Printing on a Post Processor Using UP^{3}I in an AFP Environment

Jean Aschenbrenner, Reinhard Hohensee and David E. Stone
IBM
Boulder, Colorado

Abstract

In a complex high-end printing environment, some applications require that some (special) data be printed on a post processor. This paper will discuss a new architecture which uses the Universal Printer Pre- and Post-Processing Interface (UP^{3}I) Print Data interface to print objects on a post processor in Advanced Function Printing (AFP) environments.

Two data streams are involved:

- Data is passed from the host (application) to the printer using AFP object containers.
- Data is passed from the printer to the post processor using a new extension of the UP^{3}I interface.

The printer control unit must convert from one format to the other. It must also translate locations for the post processor.

As an example, an application might print most of its data on a black/white printer and add highlight color, invisible ink, or MICR toner to the page using a post processor. The capability of each post processor can vary and the data format and control required will be different.

The architecture has been defined to allow for this variability. In the AFP environment, the format of the data is hidden within the AFP object container and does not need to be understood by the print server or printer.

Introduction

Future post processors will have more capabilities and there is a demand for more sophisticated ways of delivering data to be printed by those post processors.

Post processors have traditionally printed using MICR ink or a highlight color. Now, some print using invisible ink or multiple highlight colors. It is expected that post processors will have the capability to store multiple canned images or to parse data streams with a limited, well-defined syntax. Because the post processor must print sheets as fast as the main printer, it is not expected that it will do complex processing such as full-color processing or handling all of a print data language. Some post processors will be able to print everywhere on the sheet, whereas others may be able to print in only narrow swaths. Obviously, capabilities vary.

There are now more ways that applications want to use post processors. Printing MICR text was a common use in the past. Possible new uses include:

1. Printing bar codes in invisible ink
2. Printing highlight-colored bars which could be part of a bar chart or could be used to separate data. (The location, width and height would define what and where to print.)
3. Printing highlight-colored logos and icons. (These could be images stored in the post processor. An image ID and location would define what and where to print.)
4. Printing a limited number of canned text strings in highlight color. They could be used for marketing messages or headers. The text may be in varying fonts, font sizes and boldness. (These text strings could be stored as images in the post processor and processed as in #3.)

Currently there are two processes used by AFP printers to send data to a post processor. One is used for very simple data and the other is used for complex data.

Simple MICR data is passed to a post-processing device over a Type II Interface. The data being sent is a sequence of entries of the form:

x-position, y-position, number of bytes following, data.

X-position and y-position are given in 1/72000 of an inch.

The Type II document states: "The data (its format and content) are undefined by this specification, unless noted otherwise. The data is generated by the application program specifically for the particular post processor that is attached. The form and content is determined by the application and the needs and capabilities of the post processor."

This interface is used to send data to the Troy MICR post processor and the data is defined to be a string of text characters. There must be an understanding between the application generator and the post processor about which font is to be used, since no font information is passed over the interface.

The printer receives Intelligent Printer Data Stream (IPDS) commands. It identifies which data to send to the post processor by noting a MICR flag in the font. It does some error checking on the characters and can identify off-page errors for the printing characters. It does not print the characters but rather puts them into a buffer which is later sent to the post processor. AFP mixing rules are not followed for these characters: They do not erase data under them and they are not erased if data is placed over them.

Complex data is printed by the Infoprint Hi-Lite Color post processor using up to 3 different highlight colors. If a data object's color is specified to be Highlight #1, #2 or #3 in the IPDS data, it is printed on the post processor. For each of the three colors, the printer renders the data into a full-sized sheet bitmap. These sheet bitmaps are compressed and sent to the post processor. This can require transmitting a large amount of data so a dedicated, fast control/data interface is established directly between the printer and post processor.

In 2003 we identified a need for handling somewhat complex data, in various forms, in a flexible manner. The concept of an agreement between the application and the post processor was critical. This would allow new applications and enhanced post processor capabilities without requiring a change to the print server or printer software.

UP^{3}I had recently become the industry-standard online post-processing architecture. It specified a UP^{3}I Print Data Triplet but, originally, there was not a useful definition of how it could be used. During 2003 and early 2004, we developed a more complete definition of UP^{3}I Print Data which was modified and accepted by the UP^{3}I consortium. We also expanded the AFP architecture to allow print data from the application to be passed through the print server and printer and out to the post-processing printer.

A key part of the definition is a Print Data Format ID (PDFID) which allows multiple data formats to be used but identifies which one is used. The PDFID is transmitted with the data in the Print Data Triplet. This allows some control over the agreement between application and post processor in that the PDFID must be registered and defined within the UP^{3}I architecture. In addition, the format ID allows the format of the Print Data to be entirely transparent to the print server and the printer. Four possible uses of post-processing printers were listed above. Each one of those could be implemented using a different format id.

Print Data for the Post Processor in AFP

The application program that creates the document to be printed generates the document in Mixed Object Document Content Architecture (MO:DCA) format. The MO:DCA architecture defines the AFP page description language. The print server then converts the AFP page description language into the Intelligent Printer Data Stream (IPDS) format. The two data streams have the same constructs and concepts but the architecture terminology is slightly different. This paper will discuss the concepts only once, using generic or IPDS terminology but the reader should understand that parallel constructs apply in the MO:DCA architecture.

An AFP document contains page objects, which in turn contain data objects. These data objects define the information to be presented on the page, such as text, graphics, image, and bar code. One form of data object is a generic data-object wrapper called an object container. This object container has basically the same characteristics as other AFP objects such as image or graphic objects.

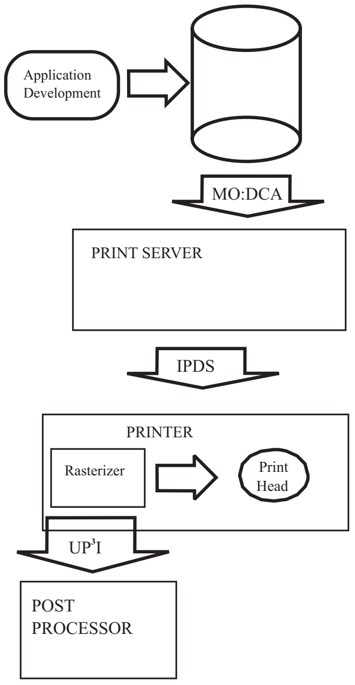

Figure 1. Flow of UP^{3}I data

The print data that is to be passed to a UP^{3}I device is generated by the application program in an object container. The object container has control (header) information and data. The data portion includes the Print Data Format ID (PDFID) and the actual data to be printed. The data portion is precisely the data that is eventually carried in a Print Data Frame on the UP^{3}I interface and its syntax is defined by the UP^{3}I specification. Thus, this data can be copied directly into the UP^{3}I Print Data Triplet and does not need to be understood by the printer.

The object container control includes:

1. An object area which has:
   - A location which is given as an offset from the page or overlay origin.
   - Width and height
   - Optionally, information that allows the object area to be colored or causes all underlying data to be erased.

2. An object presentation space which has:
   - A mapping option which specifies how the object presentation space is mapped into the object area.

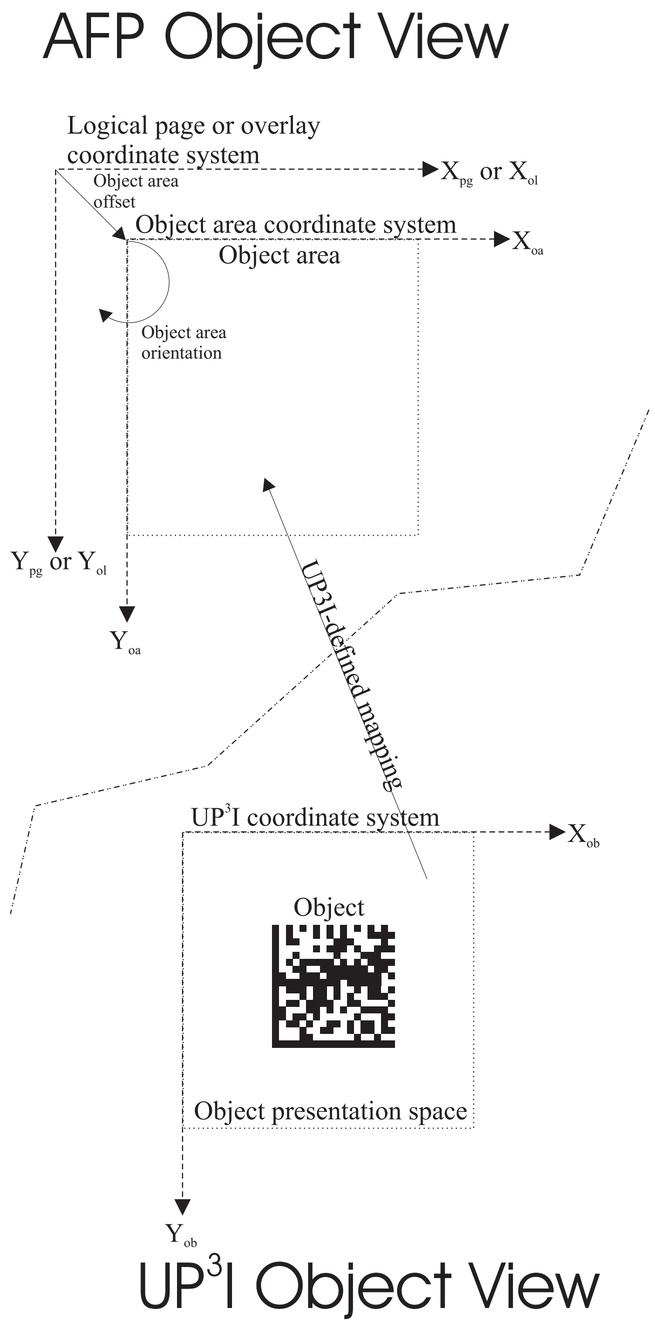

Figure 2. Conceptual Views

Mapping, Mixing and Locations

The data to be printed is generated in the object presentation space. This is then mapped into the object area. The mapping options available for other data objects include scale to fit, position and trim, center-and-trim, and others. For a UP^{3}I Print Data object container, there is a special mapping option, "defined by UP^{3}I Print Data format ID," which indicates that the mapping is defined by the post-processing printer. This is the only mapping option allowed for a UP^{3}I Print Data object container. (See Fig. 2.) The post-processing device can use the object area location however it wishes, although the use should be consistent given the PDFID. For instance, the location could be used as the center location of an image, as the bottom left start of a text string, or as the top left corner of a bar code object.

The printer is responsible for the object area of the container. It processes the object area coloring, if specified, and ensures that the AFP mixing rules are followed as it does the processing.

The post-processing printer is responsible for the object presentation space. Since the print data is presented by a UP^{3}I device after the complete page is rendered by the printer, the presentation container cannot mix with the remainder of the page data according to the standard AFP mixing rules. The AFP mixing rules require that an object is mixed with the remainder of the page data based on the order in which it is specified on the page. It would be very difficult to follow this rule since the UP^{3}I print data is rendered last due to the physical configuration of the system. Therefore, in AFP, a new type of mixing is defined for UP^{3}I print data:

The print data is developed in its own presentation space by the UP^{3}I device in accordance with the Print Data format. It mixes with the remainder of the page in a manner that is defined by the print data format.

For example, one Print Data format might define the mixing such that the data is printed with invisible ink that is transparent with respect to any underlying data. Another format might define the mixing such that, since a MICR ink is used, characters printed are opaque and cover all underlying data. The location of the object area must be sent to the post processor. However, the location of the object area is specified as an offset from the page or overlay. The location specified to the post processor must be in relation to the whole sheet. The location is specified as an offset from the UP^{3}I medium origin (which is the left corner of the leading edge of the sheet). Therefore, the printer must convert the locations. This is shown in Fig. 3. Further, the units for the offsets differ. The printer must convert from AFP L-units to 1/72000 inch (millipoints) which are the units used by UP^{3}I.

Object area width, height and orientation will be converted to UP^{3}I units and sent to the post processor but the post processor is not required to use this information. The application creating the data to be printed by the post processor should typically assume that it will be printed as received, without any rotation, since most post processors will not be able to rotate data which they print. If the post processor is able to rotate, a new PDFID can be defined that specifies support for rotation.

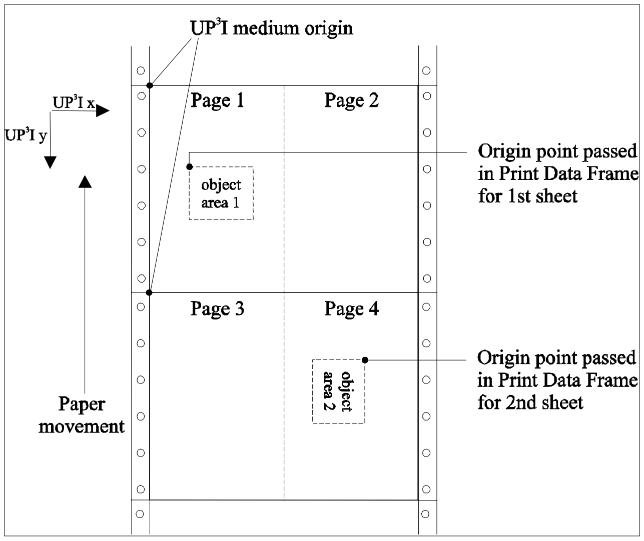
UP3I medium origin
UP3I x
UP3I y
Page 1
Page 2
Page 3
Page 4
object area 1
object area 2
Origin point passed in Print Data Frame for 1st sheet
Origin point passed in Print Data Frame for 2nd sheet
Paper movement

Figure 3. UP3I Medium and AFP Object Origins

Print Data in UP^{3}I

UP^{3}I architecture specifies that, for each sheet sent, there will be a Form Exit Frame which specifies operations to be performed on that sheet. In addition, if print data is involved, a **Print Data Frame** will be sent. The UP^{3}I spec says "This frame is typically used for additional printers … which need a small amount of print data ..." Too much data could mean that it does not reach the post processor in time to be printed.

The Form Exit Frame includes multiple triplets which are self-identifying structures consisting of length, identifier and data fields. There is one Form Exit Frame for each sheet. It is passed along sequentially to all devices in the paper path. The Form Exit Frame always includes:

- A Form Size Triplet which gives the paper length and width.
- A Page/Set/Job Triplet Triplet which holds the UP^{3}I_PAGE_ID for each logical page on this form.
- Form Finishing Operating Triplets

If print data is being sent with the sheet, the Form Exit Frame includes a Form Finishing Operating Triplet which holds the following, along with other information:

- ID of the post-processing printer device which is to receive the data
- Finishing Operation Type: Postprocessing Print (0x10)
- Finishing Operation Parameter: Print (0x0001)

The Print Data Frame is sent directly to the destination post-processing printer. The destination device does not forward it further. Thus, the Print Data Frame is destined for exactly one printing device.

The Print Data Frame includes these UP^{3}I triplets:

- Paper Sequence ID Triplet: includes IDs of sending and destination UP^{3}I devices.
- UP^{3}I_PAGE_ID Triplet: includes the UP^{3}I_PAGE_ID which identifies the page where printing should occur
- Print Location Triplet: location for print data plus front/back-side indicator
- Print Data Triplet: holds the UP^{3}I Print Data Format ID as well as the print data. Maximum size possible is 64MB. It needs to be smaller than this for performance reasons.

There may be multiple sets of UP^{3}I_PAGE_ID, Print Location, and Print Data triplets in one Print Data Frame.

The printing post processor must be told where to position the data. This information is in the Print Location Triplet. The offset of the data to be printed is expressed in millipoints (1/72000 inch) from the UP^{3}I medium origin, x (across the web) first, y second.

For future extendibility, the following additional information will be carried in the Print Location Triplet.

- Object area size: width & height in millipoints
- Object area orientation: 0, 90, 180 or 270 degrees

The printing post processor will determine if and how it uses the location, size and orientation.

The printer creates the UP^{3}I triplets. It constructs the Print Location Triplet using information from the object area control. The ID and data in the object container are inserted into the Print Data Triplet. The printer knows the ID of the destination post processor printer since this ID was reported by the post processor. It inserts that ID and the UP^{3}I_PAGE_ID into the Print Data Frame when it sends the sheet.

Currently there is only one group of Print Data Format IDs (PDFID) specified in the UP^{3}I architecture. The PDFID is 4-bytes long. The currently supported PDFID is:

X'BCy000xx' — Bar Code (BCOCA) Support

- "**xx**" represents the supported bar codes (such as Data Matrix, POSTNET, EAN 13)
- **"y"** represents the mixing rule for foreground data of the post-processing printer (opaque, transparent, or device specific).

This allows IPDS BCOCA objects to be sent directly to the post processor, which must be able to parse the BCOCA syntax.

Error Handling

The printer does not need to examine the print data in the object container and it does not need to do any checking of the data. The error handling process is complicated if an error is found by the post processor, so it is acceptable, although not required, for the printer to error check the data. It is assumed that the application is designed for a particular post processor with limited capability and that it will be possible to thoroughly test the application before doing any production runs.

Syntax and position-check errors detected (by either the printer or post processor) within a Print Data object have a well-defined exception ID. Rules have been architected to define the recovery actions required when the post processor reports errors.

The post processor will be set up to ignore errors or else to stop printing. If printing stops, operator intervention will be required. The UP^{3}I interface allows the post processor to send back some generic error indicators and a string of text which can be used to diagnose the problem.

Post Processor Identification

The printing post processor announces its presence on the UP^{3}I interface in a UP^{3}I Self Defining Field Frame. It reports its device ID and identifies itself as a Front or Reverse side Postprocessing Printer. It also tells which PDFIDs it supports. This information is passed up to the print server in the IPDS XOH-OPC (Obtain Printer Characteristics) response. Thus the server knows that the post processor is available and its capabilities.

It is possible to connect multiple printing post processors to one printer. For duplex printing, there might be two printing post processors, one for each side. The printer knows which side a particular object container is to be printed on and it knows the device ID of the post processor for that side. So it can route the UP^{3}I Print Data to the correct device.

Creation of Print Data by the Printer

This architecture does not preclude the printer from generating the print data itself, without the use of object containers, based on IPDS input. For instance, there might be a post-processing device which prints text using MICR ink. It recognizes data with the same format as that used on a Type II interface but receives it as UP^{3}I Print Data. Assume a UP^{3}I PDFID for this format has been architected. Now, the printer could be modified to identify characters which use a MICR font, create data which is formatted for the Type II interface, and copy it into a UP^{3}I Print Data Frame. Such a solution requires changes to the printer and is not as flexible as the object-container solution. But it may be useful in some situations.

Conclusion

This paper discussed a way of sending print data to a post processor in an AFP environment. The format of the data can vary and is identified by a Print Data Format ID (PDFID) which accompanies the data, but only the application and the post processor need to understand that format.

The UP^{3}I interface provides an accepted, effective way to send the print data from the printer to the post processor. The use of object containers allows a conduit for UP^{3}I Print Data through the AFP system. Preparing the data to send across the UP^{3}I interface is easily accomplished and requires only converting the print location and copying the PDFID and print data. This allows for a flexible architecture where the print server, the printer and the UP^{3}I interface do not need to change when a new data format is added.

This architecture is able to support future post processor capabilities and future application requirements.

References

1. UP^{3}I Limited, *Universal Printer Pre- and Post-Processing Interface*, Version 1.04 preliminary 5, June 2004.
2. IBM Corporation, *Intelligent Printer Data Stream Reference*, S544-3417-06, November 2002.
3. IBM Corporation, *Mixed Object Document Content Architecture Reference*, SC31-6802-06, January 2004.
4. IBM Corporation, *IBM 3900 Page Printer Advanced Function Post Processing Interface Specification*, TR-82.0520, May 1993.

Biographies

Jean Aschenbrenner is a Senior Technical Staff Member (STSM) in the Printing Systems Division of IBM in Boulder, CO. She joined IBM in 1984 and did VLSI and card design for the printer control engine. Since 1993 she has been working on microcode and is part of the rasterizer team for the high-end common control unit. She has coordinated work on color management on IBM's high-end color printers. Jean received her B.A. in Mathematics from Mount Holyoke College (1970), her M.A.T. from Wesleyan University in Connecticut (1971), her B.S. in Electrical Engineering from Colorado University (1983) and her M.S. in Computer Science from National Technological University (1994). She is a member of Phi Beta Kappa honor society and of IS&T.

Reinhard Hohensee received his BES degree from S.U.N.Y. Stony Brook in 1970, and his MSEE and Electrical Engineer degrees from Syracuse University in 1972 and 1977, respectively. He joined IBM in 1973 and had development assignments on processors and I/O attachments at the IBM Laboratories in Endicott, New York, in Boeblingen, Germany, and at the T.J. Watson Research Center in Yorktown Heights. Since 1985 Reinhard has been with IBM Printing Systems in Boulder, Colorado, where he has worked on printer controllers, printing architectures, and color printing. He is currently an IBM Senior Technical Staff Member responsible for the Advanced Function Presentation (AFP) Architecture and for color system design.

David Stone is a Senior Data Stream Architect, a 29 year veteran with IBM Corporation, and holder of advanced degrees in Mathematics, Computer Science, and Business Administration. Dave currently works with printer and document data streams, bar code description and printing, and font technologies used by IBM. Dave has worked on several development efforts within IBM including banking, tele-satellite communications, storage products, and printing. He was a member of IBM's first Print Services Facility software development team and is currently responsible for the Intelligent Printer Data Stream, Font, and Bar Code Object Content architectures. Dave holds numerous patents on print architecture.