This paper describes an architectural approach that provides information interchange across a broad spectrum of user applications and office automation offerings. Some of the architectures described herein are currently implemented in existing IBM products. These and other architectures will provide the basis for document interchange capability between products such as the IBM 5520 Administrative System, the IBM System/370 Distributed Office Support System (DISOSS), and the IBM Displaywriter System. Specifically described is a document distribution architecture and its associated data streams. Transforms can be utilized to interchange between these data streams and others.

A general overview of the architectures as opposed to a detailed technical description is provided. The architectures described are protocols for interchange between application processes; they do not address the specific user interface. The document distribution architectures utilize SNA for data transmission and communications control facilities.

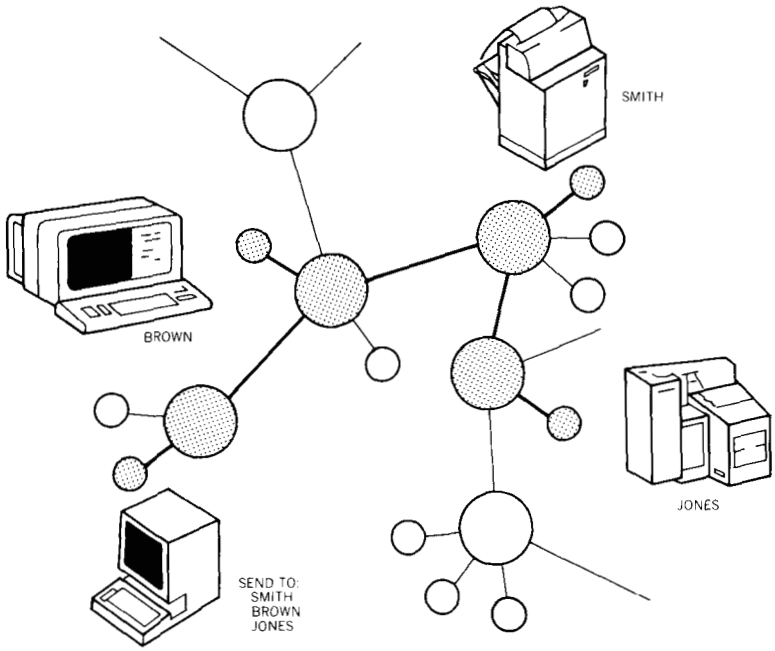
Electronic information interchange in an office environment

by M. R. DeSousa

The desire to interchange office information electronically has been with us since Samuel Morse invented the telegraph in the 1840s. Today, "electronic document distribution" is a business buzzword; professional journals and trade magazines abound with references to the automated office, electronic document distribution, and communication networks. The office-information-interchange system is fast becoming a reality.

The office-information-interchange system is envisioned to work in a network such as the one shown in Figure 1. Such a network is a complex interconnection of systems of various capabilities and a large number of different terminals, or work stations, performing an assortment of applications. A major problem that such a network presents is the variety of interfaces and data forms that must be accommodated to functionally interconnect such devices into an operational information-interchange system.

Figure 1 Office information interchange network



system requirements

Not only must the devices be functionally interconnected, but the office-information-interchange system must provide the basic capability to:

- Enter and edit information
- Distribute information
- Print (or display) information

The "enter and edit" capability allows an originator to create information that must be portable between devices of differing functional characteristics while maintaining the capability to perform additional editing and revision.

The "distribute" capability requires a set of processes that communicate directions and interchange control information and carry the information from an originator, or sender, to a recipient.

The "print" capability, in this context, implies a sender/recipient relationship where the sender wants to distribute information in its final form to the recipient. It is analogous to a traditional mail environment. It is assumed that the recipient has no need to modify or edit the information. To ensure that the information conveys its intended meaning, the sender optionally should have assurance that the information will be printed exactly as specified by the sender (Figure 2).

Figure 2 Print fidelity (Product A creates and prints a document. Because the document is in columnar format, a substitution to the font or space requirement could totally change the meaning of the information. Therefore, when Product A interchanges the document, Products B and C must print the document exactly as Product A did.)

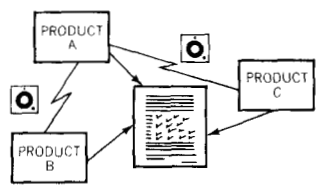


Figure 3 Information interchange system architectures

It is not necessary that each device within the network provide all three basic capabilities. For example, a device that is capable of distributing information need not be capable of creating nor printing it.

Irrespective of the configuration, the system must be easy to use if it is to be effective. The complexities of interfaces and data forms must be transparent to the users. To expect a sender of information to know the device and data-form requirements is unrealistic. The sender should be able to request that the information be distributed and should not have to be concerned about the devices used by the recipients.

The approach

A set of architectures has been defined that specifies the control of information (referred to as the data stream), the document distribution application, the communication transmissions, and any removable storage media being used to manually interchange information (Figure 3). The data stream architectures specify the form of the information by describing the syntax and meaning of allowable elements in the data stream.

The document distribution, or interchange, architecture permits information to be carried from an originator to a recipient without requiring that both be interactively communicating during the distribution process. Further, it allows an originator to send information (a document) to multiple recipients with a single distribution request. And, finally, the distribution architecture provides for services such as security, safe storage during the distribution process, and confirmation of delivery.

The architecture required for transmission control already exists, and it is IBM's Systems Network Architecture (SNA).[1-5] For information interchange via removable storage media, there is a requirement for an interchange diskette architecture. Neither SNA nor the diskette architecture requirement is described in this paper.

A way of putting these architectures into the proper perspective is to liken SNA to a postman, the distribution architecture to an envelope, and the data stream to a message or letter within the envelope.

Because of the unique functional requirements needed to perform both enter/edit and print in a single data stream, it was clear that both capabilities could not be satisfied through a single architecture. The information-creation process requires functional richness which would not easily be transformed into a presentation data stream at a printer or display. The print process requires simplicity if it is to be compatible with a wide range of output devices.

An enter and edit data stream should also be portable between differing devices of unlike functional levels. For example, if information is entered on device A and sent to device B for additional entry and editing, device B should perform identically those functions that both devices support and must recognize those functions it does not support and provide a responsible, appropriate response. With a single data stream, any reasonable approach that would permit interchange on devices of unlike functional levels would preclude guaranteed print fidelity.

A print data stream can be much simpler than an edit and enter data stream because complex editing and formatting functions are not required. This is explained in more detail later. But, with the potential combination of text and other forms of data, the print data stream becomes more complex.

specific DCAs

In order to separate the complexities deriving from differing data stream requirements, it was concluded that architectures for both revisable-form and final-form data streams were required. These architectures are generically referred to as Document Content Architectures or DCAs. The specific DCAs described are the Revisable-Form DCA, the Final-Form Text DCA, and the Final-Form Mixed Data DCA.

The revisable-form data stream is text and has not yet been transformed into its final form. The text, although sufficiently formatted for presentation, can still "flow" as the result of additions and deletions. The revisable-form data stream is used when editing text or when distributing text for revision.

This paper describes only one revisable-form data stream. This is not to imply that this DCA is the only DCA that can be used in an interchange system. Several revisable-form data streams already exist. Some of those generally used are the IBM Document Composition Facility Program Product, the IBM Generalized Markup Language (GML) Program Product, the IBM Script/370, and the

IBM 3730 Distributed Office Communications System text data stream. These data streams are found in IBM's data processing systems. Because these data streams may coexist in an information interchange system with the DCAs described herein, transforms within the system may be required to ensure information interchange of revisable information for distributed document development. IBM's other text-processing (or word-processing) products such as the Displaywriter use the Revisable-Form DCA described herein.

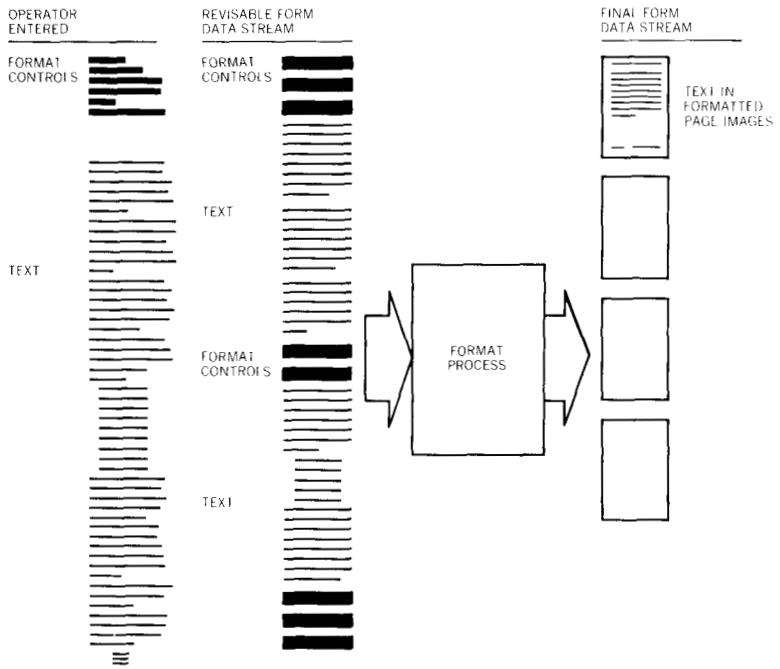
The final-form text data stream is text that has already been formatted and is ready for presentation. This data stream is used where the recipient merely reads the information and files or discards it. It allows the interpretation of graphic characters and of conventional text-processing controls such as carrier return and tab. Such IBM products as the 5520 Administrative System and the Displaywriter use the Final-Form Text DCA described later.

The final-form mixed data type data stream is also formatted and is ready for presentation. Like final-form text, this data stream is intended primarily for the read-only application but differs in that it supports combinations of data types within the same data stream, e.g., text and image.

The final-form DCAs are suitable for use as a device interface format and are independent of any specific device characteristics. All devices that participate in an information-interchange system must support a final-form DCA that guarantees print fidelity when requested.

Figure 4 shows the relationship of the revisable and final-form DCAs. In the revisable form, page-width and page-depth information are carried as general formatting controls at the start of the document and at specific locations within the data stream. Line-ending and page-ending decisions are made to permit viewing during the editing process, but the decisions are subject to change during subsequent processing. The revisable form may also include items such as margin text and pointers to external text for inclusion. The information is not sufficiently formed for final presentation, but it is well-suited for editing because a single change to the margin text, for example, can apply to the complete document. Once the document is transformed into its final form, page and line endings are permanently fixed, and the information is in page image format. Complex formatting controls such as headings and footings, page numbering, and footnotes, which were permitted in the revisable-form data stream, have been resolved and do not appear in the final-form data stream. Again using margin text as the example, when it is placed in its appropriate location on each page as text, a change to the margin text will require a change on every page. It is this transformation of complex for-

Figure 4 Relationship of revisable-form and final-form data streams



matting controls into simple printable text that makes the final-form data streams acceptable to a broad spectrum of output devices. It is this same transformation, however, that makes the final-form data streams unsuitable for editing because the originator's format "intent" is lost.

The discussion of transforms, thus far, has been restricted to the transformation of the revisable-form data stream into the final-form data stream. Because there is more than one data stream in both the revisable-form and final-form applications areas, transforms within each application may also be required. In final form, transformations to and from final-form text and final-form mixed data types will be required. For example, a text-only document distributed as a final-form text data stream will require a transformation when the output is produced on a device whose data stream is final-form mixed data types. In some instances the end user may be involved in the transformation process, but final form transforms should be transparent.

Revisable-Form DCA

The Revisable-Form DCA provides for the interchange of editable documents. Interchange is defined as the proper interpretation of the data stream, but not necessarily complete execution of the function implied or expressed. If a function is performed as speci-

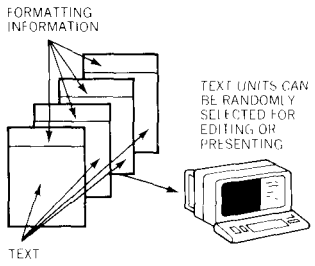
fied or is signaled when not performed, the interpretation is proper. Subsequent editing may then be used to produce a document that may be processed on either the originating or the receiving system.

The means to accomplishing interchange is defined by an architecture that enforces a standard definition of syntax and semantics for reformattable, editable text. The architecture specifies a data stream organization that:

- Guarantees interchange among implementing systems.
- Provides sufficient structure and redundancy in the data stream to support nonsequential access while allowing for identical interpretation whether processed sequentially or nonsequentially.
- Allows format declarations while remaining unformatted, thereby preserving for a user the capability to revise both format declarations and textual content.
- Provides sufficient information so that the document can be printed or displayed in its revisable form (as entered), or, through appropriate processing, in its final-formatted form, or in any form between these two extremes that a system chooses to implement.

The Revisable-Form DCA is specifically designed to support two text-processing environments; *document development* and *distributed document development*. The document development environment is the initial phase of producing a document: the entering of text and format controls into a text-processing system and the editing of individual sections of a document. The distributed document development environment is the distribution of documents to different work stations or text-processing systems because of load balancing or product capability limitations, or to allow revision at a remote location.

Figure 5 Text units in the Revisable-Form DCA

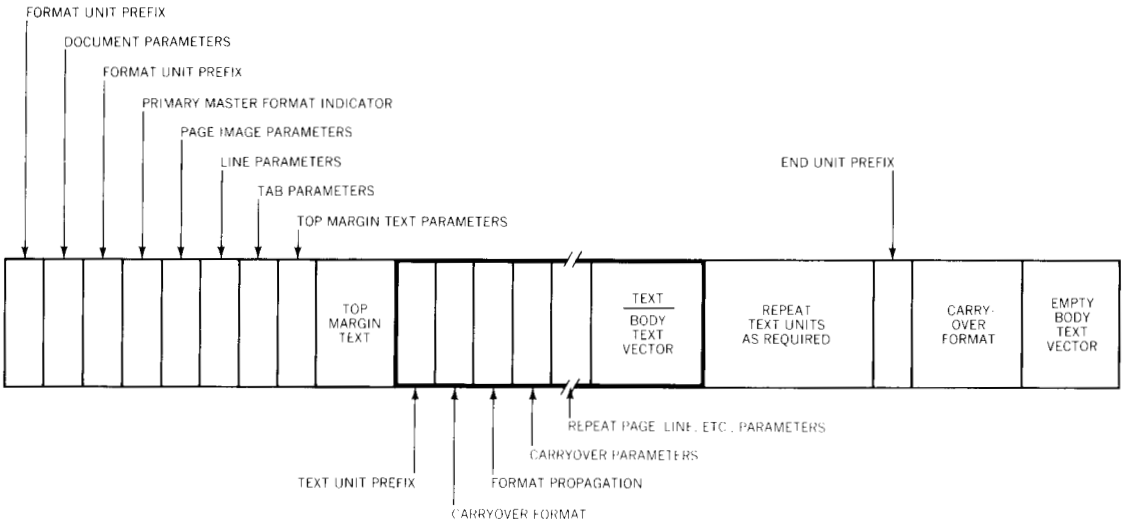


The Revisable-Form DCA supports these applications by defining a common text-processing syntax and semantics and by providing structures for random access of documents on a piecemeal basis, with formatting parameters carried as an integral part of each of these units.

The Revisable-Form DCA data stream is composed of revisable text, embedded text-processing controls, and formatting declarations in the form of structured fields. Each structured field has an *introducer* that contains the length of the declaration, its class, type, and format. This introducer makes all parts of the DCA self-describing.

The content of the revisable document is stored in pieces that are called *text units*. The concept of text units is illustrated in Figure

Figure 6 Revisable-form data stream organization



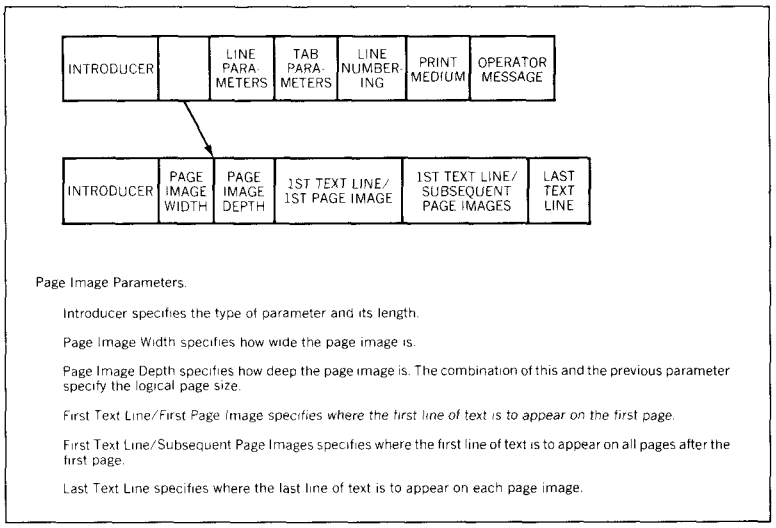
5. Each text unit starts with the formatting parameters required to format that piece of the document. Thus, an originator can select a text unit at random and can display or print that unit as it would appear if the entire document had been processed sequentially.

The revisable-form data stream organization is shown in Figure 6 with a text unit highlighted. The information that precedes the text unit can be thought of as document initialization information. Note that margin text is specified outside of the text unit. A carry-over format and empty-body text vector are appended to the end of the data stream for ease of document extension.

The structure of the data stream provides the capability for dynamic formatting while editing at a display; i.e., tab settings can be changed and the effect of the change made visible immediately. The structure also provides the capability to retain the originator's "intent" during entry or editing. Again, by way of example, if the originator adds to or deletes words from a centered statement, the modification is made and centering occurs without additional instruction from the originator.

There may be instances when a device does not fully understand the revisable-form data stream it receives because of functional differences. Because of this possibility, all participating devices must detect differences and handle them as exception conditions. This requirement means that all devices must detect and report the nature and location of the exception condition and recover from the exception condition.

Figure 7 Master format declaration



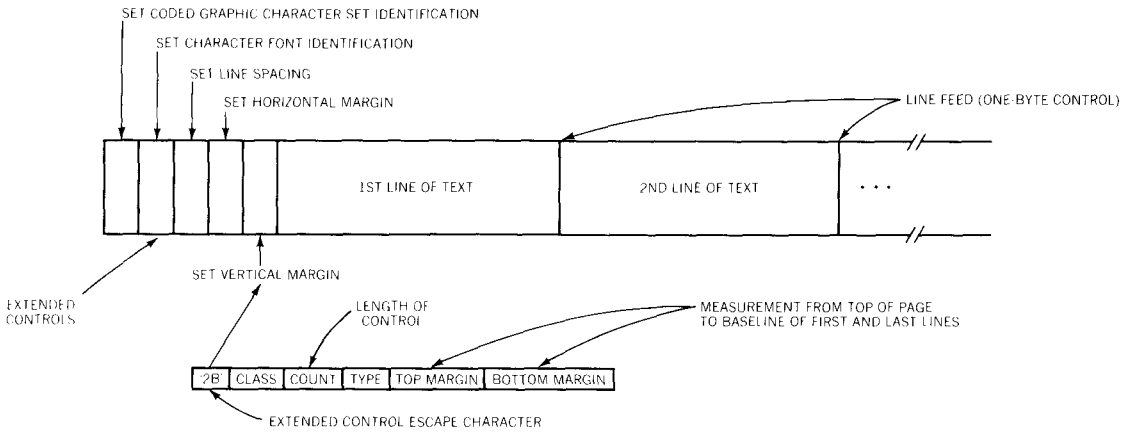
The data stream consists of standard EBCDIC one-byte controls, a set of multiple-byte controls, and a set of self-describing structured fields. It is through use of the structures that the state of the format is maintained and known anywhere within the document and that random access and processing of the revisable text is possible.

One structure—the master format declaration—is described in detail to provide some insight into the architecture. The master format is a statement of the general appearance of page images that will ultimately be put on a display or a piece of paper. Two master formats can be specified in a document: the Primary Master Format and the Alternate Master Format. Figure 7 shows the general content of a master format declaration. The introducer of the declaration specifies whether it is a primary or alternate master format. Within the figure, only the Page Image Parameters field is shown in detail.

Final-Form Text DCA

The Final-Form Text DCA specifies the representation of formatted text information for interchange using communications facilities. This architecture guarantees print fidelity when requested. It is suitable for use as a device interface format and is independent of any specific device characteristics. It provides a simple data stream structure capable of being processed sequentially by synchronous devices.

Figure 8 Final-form text data stream organization



The architecture provides the definition of text and format control function to format and print a document. *Text*, as defined here, means an ordered string of characters (graphic symbols) that are obviously suitable for the specified purpose of representing coherent information. Text is further ordered into units of composition and presentation referred to as *lines*. The lines of text, when assembled into an ordered finite collection, will comprise a presentation unit called a *page*. A single page or a group of pages will comprise a document that is the object or unit of transfer for interchange. The term *print* used here will include displaying a document on a volatile medium such as a video display, reproducing a permanent image on paper or photo-sensitive media, as with impact, ink jet, or photo printers, or recording the document image on magnetic media such as diskettes.

Control functions are designated by specific control codes within the character set used for the text string. The graphic symbols assigned to a text character set are explicitly noncoincident with any of the codes assigned to control functions to prevent obvious ambiguity. The control codes are imbedded within the text at specific positions where a control function is to be activated to produce a desired result with the document in its presented form. Control functions may also activate a state condition for a process algorithm or device action and may be used to instruct an operator about how to operate a device. All the controls that are supported in the Final-Form Text DCA are either EBCDIC formatting controls or extended multibyte controls.

Figure 8 shows the organization of the final-form text data stream. The Set Vertical Margin extended control is shown in detail.

Generally, the EBCDIC one-byte controls provide the basic functions, such as line end, backspace, and indent, and have a one-time, immediate effect. The extended controls are more global and provide for functions such as line spacing, horizontal and vertical margins, and tab setting. A few of the extended controls remain in effect until the line-end control, but most remain in effect throughout the document or until they are reset.

In Figure 2, the concept of print fidelity was introduced. The ability to guarantee exact reproduction is provided through an exception action control. A user can specify what level of deviation he will permit. If no deviation is permitted because of the nature of the information that is to be presented, the user can specify that the information must be presented as specified or the presentation must be terminated. This requirement can be placed on the entire document or at critical points within the document.

Final-Form Mixed Data Type DCA

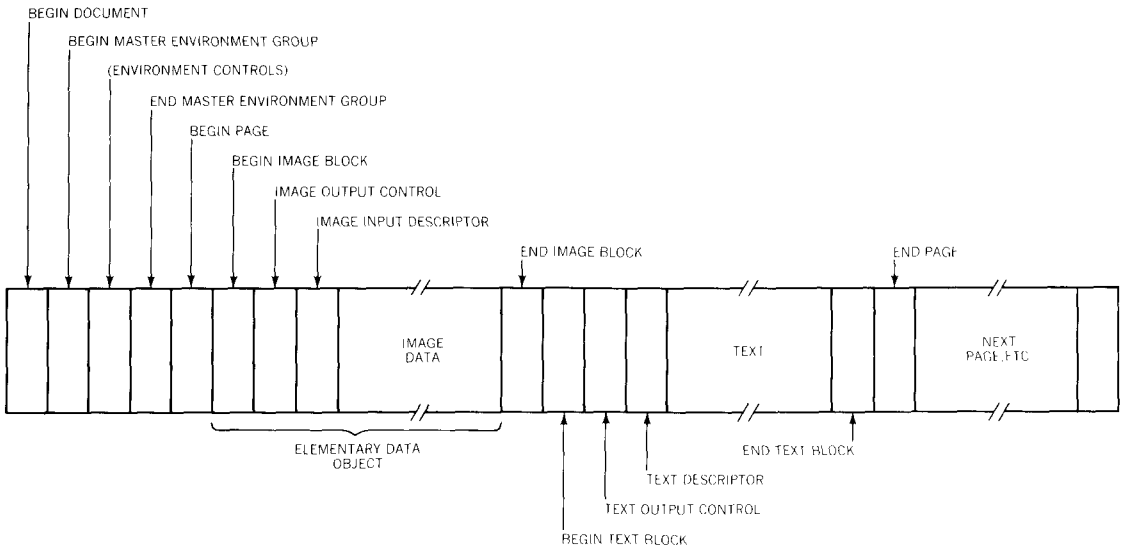
The Final-Form Mixed Data Type DCA consists of a contiguous sequence of structured fields; its structured field organization is similar to that of the revisable-form data stream. Certain of these fields can be grouped into objects that collectively form a document. The objects are bounded by appropriate begin and end structured fields. For example, a document starts with a Begin Document structured field and terminates with an End Document structured field.

The most fundamental object is an elementary data object. This object contains the information to be printed or displayed.

An elementary data object is bounded by a begin-block and an end-block structured field. Within an elementary data object, two additional varieties of structured fields can exist. One contains the actual data, for example, image raster data or text data; the other, which is a collection of structured fields, contains environment control information that describes the autonomous characteristics of the data. The conditions established by the environment control information only apply to the data within the object. The environment is "scoped" by the begin and end structured fields of the object.

An elementary data object is thus a bounded, self-contained unit that is composed of the elementary data to be presented and a complete description of the characteristics of the data. The elementary data object in this form is independent; it is never governed or influenced by data characteristics specified outside of the elementary data object. Only when the data characteristics are not explicitly specified within the elementary data object is

Figure 9 Final-form mixed data type data stream organization



the data subject to the influence of data characteristics specified outside the elementary data object.

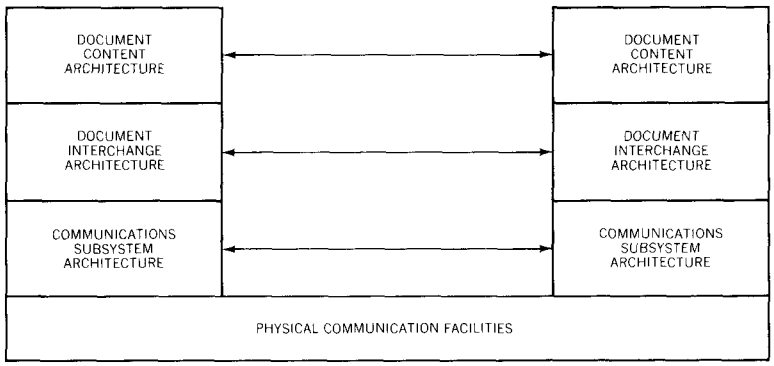
These elementary data objects become the components that form a page. A sample data stream is shown in Figure 9. Two elementary data objects are represented. Objects have a spatial relationship to the page in which they are contained but no relationship to each other.

The Final-Form Mixed Data Type DCA has three hierarchical levels: document, page, and elementary data object. Within this hierarchical scheme, control functions are applicable at particular levels. For example, the control function for media exists at a higher level than a page because multiple pages may appear on a single element of media.

As stated before, if the environment control information is explicitly specified within an object, the environment is restricted, or its "scope" is, to that object. If the environment control information is not specified, the environment for that object is taken from the next higher level in the hierarchy. This is referred to as "factoring."

To understand factoring, consider a mixed-data-type document in its simplest form, for example, a multiple-page document that is created by an inexpensive image scanner. All pages are identical in size, all pages contain one image with no other elementary data

Figure 10 DIA architectural layers



objects, and all images have identical data characteristics. Because the elementary data object environment control information for each image is identical, it can be specified once at the document level as opposed to once per image. In this example, the environment control information for all elementary data objects can be placed in a structured field called the *Master Environment Group*.

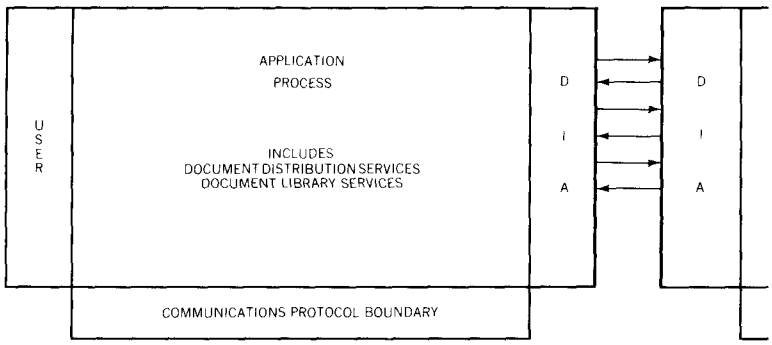
Every elementary data object that is within the scope of this Master Environment Group, but does not specify its own environment control information, uses the Master Environment Group as though the information had appeared in the elementary data object. Any elementary data object environment control information that appears within an elementary data object overrides the corresponding information in the Master Environment Group for the duration of that elementary data object. This maintains the integrity of the elementary data object as an independent entity.

The net effect is that the general case is allowed to appear at a higher level in the hierarchy. Any time it is necessary to override the general case, it can be accomplished by inserting the environment control information at the appropriate lower level in the hierarchy. Such environment control information remains in effect until the scope for that level of the hierarchy is reached.

Document Interchange Architecture

The Document Interchange Architecture (DIA) specifies how devices are to interchange intentions and data. It provides the capability to invoke the distribution services and the library services that are required of an interchange system. DIA specifies the rules and a data structure that establish the discipline for predictable information interchange between devices. DIA provides the method for systems and devices to interchange documents for a

Figure 11 DIA application process layer interfaces



variety of purposes, independent of the data types DIA contains. Interchange, as defined here, means that a document can be transported from one device to another device without change to its form or coherence. However, through the use of controls that can accompany the document, the document can be processed to produce the results defined by the sender.

DIA prescribes the exchange of information at the DCA layer and is distinct from the adjacent architectural layers: DCA and SNA (Figure 10).

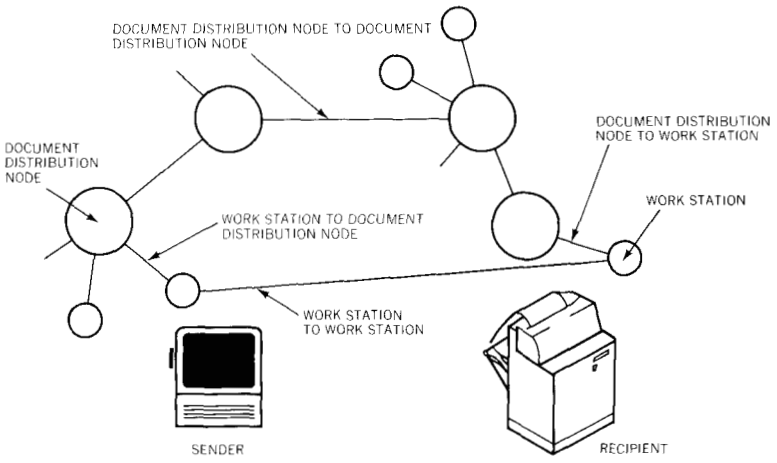
Expanding the DIA layer shows the relationship of DIA to the application process functioning at that layer. In Figure 11, the application process has an interface to the user, an interface to SNA, and an interface to another application process. The Communication Protocol Boundary provides the interface to SNA. DIA specifies the commands and the results expected on the interprocess interface; it does not address itself to either the user interface or the SNA interface. Data in DIA structures may physically pass through the Communication Protocol Boundary interface to reach the other process, but DIA defines the structure, commands, and results expected between the two application processes regardless of the physical path.

DIA is conceptually divided into an information-interchange base and various application services, as shown in Figure 12. The information-interchange base includes the structures and procedures that are common throughout the architecture (e.g., DIA session control, exception recovery, and encryption). DIA session control is the set of procedures and the commands necessary to exchange identification, authentication, functional capability, and status information that are pertinent to the application processes.

Figure 12 DIA structure

Document distribution services, illustrated in Figure 13, support document distribution through work station to document distribution node (DDN), DDN to DDN, and DDN to work station functions.

Figure 13 Document distribution services



Document distribution services also support work station to work station document interchange.

Document library services support the maintenance of documents on storage media. This facility provides the commands for requesting another process to perform operations, such as file, retrieve, and delete, on entire documents. This architecture does not address the manipulation of the internal content of a document.

The capability of the base and the parts may be subsetted to accommodate limited-function, entry-level devices that have communications facilities. This capability provides for orderly migration to more advanced devices.

The structure of DIA is extendable to other services of office automation systems.

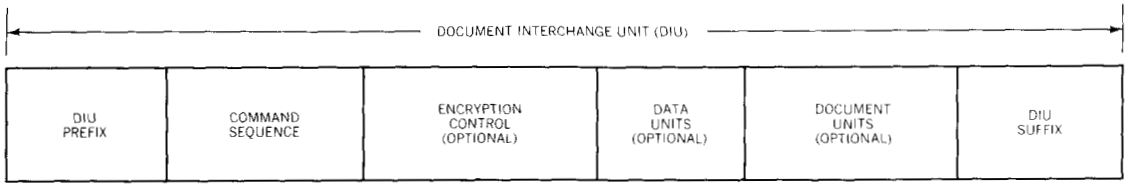
Document Interchange Unit

The Document Interchange Architecture defines a Document Interchange Unit (DIU), which is the major interchange facility between processes (Figure 14). An application process that uses DIA performs the construction of the DIU for sending to another process. The receiving application process interprets the structures of the DIU and performs the requested operations. The processing procedures are not defined by DIA; it is the responsibility of the application process to meet the requirements of the DIA interprocess interface and produce the specified results.

The DIU consists of six logical entities:

- The DIU Prefix introduces and identifies the DIU.
- The Command Sequence contains the commands for application function processing.

Figure 14 Document Interchange Unit



- The Encryption Control contains the information to encipher the data units and document units for security purposes.
- The Data Unit contains information that may be referenced by one or more commands in the Command Sequence.
- The Document Unit contains the document profile, which describes the characteristics of the document and, optionally, the content of the document.
- The DIU Suffix specifies the end of the DIU and indicates whether or not any abnormal conditions affected the DIU transmission.

The DIU, in this information interchange context, is assumed to carry documents that conform to the DCAs discussed earlier. However, the DIU, in general, can transport any type or format of the data object. The DIU carries information that explicitly identifies the object type and the characteristics necessary to process it as the sender intended. If an information interchange object does not conform to the DCAs described, a transformation may be required.

Each DIU component and subcomponent has an introducer that specifies the length and describes the semantics and syntax of that piece of data. Figure 15 shows an overall perspective of the DIU structure.

Again for further insight, the Request Distribution Command, which is used by a work station to request the distribution of a document, is shown in Figure 16. The command is schematically depicted within a DIU.

Concluding remarks

The following scenario is intended to put the various architectural components into perspective and to show how an office-information-interchange system might work. An actual working system could be considerably different from what is outlined here because of office procedures, product mix, and applications to be performed.

Figure 15 DIU overview

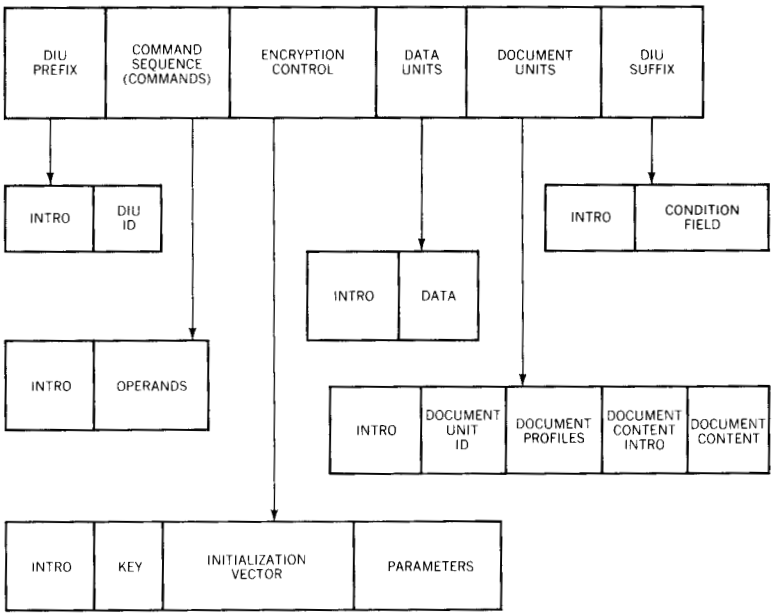


Figure 16 The Request Distribution Command

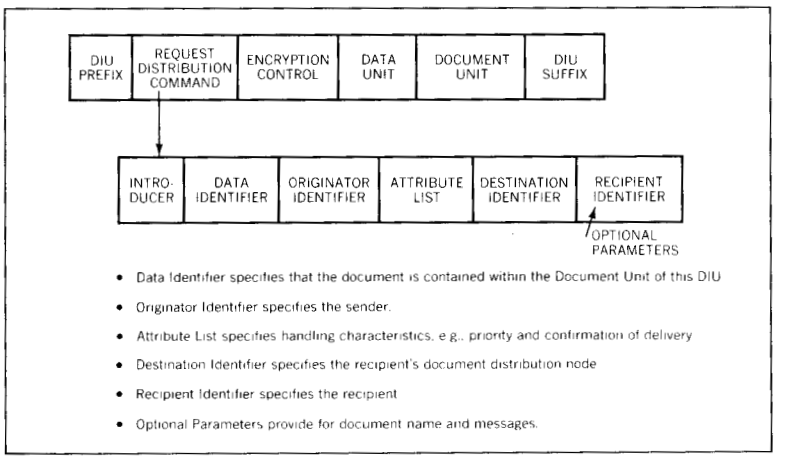


Figure 17 depicts the initial phase of information interchange—document creation. The numbered steps show the order of the operations. A document is started using the revisable-form data stream at Work Station A (Step 1). The document is moved to Work Station B before it is finished (Step 2). It can be moved via communications facilities or diskette. How it is moved will probably be influenced by the physical location of the work stations.

Figure 17 Initial phase of information interchange

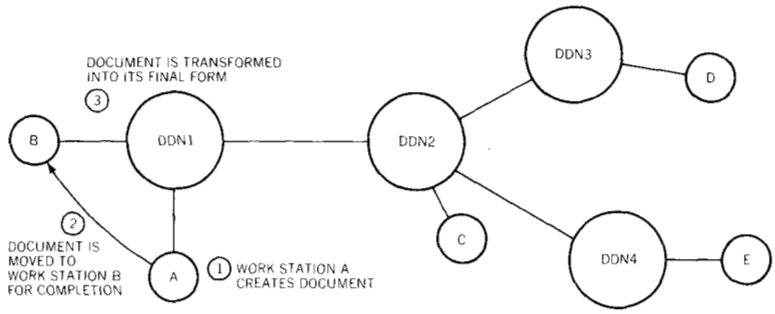
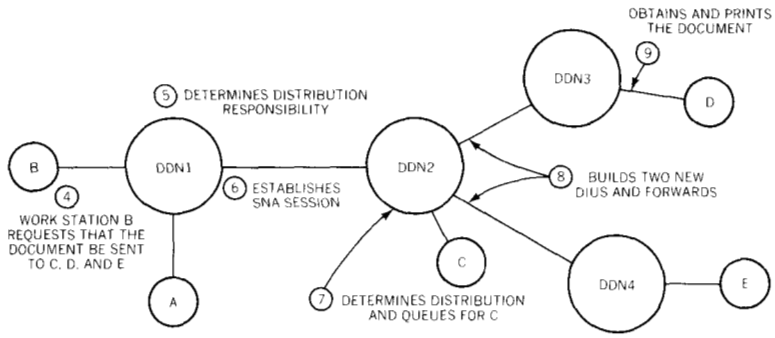


Figure 18 Document distribution information interchange



The two work stations need not be identical devices. After being moved, the document is completed and the data stream is transformed into its final form (Step 3).

Figure 18 depicts the document distribution application. Work Station B requests DDN1 (Document Distribution Node 1) to distribute the document to Work Stations C, D, and E (Step 4). DDN1 determines its distribution responsibility by examining the addresses contained in the request distribution DIU and accessing its destination directory (Step 5). DDN1 established an SNA session with DDN2. The two nodes exchange DIA session control information, and the document is sent to DDN2 (Step 6). DDN2, which has stored the document, determines its distribution responsibility. Because Work Station C is a subscriber at DDN2, DDN2 builds a delivery DIU and queues the DIU until a session is established with Work Station C (Step 7). The destination directory shows that Work Station D must be routed through DDN3 and Work Station E through DDN4. DDN2 builds two new distribute DIUs and forwards them or stores them for later transmission (Step 8). Finally, Work Stations D and E obtain the document from their controlling

nodes and print or display it (Step 9). This simplified presentation demonstrates one way in which the system could work.

summary What has been described is a set of architectures, some already implemented in existing products, that in combination with other IBM architectures will permit information interchange across a broad spectrum of user applications and office automation offerings. Initially, information interchange will be restricted to text with a limited number of devices, but over the next few years, information interchange is expected to become available across a wide offering of systems. It is also expected that data types other than text will be included, over time, for interchange.

ACKNOWLEDGMENT

The architecture developments referred to in this paper were conceived and developed by many contributors who represented several IBM products and development locations. The number of contributors is too large to list them individually, but if it had not been for the dedicated work of these people, this paper would not have been possible.

CITED REFERENCES

1. C. R. Blair and J. P. Gray, "IBM's Systems Network Architecture," *Datamation* **21,** No. 4, 51-56 (April 1975).
2. J. H. McFadyen, "Systems Network Architecture: An overview," *IBM Systems Journal* **15,** No. 1, 4-23 (1976).
3. P. G. Cullum, "The transmission subsystem in Systems Network Architecture," *IBM Systems Journal* **15,** No. 1, 24-38 (1976).
4. J. P. Gray and T. B. McNeill, "SNA multiple-system networking," *IBM Systems Journal* **18,** No. 2, 263-297 (1979).
5. *Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic*, SC30-3112, IBM Corporation; available through IBM branch offices.

GENERAL REFERENCES

J. N. Bruno, "Electronic mail: It gets there fast," *Administrative Management*, 28-70 (September 1979). Discusses the general concept of electronic mail with its inherent product-type and data-form problem.

Dun's Review (August 1979). The entire issue is devoted to the "Office of the Future."

G. H. Engel, J. Groppuso, R. A. Lowenstein, and W. G. Traub, "An office communications system," *IBM Systems Journal* **18,** No. 3, 402-431 (1979). Discusses the applications requirements of an office communications system.

IBM Administrative System, Introduction, GC23-0702, IBM Corporation (September 1979); available through IBM branch offices.

IBM Distributed Office Communications System: Host Programmer's Guide, GA33-3030, IBM Corporation; available through IBM branch offices.

The author is located at the IBM Information Systems Division laboratory, P.O. Box 1900, Boulder, CO 80302.

Reprint Order No, G321-5138.