Chapter 1. Using Form Definitions and Page Definitions

The use of form definitions and page definitions to format line data for an AFP application originated in the mainframe environment around 1983. Users of Infoprint Manager for AIX and Infoprint Manager for Windows NT and 2000 commonly use these objects. The AS/400 system has supported data streams that used these objects since V2R1 and has been able to generate them natively since V3R2/V3R7.

Before V5R1, few customers on the AS/400 system used form definitions and page definitions to format their data. There were many other options for output formatting on the AS/400 system, such as DDS and APU, which could essentially generate comparable results. The original tool for generating form definitions and page definitions on the AS/400 system, Page Printer Formatting Aid (PPFA), required a reasonably high skill level. Prior to V5R1 you could not send spooled files using page definitions to an ASCII printer using Host Print Transform.

These limitations are addressed by a number of enhancements and new products made available with V5R1. This chapter describes these changes and how they affect the use of form definitions and page definitions on the iSeries server. Basic concepts and implementation considerations for the iSeries server are also covered.

This chapter provides a very high level view of form definition and page definition processing. For additional detailed information on any of the topics described in this chapter, refer to *IBM Page Printer Formatting Aid*, S544-5284.

1.1 Form definition and page definition basic concepts.

Form definitions and page definition objects are used to enhance the formatting of line data in an AFP environment. They provide similar, but not identical, functions to the iSeries concept of using DDS and printer files to format print applications.

1.1.1 Form definition basics

The form definition contains the specifications that deal primarily with the physical attributes of the printed output. It controls page origin and orientation, overlays, copies, paper source and output bin, simplex or duplex, print quality, and multi-up. The form definition also determines whether selected fields from the input data are suppressed.

Within a form definition, you may specify one or more copy groups, and, within the copy groups, there may be specifications for one or more subgroups. These are described in the Infoprint Designer chapter.

1.1.1.1 Copy group

A *copy group* defines the current physical page within a form definition. Note that it defines the physical page and could therefore include the formatting instructions for two sides of a piece of paper. A form definition might contain only one copy group. If a different copy group is invoked, it follows that printing must start on a new physical page. An alias for copy group is *medium map*.

Common purposes for a copy group include the ability to pick from a different drawer on the printer. Another example is a switch between simplex and duplex. A final example is the printing of an overlay of standard terms and conditions on the reverse side of client copies of a purchase order. Switching between copy groups in the form definition is controlled by conditional processing in the page definition.

1.1.1.2 Subgroup

A subgroup is a subset of a copy group. Up to 127 subgroups can be constructed within any one copy group. The easiest way to remember the function of the subgroup is to think of multipart stationery (top copy, pink copy, blue copy, etc.), as commonly used on impact printers. To replicate this electronically we can use multiple subgroups to repeat each printed page as required. The difference is that each copy may be altered subtly if required.

The Infoprint Designer Chapter describes a sample application called BACK2 that specifies that each page of the original spooled file is to be printed twice. The first copy uses an Invoice overlay with a constant back containing the Terms and Conditions overlay. The second copy is drawn from a different paper drawer and uses the Packing Slip overlay. Figure 1 on page 4 illustrates the form definition source you must generate if you are coding that application using PPFA (rather than using the Infoprint Designer interface).

```
SETUNITS  10.00 CPI  6.00 LPI LINESP  6.00 LPI ;
FORMDEF BACK2
      OFFSET  0.00 MM   0.00 MM
      REPLACE YES
      COMMENT '';
 SUPPRESSION DFLT;
 SUPPRESSION PACK;
 COPYGROUP COPY1
      DUPLEX NORMAL
      CONSTANT BACK;
  OVERLAY TERMS TERMS NORASTER;
  OVERLAY PACK PACK NORASTER;
  OVERLAY INVOIC INVOIC ;
  SUBGROUP FRONT BIN 1 OVERLAY INVOIC SUPPRESSION DFLT;    /*Invoice*/
  SUBGROUP BACK OVERLAY TERMS;    /*T's & C's*/
  SUBGROUP FRONT BIN 2 OVERLAY PACK SUPPRESSION DFLT PACK; /*Packing*/
  SUBGROUP BACK BIN 2;     /*Blank */
```

Figure 1. Sample PPFA source for a form definition

A form definition is required for all AFP printing. On the iSeries server, if a form definition is not specified explicitly, one is built into the spooled file based on the printer file specifications. Form definitions are used with any printer file type, except one that is used to generate *USERASCII. The most common use of custom or user generated form definitions is with printer files that are designated as *LINE, and, in most cases, they are used along with a page definition.

1.1.2 Page definition basics

A page definition contains the specifications for the logical page layout. You can specify the positioning, rotation, suppression, and font of entire lines of data or specific fields within a line. Data fields may be printed as bar codes. Additional overlays and page segments may be placed on the page as determined by instructions in the page definition.

A page definition must have at least one page format that contains at least one subpage. The subpage contains format instructions for the line and field definitions.

There are two ways to define the mapping within a page definition:

- Traditional line data processing
- Record format line data processing

1.1.2.1 Traditional line data processing

Until recently, traditional line data processing has been the only way to define the formatting rules in a page definition. The input data must be presented in a fairly structured and predictable manner. For example, if a certain piece of information must be printed in a certain way (with a specific font at a specific location) it must be on the same relative line and position on each page in the original spooled file.

The basic element of formatting is the PRINTLINE command. Data records from the input are processed in sequence with the formatting rules in the page definition's PRINTLINE commands. Groups of lines may be processed identically by using the REPEAT subcommand.

The logic may be altered slightly by using Channel codes. For example, the character "1" in the first column is usually an indication to start a new page.

A simple Page Definition is shown in Figure 2. It was designed to simulate PAGRTT(*COR) or (*AUTO) which is commonly used on the AS/400 or iSeries server for system printing. Note the use of the CHANNEL and REPEAT subcommands in the PRINTLINE command.

```
SETUNITS 1 IN 1 IN
         LINESP 8.8 LPI;

PAGEDEF STD132
   REPLACE YES
   WIDTH 10 IN HEIGHT 7.5  IN
   DIRECTION DOWN;

FONT CR13 CS 420090 CP V10500;

PRINTLINE CHANNEL 1 REPEAT 66
   FONT CR13
   POSITION MARGIN TOP;
   ENDSUBPAGE;
```

Figure 2. Simple page definition

Fields within a line may be formatted individually. They are identified within the print record by the START and LENGTH subcommands. If a font is specified for the field, it overrides the font specified in the PRINTLINE. The placement of the field, if specified, is relative to the placement of the PRINTLINE.

Figure 3 illustrates how a PRINTLINE command can have individual fields formatted using the FIELD command. The first PRINTLINE is used to format the first line of data. The second PRINTLINE is used for the next five lines of data.

```
PRINTLINE POSITION 1 IN 1 IN FONT TNR12;
   FIELD START 1 LENGTH 4;
   FIELD START 11 LENGTH 4
      POSITION 4 IN 3 IN
      FONT HELV10;

PRINTLINE POSITION 3 IN 4 IN FONT HELV10 REPEAT 5;
      FIELD START 3 LENGTH 7 FONT TNR12;
      FIELD START 10 LENGTH 4;
```

Figure 3. FIELD formatting within a PRINTLINE

1.1.2.2 Record format line data processing

Traditional line data processing works well enough for users migrating from impact to laser since the data likely prints in predictable places on any pre-printed forms that are in use at the time. However, it has not proven to be flexible enough to meet the needs of some customers who require more control in determining exactly how the data should be printed.

Record format line data processing was developed for page definitions recently to include functions required by customers with more complex print requirements, such as financial statements for clients with multiple account types or telephone bills with multiple types of charges.

Record format line data processing includes features for:

- Automatic page management:
  - Headers, footers, page numbering
  - Grouped item overflow
  - Subheadings
  - Page overflow
- Simple Graphics:
  - Lines, boxes, circles, ellipses
  - Filled areas
- Text enhancements:
  - Delimited fields on input
  - Right and left alignment on output

The main difference is that the basic element of formatting has changed from the PRINTLINE command to the LAYOUT command. Each record of data must have a 10 byte Record ID that matches up with the corresponding LAYOUT command that is used to format it. (With V5R1 the DDS record name will be mapped automatically to the 10-byte record format name. More on DDS in 1.2.5, “DDS support for record format line data” on page 18.)The sequence of the Layout commands in the page definition is not a consideration.

There are different layout types:

Body This is used for most data, such as detail lines. A page definition may have multiple body layouts for different detail record types, such as local or long distance phone charges.

Page Header This is automatically printed on each page. You may use a default header or define your own.

Page Trailer Similar to a page header, this prints at the bottom of each page.

Group Header Group Headers precede a group of Body records. They may continue on the next page if necessary. They are in effect until a new Body Group is started or NOGROUP is specified.

Another major enhancement that is available with record format line data processing is the ability to use variable length fields in the data, separated by a character defined with the DELIMITER subcommand. The individual fields are selected by the FLDNUM subcommand rather than by the start position and length. In addition, selected characters within delimited fields may be selected by specifying the START and LENGTH with respect to the delimiter. This function may prove useful when dealing with languages that are more adept at string manipulation, such as Java.

Figure 4 on page 7 illustrates examples that use the LAYOUT command for record formatting.

```
Sample LAYOUT and FIELD Commands for Body, no delimiter characters:

LAYOUT C'layout1' BODY GROUP
   DIRECTION ACROSS
   COLOR CYAN
   OVERLAY MYOVL 0.5 0.25 OVROTATE 90;
   FIELD START 1 LENGTH 20
         ALIGN LEFT
         POSITION 0 CURRENT
         FONT FONT1;

Sample LAYOUT and FIELD Commands for BODY using field delimiter:

LAYOUT C'layout2' BODY DELIMITER C'%';
   FIELD FLDNUM 1
      POSITION 0 NEXT
      FONT FONT2;


Sample LAYOUT Command for GROUP HEADER:
LAYOUT C'headr1' GRPHEADER XSPACE .2
   IN POSITION SAME .6 IN;

Sample Layout Command for PAGE HEADER:
LAYOUT C'pghd' PAGEHEADER NEWPAGE
   POSITION .6 IN ABSOLUTE .55 IN;
```

Figure 4. Sample commands for record formatting

There are many other functions supported when record formatting line data processing is used that are not supported with traditional line data processing:

- Field delimiter and subfield
- Automatic page numbering
- Automatic page overflow management
- Simple graphics: lines, boxes, circles, ellipses
- Boxes or lines that enclose a variable number of input lines
- Right or left alignment of text fields on output
- Conditional processing testing of substrings of delimited fields

Figure 5 on page 8 shows some examples of commands for the new graphical features available with record format line data processing.

```
Draw four identical lines, 0.25 in apart.
DRAWGRAPHIC LINE POSITION LPOS CPOS +1.0 DOWN 3.0 IN
         LINETYPE DASHDOT LINEWT MEDIUM
         COLOR ORANGE COPY ACROSS 4 SPACED 0.25;

Circle, filled with dotted pattern.
DRAWGRAPHIC CIRCLE POSITION LPOS NEXT
         RADIUS 1.0 IN LINEWT MEDIUM LINETYPE SOLID
         FILL DOT04;

Ellipse
DRAWGRAPHIC ELLIPSE POSITION LPOS NEXT
         AXIS1 -2 -2 AXIS2 +2 +2 LINEWT MEDIUM
         LINETYPE SOLID;

Using ENDGRAPHIC to make variable size box.
DRAWGRAPHIC BOX GRAPHID 01
         BOXSIZE 3 IN POSITION 1.75 IN .3 IN;

ENDGRAPHIC GRAPHID 01 LPOS;
```

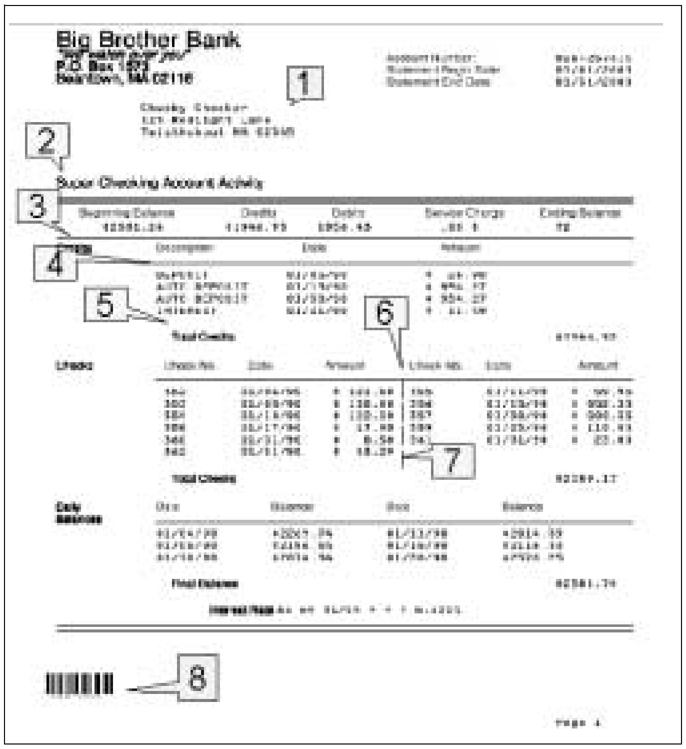
Figure 5. Examples of commands for new graphic features

Note

Neither traditional line data processing or record format line data processing provides a means to edit the data in a similar fashion to the way DDS and some other iSeries high level languages such as RPG use Edit Words or Edit Codes. Currency, punctuation or date diameter characters must be included in the original data being passed to the page definition.

An example of output created using record format line data processing is shown in Figure 6 on page 9. It illustrates the following features of record formatting:

1. Page Header *statmid* appears on each page
2. Body layout *statsum* produces the summary information. Horizontal lines are created by using the DRAWGRAPHIC command.
3. Group header *crheader* produces a “Credits” subheading. It is repeated if the group spans a page.
4. Body group *crdata* contains the credit detail lines. The amount column is right-aligned.
5. Body group *crtotal* prints the total credit information. The data is passed from the application. It is not calculated automatically.
6. A variable length vertical line in the Checks group is defined by the DRAWGRAPHIC LINE DOWN command.
7. ENDGRAPHIC commands end the vertical line.
8. Page trailer *pgenum* ends the page, prints the page number, and prints a bar code. It is printed on each page.



Big Brother Bank

P.O. Box [illegible]

Beantown, MA 02116

Super Checking Account Activity

Total Credits

Checks

Total Checks

Daily Balances

Final Balance

Interest Rate

Figure 6. Output example produced using record formatting

For a complete listing of a page definition and corresponding data for a similar example, refer to the *IBM Page Printer Formatting Aid*, S544-5284.

Note:

You cannot mix LAYOUT commands and PRINTLINE commands in the same page definition.

1.1.3 Conditional processing

You can change the formatting of different pages of your output based on rules that you define in the page definition. This is called conditional processing. Based on data that the page definition finds in a predictable location, it can choose to use a different set of rules for the physical aspect of the job, by selecting different copy groups from the form definition. For example, if the customer number begins with the letter “E”, use the copygroup that prints an English overlay; if it begins with an “F”, use the copygroup that prints a French overlay. Similarly, you may use conditional processing to select a different layout for the data, or page format, from the page definition. En example of this function is printing the data from a total or summary page differently from the way the detail pages print.

Conditional processing may also be used to force printing on a new side or a new sheet of paper in multi-up or duplex applications cases.

The Infoprint Designer Chapter describes a scenario where a different page format is used depending on the information that prints in the total box on the form. If it has the word ‘Continued’, you should use the page format called

‘CONTD’. Otherwise, you should use the TOTAL page format. With Infoprint Designer, the set up of the condition is done interactively.

The equivalent page definition commands you would enter using PPFA are shown in Figure 7.

```
CONDITION TOTALBOX START 70 LENGTH 9
     WHEN EQ
      'Continued' BEFORE SUBPAGE
      CURRENT
      PAGEFORMAT CONTD
     OTHERWISE BEFORE SUBPAGE
      CURRENT
      PAGEFORMAT TOTAL;
```

Figure 7. Sample conditional processing command

If the copy group from the form definition is affected by the results of the condition test, the word “CURRENT” is replaced by a COPYGROUP subcommand. Otherwise, the word CURRENT simply instructs you to keep using the same copy group that was in effect.

There is a slight difference in the syntax for the CONDITION command when it is used with Traditional Line Formatting versus Record Formatting. However, the overall concept and functions are similar.

1.1.4 Line data basics

The origin of line data goes back to impact printers that printed one line of information at a time. A physical tape with with holes in specific positions or channels was used to control spacing and skipping.

On the iSeries server, the term *line data* specifically refers to spooled files that have DEVTYPE(*LINE) specified. The data may look similar to the default data that is generated when one uses DEVTYPE(*SCS), but SCS supports some imbedded formatting instructions, such as font, cpi, and variable line spacing, where LINE does not.

Usually, LINE spooled files are generated with an extra byte at the beginning for the carriage control or forms control byte. In page definition terms, this is referred to as the Channel code.

The extra byte for the Channel code is mandatory if you are imbedding selected AFP instructions within the LINE data. The AFP instructions must begin with a Hex’5A’. An example of such an instruction is an Invoke Media Map to change the copy group used within a form definition. Channel codes are ignored if you are using record format line data processing, but may be necessary if you include AFP instructions in the data.

Fonts are usually defined by using the FONT subcommand in a PRINTLINE command. An alternate way is to add one additional character (after the forms control byte) to use as a font selection code. Or, in page definition terms, the character can be used as a Table Reference Character (TRC). TRCs are only supported using traditional line data processing.

In the case of record format line data processing, the first 10 characters of the record (after the carriage control byte if one exists) must contain the Record ID. The data following the Record ID may be the traditional format where the fields are determined based on their position in the line. An alternative is to separate the fields by a delimiter character.

Figure 8 shows examples of different ways to generate line data for traditional and record format processing.

```
Traditional Line Data with CC
1This is the top of the page
 Field1     Field2          Field3

Record Format data with optional diameter characters
pagehead  This is the top of the page
detail    Field1%Field2%Field3
```

Figure 8. Sample line data

If the data is referenced by FIELD, the START position in the page definition does not include the channel codes or TRCs, in the case of traditional line data processing, or the 10 character record ID used with the record format line data processing.

Version 5 Release 1 provides two new methods to easily produce record format line data. Section 1.2.4, “Java support for record format line data” on page 17 and the Printing from Java applications chapter describe how to produce record format line data spooled file from a java application program. Section 1.2.5, “DDS support for record format line data” on page 18 describes the new DDS support for record format line data.

1.1.4.1 Imbedding AFP instructions

One method of enhancing the output produced by form definition and page definition processing is to insert Advanced Function Presentation (AFP) instructions in the form of structured field records right into the data stream. This may be done with either type of formatting, traditional line data or record format line data.

This allows the application programmer to have more flexibility than is otherwise offered by the relatively constrained rules of Conditional Processing.

A spooled file that contains both data records and AFP structured fields is known as mixed. On the iSeries server, the printer file would be generated as DEVTYPE(*AFPDSLINE).

The AFP structured fields that may be imbedded in the data stream are:

- Invoke Media Map: Used to change copy group
- Invoke Data Map: Used to change page format
- Include Page Segment
- Include Page Overlay
- Presentation Text

If you are building the data stream manually, the data records must be preceded by a carriage control byte and the AFP records must begin with X’5A’. For more information on mixed data streams, refer to *Advanced Function Presentation Programming Guide and Line Data Reference*, S544-3884.

If you are using DDS to create your spooled data, the INVMMAP, Invoke Media Map, and INVDTAMAP (Invoke Data Map) keywords can be used to insert AFP structured fields in an AFPDSLINE data stream.

1.1.5 iSeries printer file keywords

When you create or change a printer file for use with form definitions and page definitions, you must consider the following parameters in the CRTPRTF or CHGPRTF Commands.

DEVTYPE
The most common way to take advantage of form definition and page definition processing is to use a printer file defined with DEVTYPE(*LINE), FORMDF(mylib/myformdf), and PAGDFN(mylib/mypagdfn).

Page definitions may also be used with printer files defined as DEVTYPE(*AFPDSLINE) if imbedded AFP structured fields are to be used.

You may specify a form definition for use with spooled files that have a DEVTYPE value of *LINE, *AFPDSLINE, *AFPDS, *SCS, or *IPDS.

FORMDF and PAGDFN
It is most common to include both parameters in the printer file. They may be library qualified. If *LIBL is used, they must be placed in libraries that PSF/400 can find for the writer, either by the job library list, a default library list, or libraries specified in the PSF Configuration Object.

If either the FORMDF or PAGDFN parameters are set to *NONE, and you specify DEVTYPE(*LINE) for printing to an AFP printer, PSF/400 builds the required object inline based on the appropriate printer file parameters.

CTLCHR
This parameter tells PSF if you are using control characters in your data, and, if so, what type.

If your application uses DDS keywords, such as SKIPA or SPACEB, or other similar high level language instructions to control skipping and spacing, specify CTLCHAR(*NONE). If the application data itself includes the control characters, you must specify CTLCHAR(*FCFC) for ANSI control characters or CTLCHAR(*MACHINE).

TBLREFCHR
If the spooled data includes a table reference character to control font selection, specify TBLREFCHR(*YES).

AFPCHARS
Use this keyword to identify up to four coded fonts to use if you select TBLREFCHR(*YES). The names to use here are the 4-character coded font names. PSF/400 adds ‘X0’ to the beginning of the name to find the font resource object.

DRAWER and DUPLEX
In most cases, the form definition and page definition parameters override equivalent functions specified in the printer file. Two exceptions to this rule are the DRAWER and DUPLEX parameters. The value specified for these parameters takes precedence over the form definition specification. To force PSF/400 to use the values in the form definition, you must specify DRAWER(*FORMDF) and DUPLEX(*FORMDF).

CVTLINDTA
This parameter specifies whether line data and a page definition should be converted to AFPDS before the data is spooled. See 1.2.6, “CVTLINDTA parameter in printer files” on page 21 for further information on using this parameter.

Other print file keywords
The following printer file keywords are *ignored* when line data is specified and a page definition is used:

- CDEFNT
- CHRID
- CPI
- FOLD
- FONT
- FNTCHRSET
- LPI
- LVLCHK
- MULTIUP
- PAGESIZE
- PAGRTT
- REDUCE

The following printer file keywords are **ignored** when line data is specified and a form definition is used:

- BACKMGN
- DRAWER (if *FORMDF is specified)
- DUPLEX (if *FORMDF is specified)
- FOLD
- FORMFEED
- FRONTMGN
- LVLCHK
- MULTIUP
- PAGRTT
- PRTQLTY
- REDUCE
- SADLSTITCH

1.1.6 Converting existing SCS spooled files to LINE
If you have an application that already produces SCS output, and you want to migrate to the use of form definitions and page definitions, it is best if you can change the printer file in the application or override it in the control language just before it is used.

If this is not possible, you may be forced to work with the SCS output after it has already been generated as a spooled file. Perform the following steps to convert an SCS spooled file to a LINE spooled file:

1. Create a physical file that has a record length of one character longer than the length of the spooled file records. This accommodates the forms control character that is used to preserve the line spacing (otherwise, all blank lines are ignored or dropped):

   ```
   CRTPF CPYSPLF RCDLEN(133) MAXMBRS(*NOMAX)
   ```

2. Copy the spooled file data to the physical file using the CPYSPLF command:

   ```
   CPYSPLF FILE(QSYSPRT)
            TOFILE(CPYSPLF)
            JOB(073040/MSHNIER/QPADEV000C)
            SPLNBR(2)
            TOMBR(QSYSPRT)
   ```

3. Create a new printer file for LINE printing. Name the page definition and form definition and set any other parameters as necessary. You must specify CTLCHAR(*FCFC) for the codes in column one to be interpreted as carriage control codes, and not as data:

   ```
   CRTPRTF FILE(mylineprtf)
            DEVTYPE(*LINE)
            CTLCHAR(*FCFC)
            DRAWER(*FORMDF)
            PAGDFN(pagedef)
            FORMDF(formdef)
            DUPLEX(*FORMDF)
            CVTLINDTA(*YES)
   ```

4. Copy the member from the physical file to the printer file. This generates a spooled file, either as DEVTYPE(*LINE) if you specify CVTLINDTA(*NO), or, if you specify CVTLINDTA(*YES), the page definition instructions are processed and the spooled file is generated as DEVTYPE(*AFPDS).

   ```
   CPYF FROMFILE(CPYSPLF) TOFILE(mylineprtf) FROMMBR(QSYSPRT)
   ```

   This generates a spooled file, either as DEVTYPE(*LINE) if you specify CVTLINDTA(*NO), or, if you specify CVTLINDTA(*YES), the page definition instructions are processed and the spooled file is generated as DEVTYPE(*AFPDS).

This process may be automated by a monitor program. For information, see the Output Queue Monitor Appendix.

1.1.7 Generating or obtaining form definitions and page definitions

Form definitions and page definitions may be obtained from a variety of sources on the iSeries server or from other platforms.

1.1.7.1 Shipped with PSF/400

A standard set of form definitions and page definitions are shipped with the PSF/400 feature. These are compatible with resources that are shipped on other IBM platforms with PSF or Infoprint Manager products. They are listed in the *AS/400 Guide to Advanced Function Presentation and Print Services Facility*, S544-5319.

1.1.7.2 Infoprint Designer for iSeries

Infoprint Designer for iSeries generates the form definition, page definition, and other resource objects used in the project and automatically places them directly in the iSeries library. No additional processing is necessary other than referencing them in the printer file.

For more information, see 1.2.1, “Form and page definitions in Infoprint Designer for iSeries” on page 16, and the Infoprint Designer for iSeries chapter.

1.1.7.3 AFP PrintSuite: Page Printer Formatting Aid/400

Page Printer Formatting Aid/400 (PPFA) is a tool for the iSeries server that takes the source form definition and page definition specifications and converts them to the corresponding AFP data stream.

To use PPFA, use an editor such as SEU to enter the source into a source physical file member. Then, use the PPFA CVTPPFTSRC command to generate the physical file members that contain the AFP versions of the form definition and page definition. Finally, use the CRTFORMDF and CRTPAGDFN command to build the iSeries objects.

You must apply PTF SF65783 to the PPFA product to be able to create page definitions that use the new Record Format Line Data features.

1.1.7.4 Resources from AIX or Windows platforms

Infoprint Manager for AIX and Infoprint Manager for Windows NT and Windows 2000 offer PPFA as optional features. There are a number of other vendor offerings also available for those platforms that use a WYSIWYG interface (similar to Infoprint Designer for iSeries) to create form definitions and page definitions.

The easiest way to import resources from a Windows platform is to use the AFP Manager that is shipped with Client Access Express with V5R1. See the AFP Manager chapter for more details.

If the files were created on an AIX system, it may be best to transfer them in binary format to a Windows PC (for example, by using FTP), and then use AFP Manager.

Otherwise, you must follow these steps to build the resources on the iSeries server:

1. Create a physical file on the iSeries server. It may have any record length you want, but you need to specify `LVLCHK(*NO)`.
2. Use the TCP/IP FTP function to transfer the resources into members of the physical file you just created. The files must be transferred in BINARY mode. Otherwise, the iSeries server attempts to perform an ASCII to EBCIC conversion on the data.
3. Once the resource data has been loaded in the physical file member, use the CRTFORMDF and CRTPAGDFN commands to create the form definition and page definition iSeries objects.

1.1.7.5 Resources from OS/390

Form definitions and page definitions can be created on the OS/390 platform using PPFA or another vendor’s products. Resources from OS/390 are in files consisting of variable length records. The iSeries server cannot handle this format directly. You must first inspect the OS/390 file to determine the length of the single longest record in the resource and then create the iSeries physical file with that length. You also need to specify `LVLCHK(*NO)`.

Once the data is loaded in the physical file use the CRTFORMDF and CRTPAGDFN command to create the iSeries resources.

1.2 Summary of V5R1 enhancements relating to form and page definitions

With some of the changes being announced in V5R1, it is expected that more customers will begin to look at form definitions and page definitions as candidates for formatting their data. The changes that affect this decision include:

- The new formatting tool, Infoprint Designer for iSeries, generates form definitions and page definitions.
- Infoprint Server for iSeries can generate index records and resource groups for spooled files in *LINE format that have a page definition specified. The index records and resources can then be used:
  - By Infoprint Server to split large spooled file into multiple PDF files
  - When viewing the file with the AFP viewer to quickly locate specific pages in the document
  - By archive products, such as On Demand or Common Server
- Enhancements to the page definition architecture to support record format line data.
- A new access class for Java to support the generation record format line data on the iSeries server.
- Changes to DDS to support generating record format line data.
- The CVTLINDTA parameter can be specified in a printer files that has DEVTYPE(*LINE) and uses a page definition. If you specify CVTLINDTA(*YES) the spooled file that is created when the application is run will be generated as fully resolved AFPDS, instead of line data. This means that applications that are developed using Infoprint Designer for iSeries, or with PPRA can now be:
  - Printed on an ASCII printer using Host Print Transform.
  - Viewed using the IBM AFP Viewer from the Client Access Navigator interface.

1.2.1 Form and page definitions in Infoprint Designer for iSeries

Infoprint Designer for iSeries is available for V4R5 and V5R1. It provides a WYSIWYG interface for creating overlays and generating form definitions and page definitions. It is described in great depth in the Infoprint Designer for iSeries Chapter. There are a some features, commands, and subcommands that are part of the architecture for form definitions and page definitions that are not supported today in Infoprint Designer for iSeries. These include:

- Record format line data
- Use of Table Reference Characters (TRCs) to select fonts
- PRTDATA subcommand in the PRINTLINE command

- Form definition subcommands dealing with finishing (stapling) and post processing equipment

Nevertheless, Infoprint Designer is likely to be the choice of many iSeries users since the WYSIWYG design lends itself to very fast application development.

1.2.2 Using Infoprint Server for iSeries

Infoprint Server for iSeries consists of a number of powerful tools for manipulating spooled file data and transforming from one data stream to another. One of the tools is a command called Create AFP Data, CRTAFPDTA. This command requires a spooled file in LINE format and an input page definition.

The output of CRTAFPDTA consists of up to four files in the integrated file system directory (IFS). The first is an AFPDS file that is a result of processing the LINE data against the page definition. The other optional files are a file containing index records with one containing copies of selected external resources and one that merges the data from the other three.

The index function can be useful when used with another component of Infoprint Server (the IPDS to PDF transform). These index records can be used to segment a large spooled file to individual components, such as single invoices or statements. This whole process is described in the End-to-End Example chapter.
.

The index records and resource groups may also prove useful when working with archive products, such as On Demand or Common Server. By including the resource group with the spooled file, you are ensured that the correct external resources, such as overlays or page segments, are being used with an old spooled file when it is retrieved.

Finally, the AFP viewer can take advantage of the index records and resource groups when you view the spooled file.

1.2.3 Record format line data processing

The underlying support for Record format line data processing has been added to PSF/400 with V5R1. It is described earlier in this chapter in 1.1.2.2, “Record format line data processing” on page 6.

Resources created on other platforms that use this function are supported on the iSeries with V5R1. Record format line data page definitions may be generated natively on the iSeries server with the PPFA/400 component of the AFP PrintSuite, 5798-AF3 (by applying PTF SF65783).

1.2.4 Java support for record format line data

In the past, the support for print from Java applications on the AS/400 system was limited and cumbersome.

With V5R1, a new class object has been introduced to the IBM Toolbox for Java that supports writing of data from Java to a printer file in LINE format. LineDataRecordWriter class allows a Java programmer to concentrate on generating the data, and the formatting can be done using form definitions and page definitions as described in this chapter.

LineDataRecordWriter writes a record in a line data format, including the record format name in positions 1 through 10. The data may be presented using a fixed layout or a variable layout may be used that uses a delimiter to separate fields.

For more details, see the Printing from Java applications Chapter.

1.2.5 DDS support for record format line data

Data Description Specifications (DDS) provides a means to define the layout and presentment of data external from the program using it. Since 1991, keywords to support AFP functions have been supported for printer files. These includes lines, boxes, host fonts resources, graphics, page segments and overlays. These elements can be placed at absolute positions on the page, or the positional values can be passed as parameters from the program.

This capability provides a great deal of flexibility to a programmer. Theoretically, the checking account document as illustrated in Figure 6 on page 9 could be produced using a high level language such as RPG, COBOL, or C. One of the challenges is that the programmer must keep track of the exact positions of any floating elements that are placed on the page, such as any graphics used to separate the different record types. These positions are passed to DDS as variables.

With the new record format line data support for DDS applications, a programmer can continue to keep the field definitions external to the program, but the logic for positioning the data and other AFP elements can be handled by the page definition. Another advantage of using DDS with PPFA record formatting is the fact that DDS supports the EDTCOD and EDTWRD, whereas PPFA does not.

1.2.5.1 How it works

When an applications writes to a printer file that is defined with DEVTYPE(*LINE) the OS/400 checks to see if the page definition that is named in the PAGDFN parameter uses record format line data commands. If it does, the format of the data written to the spooled file is modified to suite the requirements of this type of page definition:

- The record format name used in the DDS specifications is written in the first 10 characters of the output record.
- The individual fields are placed one after another in the printer file record. All data fields follow, starting in position 11. All specifications relating to position are ignored. This includes the Line and Position fields in columns 39-44, the SKIPB, SKIPA, SPACEB, and SPACEA keywords, and the POSITION keyword.

For example, consider an application that normally writes an address block to one record in the DDS, with the company name, street and city appearing on three different lines. With record format line data processing, all three fields are placed in one record in the output spooled file.

An example of some DDS specifications for a check record is shown in Figure 9. The DDS compiler requires that a value be entered in positions 42-44 of the specifications. This value would normally represent an absolute or relative position on the line. It is ignored by record format line data processing.

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7
     A          R CKDATA                     SPACEB(1)
     A                                      1' '
     A            CHECK         12          2
     A            DATE           6  0      40EDTCDE(Y)
     A                                     +0'  '
     A            AMOUNT         6  2      10EDTWRD('$   0.  ') SPACEB(1)
     A                                     +0'   '
     A            PAYTO         25         +5
```

Figure 9. Example of DDS to be used record format line data page definition

This sample contains some blank spaces inserted as constants. This is done to match the data up with the required positions as defined in a page definition that already existed. Any spaces that are required to match up the data with the page definition layout must be hard coded because the normal way of doing positioning will be ignored.

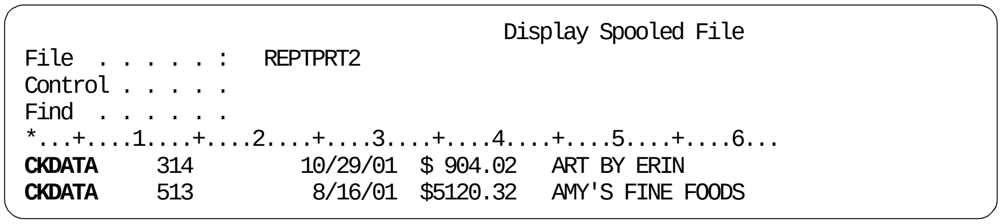
Figure 10 on page 19 shows the page definition LAYOUT command that is used against the CKDATA record. This is a subset of a page definition sample called **rept1** that is found in the *IBM Page Printer Formatting Aid: User's Guide*, S544-5284.

```
LAYOUT C'CKDATA' BODY GROUP;
  FIELD START 2 LENGTH 3 ALIGN LEFT
      POSITION 1.2 in CURRENT
       FONT varb ; /* Variable text - Check number */
  FIELD START 14 LENGTH 8 ALIGN LEFT
      POSITION .1 in CURRENT
       FONT varb ; /* Variable text - Date */
  FIELD START 35 LENGTH 25 ALIGN LEFT
      POSITION 2.0 in CURRENT
       FONT varb ; /* Variable text - Payable to: */
  FIELD START 24 LENGTH 8 ALIGN RIGHT
      POSITION 5.6 in CURRENT
       FONT varb ; /* Variable text - Amount */
```

Figure 10. Page definition layout to use against sample DDS record

The original sample, which was created for OS/390, had lower case record ID's. The sample is changed to use an upper case name for the layout because DDS does not allow for lower case record format names.

Figure 11 shows an example of spooled file records that are generated using the DDS from Figure 9. The DDS keywords EDTCDE and EDTWRD were honored, and the appropriate punctuation was added to the currency field and the date. The SPACEB(1) keywords were ignored, as was the positioning information from column 42-44.



```
                                         Display Spooled File
File  . . . . . :   REPTPRT2
Control . . . . .
Find  . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6...
CKDATA     314         10/29/01  $ 904.02   ART BY ERIN
CKDATA     513          8/16/01  $5120.32   AMY'S FINE FOODS
```

Figure 11. Spooled file after processing with record format line data

Remember that the when using START and LENGTH to define fields for record format line data processing, the start position begins after the 10-character record ID. In this case, the check number is in position 12 of the printed record, but the page definition will reference it as starting in position 2.

The following keywords will be honored by record format line data processing:

- ALIAS
- DATE
- DATFMT
- DATSEP
- DFT
- DLTEDT
- EDTCDE
- EDTWRD
- FLTFIXDEC
- FLTPCN
- IGCALTTYP
- IGCANKCNV
- INDARA
- INDTXT
- MSGCON
- PAGNBR (probably is not going to be of much use)
- REF
- REFFLD
- TEXT
- TIME
- TIMFMT
- TIMSEP

Keywords relating to formatting will be ignored:

- BARCODE
- BLKFOLD
- CDEFNT
- CHRID
- CHRSIZ
- COLOR
- CPI
- CVTDTA
- DFNCHR
- DOCIDXTAG
- DRAWER
- DTASTMCMD

- DUPLEX
- ENDPAG
- ENDPAGGRP
- FNTCHRSET
- FONT
- FORCE
- GDF
- HIGHLIGHT
- INVDTAMAP
- INVMMAP
- LINE
- LPI
- OUTBIN
- OVERLAY
- PAGRTT
- PAGSEG
- POSITION
- PRTQLTY
- SKIPA
- SKIPB
- SPACEA
- SPACEB
- STRPAGGRP
- TRNSPY
- TXTRTT
- UNDERLINE
- ZFOLD

Option indicators in position 7-16 of the DDS specification will be honored. This could pose a problem because subsequent fields on the same line will have their position altered based on whether the indicator is on or off. It is recommended that you force a blank field to print to take up the space. For example, if the amount field from the example above was not included because of an indicator, the Payee field would print in the wrong position. It is the programmers responsibility to insert blanks in the record to compensate for this. An alternative approach would be to insert characters between each field, whether they are included or not, to be used as field delimiter characters.

1.2.6 CVTLINDTA parameter in printer files

If you have a printer file that had DEVTYPE(*LINE), and a page definition is specified, the spooled file that is generated remains with DEVTYPE(*LINE) and it contains only the data. The associated spooled file attributes point to the page definition that was requested. At print time, PSF/400 takes the instructions found in the page definition and formats the data accordingly.

Spooled files that are generated with DEVTYPE(*LINE) cannot normally be sent to ASCII printers using Host Print Transform. They also can’t be viewed using the IBM AFP Viewer, which is available as part of the Client Access Express Navigator.

The CVTLINDTA printer file parameter has been added with V5R1 to bypass these two restrictions. By specifying CVTLINDTA(*YES), the line data is processed immediately against the page definition as the spooled file is generated. The resulting spooled file that is generated contains fully resolved AFPDS. This data may be printed on a PCL or PPDS ASCII printer using Host Print Transform and it may be viewed using the IBM AFP Viewer.