Architectures of Advanced Function Printing

by R. K. deBry
B. G. Platte
C. L. Berinato
J. W. Marlin

Discussed is the use of the capabilities of all-points-addressable laser (page) printers in applications involving pages composed of text, image, and vector data in a device-independent way. Also presented is the ability to describe and print complex documents composed of multiples of such pages. Provision is made for the migration of current line-printer applications to print using these new page printers. Three architectures are described that—along with an Advanced Function Printing (AFP) model—support these capabilities. Each of these architectures is described in the context of the current implementation of the Advanced Function Printing software.

A sound architectural base is fundamental to the success of any major software development. A set of well-designed, carefully specified interfaces is important to the systems developer, who must make the various components of the system communicate with one another. However, if the definition of these interfaces is important to the systems developer, it is even more important to the user of the system, who expects system interfaces to be functional, easy to use, and durable. Historically, the interfaces that have been defined for printing have been straightforward descriptions of line-oriented text data and have been relatively uninteresting. Therefore, the coding of output statements has been looked upon as some of the necessary drudgery of application program development. With the introduction of all-points-addressable laser printers and their ability to print high-quality typographic text, raster-image, and vector graphics, this is no longer true. Print interfaces have quickly become a subject of great interest.

When we were considering the architecture for Advanced Function Printing (AFP), it was clear that a well-defined and articulated set of interfaces was required. Above all else, the architectures of AFP had to provide a stable, durable base for customers to build upon. Thus, one of the major objectives in defining the architectures was to provide a structure that was extensible and would, therefore, allow the architectures to evolve over time to meet new and existing customer requirements. At the same time, these structures would have to be sufficiently flexible to adapt to changing hardware technologies. We can describe the functions initially required of the architecture in two broad categories.

A very important function required to exploit the capabilities of high-function, all-points-addressable printers is a device-independent description of documents that contain mixtures of text, image, and vector data on the same page. Text data may require the use of multiple high-quality typographic fonts, as described by Griffee and Casey[1] in this issue. Provision must be made for laying out pages of the document in any direction and with any orientation. All of this function must be provided in such a way that the printers can operate as efficiently as possible. It is not enough to be able to simply describe pages

composed of text, image, and vector data. Applications must have an interface through which they can build and manage the printing of complex, multipage documents, such as invoices or insurance forms, where some pages are duplexed, others have multiple copies, and so on. These document-related functions are an important element of a printing interface. Where they make sense, these functions must be provided on both fanfold and cut-sheet printers.

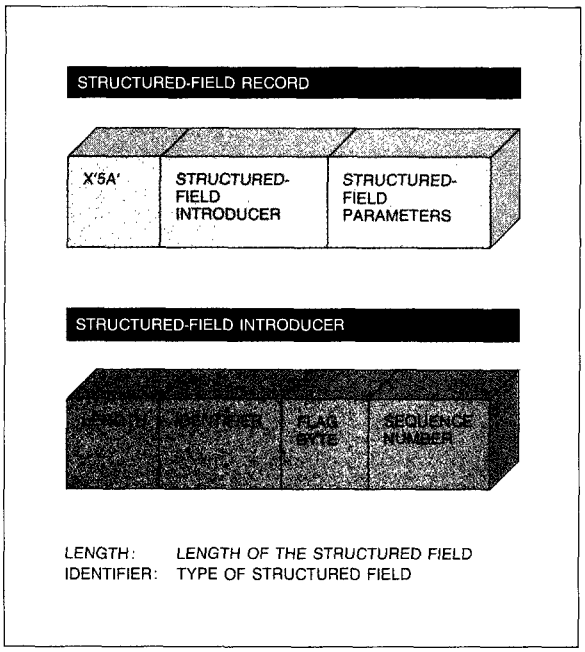
These very important considerations led us to define three distinct interfaces for AFP.

Line-printer data. The first interface is the basic line-printer data stream, as defined for the IBM 1403 printer and its successors. This type of data stream is composed of a set of fixed-length records in which each record is one print line and the first character in each record is an ANSI forms-control character or a machine-control character. This particular interface was chosen to be supported because of the number of IBM customers who use this data stream for large batch-printing applications. Tools have been provided that allow an existing application that generates a line-printer data stream to exploit the functions of Advanced Function Printing through externally defined declarations that are invoked at print time. The line-printer data stream is discussed briefly later in this paper and in the paper by deBry and Platte[2] in this issue.

Advanced Function Printing Data Stream (AFPDS). AFPDS, the architected application interface for AFP, provides a device-independent interface through which applications may describe pages composed of text, image, and graphics data.[3] IBM program products, such as Document Composition Facility (DCF),[4] DisplayWrite/370 (DW/370),[5] and Graphical Data Display Manager (GDDM),[6] generate AFPDS data streams. In addition, there are a number of utility programs, such as Page Printer Formatting Aid (PPFA),[7] and Overlay Generation Language (OGL)[8] that generate data in AFPDS format. Customer-written applications that require the capabilities of Advanced Function Printing should also be written to the AFPDS interface to keep device dependencies out of the application program. The AFPDS data stream is passed to the print-services component of the system, Print Services Facility,[9] where the Intelligent Printer Data Stream, the third architected interface of AFP, is generated.

Intelligent Printer Data Stream (IPDS). IPDS is produced by Print Services Facility and is similar in structure and data content to AFPDS, but has been bound to a specific printer. In IPDS, device-specific resources such as fonts are bound into the data stream, and device-specific error recovery and device control are provided through an IPDS dialogue with the printer. IPDS is independent of the attachment mechanism of the printer. Today, printers exist that support IPDS over System/370 channels, SDLC LU 6.2 sessions, SDLC LU 1 sessions, and BSC links. IPDS is a Systems Application Architecture (SAA) protocol produced not only by Print Services Facility (PSF) but also by GDDM, System/36, System/38, and the IBM PC LAN PrintManager.[10]

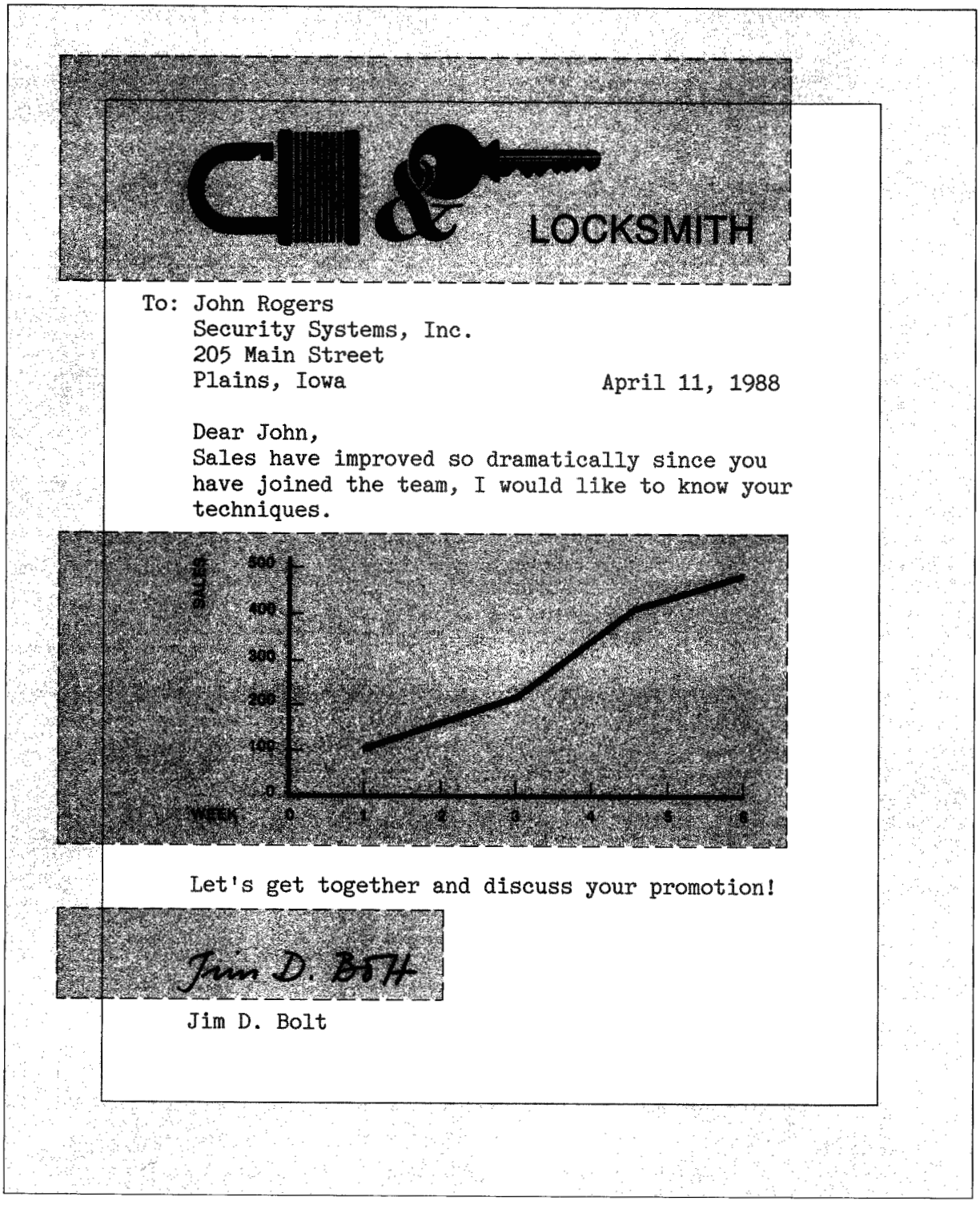
Figure 1 Structured-field definition for AFPDS and IPDS



The structure of AFP architecture

Both Advanced Function Printing Data Stream (AFPDS) and Intelligent Printer Data Stream (IPDS) are structured-field data streams. Structured fields are self-identifying, self-contained data-stream elements of the form shown in Figure 1. The structured field is composed of a length parameter, an ID, and flag bits, optionally followed by the data content of the field. The length parameter defines the length of the structured field, including the length parameter itself, and points to the next structured field in the sequence. The ID parameter defines the content of

Figure 2 Sample composed page



the structured field and indicates to the processor how the rest of the structured field is to be interpreted.

Both AFPDS and IPDS are fully paginated, object-oriented data streams. This concept, which is similar to the Office Document Architecture/Office Document Interchange Format (ODA/ODIF) standard,[11] is described in Reference 12. Consider the sample page shown in Figure 2. Note that the page is composed of elements of text, image, and graphics. These are described in the data streams as discrete objects, each having its own coordinate space onto which the object's data are mapped. The architecture definition of the object is independent of the data stream in which it is carried. Each data stream, whether AFPDS or IPDS, provides a description of the environment and defines the coordinate space on the page into which the objects themselves are placed.

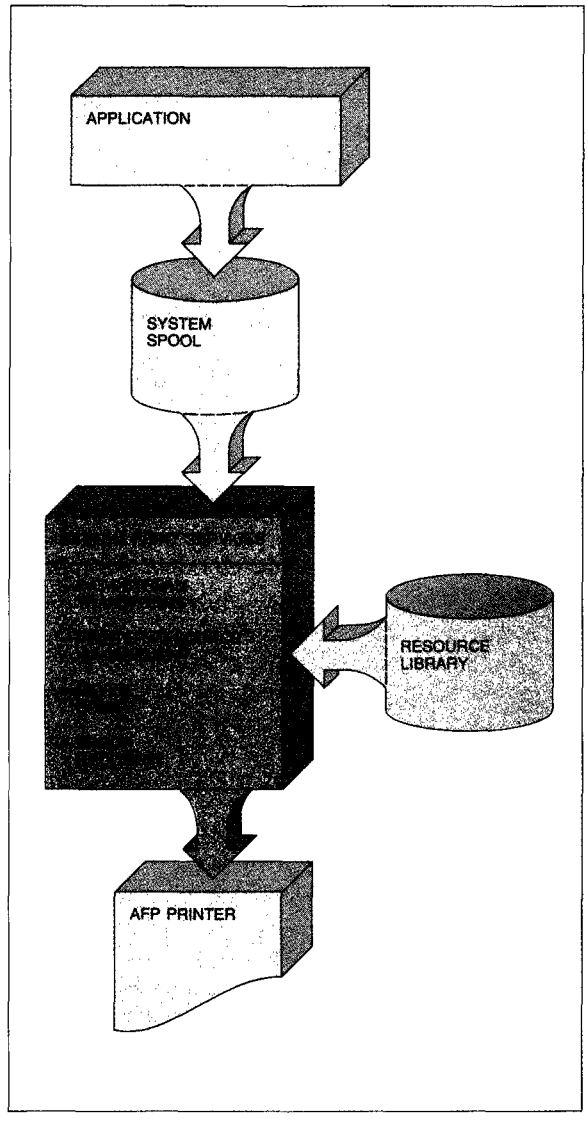
Other common architectural concepts shared between AFPDS and IPDS are the definition of printer resources (fonts, electronic forms), media controls (paper source, duplex), and presentation controls (rotation, positioning). These concepts are described in more detail in the following sections.

Figure 3 shows the AFP print model, based upon the architectures we have just discussed. This structure also illustrates some of the benefits of the AFP architectures. By providing a simple line-printer interface to AFP, existing customer applications can be implemented. The model shown also provides a way for these existing applications to exploit the capabilities of AFP without requiring the application to be rewritten.

The model also clearly separates the application layer from the printer layer. Thus, the application can be a relatively device-independent one and can be freed from the details of resource management, device control, and error recovery. These functions are provided by the print-services component of the system, which uses the more device-oriented IPDS data stream to carry on a two-way dialogue with the printer. Both AFPDS and IPDS provide for the mixing of text, image, and vector data on the same page, with similar data structures and data content. This minimizes the processing required to move from the AFPDS format to the IPDS format and ensures the integrity of the data to be printed.

Both AFPDS and IPDS are fully paginated data streams. As we discuss in later sections of this paper,

Figure 3 AFP model



each page of a document is fully described and carries with it sufficient environmental information to allow for efficient error recovery during the printing process. Fixed page boundaries and the block-structured nature of the data streams also allow certain efficiencies to be introduced into the printing process.

Finally, because the architectures of both AFPDS and IPDS are block-structured and obey strict scoping rules, new data types can easily be introduced. Migration and coexistence problems are minimized

because unrecognized data types can be passed over without destroying the integrity of the rest of the data on a page, and unsupported objects can be transformed where possible. Vector graphics and more sophisticated image-compression techniques are possible candidates for extensions to AFPDS.

Advanced Function Printing Data Stream

AFPDS is the defined application interface to AFP, and, as mentioned earlier in this paper, AFPDS is a fully paginated, object-oriented data stream. These data streams are not bound to a specific printer and are independent of the operating system environment. AFPDS is currently processed in MVS, VM, VSE, and System/36, and there is nothing in the architecture to prohibit its implementation in other key IBM operating systems. AFPDS structured fields are used to define fully composed pages, to map line data to page-printer format, and to describe mixed-line and composed-page data. We now discuss each of these applications.

Each page is a self-contained entity with its own environment.

Composed-page data streams. Figure 4 shows a typical page composed of text, image, and vector graphics data. Along the side of the page are the AFPDS structures used to describe the page. Each page is a self-contained entity, and multiple pages may appear between a begin-document structured field and an end-document structured field. Throughout this example, note the block structure of the data stream. Each object—document, page, data object, etc.—is defined by a BEGIN/END set of structured fields. The architecture includes a well-defined set of scoping rules for structured fields and for the nesting of objects in the data stream. These rules are important, because they provide for extensibility in the data stream and make certain efficiencies possible in the processing and error recovery of the data stream.

Begin document. The begin-document structured field identifies the beginning of a document that is to be processed by the system Print Services Facility (PSF).

Begin page. The begin-page structured field defines the beginning of a new page. Each page is a self-contained entity with its own environment. This page structure provides for more efficient error recovery and allows high-speed printers to pipeline page processing.

Active environment group. The active environment group defines the size of the page and identifies page segments and coded fonts to be loaded into the printing subsystem. Each of these functions is described in its own structured field, which is included within the active environment group. Later in this paper we show that the values coded into the active environment group are mapped into similar IPDS structures to set up the physical printing environment on the printing device. The active environment group has the following appearance:

BEGIN ACTIVE ENVIRONMENT GROUP
- MAP CODED FONT—identifies coded fonts to be used on the page
- MAP PAGE SEGMENT—optional parameter that identifies page segments used
- PAGE DESCRIPTOR—specifies the size of the page
- COMPOSED TEXT CONTROL—constant information for the Print Services Facility (PSF)
- COMPOSED TEXT DESCRIPTOR—text origin and orientation

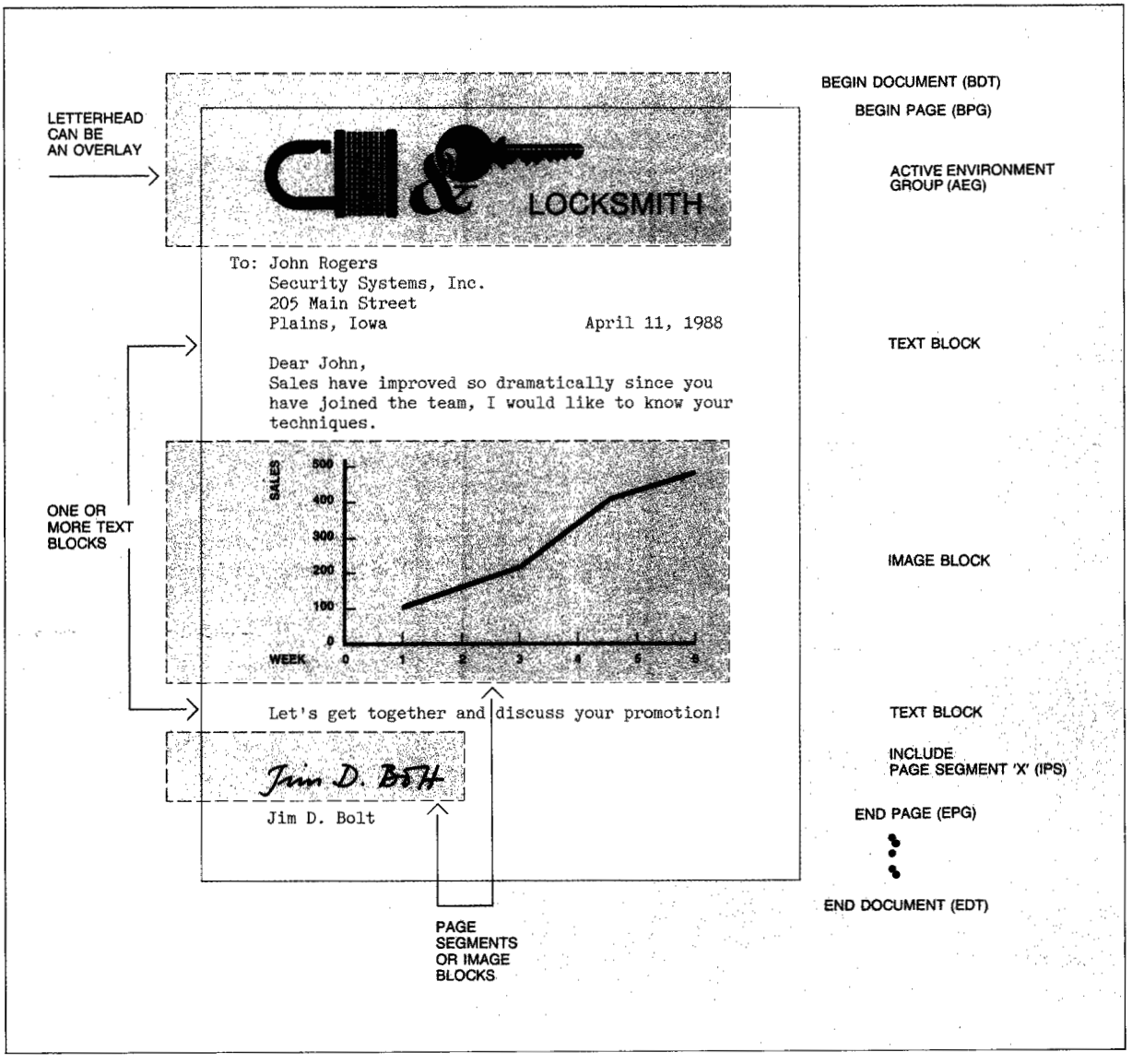
END ACTIVE ENVIRONMENT GROUP

Composed-text block. The composed-text block contains the actual data to be printed on the page, along with optional formatting controls. These controls specify functions such as the following:

- Moving the print position in the in-line direction; positioning may be in absolute terms or relative to the current position
- Moving the print position in the baseline direction; positioning may be in absolute terms or relative to the current position
- Selecting a font by referring to an identifier defined in the active environment group
- Drawing horizontal or vertical rules
- Setting text orientation

The text data are identical to those carried in the IPDS data stream. Thus, when the data stream is bound to a particular device and the IPDS data stream

Figure 4 Sample page with AFPDS



is generated, the text block does not have to be opened up and transformed. The information required for binding to a particular printer is isolated in the active environment group. The text block is made up of a set of delimiters, along with the text controls and the text data. Multiple text blocks may appear on the same page. Note that the term *in-line* refers to the direction in which characters are placed along the current line, and *baseline* refers to the direction in which lines are placed on the page. A text block appears as follows:

```
BEGIN COMPOSED TEXT BLOCK
  COMPOSED TEXT DATA
END COMPOSED TEXT BLOCK
```

Image block. An image block is similar to a composed-text block in that it is also made up of a set of

structured fields that delimit the image block, provide descriptive information, and carry the actual image data. The set of image-structured fields described here provide for a unique form of compression, where an *image cell* may be defined and then replicated across the page. Other more sophisticated forms of image compression may be instituted within the structure provided by the architecture. An image block comprises the following structured fields:

BEGIN IMAGE BLOCK
  IMAGE OUTPUT CONTROL—image origin, orientation, and scale factor
  IMAGE INPUT DESCRIPTOR—image size or image-cell description
  IMAGE-CELL POSITION—required only when image cells are used
  IMAGE RASTER DATA
END IMAGE BLOCK

Page segment. A page segment defines constant data that can be printed on different pages or in different positions on the same page. The constant data can include text (in a composed-text block) or images (in image blocks) or both. Page segments are named in the document and can be bound into the data stream in the print server, or named in the IPDS data stream and bound at the printing device. The latter technique is especially useful where a given page segment, such as a logo (i.e., logotype) is used over and over again during the printing of a document. When the page segment is prestored in the device, it does not have to be transmitted to the printer each time it is used. On the other hand, if data security in the page segment is involved (such as a signature, for example), we do not want it stored in the printer, and we send it in the data stream each time it is to be used.

If data security in the page segment is involved (such as a signature), we send it in the data stream each time it is to be used.

A page segment is described by the following structured fields:

BEGIN PAGE SEGMENT
  COMPOSED TEXT BLOCK—optional
  IMAGE BLOCKS—one or more; optional
END PAGE SEGMENT

Page segments are then referenced in the data stream with an include-page-segment structured field.

Graphics data. In Figure 4, we show a sample page composed of vector data. Such documents are often created using graphics facilities, such as GDDM, which provides an interface known as the Composed Document Presentation Data Stream (CDPDS) in which GDDM vector graphics data can be described. For applications using this interface, GDDM generates IPDS graphics objects for GDDM-driven printers, or it converts the graphics into AFPDS image blocks for PSF-driven printers. For example, DW/370 uses this interface to print GDDM graphics on a PSF printer. Because the structures of AFPDS and CDPDS are nearly identical, the graphics block defined in CDPDS may easily be supported by AFPDS at some future point in time.

End page terminates the page.

End document terminates the document.

Line data. The AFP architecture allows an application to print existing line-data streams on a page printer or invoke transforms to enhance the line data before printing. Two AFPDS control structures—Page Definitions (PAGEDEFs) and Form Definitions (FORMDEFs)—provide this function.

Page definitions. PAGEDEF defines the page format used by print services to format line data into pages. Page definitions can also contain names of fonts, page size, names of data-suppression fields, and line positioning.

These controls are applied externally to the definition of the print data stream at the time the system print services are scheduled for the print data set. A simple example of applying a PAGEDEF is to print line data in a two-up format, that is, to print two logical pages (i.e., numbered pages) side by side on a sheet of standard computer paper. This requires changing to a smaller font size, rotating the output, and repositioning the pages on the media. Page definitions are composed of standard AFPDS structured fields. Nor-

mally, page definitions are invoked by specifying the page definition as an installation default parameter for line-printer datasets or as a parameter when scheduling the line printer dataset for printing. The application program itself need not be changed.

A PAGEDEF comprises a set of data maps, as follows:

```
BEGIN PAGE MAP (PAGEDEF)
  BEGIN DATA MAP
  BEGIN ACTIVE ENVIRONMENT GROUP
  MAP CODED FONT
  MAP PAGE SEGMENT
  PAGE DESCRIPTOR
  COMPOSED TEXT CONTROL
  COMPOSED TEXT DESCRIPTOR
END ACTIVE ENVIRONMENT GROUP
BEGIN DATA MAP TRANSMISSION SUBCASE
  LINE DESCRIPTOR COUNT—number of line descriptors
  LINE DESCRIPTOR—map of line data to page
  .
  .
  .
  FIXED DATA SIZE—number of bytes of fixed data
  FIXED DATA TEXT—constant text data for page
END DATA MAP
  .
  .
  .
END PAGE MAP
```

Form definitions. A FORMDEF is required for all AFP print jobs. A FORMDEF specifies the position of a page on a form and includes one or more copy groups. A copy group provides the following functions:

- Text suppressions, if any
- Overlays to be used
- Offset stacking
- Edge marking
- Forms flash
- Horizontal paper adjustment
- Paper-bin selection
- Duplexing

Both form definitions and page definitions can be applied externally to the print data set. A FORMDEF is composed of a Document Environment Group and a set of medium maps. A Document Environment Group is similar to an Active Environment Group and has the following structure:

```
BEGIN DOCUMENT ENVIRONMENT GROUP
  MAP MEDIUM OVERLAY—identifies overlays
  MAP SUPPRESSION—identifies text suppression names
  PAGE POSITION—page offset from medium (physical page)
  MAP MEDIUM DESCRIPTOR—identifies measurement units to be used
END DOCUMENT ENVIRONMENT GROUP
```

A medium map is provided for each set of pages—called a copy group—that have similar attributes. A medium map is composed of the following structured fields:

```
BEGIN MEDIUM MAP
  BEGIN FORM ENVIRONMENT GROUP
    FORM ENVIRONMENT GROUP DESCRIPTOR
    MAP MEDIUM OVERLAY
    PAGE POSITION
    MAP MEDIUM DESCRIPTOR
    MEDIUM COPY COUNT—number of copies of each page
    MEDIUM MODIFICATION CONTROL—duplex, bin selection, etc.
  END FORM ENVIRONMENT GROUP
  .
  .
  .
END MEDIUM MAP
```

Mixed line and page data. By mixing structured fields with line data records, an application can perform the following functions:

- Change page position, specify duplex printing, change paper source, specify offset stacking, and specify number of copies.
- Change the page format—margins, line spacing, page size, and the contents of the page.
- Use text control codes in the file to change fonts within a line or page and control placement of text on the page.
- Include image data directly in the file.
- Include page segments on a page.

Table 1 lists the structured fields that can be included directly within a line-printer data stream. In a line data stream, the structured fields are identified by using the X'5A' control character on the front of the record. Line data records must be prefixed with

Table 1 AFPDS structured field that can be mixed with line data

Abbreviation and Name	Identifier (hexadecimal)	Record Length (bytes in hexadecimal code)	Description
IMM: Invoke Medium Map	D3ABCC	10	Selects the copy-group definition
IDM: Invoke Data Map	D3ABCA	10	Selects the page format
CTX: Composed-Text Data	D3EE9B	8–7FEF	Includes text on a page
Image definition controls:			
BIM: Begin Image Block	D3A87B	10	Begins an image definition
IOC: Image Output Control	D3A77B	20	Positions an image on a page
IDD: Image Input Descriptor	D3A67B	2C	Specifies the size of an image
IRD: Image Raster Data	D3EE7B	8–7FEF	Describes an image raster pattern
ICP: Image Cell Position	D3AC7B	14	Specifies size, position, and repeating of image cells
EIM: End Image Block	D3A97B	10	Ends an image definition
IPS: Include Page Segment	D3AF5F	16	Specifies a page segment

a valid carriage-control character when line and page data are being mixed.

Intelligent Printer Data Stream

IPDS is an IBM data stream for all-points-addressable printing[13] and has been defined as an SAA protocol. PSF accepts the AFPDS data stream and generates IPDS. IPDS is bound to the printer in the sense that the IPDS data stream contains device-specific resources, such as fonts, and device-specific controls and recovery sequences. This data stream provides for a two-way dialogue between the printer and print services for managing resources, querying device characteristics, and handling certain recovery situations. The following are some of the characteristics that make IPDS unique among printer data streams.

IPDS is a fully paginated, object-oriented data stream. Applications that create each type of source data (for example, graphics, image, and text) may be independent of one another. IPDS allows the output of these independent applications to be merged at the printer. Fixed page boundaries provide for efficient error recovery and allow for the design of high-speed printers that are capable of pipelining the processing of page data. Because IPDS is independent of the carrying protocol, the same data stream can be carried to channel-attached printers, local area networks, or any other type of networking protocol that supports the transparent transmission of data.

IPDS transfers all data and commands through self-identifying structured fields, many of which are similar to those defined in AFPDS. Additionally, controls are provided for the dynamic management of downloaded and resident resources, such as overlays, page segments, and fonts.

Other device-control functions provide for duplexing, operator panel displays, paper sourcing, and finishing. The same controls further provide the comprehensive handling of exception functions, so that users can control levels of document precision. The range of document precision is from output exactly as requested to output that is the best the printer can do.

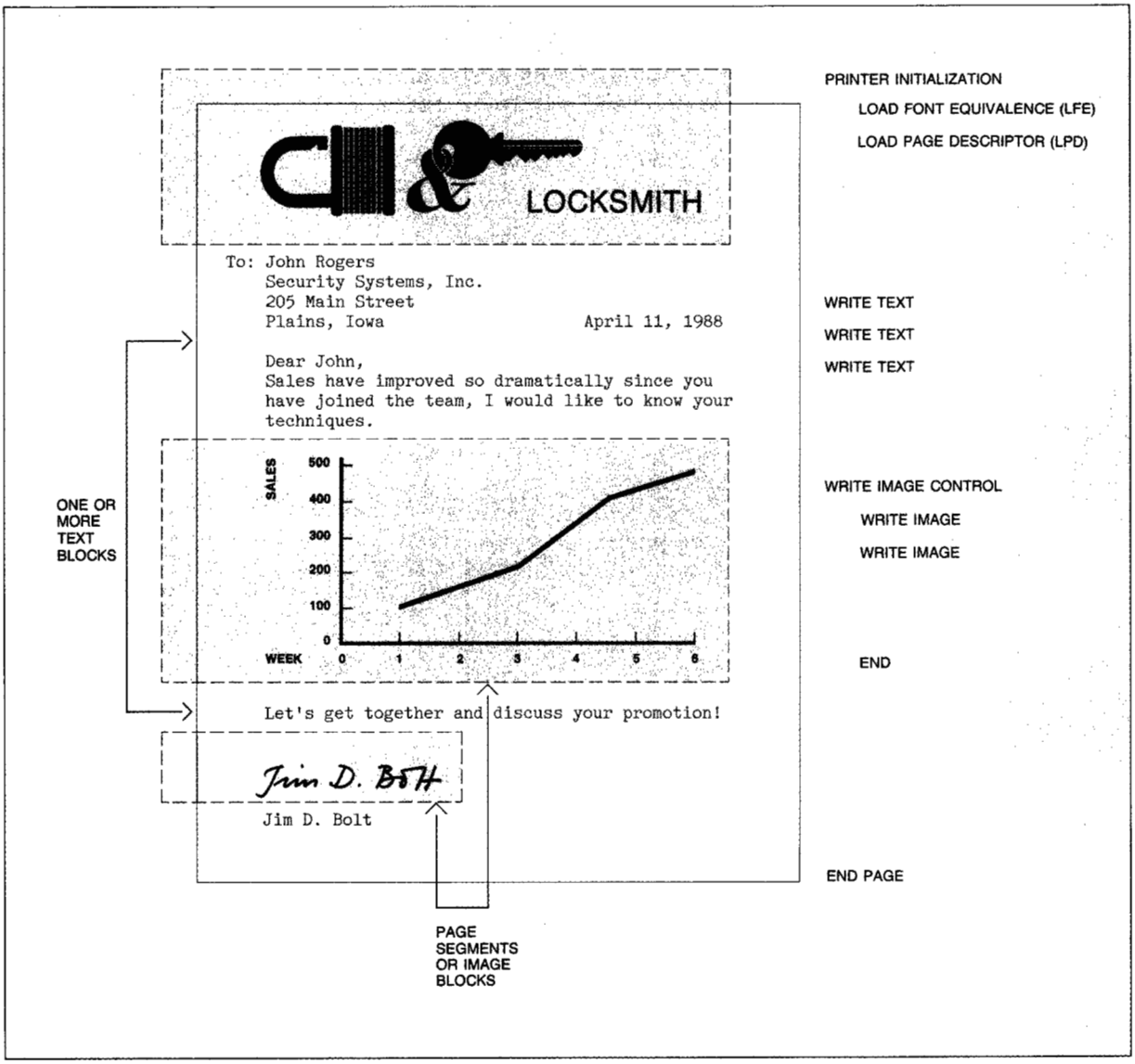
Finally, IPDS provides a complete acknowledgment protocol at the data-stream level. This protocol helps synchronize host and printer processes and exchange query–reply information, and it returns detailed error information.

Consider Figure 5, which shows the same page as in previous examples, but with the IPDS data-stream constructs that are used to print it. Note the similarities between this data stream and the AFPDS example discussed in the previous section.

Printer initialization

Before any printing can take place, the host must specify certain parameters and conditions for the printer. A typical printer initialization sequence might be composed of the following commands, which are passed to the printer as structured fields:

Figure 5 Sample page with IPDS



Sense Type and Model. The host sends this command to the printer, which senses the IPDS functions that are implemented by the printer.

Acknowledge. Recall that IPDS defines a two-way communications path between the host and the printer. In this case, the acknowledge structured field provides the host with the requested IPDS implementation information.

Set Home State. IPDS defines a state-machine implementation that must be followed by any IPDS printer. The initializing commands described here require the printer to be in *home state.*

Load page descriptor. This command sets print characteristics for the logical page, including page size, initial text coordinate positions, text direction, text margin, intercharacter adjustment, baseline increment, font ID, and text color.

Load Page Position. This command positions the upper left-hand corner of the logical page with respect to the top-of-form setting, which positions the logical page on the physical page.

Load Copy Control. This command specifies the number of copies to be produced, whether to print simplex or duplex, the overlays that are to be included on each copy, and the suppressions that are to be activated for each copy. Suppression allows data to be selectively omitted during printing.

Load Font Equivalence. This command maps local font IDs from within the text or graphics data to either loaded font IDs or global font IDs used for resource management.

IPDS data objects. Once the printer has been initialized, it is ready to receive the page images to be printed. The sequence of structured fields required to print the example page might be as follows:

Begin Page. This structured field puts the printer into page state.

Write Text. This structured field sends text data to the printer. The text data are identical to the data contained within an AFPDS composed-text block. When the printer is in page state, the text following this command prints on the current page.

Include Page Statement. This command causes a previously stored page segment to be merged with the current page. An identifier in the structured field names the page segment to be merged.

Write Image Control. This command causes the printer to enter the image-block state.

Write Image. This command sends a raster image to the printer. The image data that are transmitted as part of this structured field are identical to the image data contained in an AFPDS image block.

End. This command ends image state.

End Page. This command terminates page state and returns the printer to home state.

Concluding remarks

The architectures of Advanced Function Printing (AFP) have been described, and the following three architectures have been defined:

- Line Printer Data Stream (LPDS)
- Advanced Function Printing Data Stream (AFPDS)
- Intelligent Printer Data Stream (IPDS)

Each of these architectures plays an important role in Advanced Function Printing. The line-printer interface provides a way for existing applications to print on an AFP printer and exploit the capabilities of the printer without affecting the application program. The AFPDS data stream is an application interface to AFP that allows an application to be device-independent, shielding the application from device-specific functions such as device control and error recovery. IPDS provides for an efficient, two-way communication link between printer and print services. It is bound to a specific printer and provides detailed control over print resources, device control, and error recovery.

The commonality between AFPDS and IPDS minimizes the processing step in transforming AFPDS into IPDS, and ensures the integrity of the data to be printed. Page definitions and form definitions are control structures that apply across all three environments and provide added functions beyond those defined in the data streams.

Cited references

1. A. W. Griffee and C. A. Casey, "An introduction to typographic fonts and digital font resources," *IBM Systems Journal* **27**, No. 2, 206–218 (1988, this issue).
2. R. K. deBry and B. G. Platte, "Advanced Function Printing: A tutorial," *IBM Systems Journal* **27**, No. 2, 219–233 (1988, this issue).
3. *Print Services Facility Data Stream Reference*, SH35-0073, IBM Corporation; available through IBM branch offices.
4. *Document Composition Facility and Document Library Facility General Information Manual*, GH20-9158, IBM Corporation; available through IBM branch offices.
5. *Introducing DisplayWrite/370*, GH12-5170, IBM Corporation; available through IBM branch offices.
6. *Graphical Data Display Manager General Information Manual*, GC33-0100, IBM Corporation; available through IBM branch offices.
7. *Page Printer Formatting Aid User's Guide and Reference*, G544-3181, IBM Corporation; available through IBM branch offices.
8. *Overlay Generation Language User's Guide and Reference*, SH35-0079, IBM Corporation; available through IBM branch offices.
9. *Print Services Facility User's Programming Guide for VM*, S544-3512, IBM Corporation; available through IBM branch offices.
10. *IBM LAN Print Manager Reference Guide: IBM PC Network*, S544-3112, IBM Corporation; available through IBM branch offices.
11. *Information Processing—Text and Office Systems Document Architecture (ODA) and Interchange Format*, ISO/DIS 8613, International Organization for Standardization (1985).

12. R. K. deBry and L. J. Hash, "Device Support of Multiple Information Types," *Proceedings of the IFIP TC 6 International Symposium on Computer Message Systems*, Washington, DC, September 1985, pp. 195–200.
13. *Intelligent Print Data Stream Reference*, S544-3417, IBM Corporation; available through IBM branch offices.

Roger K. deBry *IBM Information Products Division, 6300 Diagonal Highway, Boulder, Colorado 80301.* Dr. deBry received his Ph.D degree from the University of Utah in 1972 in electrical engineering. He subsequently took an assignment in Kingston, where he was responsible for the IBM 3270 display data-stream architecture definition, and in 1979 received an Outstanding Innovation Award for his work. In 1985 he published the book *Communication with Display Terminals*, McGraw-Hill Book Company, Inc., New York. Dr. deBry worked for two years on the Corporate Programming Staff in Purchase, New York, where he focused on architecture and system design issues relating to device support. During this assignment, he became interested in software support for laser printers. As a result, at the completion of his corporate staff assignment he moved to Boulder as technical assistant to the programming center manager. Dr. deBry has been in this position for two years, and has been responsible for developing a printer software strategy. He is a member of ACM and an active participant in IFIPS, where he is the U.S. representative to IFIPS, TC2, Programming.

Brian G. Platte *IBM Information Products Division, 6300 Diagonal Highway, Boulder, Colorado 80301.* Mr. Platte joined IBM in 1970 as a programmer, and is currently a senior programmer in the Page Printer System Design Department, where he is responsible for page printer system design. For the past nine years he has worked on APA printers and Advanced Function Printer software. Mr. Platte's responsibilities have included IBM 3820 software support, page printer software planning, system design, and data-stream architecture. Prior to working on printers, he worked on the IBM 3850 Mass Storage System, where he was responsible for the microcode. Mr. Platte earned a B.S. degree in mathematics at the Central Michigan University, Mount Pleasant, Michigan. He received an M.S. degree in computer science from the University of Colorado, Boulder, in 1975. Mr. Platte has received two patents, one of which was for the IPDS architecture, which also earned him his first Invention Achievement Award, an IPD President's Award, and an Outstanding Technical Achievement Award.

Carol L. Berinato *IBM Information Products Division, 6300 Diagonal Highway, Boulder, Colorado 80301.* Ms. Berinato joined IBM in 1976 in Boulder as a programmer in product test, where she was part of the team testing the IBM 6670 printer. Today, she is a development programmer in the Boulder Programming Center and manages the architecture department, which has responsibility for the Intelligent Printer Data Stream (IPDS) architecture for Advanced Function Printing and the corporate font architecture. Ms. Berinato graduated in 1973 with a B.A. in philosophy and religion from Boston University, and in 1976 received an M.S. in digital systems design from Arizona State University, Tempe.

James W. Marlin *IBM Information Products Division, 6300 Diagonal Highway, Boulder, Colorado 80301.* Mr. Marlin is an advisory systems analyst in the IBM development laboratory in Boulder. He has held several development and managerial positions in areas related to IBM systems architecture, including display programming advanced technology, communication systems programming, industry systems development, and office systems interchange architecture. He is the recipient of an IBM Outstanding Innovation Award and an IPD Division Award for his contributions to IBM data-stream strategy and development. Mr. Marlin was the lead architect for the development of the Intelligent Printer Data Stream architecture. He received a B.S. in mathematics from the University of Washington, Seattle, in 1961.

Reprint Order No. G321-5322.