# The Image Object Content Architecture

by Y. Hakeda

***Technical advances to image processing and the availability of the resulting technologies at reasonable cost have helped to promote the use of images in office, engineering, and scientific environments. As evidence of this use, a wide variety of applications and products designed for image processing have been introduced into the market in recent years. In order to encompass different applications and products in a single image processing system and to allow image data to be exchanged and interpreted consistently throughout the system, IBM has introduced the Image Object Content Architecture (IOCA). This paper discusses requirements for the architecture, concepts of the architecture, use of the architecture in the different data stream environments used by image processing systems, and the IOCA function sets that have been defined for interchange within Systems Application Architecture™ environments.***

In March 1987, IBM announced Systems Application Architecture™ (SAA™),[1] a collection of interfaces, conventions, and protocols for IBM software to attain consistency in applications across the MVS/TSO/E (Multiple Virtual Storage/Time Sharing Option Extensions), CMS/VM (Conversational Monitor System/Virtual Machine), OS/400® (Operating System/400®), and OS/2® (Operating System/2™) computing environments. One of the three major elements of this announcement was Common Communications Support (CCS),[2] which has the objective of providing a consistent set of protocols for communications among applications, and for storing, retrieving, and moving information through a communications network.

On May 16, 1989, IBM announced a number of CCS architectures that enable the interchange of information, including images, text, and graphics, within the Mixed Object Document Content Architecture-Presentation (MO:DCA-P),[3] Intelligent Printer Data Stream (IPDS),[4] and 3270 Data Stream (3270 DS)[5] environments. One of the newly announced architectures was the Image Object Content Architecture (IOCA).[6,7] The purpose of IOCA is to provide a single and consistent method for all applications and products involved in image processing to represent images, particularly when carried within the three data stream environments.

In this paper, the requirements for the architecture, the concepts of the architecture, its use within the data stream environments used by image processing systems, and the IOCA function sets that have been defined for interchange within the SAA environments are described.

## Background

Image processing has recently become affordable and "usable" because of a number of major changes in the computing environment. Without such changes,

image processing would still remain an advanced technology item within the confines of research laboratories.

Image is a data type that often tends to consume relatively more auxiliary storage and computer memory in comparison to other commonly known data types such as text and graphics. One reason for its use of more resources is the need to handle individual pixels,[8] which comprise the image, as contrasted to characters, lines, and circles, which are often symbolized in shorthand form. The pixels form a progression of horizontal rows and vertical columns which make up the image. As the distance between rows and between columns decreases, the grain of the image becomes finer, and the image usually becomes sharper. As the size of the image increases, more rows and columns are often recorded as a part of the image. Finally, representation of gray-scale and color images carries the additional requirement that more bits of data be recorded to represent each pixel. The end result is often use of more auxiliary storage and computer memory because of the amount of space required for each image. Recent availability of storage devices and high-speed memory at reasonable costs has therefore been one of the factors in making image processing commercially feasible.

**Performance plays a key role in determining the feasibility of image processing.**

Performance also plays a key role in determining the feasibility of image processing. If the size of the image itself is large, faster CPUs are required for processing large amounts of image data at high speeds. Otherwise, the image processing performance may become quite intolerable. However, recent trends for faster processors at affordable prices have helped to make image processing a viable offering.

Progress in image technology, such as in the area of image compression, has also helped to pave the way for the use of images. Without image compression, each pixel of a black and white image would normally be represented as a single bit of data. With the use of image compression techniques that exploit certain characteristics of the scanned image, multiple pixels may now be represented as a single bit of data, decreasing the required size of auxiliary storage or computer memory for an image typically by a factor of eight to twenty or more.

The development and commercial use of high-resolution displays and printers has also helped make images readily available. Only with high-resolution equipment can an image be displayed or printed at acceptable quality.

Finally, the general availability and use of programmable workstations has helped in creating an environment now fit for the use of images. Until recently, all points addressable (APA) displays were only available at computer centers and used for data entry. Now, programmable workstations have become inexpensive and fast and appear on desks within many business offices, not to mention on the laps of many traveling business professionals.

These factors are only some of many that have facilitated the recent surge in the use of images. Applications now using images are found in bank check processing, signature verification, and engineering drawings, and in the insurance, medical, and manufacturing industries, to name a few.

### Purpose and objectives

As more and more applications use images, the exchange of images and the sharing of image databases among different applications begin to play a key role within an enterprise. If there is no standard single image architecture, an image application cannot exchange images with another application without the intermediate step of converting the retrieved image to a form recognized by the recipient. Conversion requires knowledge of both the format by which the image was created and the format recognized by the recipient, as well as what could amount to a large amount of time and code to perform such intermediate conversions. As an example, discrepancies between the image format created by an input device such as a scanner and the image format recognized by an output device such as a display or

a printer often require that the image be decompressed, reformatted, and recompressed prior to the actual viewing or printing of the image. The same problems may be encountered when retrieving images from the database of a particular image application and sending the image to yet a different image application. Both are highly conceivable scenarios within an enterprise.

One of the main reasons that such problems occur is that without standardization, applications normally tailor their data representation to match the minimal needs of the application. Such data are usually structured into a form that will provide maximum performance to the application by being stripped of all superfluous constructs, which are taken for granted when processing. Yet such data can only be recognized within the realms of a single application.

It is also likely that different software and hardware components will contain similar functions, but, having been built around different applications, the components create and recognize only the image format unique to the application. Therefore, without having a single format for images as a standard, exchange of pertinent image information between different image applications and products is a tremendous chore, requiring much consumption of time, code, and even human resources to perform the necessary conversions.

To prevent these potential problems, IBM has introduced the Image Object Content Architecture. The primary purpose of IOCA is to provide a standard method by which all IBM image applications and products involved in image processing can represent images. This method would provide conventions and directions—for current as well as future products—for the processing and interchanging of images among different applications within and among enterprises.

An example of an image processing system with IOCA is shown in Figure 1.

The key design objectives in the development of the architecture were application and device independence, extendability, intactness in the different data streams, and a subset/superset relationship of functions. Following is a discussion of each objective:

- Provide an architecture that is rich enough to be capable of representing image information by any image application or image device and yet be application- and device-independent—An architecture must be rich enough to provide image applications and image devices with the necessary constructs and functions needed for image processing. It must also accomplish this without incorporating application- or device-unique features recognized only by a particular product.

- Provide extendability—As image technologies progress further, it should be possible to make additions to the architecture easily and in such a way so as not to affect the design of the entire architecture or even a significant portion of the architecture.

- Provide an image description that is flexible enough to exist intact in the MO:DCA-P, IPDS, and 3270 Data Stream environments—The three data stream environments used by IBM image products and systems are MO:DCA-P, IPDS, and the 3270 Data Stream. The MO:DCA-P environment is used to describe documents that are interchanged among IBM systems. Each document consists of one or more pages with each page of the document containing an IOCA image, presentation text, graphics data, or a combination of the three. IPDS is used to send IOCA images, presentation text, and graphics data to a printer. The 3270 Data Stream is used to display IOCA images and graphics data on a nonprogrammable workstation.

  In each of these three data stream environments, the description of the IOCA image must be structured such that it will exist intact. Much of the purpose in having an architecture is wasted if conversions are necessary when imbedding IOCA images into the different data streams.

- Provide subsets and supersets—Different image applications require different levels of complexity to describe images. Quite often, such differences stem from the exploitation of various image processing characteristics supported by the application, such as the ability to handle color images as opposed to being able to handle only bilevel (e.g., black and white) images. Each function set should be defined as a subset or superset of other sets, maximizing the possibility of interchange among products with different image processing capabilities. Thus all products that support higher-function sets would be capable of processing images of lower-function sets. An example of this is the support of bilevel images in addition to color images within a higher-function set.

**Figure 1 Example of an image processing system**

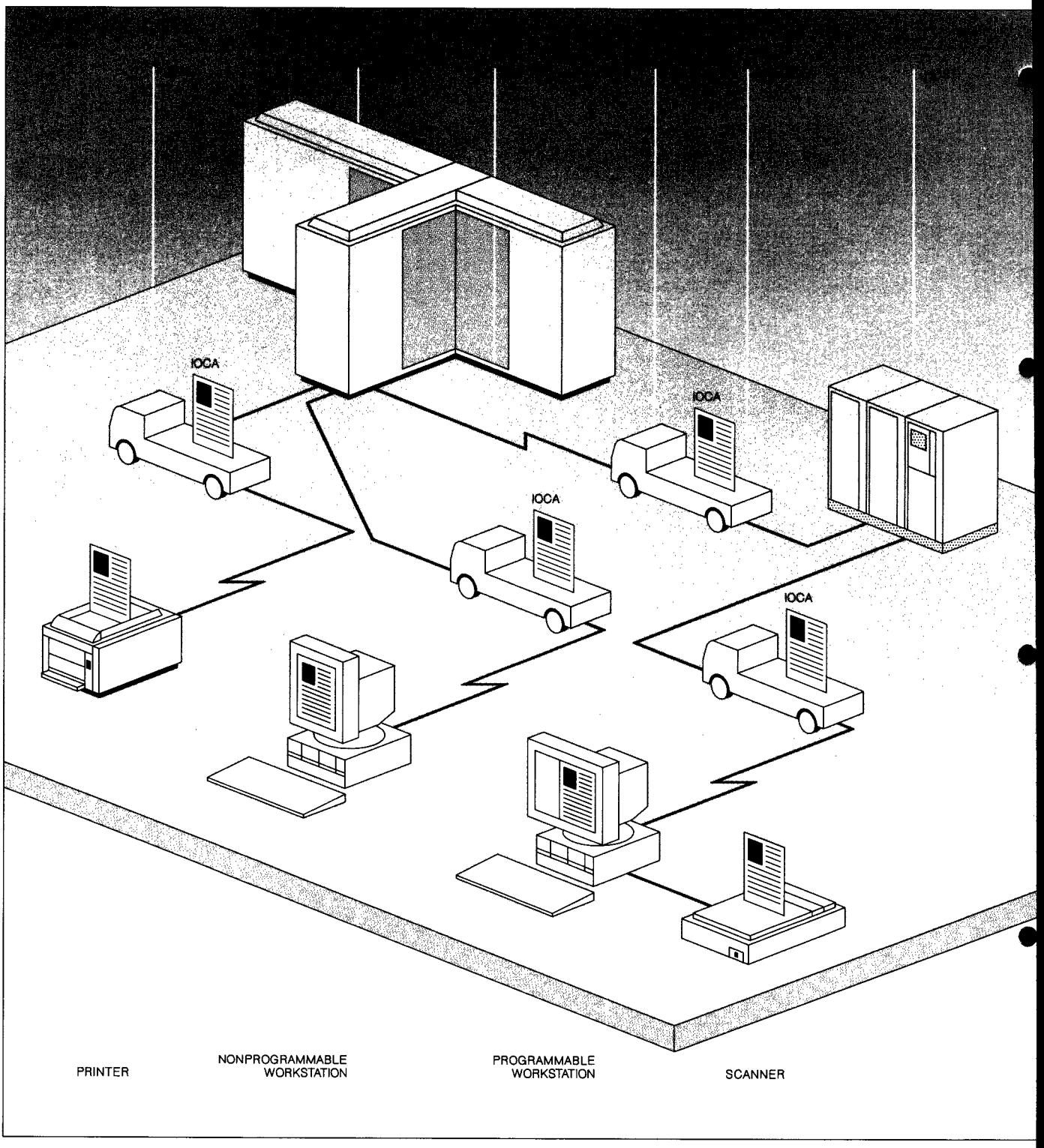
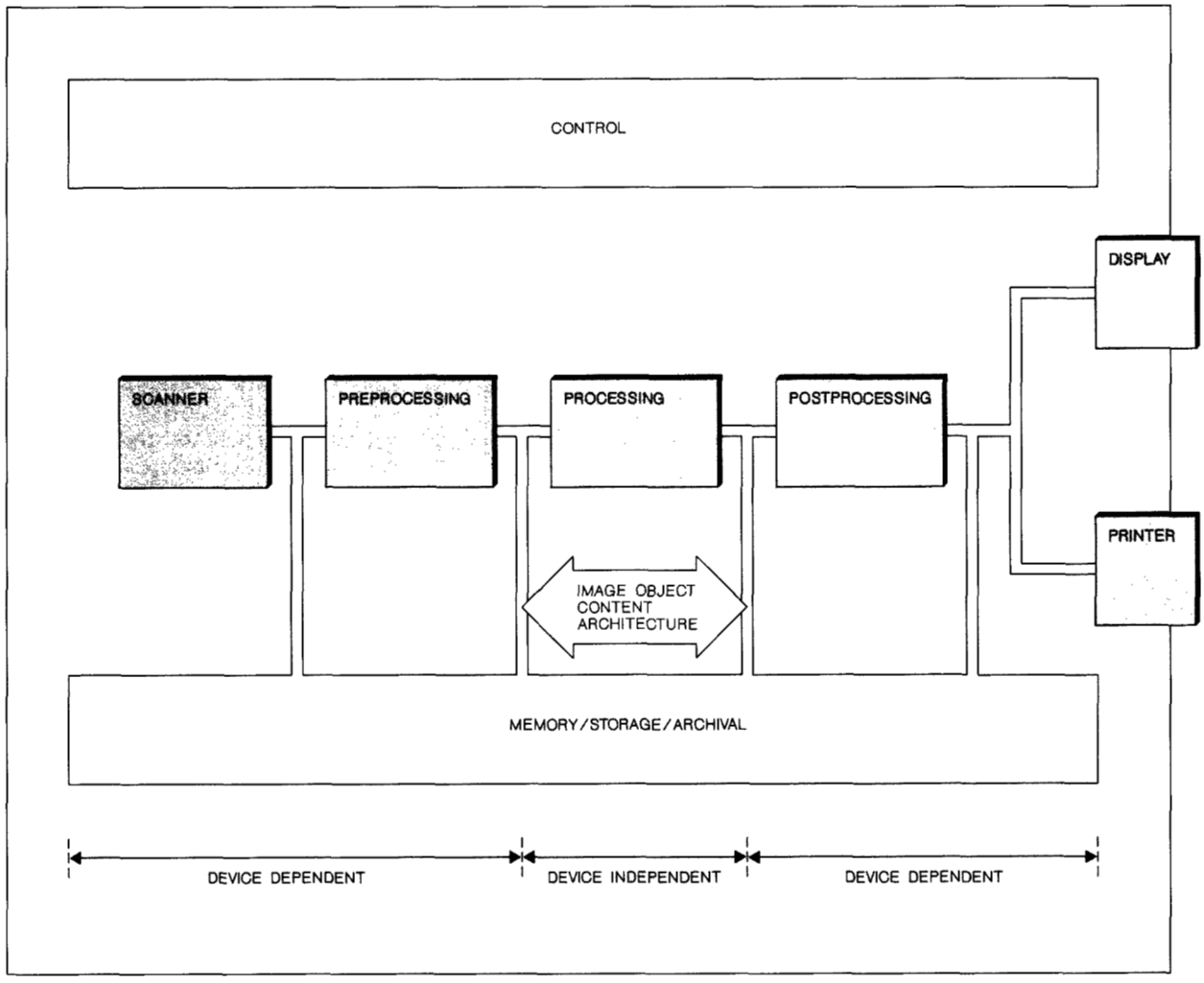


Figure 2 Steps in image processing



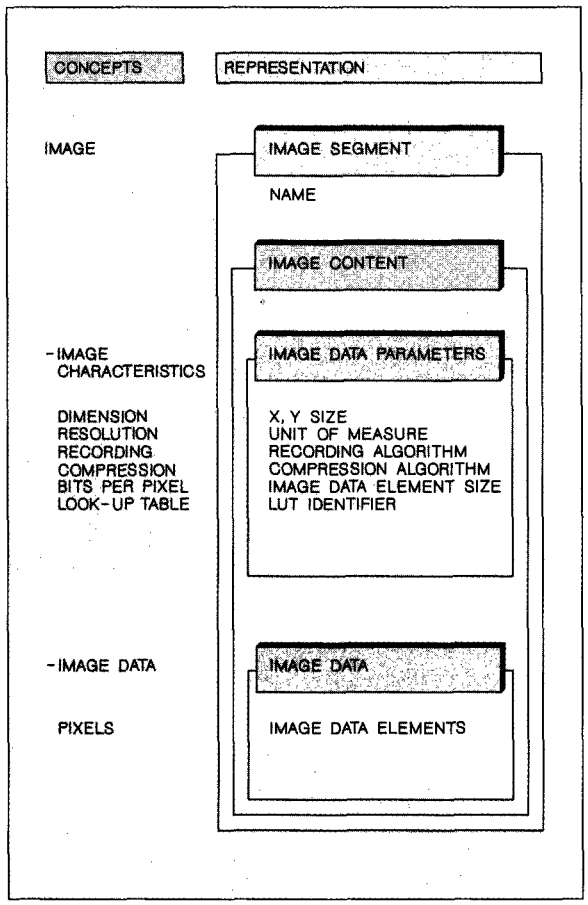

## Scope of the architecture

In order to design an architecture that may be used by any and all image applications and devices, let us first look into the steps involved in image processing.

All image applications or image processing systems must be able to scan, store, retrieve, display, print, or exchange (send and receive) images within or across image applications, or perform some combination of these activities. The activities can be generalized as create, exchange/store/retrieve, and output. These processes culminate in the term *image processing*. A model of image processing is depicted in Figure 2. Details of each of the five steps defined by the model are described next.

1. Creation—A scanner or similar input device as well as a program may be used to create an image. The creation step is supported by many types of devices and technologies such as scanners and video cameras. The resulting image created during this step will normally contain device-dependent information.
2. Preprocessing—In the model, preprocessing is the gateway from the input devices. In this step, device-dependent information is removed from the image. For example, if the image was created by scanning a document, the end-of-scan-line code that was inserted during the creation step (step 1) is removed. As a result, an image is generated with such characteristics as resolution

Figure 3 Image representation




and size, which do not depend on the method by which the image was captured.

3. Processing—Processing is performed on the device-independent image. Processing includes actions such as storing the image, retrieving the image, distributing the image, modifying the image, and using the image in an application (e.g., character recognition). The image is in an interchangeable form where all device characteristics are absent. Images in this form can be passed to another image system or environment and interpreted consistently.
4. Postprocessing—Postprocessing is the gateway to applications that support output devices. The required device-control information is inserted, and a device-dependent image is generated. There is one postprocessing step for each type of device.
5. Output—This step presents the image to the user and is controlled locally by the device. The device can be a display, printer, or other type of output device.

The scope of IOCA is bound by step 3. At this step, the image is free of all application and device-unique features. It is thus in a universally recognizable form, unbiased by special features inserted for performance or for other reasons unique to a particular application or device. It is thus in an optimum form for standardization.

The term *IOCA processor* may be used to describe the processing that occurs at this step. All other steps in the image processing model deal with device-dependent features and are thus called *controlling environments*. The IOCA processor can be thought of as interfacing with the controlling environment via a data stream, communication line, or I/O device.

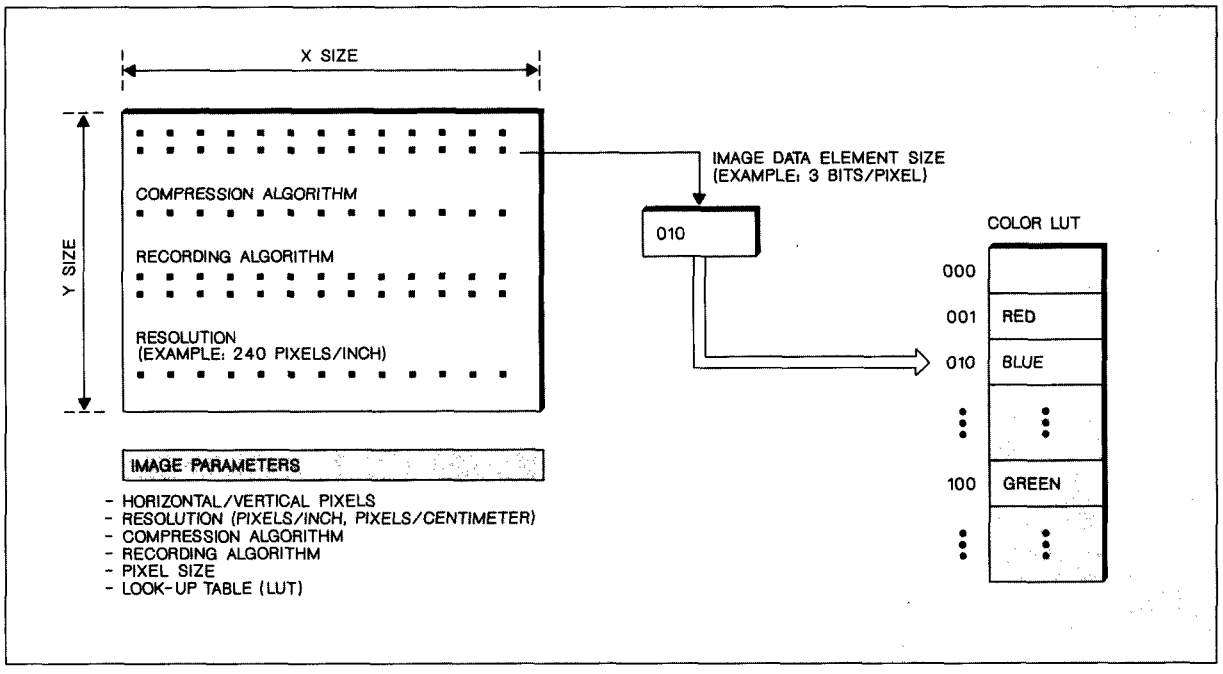
### Architecture description

The base structure used to represent an image is called the *image segment*. The image segment is the IOCA structure that serves as both input to the IOCA processor and output from the IOCA processor. It is also the structure used to represent images that are passed to other image applications. The structure of the image segment consists of image data parameters and image data.

The image data parameters are used to describe the following image characteristics which are required for the representation of images (Figure 3):

- Image dimension and resolution—It is necessary to know the number of pixels in the image in both the horizontal and vertical directions. Reconstructing the image to its original size also requires that the distance between pixels in both the horizontal and vertical directions be known. This distance is the resolution of the image. If an image is scanned at a resolution of 300 pixels per inch and is printed by a printer at 240 pixels per inch without any modification in resolution, the original image will be printed pixel for pixel resulting in an image enlarged by a ratio of 5:4, which is the scanning-to-printing resolution ratio. If the image is to be printed at its original size, the pixels in the scanned image must be reduced by a ratio of 4:5, which is the printing-to-scanning resolution ratio.

**Figure 4 Image parameters**

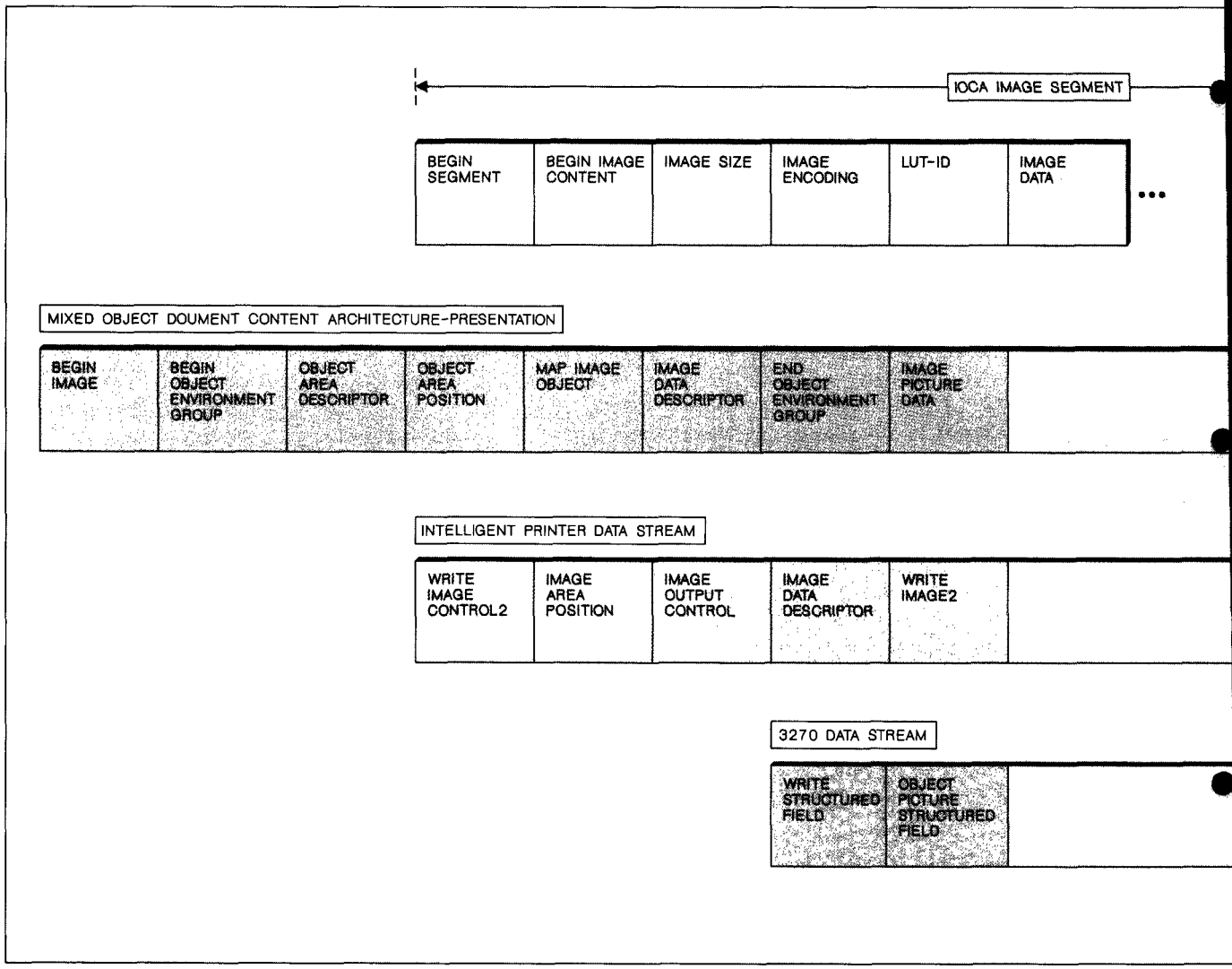



- Recording sequence—Images are often recorded from left to right and from top to bottom. However, implementations may at times record from left to right but from the bottom up. Such information is characterized as a part of the recording algorithm. Support for both forms of recording is defined by the architecture.
- Compression—Compression is the technique used to decrease the size of the image data. Without knowledge of the compression technique used to compress the data, reconstruction of the image would be virtually impossible by another application or device. The two compression algorithms currently supported by the architecture are the IBM Modified CCITT (International Telegraph and Telephone Consultative Committee) Modified READ algorithm (IBM MMR)[9] and the CCITT T.6 Group 4 Facsimile Coding Scheme[10] in addition to the support of uncompressed data.
- Bits per pixel—The number of bits that comprise each pixel may vary, depending on whether the pixel represents a bilevel, gray-scale, or color image. The "image data element size" is used by this architecture to determine the number of bits used to represent each pixel for a particular image (image data element is synonymous with pixel).
- Pixel structure—Two methods exist for representing color information for a color image. One is the interpretation of the color of a pixel based on the combination of colors represented intrinsically by the value of the pixel. A particular color model such as RGB (Red, Green, Blue) is chosen, and a number of the bits of each pixel are used to define the intensity of a particular color element (such as red, green, or blue). The second method is described in the next characteristic.
- Look-up table—The second method for representing color information is to use a look-up table (LUT). The pixel value is interpreted as an index in a table. The data at the indexed location in the table is the color value for the pixel. An example of this method is shown in Figure 4 where a pixel value of "010" represents a "blue" pixel.

Each of the image data parameters and the image data are embodied in the form of self-identifying, self-contained structures. Thus, the design of all of the structures is simple since each proves to be an entity of its own and can easily be extended. The syntax and semantics of each structure are defined by the architecture, providing consistency of generation and interpretation by all IOCA supporting products.

As image technologies make further progress, future expansion of the architecture necessitated by new requirements may be accomplished by enhancing

**Figure 5 IOCA in MO:DCA-P, IPDS, and 3270 Data Stream**



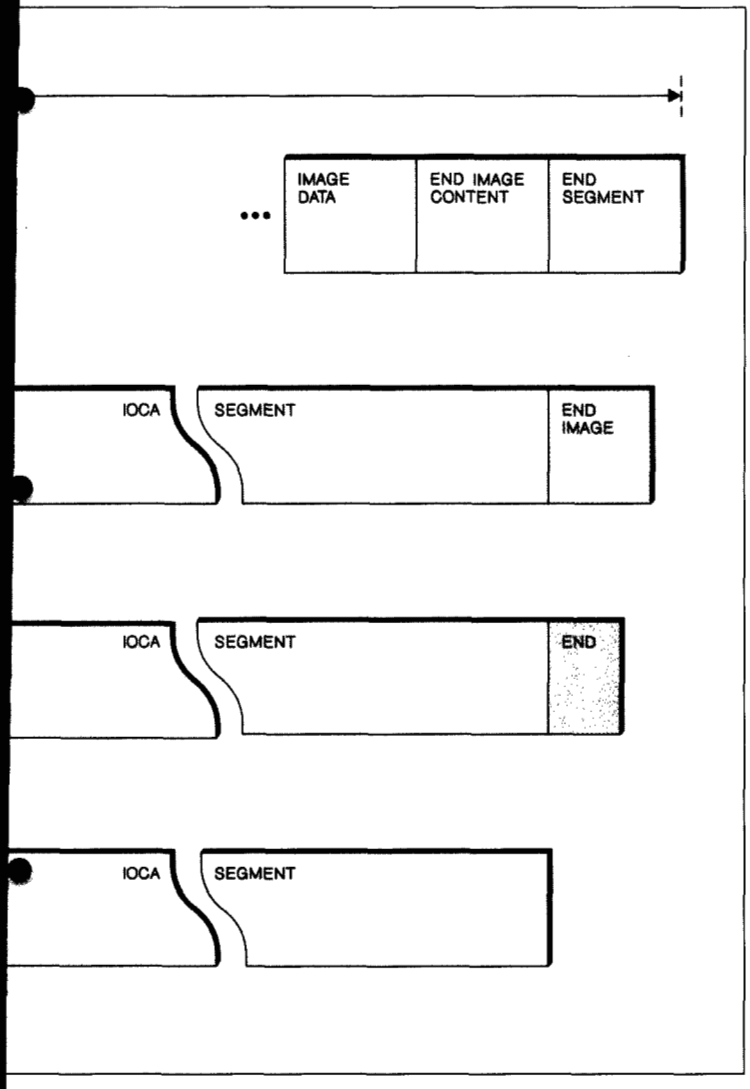

currently defined structures and defining new structures to the architecture. Both ways may be realized without affecting the design of the architecture.

The same image segment is carried intact within the three data stream environments used by image systems, as described next.

### Relationship to data stream environments

As described earlier, the three data stream environments used by IBM image processing systems are:

- Mixed Object Document Content Architecture-Presentation—MO:DCA-P is used to interchange documents containing images or images and text between application programs within or among image processing systems. The architecture is intended to present the contents of the document on a workstation or on a printer.
- Intelligent Printer Data Stream—IPDS is used to send image data or image and text data that have been created by an application program to an APA printer. Images may be printed by themselves or along with text.




- 3270 Data Stream—The 3270 Data Stream is used to transmit data between an application program and a workstation or printer. Within an image processing system, a nonprogrammable workstation may be used to display both images and alphanumeric data with this data stream.

The primary reason that IBM has separated the description of the objects (such as image and text) and the data stream environments is so that the definition of the imbedded data type is independent of the data stream by which it is carried. Figure 5 is an example of the same IOCA image segment carried within each of the three data stream environments.

### IOCA SAA function sets

IOCA has defined two function sets as a part of SAA. The function sets have been named Function Set 10 (FS 10) and Function Set 20 (FS 20). Both are intended for the interchange of images within and among the four operating environments as defined in SAA: MVS/TSO/E, CMS/VM, OS/400, and OS/2.

FS 10 enables the interchange of bilevel images. The two compression algorithms currently supported by this function set are the IBM Modified CCITT Modified READ algorithm and the CCITT T.6 Group 4 Facsimile Coding Scheme in addition to uncompressed data. This function set is a proper subset of FS 20 and is currently supported by the ImagePlus™ system.

FS 20 enables the interchange of up to 24 bits-per-pixel color as well as gray-scale and bilevel images. Compression is applicable only to bilevel images. The choice of compression algorithms is the same as for FS 10.

All products that conform to one of these two IOCA function sets are also classified as either an IOCA generator or receiver.

A product that is classified as a *generator* of a particular function set accepts only a subset of the IOCA parameters and values that are defined in the corresponding function set definition and nothing else. Thus, all accepted or generated parameters and values remain within the bounds of the defined function set.

Products that are classified as a *receiver* of a particular function set must be capable of accepting any IOCA image that conforms to the corresponding function set, although it may use only a subset of those parameters and values to generate images. A receiver will therefore have the advantage of being able to receive any IOCA image that is generated by a generator of the same function set.

### Summary

The need for a single consistent mechanism for representing images by image processing systems resulted in the development of Image Object Content Architecture, as described in this paper. The IOCA requirements have been discussed, the concepts de-

scribed, its relationship to the data stream environments used by image processing systems explained, and the function sets defined for SAA introduced.

IOCA will continue to be enhanced as new technologies and new requirements for image processing are encountered by systems that use images.


## Acknowledgments

The author wishes to acknowledge all of the architects and product representatives who contributed to the development of this architecture. In particular, special gratitude is owed to those who helped prepare this paper: Takeharu Mizukoshi for his timely reviews; Eihiko Tokunaga for his management guidance; Von Tucker, Dave Stone, and Andy Maholick for providing details on MO:DCA-P, IPDS, and the 3270 Data Stream, respectively; and Lauren Kingman for reviewing the drafts and for providing the author with the opportunity to publish this paper.




## Cited references


1. *Systems Application Architecture: An Overview*, GC26-4341, IBM Corporation; available through IBM branch offices.
2. *Systems Application Architecture: Common Communications Support Summary*, GC31-6810, IBM Corporation; available through IBM branch offices.
3. *Mixed Object Document Content Architecture Reference*, SC31-6802, IBM Corporation; available through IBM branch offices.
4. *Intelligent Printer Data Stream Reference*, S544-3417, IBM Corporation; available through IBM branch offices.
5. *3270 Information Display System Data Stream Programmer's Reference*, GA23-0059, IBM Corporation; available through IBM branch offices.
6. *Image Object Content Architecture Reference*, SC31-6805, IBM Corporation; available through IBM branch offices.
7. *Architectures for Object Interchange*, GG24-3296, IBM Corporation; available through IBM branch offices.
8. R. M. Helms, "Introduction to Image Technology," *IBM Systems Journal* **29,** No. 3, 313–332 (1990, this issue).
9. K. L. Anderson, F. C. Mintzer, G. Goertzel, J. L. Mitchell, K. S. Pennington, and W. B. Pennebaker, "Binary-Image-Manipulation Algorithms in the Image View Facility," *IBM Journal of Research and Development* **31,** No. 1, 16–31 (January 1987).
10. "CCITT Recommendation T.6," *CCITT T Series*, Volume VII - Fascicle VII.3, International Telecommunication Union, Geneva (1985).



**Yuji Hakeda** *IBM World Trade Asia Corporation, 1623-14, Shimotsuruma, Yamato-shi, Kanagawa-ken 242, Japan.* Mr. Hakeda is a senior associate programmer in the Office and Image Systems group, Manufacturing and Development, IBM Asia Pacific. He joined IBM Japan in 1982 as a development programmer and participated in the development of products such as the IBM 8815 Scanmaster I and the IBM 3193 Display Station. In 1984, he became a member of the architecture group and served as lead architect in the development of the Image Object Content Architecture. Since 1989, he has been involved in the design of facsimile systems and other image-related systems. Mr. Hakeda received his B.A. in computer science from Columbia College, Columbia University.



Reprint Order No. G321-5403.