functions either by reading AFP pages from the AFP input file or adding new AFP pages to be composed and then process different pages in an interleaved manner once each page is initialized, all of the AFP data streams will be kept in memory buffers in page-level. After you processed and counted all of the pages of a page-group, you can put your pagination text, OMR, or barcode on each page just before you close the page and write into the AFP output file with the "Close Page" function.

MakeAFP Weaver provides a special variable $Page, with which you can switch to any AFP page directly regardless of if it is to be opened or already opened.

C Sample:

```
/************************************************************************/
/* This sample shows how to mask an area, capture an index value        */
/* as the part of string for add a barcode, and add pagination text     */
/*                                                                      */
/* Indexed AFP was encoded in CP-037, USA EBCDIC                        */
/************************************************************************/

int main( )
{
  unsigned int i, grpPages, pageSN = 0;
  char tmp[80], policyNo[20];

  $MaxPaging = 50;            // Maximum paging is up to 50 pages

  SetUnit(IN_U600);           // Set default unit to inch

  Start();                    // Start initiation, open default input,
                              // output and definition files, retrieves
                              // AFP resources, allocate memory

  Encoding("ibm-037","ibm-437");   // AFP – CP037, PC - CP437

  OpenDoc();                  // Open AFP document

  while ($Edt == 0)           // Until end of AFP document
  {
    $Page = 0;                // Reset AFP page buffer number

    do {

      $Page++;                // Point to next AFP page buffer

      GetPage();              // Get a page from existing AFP file

    } while ($Eng == 0);      // Until end of each page group


    // Now got all pages of a page group, now it is
    // ready to compose the new AFP output

    grpPages = $Page;         // keep total number of pages per group
    for (i = 0; i < grpPages; i++)
    {
      $Page = i + 1;          // Point to page buffer number to be opened again

      InclPseg("S1OWL", 0.3, 0.25);    // Add a new page segment image

      MaskArea(5, 0.4, 2, 0.75);       // Mask an area on every page

      sprintf(tmp, "Page %d of %d", $Page, grpPages); // Generate pagination

      Font(1);  Pos(8, 0.45);  Rtxt(tmp);             // right align pagination

      sprintf(tmp, "%06d", ++pageSN);  // generate page serial number

      Font(2);  Pos(0.2, 10.8);  Ltxt(tmp);  // With MakeAFP Weaver, you can
                                             // use a font encoded in ASCII
                                             // directly

      GetIdx("Policy", policyNo);            // MakeAFP Weaver does auto-
                                             // conversion with the
                                             // encoding names defined with
                                             // Encoding() function

      sprintf(tmp, "%d %d %s", pageSN, $Page, policyNo);

      BarCode(CODE128, tmp, 0.3, 2, 2, 0.2, DEG90);  // Add 1D barcode 128
      DataMatrix(tmp, 5.4, 0.8, 0.4, 0.4);           // Add 2D DataMatrix

      ClosePage();              // Close AFP page, write each page to AFP file
    }
  }

  CloseDoc();                   // Close AFP document and AFP output file

  #ifdef      _DEBUG
    ViewAFP();                  // Only view AFP output in debug mode
  #endif

  return 0;
}
```

Adding Page Group Indexes

The AFP document pages can be organized into smaller, uniquely identifiable units, called page groups. Each group represents a logical multi-page bundle. MakeAFP lets you divide a large AFP output document into individual page groups, by inserting AFP indexing structured fields in the AFP file that define the group boundaries.

With the “Begin Index” and “End Index” functions, you can define the start and end of index page group boundaries within an AFP output document, so the statement pages that belong to each client can be quickly navigated and retrieved by the AFP viewer and AFP archiving system, and can be used by MakeAFP reprint and sorting utilities.

With the “Put Index” function, you can identify a group of pages with the indexing tag containing an attribute name and value. For instance, an “Account Number” attribute is associated with the account number of each customer. This type of tag is called a group-level tag since it is associated with a group of pages.

In addition, MakeAFP Weaver allows you to generate a group of AFP index object file, AFP resources file and AFP document file that can be loaded into IBM Content Manager OnDemand directly in high performance. Refer to *MakeAFP Weaver Reference* Chapter 2 for more details about using MakeAFP definition parameters RESTYPE and INDEXOBJ.

For group-level AFP indexing, a “Begin Index” function and “Put Index” functions must be called before writing the first page of each page group, and “End Index” must be called after writing the last page of each page group.

C Sample:

```
/***************************************************************************/
/* This sample shows how to capture a trigger by an overlay name and      */
/* data fields from page 1, then add AFP indexes and barcode to AFP output */
/*                                                                         */
/* Non-indexed AFP input document was encoded in CP-037, USA EBCDIC       */
/***************************************************************************/

int main( )
{
   unsigned int i, grpPages, pageSN = 0;
   char tmp[80], mobileNo[20], custName[60];
   bool bog = 0;

   $MaxPaging = 50;          // Maximum paging is up to 50 pages

   SetUnit(IN_U600);         // Set default unit to inch

   Start();                  // Start initiation, open default input,
                             // output and definition files, retrieves
                             // AFP resources, allocate memory

   Encoding("ibm-037","ibm-437");

   OpenDoc();                // Open AFP document

  $Page = 1;                 // Set AFP page buffer number to 1 for the first
                             // page of AFP file

   GetPage();                // Get first page of AFP file

   while ($Edt == 0)         // Until end of AFP document
   {
      GetField(660, 1080, custName);      // Get customer name

      GetField(4050, 900, mobileNo);      // Get customer mobile number

      do {

         $Page++;             // Point to next AFP page buffer

         GetPage();           // Get next page

         // detecting if it is the first page of next page group,
         // overlay O1OVL1E only used by first page of each page group
         bog = TriggerOvly("O1OVL1E");

       } while (!bog && !$Edt);      // Until beginning of next page group or
                                     // End of AFP file

      bog = 0;                       // Reset it for next group

      // Now got all pages of a page group and first page of next group, now it
      // is ready to process new AFP output

      if (!$Edt)                     // If not end of AFP document
         grpPages = $Page -1 ;       // Keep total number of pages per group,
                                     // need to minus 1 page of the first
                                     // page of next group

      BgnIdx(mobileNo);         // Auto-converts ASCII to EBCDIC for indexes
      PutIdx("Customer Name", custName);
      PutIdx("Mobile Number", mobileNo);

      for (i = 0; i < grpPages; i++)
      {
         $Page = i + 1;              // Point to page buffer number to be opened

         sprintf(tmp, "%d %d %s", ++pageSN, $Page, mobileNo);
         BarCode(CODE128, tmp, 0.25, 2.2, 2, 0.2, DEG90);   // Add 1D barcode

         ClosePage();                // Close AFP page, write to AFP file
      }

      EndIdx();                      // End of group level index

      MovePage(1, grpPages + 1);     // As we got first page of next group
                                     // beforehand, now need to move its contents
                                     // to page buffer 1 for the next page group

      $Page = 1;                     // Reset page buffer to 1 for next page group
   }

   CloseDoc();                       // Close AFP document

   return 0;
}
```

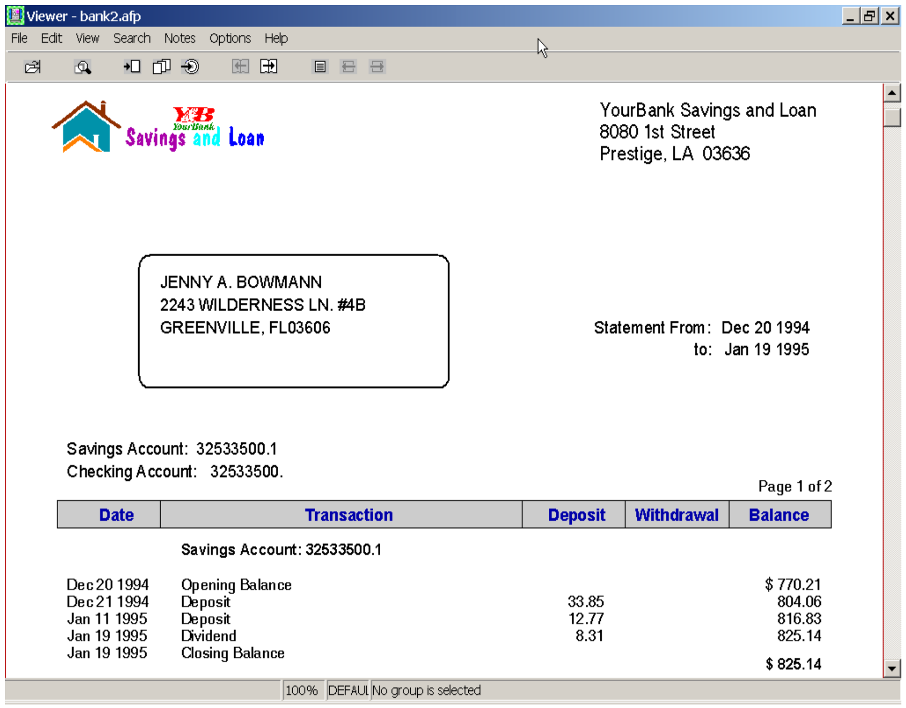
Viewing AFP File

MakeAFP Weaver provides the “View AFP” function with which you can immediately call an AFP viewer to view the AFP output file you just generated in debug or execute mode, this will assist you greatly during your development.

AFP viewer for Windows can be integrated easily with MakeAFP Weaver by the “View AFP” function so that you can view a newly generated AFP file immediately during your development.

With Windows Explorer, you can select “Tools --> Folder Options --> File Types --> New” to associate the AFP type file to an AFP viewer. Once you defined the new AFP file type, MakeAFP Weaver will be able to call your AFP viewer to view the generated AFP file.

The “View AFP” function must be called after the “Close Document” function.


Viewer - bank2.afp
File Edit View Search Notes Options Help
YB YourBank Savings and Loan
100% DEFAUL No group is selected

YourBank Savings and Loan
8080 1st Street
Prestige, LA 03636

JENNY A. BOWMANN
2243 WILDERNESS LN. #4B
GREENVILLE, FL03606

Statement From: Dec 20 1994
to: Jan 19 1995

Savings Account: 32533500.1
Checking Account: 32533500.

Page 1 of 2

Date	Transaction	Deposit	Withdrawal	Balance
	Savings Account: 32533500.1			
Dec 20 1994	Opening Balance			$ 770.21
Dec 21 1994	Deposit	33.85		804.06
Jan 11 1995	Deposit	12.77		816.83
Jan 19 1995	Dividend	8.31		825.14
Jan 19 1995	Closing Balance			$ 825.14

C Sample:

```
void main( )
{
  Start();                // Start initiation, open default input,
                          // output and definition files, getting
                          // AFP resources and font information
  OpenDoc();
      :
      :
  CloseDoc();             // Close AFP document and close its file

  #ifdef _DEBUG           // Only view AFP in debug mode

    ViewAFP();            // view AFP file just generated, it must
                          // be called after CloseDoc() function
  #endif
}
```

Working with MakeAFP Definition File

MakeAFP definition file is used to specify your AFP resource names, TrueType / TrueType Collection / OpenType fonts, the locations of resource directories, and whether you want to transfer and retrieve AFP resources from the input AFP file and resource directories, and whether put the resources required inline within your AFP document file or put them into a separate AFP resource file.

Once the MakeAFP definition file is defined for your program, you can call it with the command-line argument flag "-d definition_file" when you run your program in batch mode.

Definition Sample 1:

```
restype=all,inline           // Transfers and retrieves all type of resources from
                             // input AFP file and resource directories, then put
                             // them inline within AFP output file
fdeflib=c:\makeafp\reslib    // Form Definition directory
ovlylib=c:\makeafp\reslib    // Overlay directory
pseglib=c:\makeafp\reslib    // Page segment directory
fontlib=c:\makeafp\reslib    // Font directory
font1=czh200,t1000437,12.7   // Font 1, outline AFP font, point size is 12.7
font2=czh200,t1000437,11     // Font 2, by character set and code page
font3=xzn200,10              // Font 3, by coded font name
font4=c0d0gt10,t1000437      // Font 4, AFP raster font
pseg=s1yblogo                // AFP Page Segment
ovly=o1bank1                 // AFP Overlay
```

Definition Sample 2:

```
restype=fdef,ovly,pseg,objt  // Only transfers and retrieves form definitions,
                             // overlays, page segments and data-object from input
                             // AFP file and resource directories, then put
                             // them into a separate AFP resource file

reslib=c:\makeafp\reslib;d:\ipm2000\reslib   // Resource paths for overlays,
                                             // page segments, fonts and
                                             // form definitions
font1=czh200,t1000437,12
font2=czh200,t1000437,11.5
font3=xzn200,10
font4=c0d0gt10,t1000437
ovly=o1bank01
objt=flower,jpeg                             // non-AFP image in JPEG
                                             // format
```

Definition Sample 3:

```
restype=none                 // Specifies that no resource file be created &
                             // no AFP resources inline in AFP output file
fdeflib=c:\makeafp\reslib
ovlylib=c:\makeafp\reslib
pseglib=c:\makeafp\reslib
fontlib=c:\makeafp\reslib;c:\winnt\fonts   // Font directories of AFP fonts
                                           // and OpenType/TrueType fonts
font1=czh200,t1000437,12
font2=czh200,t1000437,11
font3=xzn200,10
font4=airialuni.ttf,T1000437,12            // Specifies using a TrueType font
                                           // "Arial Unicode MS", font size
                                           // 12, and encoding is ASCII codepage
                                           // T1000437 for USA English
```

Installing MakeAFP Weaver and Learning from Samples

Prerequisites

- Windows 7 or above.
- Visual Studio C++ 6.0 or above for using C/C++ programming.
- MakeAFP Viewer, MakeAFP Workbench, AFP Workbench Viewer, or IBM Content Manager OnDemand Client.

Installing MakeAFP Weaver

MakeAFP Weaver installation is easy:

- Log in as an Administrator user.
- Run setup package you received.
- Follow the instructions on the installation screens to install the package, the destination folder is **c:\makeafp**.

Learning from MakeAFP Weaver Samples

MakeAFP supplied samples are installed in the path c:\makeafp\afpweaver. You can run c:\makeafp\afpweaver\test\demo.bat directly for batch testing, or click any of the Visual Studio 6.0 workspace project files *.dsw in the subdirectories of c:\makeafp\afpweaver\VC6\ if you are using Visual Studio C++ 6.0, or click any of the solution files *.sln in the c:\makeafp\afpweaver\VC7\ and c:\makeafp\afpweaver\VC8\ subdirectories if you are using Visual Studio C++ 2003 or 2005, and then press the F5 key to execute the program or press the F10 or F11 (go into sub-function) key to run the program in a step by step debugging mode to familiarize yourself with MakeAFP Weaver.

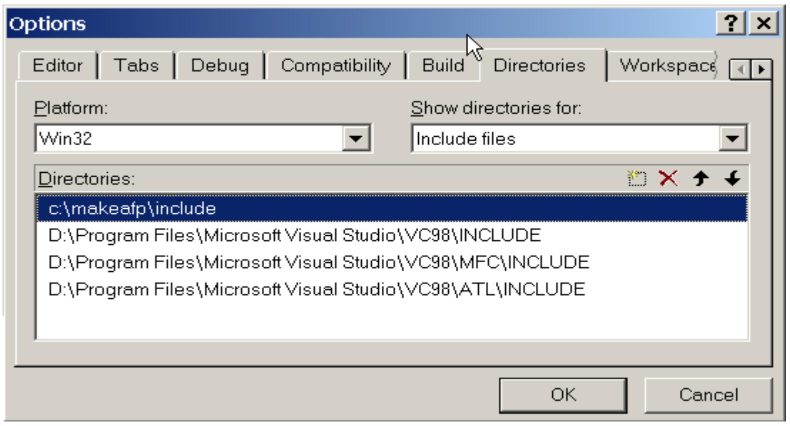
MakeAFP Weaver samples for VB.NET are provided in path c:\makeafp\afpweaver\VB.NET.

Compiling and Running Under Visual C++ V.60

To compile and run a MakeAFP Weaver program, you need to create a project in Microsoft Visual C++ that makes use of the MakeAFP library for Windows 2000/XP/2003. You can use the steps here as a reference If you are using Visual Studio C++ 6.0.

Adding MakeAFP Include Files' Directory

With Visual Studio C++ 6.0, select "Tools" menu --> "Options" --> "Directories" --> "Include files", to add directory c:\makeafp\include where the MakeAFP header file afpweaver.h was installed.



Platform: Win32

Show directories for: Include files

Directories:

c:\makeafp\include
D:\Program Files\Microsoft Visual Studio\VC98\INCLUDE
D:\Program Files\Microsoft Visual Studio\VC98\MFC\INCLUDE
D:\Program Files\Microsoft Visual Studio\VC98\ATL\INCLUDE

Or just put the following coding at the beginning of your program:

```
#include "c:\\makeafp\\include\\afpweaver.h"
```

Creating Your New Project

- With Visual Studio C++ 6.0, select “File” --> “New” --> “Projects” -->"Win32 Console application" to create your new project.
- From the “Build” menu, select “Set Active Configuration” to switch to either define release or debug configuration.
- From the "Build" menu, select "Set Active Configuration…” to open the " Set Active Project Configuration" dialog box, then select **release** configuration.
  - From the "Project" menu, select "Settings" to open the "Project Settings" dialog box, then select the "C/C++" tab.
  - From the "Category:" pull-down menu in the "C/C++" tab, select "Code Generation", then from the "Use run-time library:" pull-down menu, select **"Multithreaded DLL"**.
- From the "Build" menu, select "Set Active Configuration…” again to open the " Set Active Project Configuration" dialog box, then select **debug** configuration.
  - From the "Project" menu, select "Settings" again to open the "Project Settings" dialog box, then select the "C/C++" tab.
  - From the "Category:" pull-down menu in the "C/C++" tab, select "Code Generation", then from the "Use run-time library:" pull-down menu, select option **"Debug Multithreaded DLL"**.
- From the “Project” menu, Select “Add To Project” --> “Files...” --> select Files of type as Library Files (.lib) --> then insert library file c:\makeafp\lib\afpweaver.lib into your project.

  Or, select the "Link" tab in the "Project Settings" dialog box, then from the "Category:" pull-down menu in the "Link" tab, select "Input", in the "Object/library modules:" entry field, enter the name of the module: afpweaver.lib. In the "Additional library path:" entry field, enter the path c:\makeafp\lib in which afpweaver.lib was installed.

  Or just put the following coding at the beginning of your MakeAFP Weaver program:

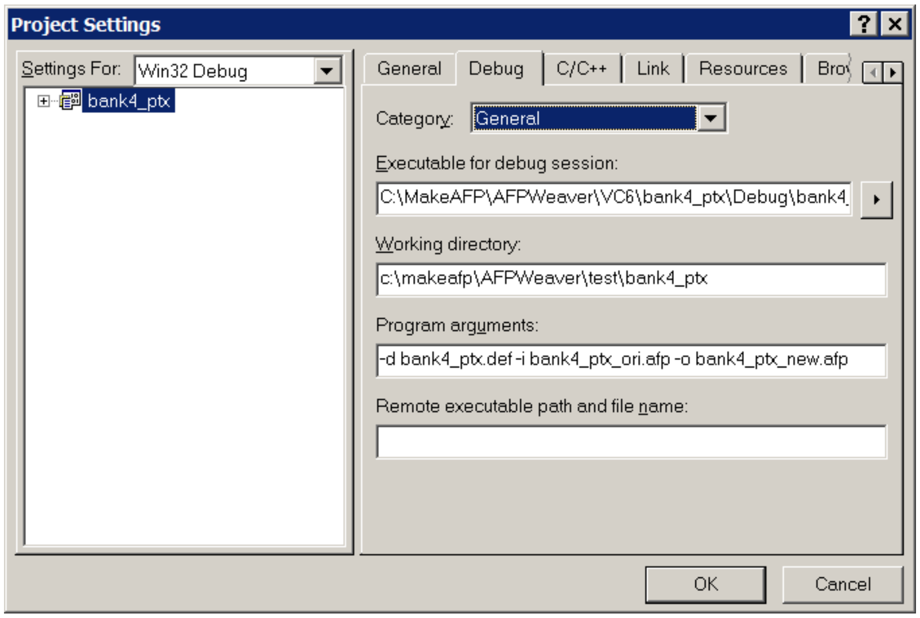
```
#pragma comment (lib,"c:\\makeafp\\lib\\afpweaver.lib")
```

Running MakeAFP Weaver program in Debugging Mode

During your development, you can run the program in debug or execute mode with your Visual Studio C++, and you can view your just generated AFP file immediately once you called the “View AFP” function within your program.

On Visual Studio C++ 6.0, From the "Build" menu, select "Set Active Configuration…” again to open the " Set Active Project Configuration" dialog box, select **debug** configuration and return, then select Project menu --> Settings --> Debug, to define your working directory and command arguments. In the “Working directory” entry field, enter the path name of your working directory where you keep your input testing file and MakeAFP definition file, and in the “Program arguments” entry field, enter arguments by MakeAFP parameters syntax as:

`-d` *`definition_file`* `-i` *`input_afp_file`* `-o` o*`utput_afp_file`*.



Category: General

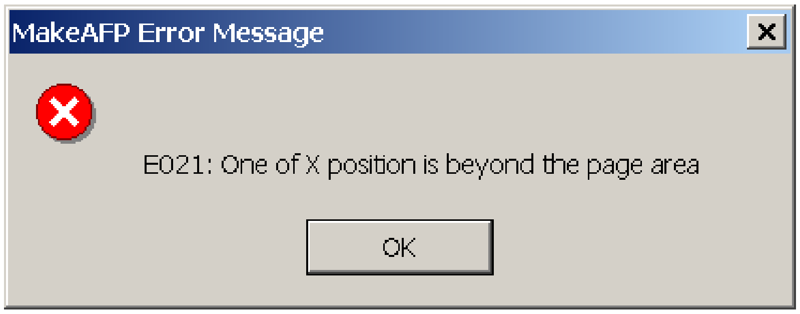
Executable for debug session: C:\MakeAFP\AFPWeaver\VC6\bank4_ptx\Debug\bank4_

Working directory: c:\makeafp\AFPWeaver\test\bank4_ptx

Program arguments: -d bank4_ptx.def -i bank4_ptx_ori.afp -o bank4_ptx_new.afp

Remote executable path and file name:

MakeAFP Weaver reports the error message and stops your program if there is any problem detected, while you are in development debug or execute mode.



E021: One of X position is beyond the page area

MakeAFP Weaver invokes the AFP viewer automatically to view the AFP data stream generated once any error message has taken place so that you can see both error messages and AFP output to analyze the problem.

Languages Bindings for Visual C#

The usage of MakeAFP Weaver with Visual C# is essentially the same as with C or C++.

To use MakeAFP Weaver with C#, you need to add the c:\makeafp\include\AFPWeaver.cs file to your project, via menu "Project" --> "Add Existing Element…".

MakeAFP Weaver samples for C# are provided in path c:\makeafp\afpweaver\c#, click a solution files *.sln and then press the F5 key to execute the program or press the F10 or F11 (go into sub-function) key to run the program in a step by step debugging mode to familiarize yourself with MakeAFP Weaver.

Languages Bindings for Java

The usage of MakeAFP Weaver with Java is essentially the same as with C or C++.

To use MakeAFP Weaver with Java you need to add the c:\makeafp\include\afpweaver.java file to your project.

MakeAFP Weaver samples for Java are provided in path c:\makeafp\afpweaver\java for the NetBeans graphical IDE.

With NetBeans, you can open MakeAFP Weaver Java projects, by selecting File --> Open Project… --> select path C:\MakeAFP\AFPWeaver\Java\NetBeans, then select one of project or all of the projects.

Chapter 2. Running MakeAFP Weaver in Batch Mode

Once you have completed the development of your custom MakeAFP Weaver program and generated the executable module in the release mode, you can run your program in high performance by its command-line.

Running MakeAFP Weaver in Batch Mode

You can run MakeAFP Weaver in batch mode by using the following command syntax:

```
program_name -d definitionFile [-i inputAfpFile] [-o outputAfpFile] [-l]
```

"-i input_afp_file" is optional, you may read the AFP data stream from the STDIN (Standard Input) piped from the output of another application directly, like from UNZIP, UNRAR, and 7-ZIP, etc.

"-o output_afp_file" is optional, you may write the AFP data stream to STDOUT(Standard Output) as another application input, for example to a print submit command directly.

"-l" is optional, if you want to output the error message to STDERR (Standard Error) in DOS command-line mode, default is pop-up the error message by a message window.

Reading the Compressed or Encrypted AFP Input File

While you are running your MakeAFP Weaver program in batch mode, you can either read AFP from an AFP input file if *"-i input_afp_file"* flag parameter is specified, or read the input AFP data stream directly from the output of another application program on the fly in high speed via the system pipe.

Example 1: Reading WinZip AES 256-bit encrypted AFP input file

Assuming your MakeAFP Weaver program is stmt1.exe, and your zipped AFP input file smt1.zip is encrypted and compressed by PKZIP AES 256-bit encryption with password *rainbow1*, you can process it directly on the fly with the MakeAFP munzip utility:

```
munzip -s -p rainbow1 stmt1.zip | stmt1 -d stmt1.def -o stmt1_new.afp
```

Example 2: Reading input AFP file compressed in ZIP 2.0 standard format

Assuming your MakeAFP Weaver program is stmt2.exe, and your zipped input AFP file smt2.zip is compressed by Info-ZIP in ZIP 2.0 standard format, you can direct process it on the fly with the Info-ZIP utility UNZIP:

```
unzip -p stmt2.zip | stmt2 -d stmt2.def -o stmt2_new.afp
```

Example 3: Reading 7-ZIP formats input AFP file

With 7-Zip freeware from www.7-zip.org, you can read compressed formats of 7Z, ZIP, CAB, RAR, ARJ, GZIP, BZIP2, TAR, CPIO, RPM, and DEB, it also supports its own AES 256 bits encryption and its LZMA compression algorithm offers ultra-high compression ratio.

Assuming your MakeAFP formatting program is stmt3.exe, and your input AFP file smt3.7z is encrypted and compressed by 7-Zip utility with password *rainbow2*, you can read 7-Zip encrypted data directly on the fly with the command:

```
    7zip e -prainbow2 -so stmt3.7z | stmt3 -d stmt3.def -o stmt3_new.afp
```

Example 4: Reading RAR format input file

With UNRAR freeware from www.rarlab.com, you can decompress RAR format compressed and encrypted files.

Assuming your MakeAFP formatting program is stmt4.exe, and your input file smt4.rar is encrypted and compressed by RAR shareware or WinRAR utility with password *bigsnow5*, you can read RAR encrypted AFP file directly on the fly with the command:

```
  unrar p –ierr –pbigsnow5 stmt4.rar | stmt4 -d stmt4.def -o stmt4_new.afp
```

MakeAFP Weaver Automation

With powerful MakeAFP automation utility AutoMakeAFP, once your AFP input file, or compressed & encrypted AFP input file is received, it will automatically be dispatched to the appropriate AFP applications for subsequent processing, such as AFP enhancements by your MakeAFP Weaver program, AFP sorting by postal code and mail-piece, print submits, etc. AFP input files can be either automatically deleted for a security reason or retained into the appropriate directory after it is processed.

Refer to *MakeAFP Utilities User's Guide* for more details about the AutoMakeAFP utility.

Chapter 3. Working With Form Definition

This chapter describes the form definitions that are supplied With MakeAFP Weaver and contains some information on how to use AFP form definition with MakeAFP Weaver.

MakeAFP Supplied Form Definitions

A form definition specifies how the printer controls the processing of the physical sheets of paper, such as input paper bin switching, simplex or duplex printing, N-UP partitions, color rendering, and CMR (Color Management Resource) association for the whole print job or group pages, etc.

MakeAFP supplied AFP form definition object resources are stored in path c:\makeafp\reslib, PPFA source codes are stored in c:\makeafp\ppfa directory for your reference.

All of the form definitions supplied by MakeAFP are in across print direction with (0, 0) offset.

Form Definitions for cut-sheet printers, with multiple input paper bins (1, 2, 3, 4, 5, envelope, manual) defined:

Form Definition	Presentation	Sides	N_UP	Page Placement
F1LCD	Landscape	1, **2**	1	Default
F1LCDN2	Landscape	1, **2**	2	Default
F1LCS	Landscape	**1**, 2	1	Default
F1LCSN2	Landscape	**1**, 2	2	Default
F1LCT	Landscape	1, **Tumble**	1	Default
F1LCTN2	Landscape	1, **Tumble**	2	Default
F1PCD	Portrait	1, **2**	1	Default
F1PCDN2	Portrait	1, **2**	2	Default
F1PCDN2A	Portrait	**2**	2	Page 1 at 1 front Page 2 at 2 back Page 3 at 2 front Page 4 at 1 back
F1PCDN2B	Portrait	**2**	2	Page 1 at 1 front Page 2 at 2 front Page 3 at 1 back Page 4 at 2 back
F1PCDN2C	Portrait	**2**	2	Page 1 at 1 front Page 2 at 2 front Page 3 at 2 back Page 4 at 1 back
F1PCDN3A	Portrait	**2**	3	Page 1 at 1 front Page 2 at 2 front Page 3 at 3 front Page 4 at 1 back Page 5 at 2 back Page 6 at 3 back
F1PCDN3B	Portrait	**2**	3	Page 1 at 1 front Page 2 at 3 back Page 3 at 2 front Page 4 at 2 back Page 5 at 3 front

				Page 6 at 1 back
F1PCS	Portrait	**1**, 2	1	Default
F1PCSN2	Portrait	1, 2	2	Default
F1PCT	Portrait	1, Tumble	1	Default
F1PCTN2	Portrait	1, Tumble	2	Default
F1PCTN2P	Portrait	Tumble	2	Page 1 at 1 front Page 2 at 1 back Page 3 at 2 front Page 4 at 2 back

** In the Sides column, the bolded value is the default defined by the default copy group.*

Form Definitions for continuous form (also call fan-fold) printers:

Form Definition	**Presentation**	**Sides**	**N_UP**	**Page Placement**
F1LFD	Landscape	2	1	Default
F1LFS	Landscape	1	1	Default
F1LFSN2	Landscape	1	2	Default
F1PFD	Portrait	2	1	Default
F1PFS	Portrait	1	1	Default
F1PFSN2	Portrait	1	2	Default
F1PFDN2A	Portrait	2	2	Page 1 at 1 front Page 2 at 2 back Page 3 at 2 front Page 4 at 1 back
F1PFDN2B	Portrait	2	2	Page 1 at 1 front Page 2 at 2 front Page 3 at 1 back Page 4 at 2 back
F1PFDN2C	Portrait	2	2	Page 1 at 1 front Page 2 at 2 front Page 3 at 2 back Page 4 at 1 back
F1PFDN2T	Portrait	Tumble	2	Page 1 at 1 front Page 2 at 1 back Page 3 at 2 front Page 4 at 2 back
F1PFDN3A	Portrait	2	3	Page 1 at 1 front Page 2 at 2 front Page 3 at 3 front Page 4 at 1 back Page 5 at 2 back Page 6 at 3 back
F1PFDN3B	Portrait	2	3	Page 1 at 1 front Page 2 at 3 back Page 3 at 2 front Page 4 at 2 back Page 5 at 3 front Page 6 at 1 back

For concept and functions of form definition, refer to IBM *Page Printer Formatting Aid User's Guide* for more information.

How to Handle Copy Group in a MakeAFP Weaver Program

A single form definition contains one or more several subsets of page controls, called *copy groups*. Copy groups define each page that can be used in the printing job. For example, when we are printing some pages in duplex mode, we need a copy group that is defined using both sides of the paper.

With the parameter *FDEF=fdefname* defined in your MakeAFP Weaver Definition file, you can specify which form definition is to be used with your MakeAFP Weaver job; and with the MakeAFP Weaver "Copy Group" function call in your program, you can invoke a new copy group previously defined within your form definition directly.

With GetPage(true) function you can read an AFP page and remove the current copy-group previously inserted in front of this AFP page.

The following example illustrates how to invoke a copy group defined in a form definition directly:

FORMDEF F1LCD defined in MakeAFP definition file:

```
restype=all,inline
fdef=F1LCD                    // The FORMDEF object is compiled from
fdeflib=c:\makeafp\reslib     // the following PPFA source code
ovlylib=c:\makeafp\reslib
pseglib=c:\makeafp\reslib
fontlib=c:\makeafp\reslib
         :
```

PPFA source code for form definition F1LCD:

```
FORMDEF LCD                   // this form definition defined two
  PRESENT LANDSCAPE           // copygroups, first one for select tray
  DIRECTION ACROSS            // 1 for duplex, second one for select
  N_UP 1                      // tray 2 for simplex
  OFFSET 0 0
  REPLACE YES;

   COPYGROUP F2LCD1
     BIN 1                    // input paper bin is 1
     DUPLEX NORMAL;           // duplex printing

   COPYGROUP F2LCS2
     BIN 2                    // input paper bin is 2
     DUPLEX NO;               // simplex printing
          :
```

MakeAFP Weaver program source code:

```
void main( )
{
  Start();
  OpenDoc();
  SetUnit(MM_U600);
    CopyGroup("F2LCS2");        // Subsequent pages will be printed on
                                // the paper from bin 2 in simplex mode
                                // defined by copy group F2LCS2
    GetPage(true);              // Remove current copy-group form input AFP
          :
    ClosePage();
          :
    GetPage(true);              // Remove current copy-group form input AFP
          :
    ClosePage();
          :
    CopyGroup("F2LCD1");        // Subsequent pages will be printed on
                                // the paper from bin 1 in duplex mode
                                // defined by copy group F2LCD1
    GetPage(true);              // Remove current copy-group form input AFP

          :
```

```
  ClosePage();
         :
  OpenPage(210, 297);          // Add a new AFP page
         :
  ClosePage();
CloseDoc();
```