A wide variety of products for the office is now available, permitting increased automation of office procedures. To realize their full potential, these products must be able to exchange information and control requests with one another. A family of architectures has been defined to satisfy this need. This paper provides an overview of this family of architectures, including their relationship to one another. One member of this family, the Document Interchange Architecture, is described in some detail. An example illustrates use of the family of architectures in an office environment.

The Document Interchange Architecture: A member of a family of architectures in the SNA environment

by T. Schick and R. F. Brockish

Distributed processing is widely recognized as a highly effective general-purpose organization for a computer system[1] and is currently a prime method of automating office procedures. A structure has been defined that provides interchangeability of information and control requests among the distributed processing facilities in an office. The structure consists of several functional layers. This multiple-layered structure defines the Systems Network Architecture (SNA) environment. The benefits of a layered structure are well understood, modular development being an important consequence. Much has been published about SNA; therefore, except for the application layer, we will not describe it here. A detailed description of SNA is provided in References 2 and 3.

The application layer consists of a family of several architectures: Document Interchange Architecture (DIA), Document Content

Architecture (DCA), and Graphic Codepoint Definitions (GCD). An architecture required to develop the full potential of a distributed processing system is now available as DIA. It is a protocol, or language, to be used among applications and permits a variety of applications, or processes, to exchange information and to make control requests of one another in a consistent manner. Defined specifically to satisfy this need, DIA is described in detail in this paper along with its relationship to other members of the family of architectures.

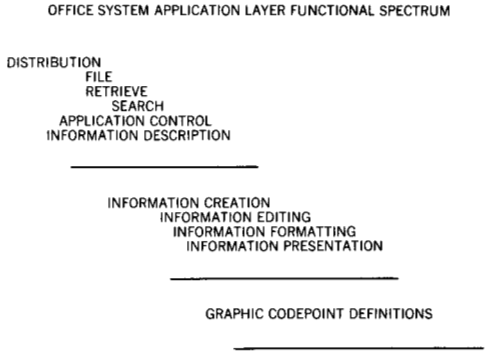
The major objective for developing DIA is to provide the means by which distributed office-application processes interchange requests that act as commands to the system, as, for example, DISTRIBUTE this document, RETRIEVE a particular document, or SEARCH for a document. These requests must be understood by the receiving process exactly as they were intended by the sending process.

DIA provides a general means by which control requests and information may be interchanged among a wide range of machines using a single application-level interchange language. This provision not only permits a centralized process, or product, to interact with many apparently different products in a consistent manner, but it also permits these different products to interact with one another directly using the same interchange language. If there were no common language, each product would need to negotiate a language with every other product with which it interacts. The language problems of building the Tower of Babel would be encountered in a new form.

Office automation is one of the most recent computer applications with a distributed processing orientation. Although DIA could be used as a general-purpose application-to-application protocol, it is oriented to the procedures of the modern office. DIA is only one of several architectures so oriented. Among others are Document Content Architecture (DCA) and Graphic Codepoint Definitions (GCD). DIA encapsulates the information structured according to these architectures. Collectively these architectures are defined to provide interchangeability of information as well as control requests, and to simplify both the system's and the user's view of them.

These architectures form a family and are tailored to the various aspects of an office system. They provide the required system interfaces. These interfaces are intentionally defined to be distinct from the user interfaces of the system, which must be optimized for human factors. The family of architectures is being used by such IBM products for the office as the Distributed Office Support Facility (DOSF),[4] the 5520 Administrative System,[5] and the Distributed Office Support System (DISOSS).[6] Later in this paper a scenario appears that describes the use of this family of architectures in an office environment.

Figure 1 Office system spectrum



A family of architectures

The office contains many different products and types of media. To achieve their full potential there must be a means to permit these products to interact with one another. The previously mentioned family of architectures was defined to satisfy this requirement.

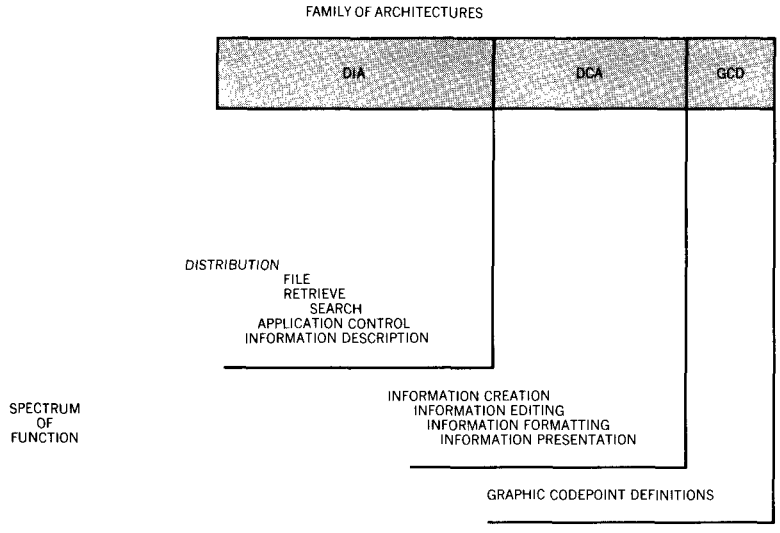
There is a spectrum of office functions that includes the creation, revision, distribution, and storage and retrieval of information. This spectrum is open-ended, thus permitting the introduction of new functions, particularly those that become feasible as a result of new technologies being developed. This apparently unstructured environment has been ordered into logically distinct functional sets.

Figure 1 indicates the office system functional spectrum within the application layer. One functional set deals with the manipulation of units of information as transparent entities. The envelope is a commonplace medium which is used to convey control requests to manipulate information and its contents as transparent entities. A different set of functions is concerned with the internal form of these units of information. Yet another set of functions has to do with the description of the units of information.

In structuring a family of architectures to support the office environment, we can relate each architecture to a specific, logically distinct, set of the functional spectrum. This relationship is important because each functional set should be able to be defined, developed, and enhanced with minimum regard to the changes and improvements of the other functional sets in the family. For example, several DCAs have been defined, including the DCA used by the IBM 5520, SCRIPT, and others. The DIA processes are unaware of the specifics of the structure of the information content. This permits both DIA and the DCAs to develop without undue interaction.

This structure is referred to as a *family* of architectures rather than a *set* of architectures because each one is designed to fit in with the

Figure 2 Family of architectures and spectrum of function



others, thus relating them to one another. The functions addressed by each architecture are bounded to minimize overlap and potentially conflicting constructs. Additionally, the structure of these architectures correlates with the structuring of an office system into its logical categories to simplify its overall conceptualization and, therefore, its implementation. The family of architectures is open-ended to permit the introduction of new architectures to support the automation of new sets of functions as they are developed or become technologically feasible.

Although these architectures are functionally bounded and logically distinct, they are defined to be functionally extendable within their respective logical bounds. This characteristic is essential for a technical spectrum that is dynamically growing. This architectural structuring provides for the orderly integration of any new technical solutions that are introduced. Figure 2 indicates the office system functional spectrum as it relates to the family of architectures.

summary of architectures

Each member of the family of architectures is now briefly summarized.

Document Interchange Architecture (DIA) is defined to provide the application-to-application protocols, semantics, and syntax. It deals with the manipulation of information without regard to the information content or the connectivity of the system.

DIA consists of several services. Document Distribution Services are concerned with the functions of distribution and delivery of information. Document Library Services are concerned with the functions of

filing, retrieving, and searching for information. Application Process Services have to do with the functions of an office system, such as formatting information, modifying information, initiating programs to be executed, and other general-purpose office system functions.

DIA also defines a document profile, a structured form to convey the characteristics of information or a document. The document profile conveys information such as the subject and date of a document, as well as the internal DCA level of the document. DIA is discussed extensively later in this paper.

The *Document Content Architectures* (DCA) are defined to provide consistent internal structuring of information or documents.[7] They deal with the functions required to be performed on the document contents such as pagination, highlighting, headings, footings, and centering. Different levels of DCA are defined, including Revisable Form, which defines the structure of an editable or formattable document, and Final Form, which defines the structure of a formatted document or a final form device-independent document.

Graphic Codepoint Definitions (GCD) address the problem of having many more graphics than can be defined by the 256 bit combinations, or codepoints, of a byte. Thousands of graphics have been defined, including those for the natural languages such as the Latin, Slavic, Oriental, and Mid-Eastern languages, and for various technical languages. Therefore, it has been necessary to identify various sets consisting of 256 graphics. Each set is called a codepoint page.

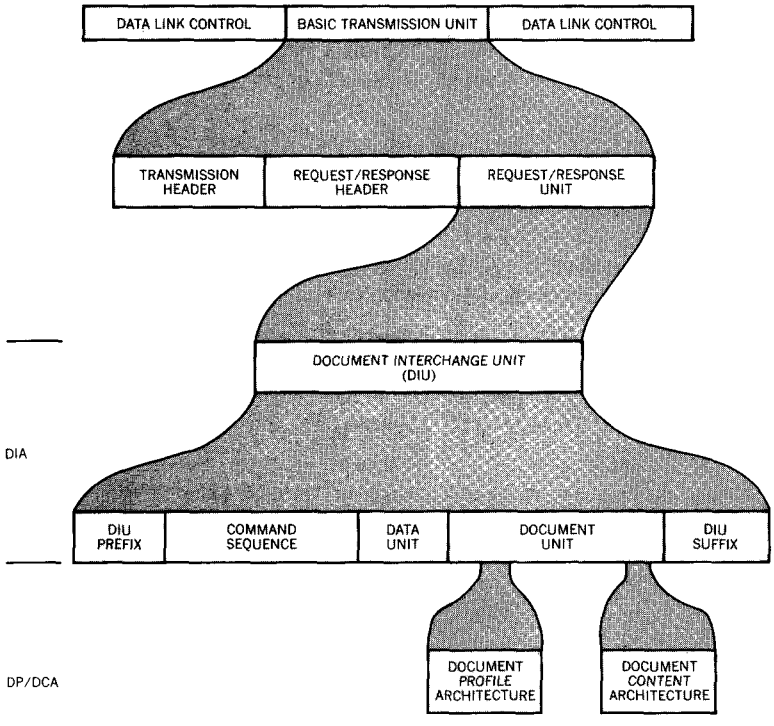
GCD defines a set of codepoint pages to ensure the consistent understanding and translation of the many graphics. Furthermore, GCD defines the means to move from one codepoint page definition to another in an orderly manner.

GCD deals with the problem that resulted from conflicting codepoint definitions having evolved for many graphics. For example, the codepoint assignment in the definitions used in the United States and England is identical for both the dollar sign and the pound sign. Therefore, it is possible to generate a document in the United States that states "I owe you $1000.00" and have it transmitted to England where it will be printed as £1000.00 with the pound sign graphic rather than the dollar sign graphic. Clearly this transmission does not maintain the integrity of the intent of the information exchange, whereas it does maintain the integrity of the information as represented by the bits of the information flow.

information flow overview

Figure 3 presents a broad overview of how each of these architectures relates to the flow of information within a system. Scanning the diagram, we see that a Document Interchange Unit is enveloped by lower-layer SNA headers and trailers. (See References 2 and 3 for appropriate details.)

Figure 3 Information flow



This view of the relationships between the units of data handled by the separate layers is highly simplified. In reality there is no one-to-one correspondence between the units of data. Architecturally, one layer's unit of data may contain fractions or multiples of another layer's unit. For instance, many Basic Transmission Units may be required to transmit one Document Interchange Unit (DIU). The DIU, which is transported transparently by the lower layers, is defined by DIA and is discussed later in this paper.

This paper concentrates on DIA and its relationship to the other members of the family of architectures.

Document interchange environment

All of the tasks that can be automated in the modern office cannot be performed locally within a stand-alone word processor. The information available in one office must be shared with people in other offices, either nearby or across the country. The Document Distribution Services of DIA provide the functions for utilizing data processing resources in the distribution of information among offices using different products.

Data processing resources are advantageous in office systems when large amounts of data are being handled. Individual word processors have limited quantities of data storage for retaining large quantities of information. Also, access to locally retained information is limited to relatively few people. The Document Library Services of DIA allow for the retention of information in pools, or libraries, on larger data processing systems whose storage capacities are apt to be extremely large and more economical than local storage. These systems support efficient searching and retrieval of information that is accessible to numerous individual work stations while at the same time offering protection against unauthorized access to the information. The document libraries can be centralized or dispersed at geographically separated data processing systems. Thus, DIA facilitates the integration of word processors with data processing systems to satisfy requirements of the automated office.

DIA specifies the semantics and structures with which to interchange both intentions and data between programmed processes in this environment. DIA does not define the user interfaces through which people use products. User interfaces are particular to individual products. DIA allows those individual products to interact with other products having the same or different user interface capabilities, to accomplish the various tasks that arise in the office environment.

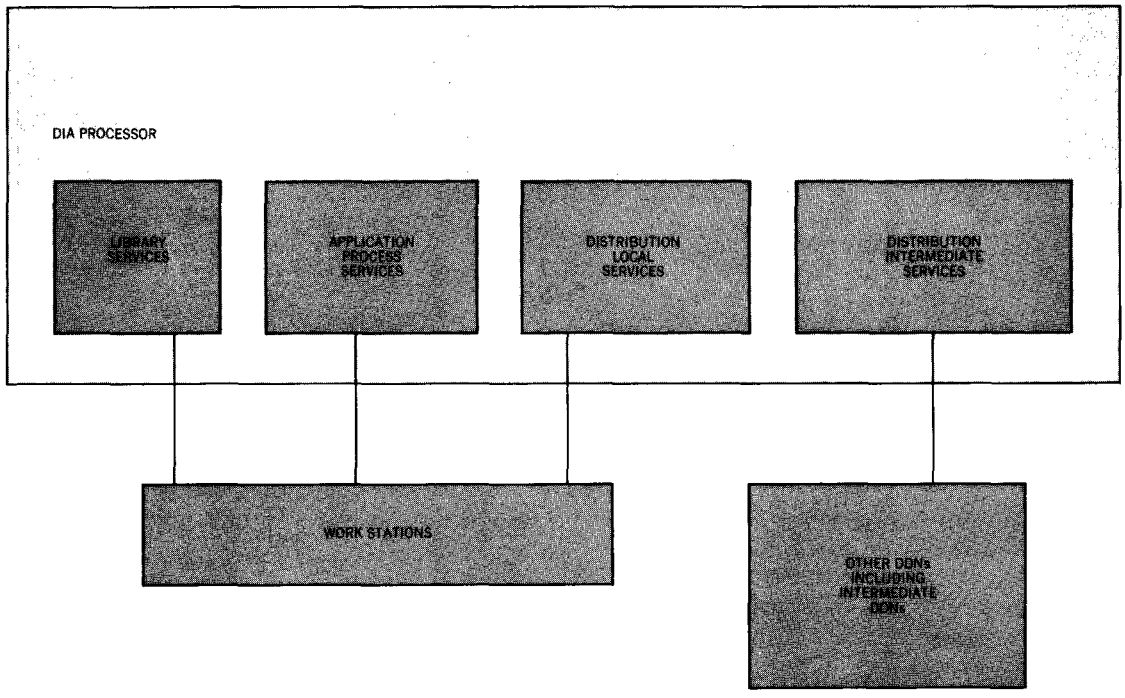
The physical system to which DIA applies is made up of work stations and Document Distribution Nodes (DDN) as shown in Figure 4. Work stations are processors with input and output facilities that serve application processes or people who are sources and recipients of documents. Work stations are Source Nodes and Recipient Nodes in the DIA system. DDNs are facilities with storage, processing, and communications capabilities to attach and support numerous work stations. DDNs communicate with each other to perform safe storage and forwarding of documents through the distribution system. DDNs may also provide the storage and processing to maintain document libraries. The work stations and DDNs are interconnected by communications lines and data transmission equipment.

We next describe the functional semantics and the structural syntax of DIA.

Definitional consistency across the DIA spectrum

DIA is defined for a broad spectrum of functional capability. The semantics, syntax, and protocols are defined with both conceptual integrity (consistent understanding of the functions to be performed), and consistency of the constructs (consistent formats and grammar of the language). This is essential to provide a design of simplicity and, therefore, comprehensibility.

Figure 4 DIA system structure overview



The semantics for each object, such as a command or an operand, are defined in the same way wherever that object appears across the entire architecture. The identification of users, documents, and system resources are always consistent. The semantics of all instances of operations, regardless of the context, are identically defined.

The syntax of architectural constructs appears in a consistent manner. Constructs are self-defining and variable in length. The advantages of this structure are related later in this paper.

The primary construct of the architecture, called a Document Interchange Unit (DIU), is the unit of interchange used across the spectrum of the architecture. All DIUs consist of up to five entities called Prefix, Command Sequence, Data Unit(s), Document Unit(s), and Suffix. Whenever these appear, they are syntactically and semantically consistent.

The Command Sequence consists of one or potentially multiple commands which may appear in any order and are executed in the order of their appearance. Existing implementations constrain the Command Sequence to a single command. Each command consists of any number of operands. Many of these operands may appear in different commands. They are defined in an identical manner across all commands in which they may appear.

Use of the command protocols of the architecture is consistent across the services of the architecture. Such consistency is important for simplicity of implementation as well as being able to predict the outcome of related events.

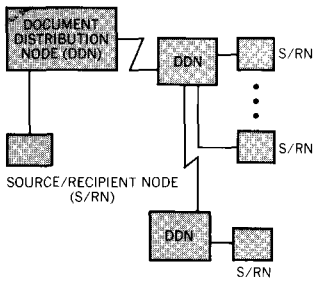
Definitions of profiles are also consistent when used across the several services of DIA. The Document Profile is used and the fields are defined identically regardless of whether they are used in conjunction with a Distribution Service or a Library Service.

DIA functional definition summary (semantics)

DIA services

DIA consists of several services that include Document Distribution Services, Document Library Services, and Application Processing Services. The architecture is defined such that services may be broken up into logical subservices. Furthermore, as the need arises, new services may be added to the architecture.

Figure 5 Document distribution system



Document Distribution Services (DDS) support the logically distinct functions of distribution requesting, storing and forwarding, and delivery in a document distribution system. DDS collectively provide for the asynchronous distribution of information. That is, they do not require that the originator and any of the recipients be in session with one another when the distribution of information is initiated. Figure 5 schematically defines the components of a document distribution system as supported by DIA.

DDS support falls into three categories: the exchange of documents directly between the source and the recipient nodes, their exchange of information to and from a DDN, and the exchange of information among DDNs.

Within the document distribution system, products implementing DIA provide functions such as confirmation of delivery, priority routing, delivery, and the mapping of logical names to communications routes. The distribution is optimized by carrying only one copy of the document on common paths in the system. In addition to the primary document, supplemental messages relating to the document are also distributed. Messages may also be distributed without a document. Information on the status of documents within the system is defined for return to the source of the distribution.

The Document Library Services (DLS) include those functions that support the storage and retrieval of information. The functions defined include the filing, searching, retrieving, and deleting of information.

The Application Processing Services (APS) include those functions that facilitate office system applications and support the other DIA

services. This set of DIA services supports functions such as formatting a document from revisable form to final form, modifying descriptive information, and permitting one application to request another to execute an identified program.

Interchange Document Profile

The Interchange Document Profile (IDP) provides a construct that characterizes its related document. The profile, a construct distinct from the related document, contains document descriptive information such as author, subject, data, addressee(s), and other information that describes a document from the user's point of view. The profile also contains document descriptive information relevant to the system, such as the DCA level of the document.

DIA-Session definition

At the application layer, the logical connection between two DIA processes is a DIA-Session. The protocols for the DIA-Session prescribe the DIA command requests and their replies to accomplish a given function.

A DIA-Session is initiated by a SIGN-ON exchange that identifies the participating processes, the role each is to perform, and the set of functions to be used. The DIA-Session normally is concluded by a SIGN-OFF command.

DIA command definition

The essence of DIA is the set of commands that define the functions to be performed when they are exchanged between DIA applications. The DIU could contain one or potentially multiple commands contained within a Command Sequence. Existing implementation versions constrain the Command Sequence to only one command.

The existing DIA commands as well as those introduced over time are intended to be relatively low-level functional primitives. Here the term functional primitive is defined so as to reflect the necessary trade-offs made when defining a command set. For example, the System/370 commands are relatively primitive. The function of a multiply command can be executed using a series of add commands. Although the command repertoire would be more primitive without the multiply command, the efficiency of a System/370 program containing a multiply function would be significantly reduced. The same is true for the command repertoire of DIA. Although DIA commands, such as FILE and SEARCH, are not primitive, they do represent an attempt to provide a set of relatively primitive commands to satisfy the requirements.

Defining these relatively primitive commands is basic to the realization of the potential of the architecture. Permitting these commands to appear in differing sequences, or programs, allows for extensibility. This open-ended functional capability is analogous to the huge functional capability provided by the limited number of relatively primitive commands defined for the various CPU architectures such as that of System/370.

DIA commands include several Library Service Commands such as FILE, which preserves identified documents in the library for an authorized document owner, and RETRIEVE, which returns a library copy of the identified document to an authorized document requestor. A brief description of the semantics of other DIA commands is related in the Appendix.

New functions may be added by defining and incorporating new DIA services, new commands, and new operands, as required. These services, commands, and operands must be defined within the framework of the architecture.

classes of commands

There are two classes of commands by which reply protocols are defined for DIA:

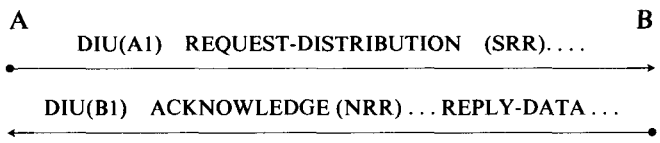
1. No-Reply-Required (NRR) Command Class—This command class is defined to let one application convey information to another application without that application necessarily replying to the command.
2. Synchronous-Reply-Required (SRR) Command Class—This command class is defined to let one application ask another to perform a function with that application having to reply to the request before any other interchange of requests is permitted.

The NRR Command Class is used for any command that does not require a command in reply from the receiver. If the command sent in the NRR class is a request for an operation that normally requires return of results, the results will be returned with an appropriate command in the NRR class. No explicit synchronization nor correlation is done for this type of command exchange. Only an ACKNOWLEDGE command with an exception condition code is allowed to reply to and reference an NRR command. The sender of the NRR command is not required to correlate the exception reply to the original command. The exception information is for statistics logging or whatever use a sender chooses to make of it. For example:

Here process A sends a STATUS-LIST command for information with no reply needed.

The SRR Command Class is used for any command that requires a command in reply as the next command sent by the receiver. The SRR command sender may not send any other commands until the replying command has been received. The replying command may be an ACKNOWLEDGE command or any other appropriate command that has a CORRELATION operand correlating it with the SRR command. Should the DIA-Session be terminated while a reply to an

SRR command is still outstanding, that reply is lost and neither DIA-Session partner is required to perform any further action on behalf of that command. The command must be issued again on another DIA-Session. For example:



In this case, Process A submits a document for distribution. Process B executes the request, acknowledges the request, and returns the identification of the distribution in the reply data.

A command reply is a command that contains the CORRELATION operand. This operand both indicates that the command is a reply to a previous SRR command and provides the data necessary to correlate that reply to its corresponding request.

function sets

Commands fall into groupings according to the services that they normally support. Some commands support more than one set of services although they are defined within one group. Other commands support all sets of services and are grouped together as being applicable across services. The commands can be further categorized by their use between different types of products in the interchange environment. This latter categorization defines the term *function set.*

Because no single product requires all of the function available in DIA, the total DIA facilities have been divided into function sets. The function sets have been defined such that a single one contains all of the DIA facilities required to support a given level of interchange for a given set of services. The function sets may be used in combinations. The specific function sets to be used during a DIA-Session are specified with the SIGN-ON command.

The definition of function sets allows products to implement only those functions applicable to their purposes. Thus, each product may tailor its implementation to its environment.

For Document Distribution Services, function sets of commands are defined to support the input of documents from work stations to DDNs, the exchange of documents among DDNs, the output of documents from DDNs to work stations, and the exchange of documents directly between work stations.

For Library services and Application Processing Services the function sets coincide with the commands within each of the services.

An example of a document distribution function set is the one for DDN to Recipient Node. The following commands are supported in this set. They support the delivery of documents and status to the work station.

OBTAIN—Requests delivery of documents.
DELIVER—Delivers documents.
LIST—Requests a list of available documents and status.
STATUS-LIST—Notifies the work station that documents or status are available for delivery.
CANCEL-DISTRIBUTION—Requests the cancellation of documents.
ACKNOWLEDGE—Acknowledges the receipt and conclusion of a request.
SIGN-ON—Initiates the DIA-Session, identifies user, function set(s), etc.
SIGN-OFF—Concludes the DIA-Session.

exception handling and recovery

An objective of DIA is to provide a means by which two application processes exchange information in a reliable manner. The processes must be capable of recognizing when a Document Interchange Unit (DIU) is received with errors that are detected at a lower layer. DIA is dependent on the fact that the layers of support below the application level will notify the application process when a permanent error exists. DIA assumes that all error procedures have been attempted by the component layers below the DIA process and that unrecoverable errors are presented to the process for final disposition.

The damage assessment process for a DIU with a permanent error is to interpret and evaluate the DIU entities to determine if the request can be successfully executed. If exceptions are detected and all of the recovery techniques have been unsuccessful, the DIU cannot be reliably processed, and the receiving process must terminate DIU processing.

The DIU process is composed of algorithms that evaluate the DIU syntax to determine which functions of the DIA repertoire are to be performed. This evaluation may discover syntax and semantic exceptions. The syntax exceptions are deviations from the structure formats of the architecture, missing parameters, parameter values outside permitted ranges, and DIU entities incorrectly encoded. Semantic exceptions are usually detected when a requested operation is activated and some violation prevents successful completion.

There is a distinction between errors and exceptions. Errors are those failures that occur at the lower layers. Exceptions are DIA architectural violations that are detected in a DIU without, necessarily, an indication of a failure by a system or application component.

The syntax and semantic exceptions that prevent a requested DIA application process from being normally completed must be reported

to the sender of the DIU. The ACKNOWLEDGE NRR command with an EXCEPTION CODE operand is used to report exception conditions detected by the receiver of a DIU. If there is an exception condition detected in the DIU containing the acknowledgment of the sender's request, the DIU processing is terminated because the sending of valid ACKNOWLEDGE commands is fundamental to a DIA process. Exceptions that are detected by a sender that prevent successful transportation of a complete DIU are specified in a form of the DIU suffix.

The EXCEPTION CODE operand is structured to contain the Exception Class and the Condition Code and to identify the DIU Exception Object in which the incident is detected. The Exception Class and the Condition Code describe the exception condition that has been detected. Since exceptions can occur in any of the DIU objects, the Exception Object code identifies the element in which the exception is detected. Because the nature of exception conditions can sometimes be obscure, the EXCEPTION CODE operand includes a field for the actual object containing the detected exception.

The receiver of a DIU that contains an exception may recommend a recovery action to the sender of that DIU. This recommendation is carried by the ACKNOWLEDGE command in the RECOVERY-ACTION operand. The receiver of the RECOVERY-ACTION operand is not bound to the recommended action. If the recommended exception action is not done, the subsequent recovery may be unacceptable to the sender of the recommendation and therefore cause the session to be terminated. The following recovery actions may be recommended.

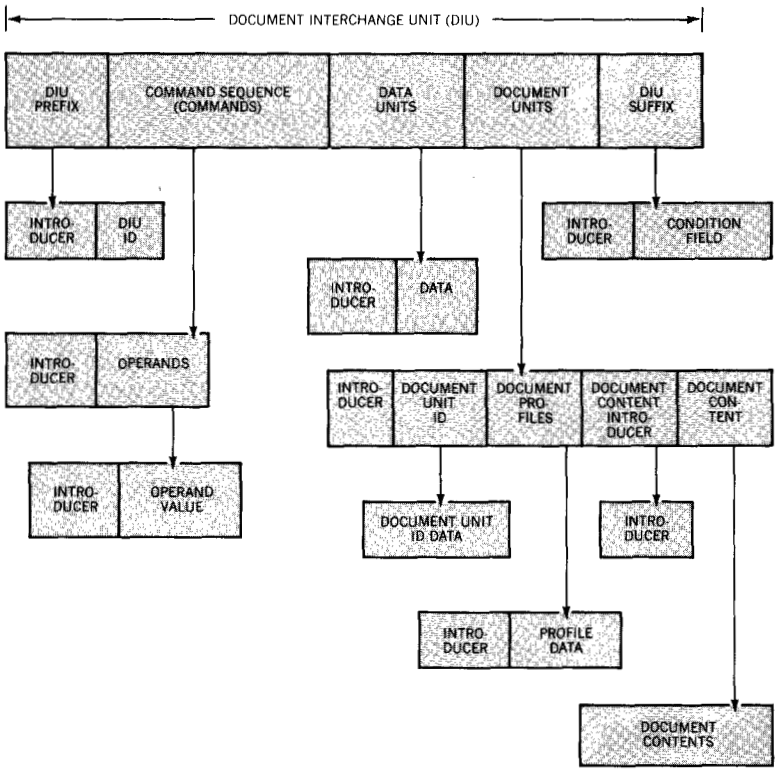
- None—recovery action is to be determined by the sender of the offending DIU.
- Resend—the sender of the offending DIU should resend that DIU as the next DIU sent after receiving the ACKNOWLEDGE command containing this value.
- Skip and resend—the sender should resend the offending DIU after all other DIUs scheduled for this DIA-Session have been sent.
- Postpone—the offending DIU should be sent on a subsequent DIA-Session.
- Cancel—the offending DIU should never be resent.

If the recovery action does not appear on an ACKNOWLEDGE command, it is the same as if it were specified as none. Recovery action values are ignored on a normal ACKNOWLEDGE command.

DIA structural definition (syntax)

Constructs of the services of DIA are self-defining and variable in length. Generally, these characteristics permit fields to appear in any order and fields of no interest to a particular product to be omitted.

Figure 6 Document Interchange Unit structure overview



Functions, defined by commands, may be supported or not, according to the functional objectives of the product. A parameter on a particular command may be supported according to whether or not it is optional and, if it is optional, whether or not its use is defined by the product.

Furthermore, the fields of the architecture are defined to be variable in length. Variableness is provided by using a length field as part of each element's introducer and by using the optional segmentation capability which permits elements to consist of an unlimited number of segments, each of which may be as large as 32 767 bytes in length.

The SNA transport network deals with one type of segmentation which is required to optimize the size of the units of information moved in the network. It is distinct from the application process segmentation. The fields in a DIU known only to the application process are constrained to 32K bytes. The DIA segmentation permits these application-level fields to have extensions, or multiple segments. Figure 6 presents an overall perspective of the architectural structure.

The above-mentioned syntactic definitions are important because they help permit individual products in an office to interact with one another using the same language. A functionally rich product may be able to utilize the full potential of the architecture, whereas a product designed to provide less capability may constrain its use of the architecture. DIA permits each of the product types to be optimized to its respective unique requirements while it also permits the full range of product types available for the office to interact within the same system.

Document Interchange Unit

DIA defines a DIU as the interchange structure between DIA processes. It consists of the following logical entities: a DIU prefix, one or more commands comprising a Command Sequence, Data Units and Document Units as options, and a DIU suffix, as shown in the top of Figure 6.

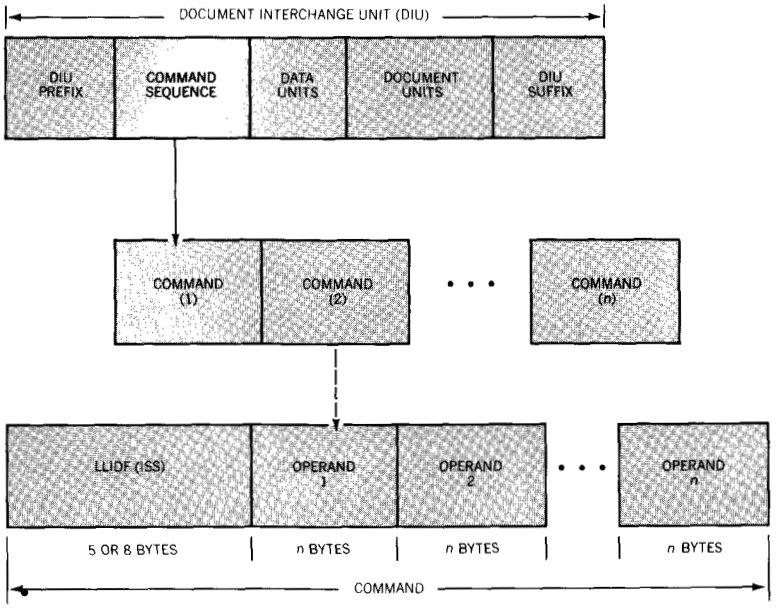
Structures within these entities are referred to as components. Examples of components are command operands and document profiles. Entities and components are self-defining and consist of several fields.

All DIU entity and component constructs contain a mandatory introducer consisting of a two-byte length field (LL) and a three-byte identifier and format (IDF) field that are followed in some cases by an optional extension (ISS) consisting of a one-byte indicator (I) and two-byte sequence number (SS), where

- The length field consists of a two-byte unsigned binary value from 5 to 32K (32 767) indicating the number of bytes in the construct.
- The IDF field consists of three bytes. The first two bytes (ID) identify the entity or component and its semantic definition; the third byte, F, identifies the structural and syntactic representation of the construct. For example, the ID indicates that this entity is a FILE command and F represents the specific operand structure of the command.
- The ISS extension field consists of three bytes. The first, referred to as I, is a byte of indicators used to specify options such as segmentation; the second and third bytes, referred to as SS, are a two-byte sequence number. The ISS field is an optional extension to the mandatory LLIDF introducer. Its presence as part of the introducer is indicated by the high-order bit of the F byte.

DIU commands, Data Units, and Document Units may be segmented to accommodate processes with limited buffer sizes, data lengths greater than 32K bytes, and situations where the total length is not known until the end of the data has been transmitted. Segmentation is controlled through the bit settings within the I byte.

Figure 7 DIA command



DIU prefix

The DIU must be introduced by a prefix which begins and identifies the DIU. The DIU prefix consists of an introducer and a DIU ID field. The DIU ID field permits a DIU sent in reply to identify the DIU to which the reply correlates.

Command Sequence

In existing implementations, the Command Sequence contains one command that prescribes operations to be performed. Syntactically, the Command Sequence could contain up to 255 commands. In a DIU there is only one Command Sequence, but it may be extensive according to the complexity of the request. Each command directs the process to which the DIU is sent to perform some function or act on data in a specific manner. Execution of commands within a Command Sequence is the responsibility of the receiver of the DIU. The operations requested by the commands must be performed in the order in which the commands appear. Commands appear within the Command Sequence of the DIU as shown in Figure 7.

The command consists of an operator, or introducer, and zero or more operands. The first two bytes, referred to as its LL bytes, indicate the length of the entire command including all of its operands. The next two bytes, referred to as its ID bytes, define the semantics and reply protocol for the command. The fifth byte, referred to as the F byte, indicates the operands that are defined for the command, with respect to their mandatory or optional appearance, and allowed types. DIU commands may be segmented.

operands

A DIA operand either contains or references data that are used in the execution of the command in which the operand appears. The syntax and semantics of each operand are consistently defined for all commands in which the operand appears.

Each DIA operand consists of an introducer and an operand value. The ID bytes indicate that this element is an operand, define the semantics of the operand, and indicate that the data reside in a Data Unit or a Document Unit, or are contained as immediate data in the operand value field.

When the ID bytes indicate that the operand contains immediate data, the operand value field contains the data to be used in performing the operation of the command. Examples of operands are Password, Message, etc. When the ID bytes indicate that the data semantically defined by this operand reside in a Data Unit or Document Unit, the operand value field contains a binary position number with a value greater than or equal to one. This position number is used to identify the specific Data Unit or Document Unit being referenced by its relationship with other Data or Document Units within the current DIU. Currently, the implementations of DIA provide for only one Document Unit within a DIU. When the operand data resides in a Data Unit, the IDF bytes of the Data Unit indicate the semantics and structure of the operand data that it contains.

Operands are defined to contain immediate data values or to reference a value contained within a data unit. The architecture is defined with this flexibility recognizing that in-line information is processed with some efficiency, yet if the operand value is lengthy, or used by more than one command in a DIU, then efficiencies are accrued when the operand value is indirectly referenced by a command, i.e., the data are not in line.

Operands are defined distinct from commands so that they may be used identically in more than one command. For example, the Recipient-Address operand is used in the REQUEST-DISTRIBUTION, DISTRIBUTE, and DELIVER commands as well as in several others.

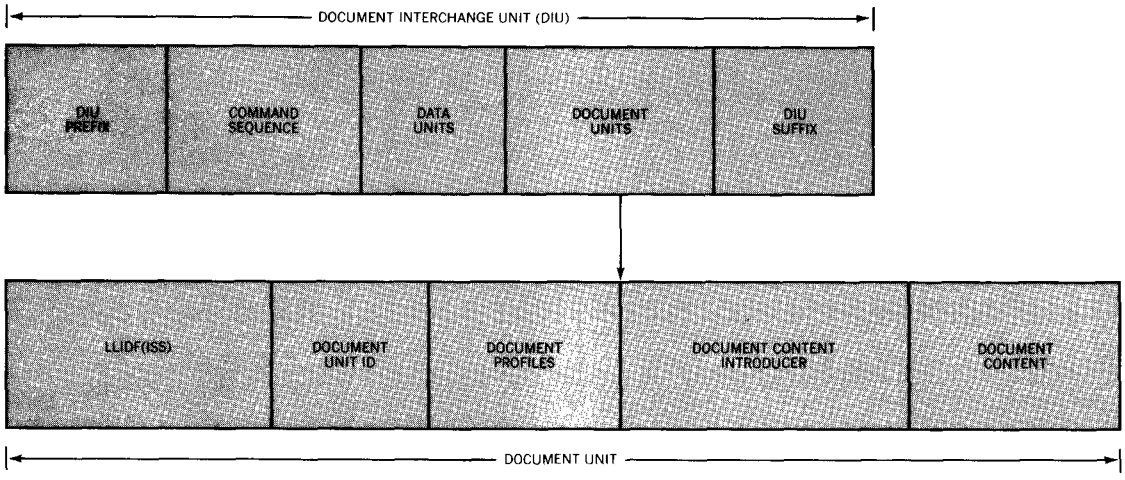
Data Unit

A Data Unit contains information, such as a distribution list, that is referenced by operands in one or more commands in the Command Sequence of the DIU containing the Data Unit. Information common to multiple operands, in the same or multiple commands, may be contained in a Data Unit and be referenced by those operands. Information which is unwieldy as immediate data or which is not available at command generation time may be placed in a Data Unit and referenced by an operand of a command.

Document Unit

The Document Unit is an entity consisting of an introducer, a Document Unit ID field, a Document Content Introducer, and, as options, Document Profile(s) and the contents of the document itself as shown in Figure 8.

Figure 8 DIA Document Unit



The Document Unit introducer's LL bytes indicate the length of the entire Document Unit. If the length is greater than 32 767 or not known at the time the length field is constructed, the Document Unit may be segmented. The ID bytes indicate that this is a Document Unit and denote its type. The F byte is encoded to indicate the presence or absence of the optional ISS introducer extension.

The specific content of a Document Unit is dependent on the requirements of each of the interchange document types. Therefore, the DCAs specify the required Document Unit composition. The Document Unit ID identifies the architecture of the document contents, DCA, contained within this document unit.

A document profile contains information relating to or describing a document. All information in a document profile applies to the entire document. The Interchange Document Profile (IDP) is defined for general interchange between processes that use DIA. The document profile is defined because of the need to deal with a document without necessarily examining the contents of the document, which may consist of noncoded information or which may be defined according to an unfamiliar internal structure.

The form of the document content is, in most cases, defined externally from DIA. DIA defines a structure for interchanging the document contents and is not concerned with the controls and format of the content itself. DIA conveys the type of document in the Document Unit ID. The Document Content Architectures are intentionally distinct from DIA so that the DIA process need not be aware of the different DCAs.

There are some instances where DIA is the defining architecture for the contents of a Document Unit. In these cases the document content is used specifically to support a DIA function. Examples of instances of the definition of document content by DIA are for transporting document distribution status data or the Library Services Document Descriptor.

suffix

The end of a DIU is explicitly defined by the required occurrence of a DIU suffix. A DIU suffix indicates the following circumstances.

- Normal termination of a DIU, Type 1. The sending of the DIU proceeded and completed normally.
- Abnormal termination of a DIU, Type 2. The sending application encountered an unrecoverable DIA exception situation during the sending of the DIU. This can occur in the sending of large DIUs for which transmission is begun before the entire DIU is constructed.

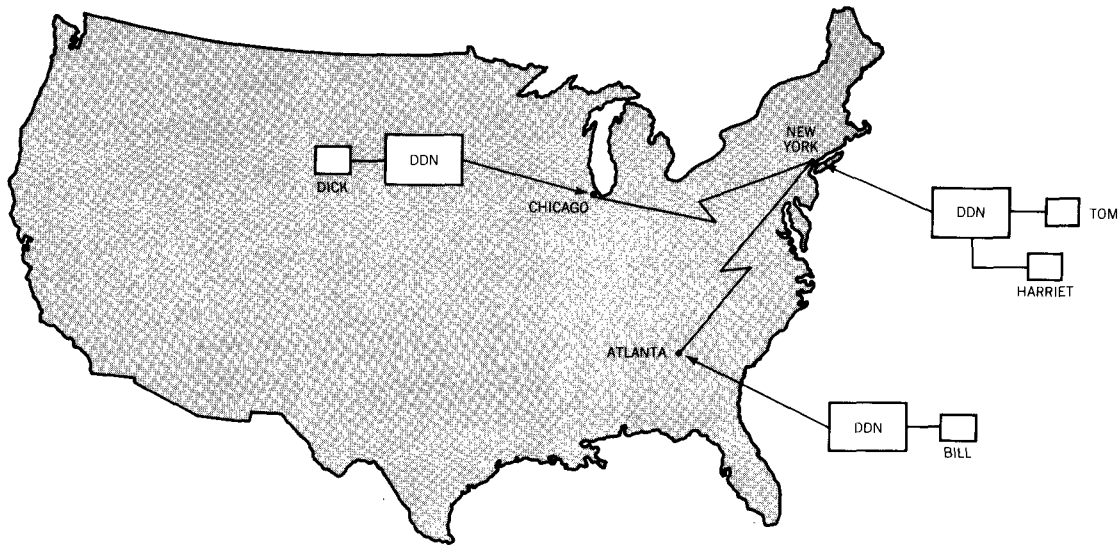
The suffix ID bytes indicate that this entity is a DIU suffix of a specific type. The specific types denote the circumstance for the use of the DIU suffix.

Use of the family of architectures for interchange

The following scenario demonstrates the application of the family of architectures described in this paper. For this example, assume that an author, Bill, in Atlanta is utilizing a work station that attaches to a document distribution system. He has created and edited a document in Revisable Form DCA. The FORMAT command has been used to put the document into Final Form DCA. The Interchange Document Profile associated with the document lists BILL as the author along with other pertinent information about the document. He is ready to distribute the document to three recipients, two of whom, TOM and HARRIET, are at an office in New York City and the other, DICK, is at an office in Chicago. The company's document distribution system is configured as shown in Figure 9.

Through the user interface provided by the work station, Bill indicates the desire to distribute the specific document to Tom and Harriet in New York and to Dick in Chicago. The work station program initiates an SNA communications session with the company's Document Distribution Node (DDN) serving the Atlanta office. When the communications session is established, the work station program sends a DIU containing a SIGN-ON command to open a DIA-Session with the DDN. The work station sends a DIU containing both a REQUEST-DISTRIBUTION command indicating the recipients, New York (Tom, Harriet) and Chicago (Dick), and the document. The DDN replies with a DIU containing an ACKNOWLEDGE command that means it understands the request, has the document safely stored, and will proceed to initiate the distribution. Since this work

Figure 9 Sample document distribution system



station has no more outstanding requests to send at this time, the work station sends a SIGN-OFF command terminating the DIA-Session and deactivates the SNA session.

The DDN, in turn, establishes an SNA session with the company's DDN in New York, and after establishing a DIA-Session sends a DIU containing both a DISTRIBUTE command indicating the recipients, New York (Tom, Harriet) and Chicago (Dick), and the document. The DDN replies with an ACKNOWLEDGE command indicating that it has safely stored the requesting DIU and will proceed with the request. The New York DDN has a similar dialogue with the Chicago DDN to forward the document to Chicago (Dick.)

The New York DDN also recognizes that it supports Tom and Harriet. It invokes the program at the New York DDN to service local users and enqueues the document for delivery when Tom and Harriet initiate an SNA session and a DIA-Session to receive mail. Tom, known to the distribution system as TOM, indicates through his work station and its user interface that he wants to know what "mail" is queued for him. The work station establishes an SNA session and a DIA-Session with the New York DDN and sends a DIU containing a LIST command. Upon receipt of the LIST command, the local distribution program in the DDN gathers together a list of information about documents that have arrived for TOM.

The information is delivered to the work station in a DIU containing a DELIVER command and a Document Unit containing the data about

the documents. The work station sends an ACKNOWLEDGE command for the DELIVER command to the DDN and presents the information to Tom, who can then select the document for delivery. When Tom has indicated which document he wants delivered, the work station sends to the DDN a DIU containing an OBTAIN command identifying the document to be delivered. The DDN replies with a DIU containing a DELIVER command and the requested document.

The work station acknowledges the DELIVER command and, depending on the capability of the work station, allows Tom to view the document or print it as hard copy. If Tom views the document and decides that he wants to retain it for future reference, he indicates to the work station that he wants this document filed under a name he chooses. The work station sends a DIU containing a FILE command and the user-assigned document name to the DDN. The library service program of the DDN files the document, assigns it a unique name, and acknowledges the FILE command. Depending on the implementation of the DDN, the FILE command may have been done automatically prior to the DELIVER command. In this case, Tom would not have to explicitly file the document to access it for reference. He would however, be expected to delete the document when it is no longer of use to him.

At some later date, Tom decides to reference the document. Using the user interface of the work station, he indicates that he wants to retrieve a document. He has forgotten the name of the document but recalls that the author was Bill. Using the user interface, he causes the work station to establish an SNA session and a DIA-Session with the DDN. The work station sends a DIU containing a SEARCH command indicating that the user wants information on all documents to which he has access and for which Bill is indicated as the author in the document profile.

The library services program searches the profiles of Tom's documents and delivers a Document Descriptor Document containing profile information about all documents authored by Bill. The work station presents that information to Tom, who can then pick the document that he wants based on date or other indicative data returned. The work station then sends a DIU containing a RETRIEVE command specifying the document desired. The DDN replies with a DELIVER command and the document. The work station then makes the document available for viewing or printing.

In the meantime, Dick and Harriet are meeting their individual needs by using their work stations to have similar DIA-Sessions with their DDNs.

This example has shown how individual architectures in the family of architectures are used to satisfy tasks encountered in the office environment. Each individual user has a different exchange of

requests and information to accomplish his or her required tasks. This is one of the reasons that the architectures defining the interchange between products support a wide variation of control sequences and data types.

Summary

DIA, although oriented to an office systems application, can be a general-purpose language used to interchange information and control requests among applications in a distributed processing environment. DIA is defined to be both semantically and syntactically extendable in order to have a long life.

DIA is one of several architectures that comprise a family of architectures oriented to the requirements of the automated office. The family of architectures provides structure to the office functional spectrum. This functional spectrum is deliberately defined to be open-ended, anticipating the introduction of new technologies and new solutions to the system needs of the office. The architectures in this family are structured among one another to minimize functional overlap and to permit them to be developed independently.

It is this family of architectures that collectively provides for the interchange of both information and control requests among the many word processors, printers, and various other machines in an office. This family of architectures permits the many distinct items of equipment used in an office to comprise an office system.

ACKNOWLEDGMENTS

The architectures that comprise the family referred to in this paper were conceived and developed by many contributors who represented several organizations in the IBM laboratories in Hursley, England; Sindelfingen, Germany; and Fujisawa, Japan; and in those located in the United States in Austin, Texas; Boulder, Colorado; Kingston, New York; Poughkeepsie, New York; Raleigh, North Carolina; Rochester, Minnesota; and Santa Teresa, California. The dedicated work of these people is reflected in this paper.

Appendix

This appendix contains a brief description of each of the DIA commands.

DIA distribution services commands

DIA Document Distribution commands support asynchronous document distribution and direct source-to-recipient document exchange processes. They provide the functions necessary to perform distribution from initiation of the distribution to delivery of the document and the return of status regarding the distribution.

- The REQUEST-DISTRIBUTION command transports a document from a source node to a Document Distribution Node (DDN) for distribution to specified recipients. The document may be contained in the DIU Document Unit or in the command recipient's Document Library.
- The DISTRIBUTE command transports a document in the store and forward system from one DDN to another DDN.
- The DISTRIBUTION-STATUS command returns, through the store and forward system, status information on the progress of a previously distributed document.
- The STATUS-LIST command notifies the recipient that one or more documents is available from the Distribution System or that information about the progress of distributed documents is available.
- The LIST command requests the receiver of the DIU to return to the requestor a list of documents available from the Distribution System for this recipient or a list of the status of documents previously distributed.
- The OBTAIN command requests the receiver of the DIU to return one or more documents scheduled for delivery to the requestor.
- The DELIVER command transports a document from a DDN to a Source Node or a Recipient Node in response to the LIST, OBTAIN, or RETRIEVE commands. The DELIVER command is also used to transport a document directly from the Source Node to a Recipient Node without utilizing the store and forward system.
- The CANCEL-DISTRIBUTION command cancels system retention of document status or cancels the distribution or delivery of documents.

DIA library services commands

DIA Document Library Services commands provide the functions for maintaining user documents in a document library on a data processing system.

- The FILE command preserves the identified document in the library for an authorized document owner.
- The RETRIEVE command returns a library copy of the identified document to an authorized document requestor.
- The SEARCH command locates the documents in the library that have the characteristics specified by the search-data operand. It creates and preserves a named list of references or pointers to the search-selected documents.
- The DELETE command permanently removes access to the identified document for the delete requestor. A document that has two or more owners will be removed from library storage when all of the owners have requested that it be deleted.

DIA application processing services commands

Application Processing Services define commands for requesting the execution of tasks by another process.

- The EXECUTE command requests the destination to invoke the named process for execution.
- The FORMAT command requests the destination to execute the named formatting process using the identified document for the format input object.
- The MODIFY command is used to revise document control information fields. A field must be uniquely identified as a specific occurrence of a DIA parameter.

DIA commands applicable across services

These commands, also known as DIA-Session control commands, are used to establish, maintain, and terminate dialogues between processes using DIA.

- SIGN-ON establishes a DIA-Session, determines the functions of the interchange architecture to be used in that session, and gives two processes the ability to validate each other's authority to exchange information.
- SIGN-OFF concludes a DIA-Session. The conclusion can be unequivocal or negotiated.
- The SET-CONTROL-VALUE command provides the ability for one DIA process to establish, change, or delete the value associated with a control variable that is stored at another DIA process.
- The ACKNOWLEDGE command notifies the DIU sender that the requested DIU command operation is concluded with either a normal or an exception response.

CITED REFERENCES

1. A. L. Scherr, "Distributed data processing," *IBM Systems Journal* **17,** No. 4, 324–343 (1978).
2. *Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic,* SC30–3112, IBM Corporation; available through IBM branch offices.
3. *Systems Network Architecture: Concepts and Products,* GC30–3072, IBM Corporation; available through IBM branch offices.
4. *IBM Distributed Office Support Facility: Document Transmission Function Guide,* SC27-0548, IBM Corporation; available through IBM branch offices.
5. *IBM 5520 Administrative System: System/370 Host Attach Programmer's Guide,* SC23-0710, IBM Corporation; available through IBM branch offices.
6. *IBM Distributed Office Support System General Information,* GH12-5124, IBM Corporation; available through IBM branch offices.
7. M. R. DeSousa, "Electronic information interchange in an office environment," *IBM Systems Journal* **20,** No. 1, 4–22 (1981).

T. Schick is located at the IBM Communication Products Division laboratory, P.O. Box 12275, Research Triangle Park, NC 27709; R. F. Brockish is located at the IBM Information Products Division laboratory, P.O. Box 1900, Boulder, CO 80302.

Reprint Order No. G321-5168.