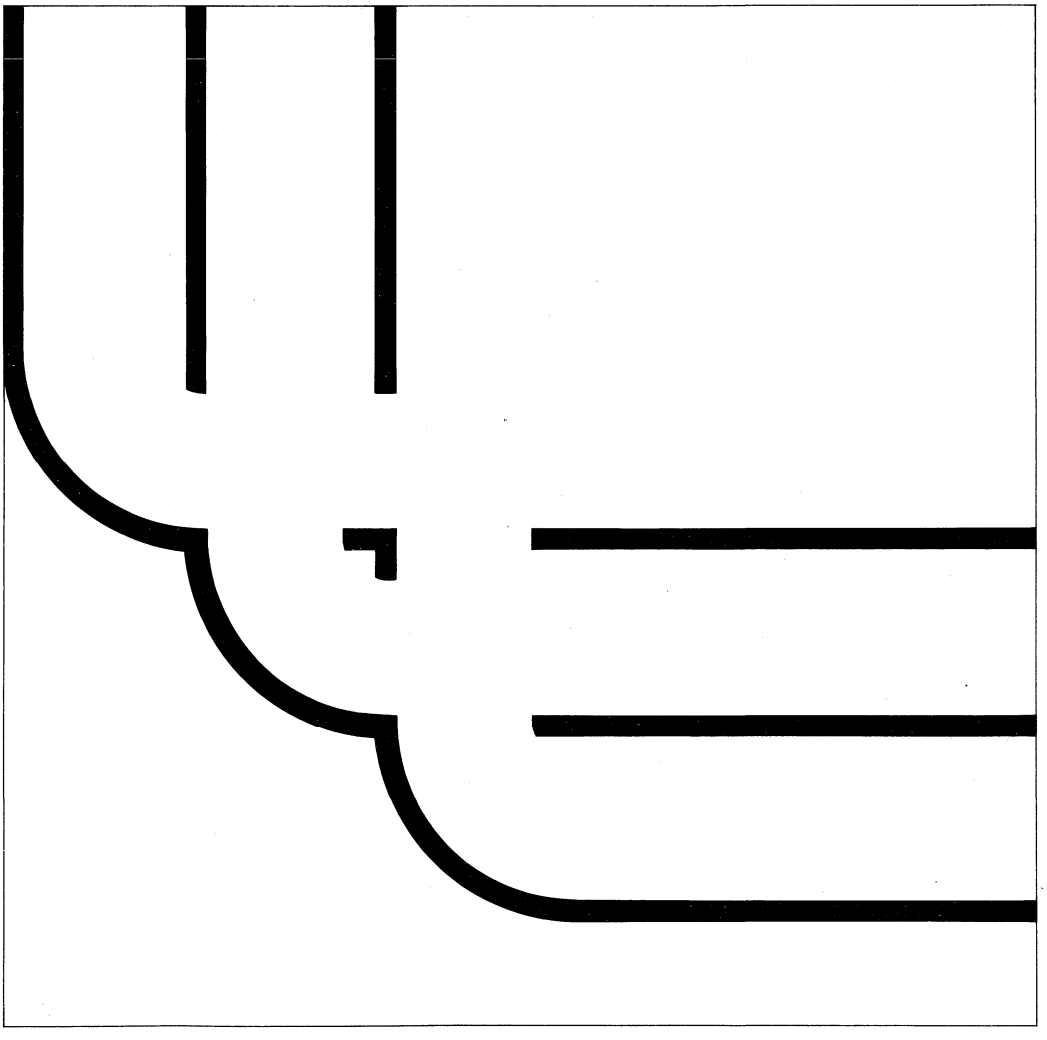
Application System/400

GC41-9802-00

System Concepts

Version 2



General

Take Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page vii.

First Edition (April 1991)

This edition applies to the licensed program IBM Operating System/400, (Program 5738-SS1), Version 2 Release 1 Modification 0, and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the proper edition for the level of the product.

Order publications through your IBM representative or the IBM branch serving your locality. Publications are not stocked at the address given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, you may address your comments to:

Attn Department 245
IBM Corporation
3605 Highway 52 N
Rochester, MN 55901-7899

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you or restricting your use of it.

Contents

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

AD/Cycle	Application System/400
AS/400	C/400
COBOL/400	DATABASE 2
DB2	Distributed Relational Database Architecture
DRDA	FORTRAN/2
FORTRAN/400	IBM
MVS/SP	Netview
OfficeVision/400	Operating System/2
Operating System/400	OS/2
OS/400	Personal System/2
Presentation Manager	PS/2
RPG/400	SAA
Series/1	SQL/400
SystemView	System/370
System/390	Systems Application Architecture
VTAM	400

The following term denoted by a double asterisk (**) in this publication, is a trademark of another company as follows:

RM/COBOL-85	Ryan McFarland Corporation

This publication could contain technical inaccuracies or typographical errors.

This manual may refer to products that are announced but are not yet available.

Information that has changed since Version 1 Release 3 Modification 0 is indicated by a vertical bar (|) to the left of the change.

This publication is for planning purposes only. The information herein is subject to change before the products described become available.

This publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Programming Interface

The *System Concepts* manual is intended to give the customer a basic knowledge of the AS/400 system features as they relate to general data processing tasks. It primarily contains overview information about topics such as: object and data management, communications concepts, integrated database functions, programming compatibility, application development, and system management and recovery. It contains no programming interfaces for customers.

About This Manual

The information in this manual is intended for programmers who wish to have a better understanding of the basic concepts of the IBM Operating System/400 (OS/400), users who wish to maintain application programs on the AS/400 system, and users responsible for operating and maintaining the AS/400 system. This manual discusses object management, data management, work management, integrated database, communications, programming environment, security, and other general programming concepts as they relate to the AS/400 system functions and utilities.

Use this manual to get a general understanding of how the AS/400 system can be used. Topics are discussed at a very conceptual level, and strategies for using the functions are provided. References are made in each chapter that direct you to information to help you complete specific tasks related to the concepts discussed in the chapter.

You may need to refer to other IBM manuals for more specific information about a particular topic. The *Publications Guide*, GC41-9678, provides information on all the manuals in the AS/400 library.

Chapter 1. Introducing the AS/400 System

The AS/400* system is a family of midrange computers based on a single software architecture. It provides many integrated features that form the foundation of the computer system. The Operating System/400* (OS/400*) licensed program provides a comprehensive, fully integrated set of batch and interactive work management functions that make processing application programs efficient and productive. Built-in data management functions provide a full range of data description capabilities and a consistent interface for access to data. All data can reside in a single, integrated relational database, with query functions that make information readily available. These functions, combined with a wide range of high-level language compilers and utilities, provide users with highly productive application development tools. System utilities and system management functions, such as message handling, spooling, diagnostic support, and electronic customer support, make operating the system convenient to use and easy to understand. Business operations are complemented by the integrated office enablers and sophisticated communications components of the system.

This chapter briefly discusses the topics listed in the following table. If you would like more information on the topic, and there is more information available in an IBM manual, the manual is listed in the right column is a good first reference.

Topics	First Reference
Operating system Licensed programs Hardware	*System Introduction*, GC41-9766
System architecture	No additional manuals

Operating System

The OS/400 licensed program supports the IBM* AS/400 system. It controls the operation of programs and provides services such as controlling resources, scheduling jobs, controlling input and output, and managing data. The OS/400 program is designed to complement and extend the advanced capabilities of the AS/400 system to provide fully integrated support for interactive applications. To supplement the full range of the interactive environment, the AS/400 system also processes multiple batch applications at the same time.

The AS/400 system supports three operating environments:

- The OS/400 program supports the functions of the AS/400 system as well as both the System/36 and System/38 environments
- The System/36 environment supports System/36 application programs and procedures
- The System/38 environment supports System/38 application programs

Programs can include data, procedures, or programs from any operating environment. For example, an OS/400 control language program could use a System/36 procedure in the System/36 environment or run a System/38 program from the System/38 environment.

Many of the functions of the OS/400 program are directly applicable to interactive data processing. Among these functions are:

- Database support to make up-to-date business data available for rapid retrieval from any work station
- Work management support to schedule the processing of requests from all work station users
- Application development support that allows online development and testing of new application programs to run at the same time as normal production activities
- System operation support that allows the user responsible for system operations to perform work from the display station using a single control language, complete with prompting and help for all commands
- Help and index search support that allows users to request online information on a wide variety of topics
- Message handling support that allows communication between the system, the user responsible for systems operations, work station users, and programs running in the system
- Security support to protect data and other system resources from unauthorized access
- Service support that allows service personnel to diagnose problems and install new functions with minimal affect on the normal flow of work

The system can be set up and installed using system defaults for basic functions. As the needs of the business grow, the use of controls and functions can be increased without disrupting applications that are already installed on the system.

Figure 1-1 on page 1-3 shows the relationship of operating system functions within the AS/400 system structure. For example, most database management is spread throughout different layers of the system, but the control language command analysis and interpretation are completely provided by the operating system. Notice how the layers interface and support the user's interface to the system. The help and command interface are available through all of the OS/400 program layers of the system, including the programming environments and licensed program utilities.

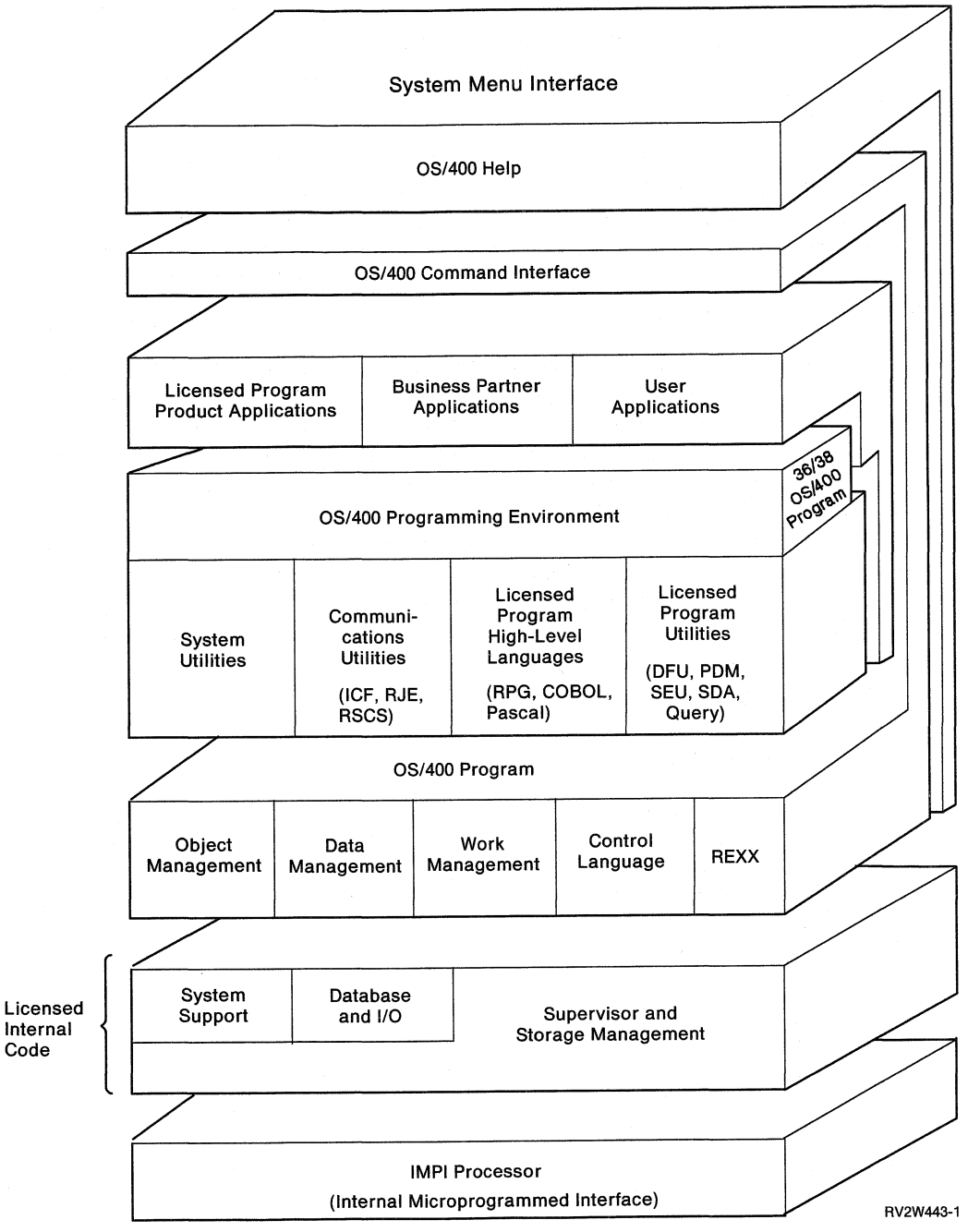


Figure 1-1. AS/400 System Structure

OS/400 functions are accessed either through the use of a comprehensive set of menus or through the control language (CL). Other AS/400 licensed programs, such as high-level languages and the Application Development Tools, also use OS/400 menus and CL.

The AS/400 system is controlled through a single, consistent control language that is supported by the operating system. The control language provides the operations normally associated with controlling the operation of a system, such as:

- Controlling the operation of input and output devices attached to the system
- Submitting batch jobs
- Ending a session with the system

In addition, many advanced functions used in data processing are provided. For example, data files and programs are created, the running of programs is controlled, and work station users can communicate with each other by using functions requested through the control language.

Although the control language is the interface through which the functions of the operating system are controlled, it is not the only interface available to the user. Specialized interfaces are available, such as data description specifications (DDS) and interactive data definition utility (IDDU), through which data in the system is described. The data is accessed and updated by high-level language programs using OS/400 functions.

The OS/400 program and other licensed programs provide the interface between the system user and the AS/400 system. These licensed programs capitalize on the advanced features of the AS/400 system, provided by both hardware and licensed internal code. Many functions that have traditionally been performed by system control programs are integrated into the licensed internal code so that they can be performed more efficiently. The interface between these functions and the system user is provided by the licensed programs that use the functions. Regardless of the function, the user does not need to be concerned about where the function is performed because the OS/400 program provides a single, consistent interface to all the system functions. Figure 1-2 shows a comparison between the traditional system design and that of the AS/400 system. With the added support provided by the system, the amount of programming effort can be reduced.

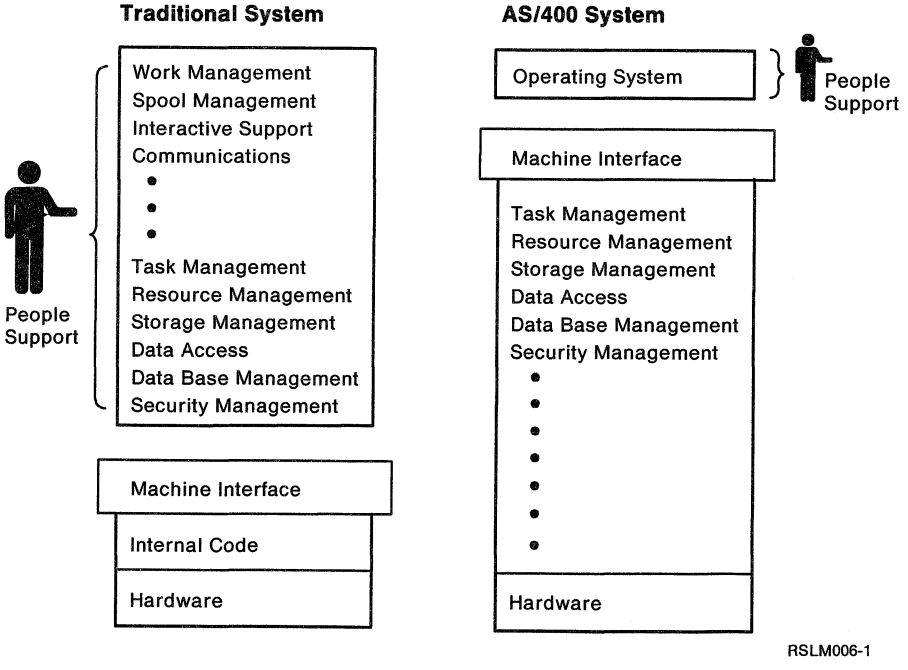


Figure 1-2. Interface to System Functions

Object Management: The term *object* refers generally to named items (such as programs or files) that are stored in the system. The object management functions allow objects to be grouped and arranged in the system. The object management functions allow users to create, update, and delete objects by name, without needing to specify the exact storage locations of the objects.

Work Management: The work management functions provide the framework through which the system and all the work performed on the system are controlled. These functions support an environment running more than one computer at a time and manage competition between jobs for main storage and other system resources. The work management functions allow work to be submitted by the user, presented to the machine to be processed, and controlled by the user responsible for systems operations.

Data Management: The data management functions support documents, database files, and device files. Data management for documents and database provides the functions required for creating and updating database files and performing input and output operations on them. Data management for the devices provides input and output operations for both local and remote devices attached to the system, including many unique functions to support the display and printer devices.

Application Development: A programmer can develop application programs interactively from a work station using the Application Development Tools licensed program, OS/400 CL commands, or the cross-platform Application Development/Cycle (AD/Cycle*) set of tools. The development activities include:

- Defining files
- Entering source programs
- Compiling programs at the same time with normal system operations without interrupting the normal flow of work on the system
- Testing programs in a protected environment so that production files can be used as input by a program being tested, but are protected from being changed by the program
- Alternating between two or more interactive jobs during the same session, such as reviewing a display of a compiler listing and reviewing the value of program variables
- Debugging a program online using the OS/400 program-provided functions to locate program errors
- Correcting the program source and recompiling the program

As shown in Figure 1-3, application development on traditional systems often requires programmers to learn multiple development tools. On the AS/400 system, all programming development services are integrated into a single application development environment.

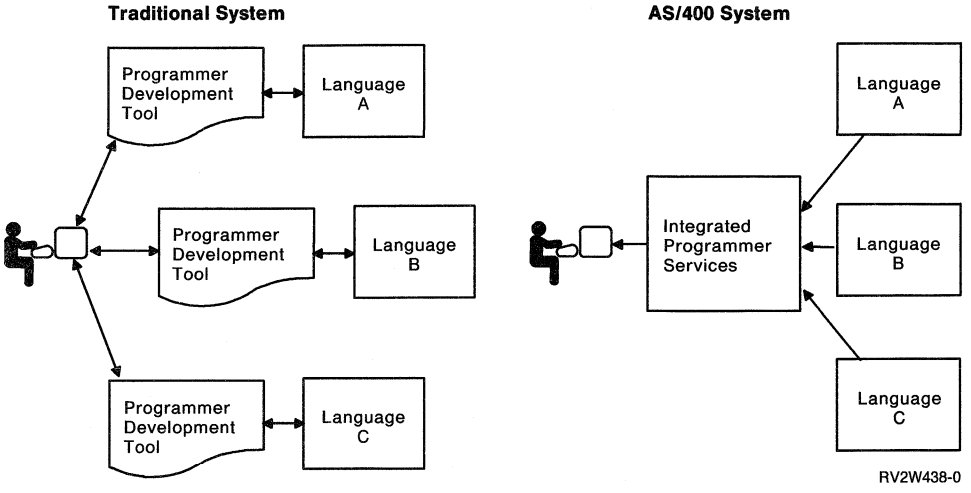


Figure 1-3. Program Development Environment

The AD/Cycle framework on the AS/400 system further integrates the development process across the product life cycle. Specifically, the AD/Cycle framework provides products that, when used together, form a framework for developing new applications and maintaining existing applications efficiently. This framework:

- Extends the IBM Systems Application Architecture* (SAA*) platform to application development
- Integrates application development tools
- Enables centralized sharing and management of application development information
- Supports enterprise and technological evolution of methodologies, processes, and tools

System Management: The AS/400 system integrates most major functions by making them a part of the operating system. For example, a user can control the operations of jobs and subsystems, respond to system messages, perform save and restore operations and so on. These operations can be performed from any work station by authorized users and are not restricted to a single person. Figure 1-4 shows the segregated functional design of traditional systems in comparison to the integration of functions on an AS/400 system.

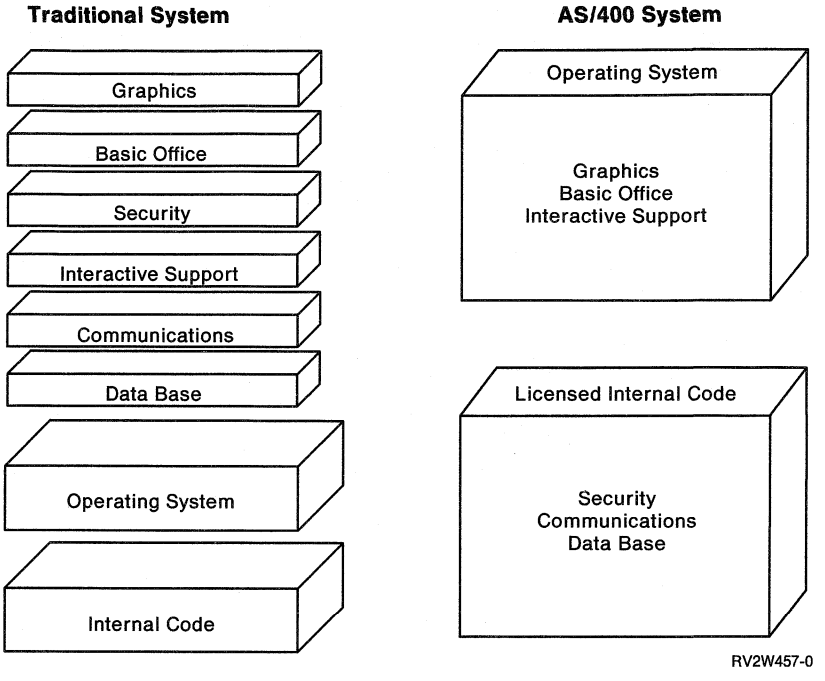


Figure 1-4. Integrated System Management

Control Language: While the menu system is the primary interface to the OS/400 program functions, the control language is also available to directly access system functions and can be used at the same time by users from different work stations. A single control language statement is called a command. Commands can be entered:

- Individually from a work station
- As part of batch jobs
- As source statements to create a control language program
- As part of a Procedures Language 400/REXX (REXX) program

To simplify the use of the control language, all the commands use a consistent naming convention. In general, the first three letters refer to the action to be taken, the next three refer to the object of that action, and the last characters, if any, provide an additional descriptor of the task to be performed. For example, the WRKJOBD command tells the system that the user wants to work with a job description. In addition, the operating system provides prompting support for all commands, default values for most command parameters, and syntax checking to ensure that a value is typed correctly before the function is performed. Thus, the control language provides a single, flexible interface to many different system functions.

Procedures Language 400/REXX: The SAA program defines a procedures language that can be used on all SAA platforms. REXX is the AS/400 system implementation of that SAA definition. As an integral part of the OS/400 program, it supports the level 2.0 SAA language definition, with the exception of natural stream support. (For more information on the SAA program, see "SAA Support on the AS/400 System" on page 9-11.)

Communications: The communications structure supports multiple architectures in a flexible and extendable fashion, by supporting multiple, communications architecture implementations and the sharing of physical resources. Documents, data, and files can be exchanged with remote systems as well as allowing remote users to access files and application programs on the AS/400 system.

Additionally, the retail communications support provides a user interface to send and receive files between an AS/400 system and a retail controller.

Licensed Programs

Licensed programs provide additional function to help the user operate the AS/400 system efficiently and to develop application programs. They range from special programming languages and programmer utilities to application enablers such as PC Support/400, providing an AS/400 interface for personal computers, and OfficeVision/400*, with its electronic mail and word processing features.

Languages

Several programming languages are available which allow the application programmer to choose the language most appropriate to the application. The following are some of the available languages:

AS/400 Pascal
AS/400 PL/I
AS/400 BASIC
C/400*
COBOL/400*
RM/COBOL-85**
FORTRAN/400*
RPG/400*

Utilities

Utilities are designed to help the application programmer create programs and to help the system operator manage the system. Some available utilities include:

- Application Development Tools helps the programmer design screens and menus, create programs, create and edit source files, and generally increase programmer productivity. The following describes some of the available functions:
  - Programming development manager (PDM) provides programmers an easy-to-use menu interface, each of which can also be accessed directly from the command line. For example, STRSEU typed on the command line starts the source entry utility function. Such flexibility allows programmers to move easily between phases of program development, such as editing, compiling, and debugging.
  - Source entry utility (SEU) provides the capability to enter and edit application program source.

- Screen design aid (SDA) provides assistance for creating screens.
- Advanced printer function utility (APF) supports advanced functions printing on the IBM 5224 and 5225 printers, including forms generation and alternate character capability.
- Report layout utility (RLU) enables programmers to create and edit report images described in data description specifications (DDS) on the AS/400 system.
- Data file utility (DFU) helps users add or change records in a data file.

- Business Graphics Utility (BGU) produces various graphic representations of the data in line, bar, and pie charts from existing data.
- Advanced Function Printing Utility/400 provides menu-driven functions that allow users to design, create, and manage overlays and page segments for IPDS (intelligent printer data stream) printing.
- Performance Tools helps the system operator fine tune system operations, such as batch processing, to achieve the highest level of performance from the system within the user's requirements.
- Query organizes data for creating reports and summaries from existing database files.
- System/38 Utilities provides support for programs, such as System/38 Data File Utility and System/38 Query, that run in the System/38 environment.

Hardware

The AS/400 system insulates the users from the characteristics of the underlying hardware by use of a layered architecture. Various models of the AS/400 family of midrange computers are available to meet the needs of all sizes of business enterprises. However, a single operating system supports the entire product line. This means programs can be run on any AS/400 system and moved between systems without change.

The machine product is made up of three layers, with a high-level machine interface (MI) separating the programmer from the detailed implementation.

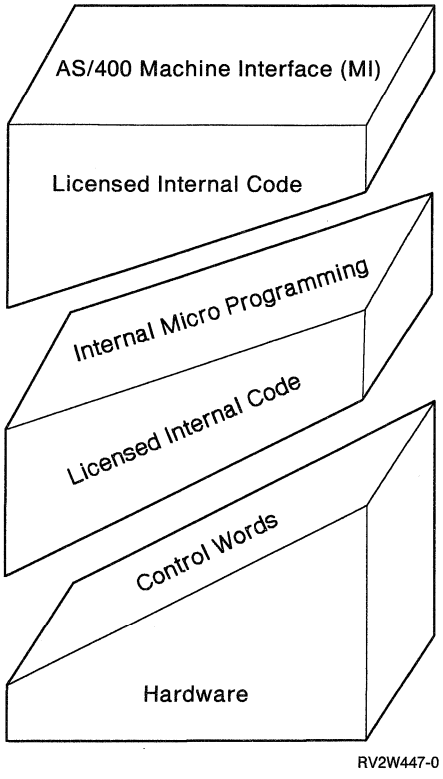


Figure 1-5. AS/400 Machine Product

Machine Interface (MI)

The MI is supported by the upper layer of licensed internal code, which contains two classes of support.

- One class is operating system functions, such as storage management, data management, and I/O support.
- The second class is the translator, which converts MI instructions into instructions at the internal microprogramming interface (IMPI) level. The conversion performed by the translator is analogous to an optimizing compiler. Individual MI instructions are converted into one or more sequential IMPI instructions or into calls to internal routines, which are sets of IMPI instructions that process the requested function. Like the AS/400 operating system, the MI instruction set is object oriented.

Internal Microprogramming Interface (IMPI)

The IMPI is supported by a second layer of licensed internal code which interprets the IMPI instructions. The IMPI also consists of two classes of support which distribute some of the functions between them.

- One class is the operating system support functions, such as storage management, security, and database integrity, task sending, task and message queuing, and I/O processing. These functions are written in vertical licensed internal code (VLIC).
- The second class consists of the traditional computational and branching instructions and extended IMPI functions. IMPI instructions are interpreted by the next lower level of microprogramming called horizontal licensed internal code. The interpretation is supported by horizontal licensed internal code rou-

tines, consisting of one or more horizontal licensed internal code instructions called *control words*. The system processor or hardware directly decodes and processes the horizontal licensed internal code control words. The IMPI instructions are a set of register, storage, and branching instructions.

System Hardware

The system hardware includes the processor and main storage, the I/O devices and controllers, and the racks, cables, and connectors that make up the AS/400 system. The hardware design allows system components to be located throughout the enterprise to meet the needs of the work place. System components, such as additional racks, I/O controllers, and storage and work station devices, can be added incrementally without reconfiguring the entire system.

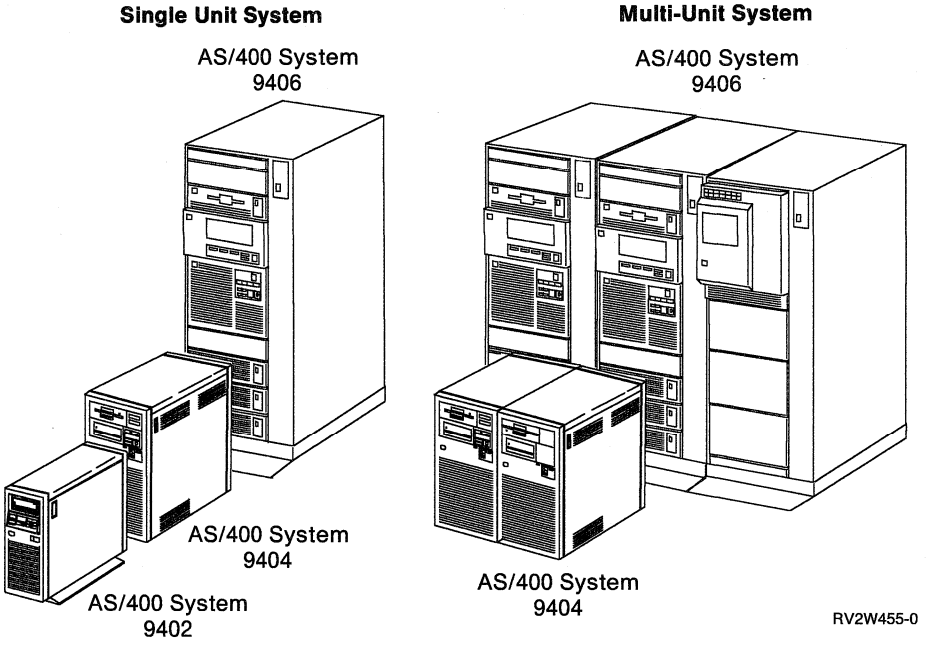


Figure 1-6. AS/400 System Configurations

System Bus I/O Architecture

The AS/400 system is designed around the system bus I/O architecture, which connects the I/O processors to the system processor. An I/O processor communicates with the system processor and controls the devices attached to it. Each I/O processor must have the correct licensed internal code loaded to communicate with the OS/400 program.

Each I/O processor has programmed instructions that run tests on its own hardware and I/O devices when the system is powered on to ensure that all devices are working correctly. But to be fully operational, each I/O processor must also load additional internal code into its storage to support its functions. For example, I/O controllers with disk, tape, and diskette devices must be able to load themselves either from the system main storage or from a tape that the operator has mounted. All of the I/O controllers attached must be able to download their licensed internal code, which is stored in the system.

Bus Control

The system I/O bus connects the service processor, the system processor, and the I/O processors, as shown in Figure 1-7. Each bus contains one bus controller, which provides control over bus arbitration and error detection and recovery.

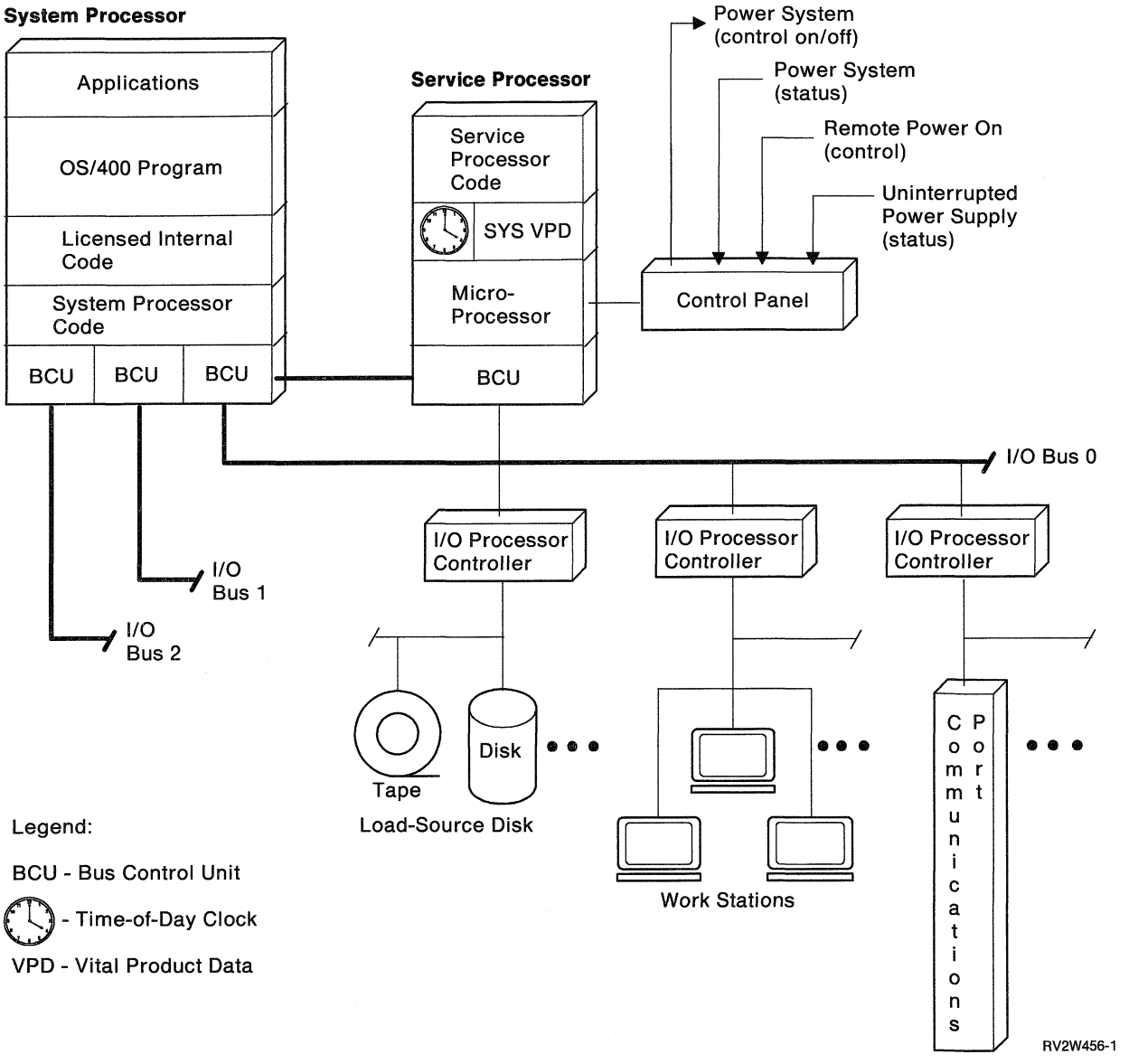


Figure 1-7. High-Level View of AS/400 Hardware and Software

The bus I/O architecture provides expandability for future system growth. Additional I/O controllers and devices are totally managed by the system processor for the end user. When a new I/O controller is added, it indicates its presence when the system is started by passing self-identifying vital product data. If the I/O controller has an I/O adapter or devices associated with it, their presence is also indicated. This information is passed to the operating system, which automatically includes the new hardware vital product data (VPD) in the system resource tables during automatic configuration. The bus I/O architecture allows the user to add new devices (tapes, diskettes, locally attached work stations, printers, and so on) to existing I/O processors, without interrupting system operations.

Flexible design of the I/O bus architecture, plus innovative design and distribution of function between the I/O processor hardware and programmed instructions, and the system processor, gives the user better performance while allowing concurrent operations between many devices.

AS/400 System Architecture

The architecture of the AS/400 system distributes functions across the elements of the system including how the system organizes work and information to facilitate operations. Figure 1-8 shows the basic system architecture.

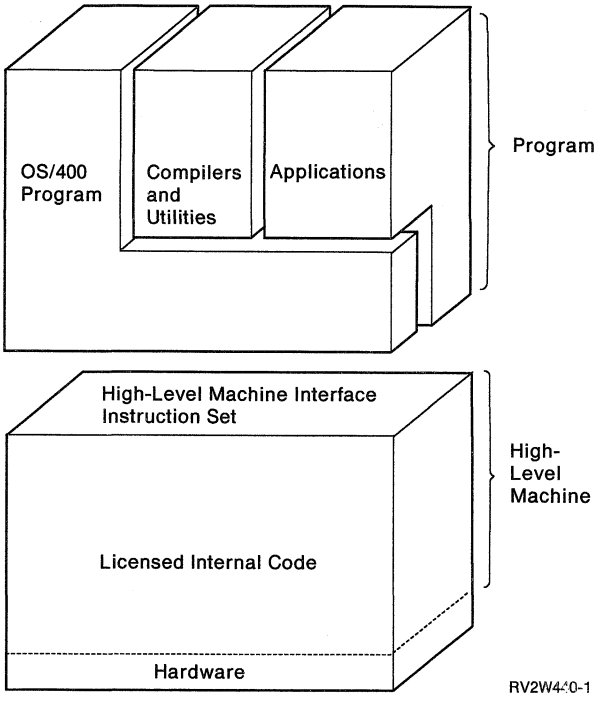


Figure 1-8. The AS/400 System Architecture

With the machine interface provided by the licensed internal code, the AS/400 system can adapt easily to new hardware and software technologies without making obsolete the existing application programs.

Object Orientation

The object-based architecture of the machine is fundamental to the overall design of the functions provided by the AS/400 system. Each type of object on the AS/400 system has a unique purpose within the system. Each has an associated set of commands with which to process that type of object. The object-oriented architecture provides a common framework for efficiently handling information and work on the system.

Using this object orientation, machine interface instructions can handle everything in a consistent manner. Each object is recognized by the system by its type, which determines how it can be used. Complex system components combine several types of primary objects to provide composite objects. (For example, a complex command can call a program consisting of several simple commands.) These composite objects are the constructs generally visible to the user; they are easier to understand and control because the complexity is handled by the system. For

example, a physical file is a user construct that is made up of a data space object that stores the data, a cursor object that provides addressability into the data space, and optionally, a data-space-index object that provides logical access to records stored in the data space. By combining primary objects, system integrity can be achieved because proven functions are used, and system performance can be improved by carefully tuning highly used functions.

Some objects are shipped with the system or created by the OS/400 program. They include objects such as the subsystem description for interactive and communications work and device descriptions created by the system during automatic configuration for detected devices. The system uses the objects to track operations and manage work submitted directly by the user or by the application programs. The system operator or user can manage these objects through programs and CL commands.

Users can also create objects to help manage their work on the system. These can include libraries to organize files, programs to manipulate those files, and even customized spelling aid dictionaries. The object management functions provided by the system help the user manage these objects.

Storage

The system uses storage as the work space for all the tasks requested either by users or programs. Storage management is handled by the system. As requests are made, objects are moved into the main storage.

Single-Level Storage

The AS/400 system is a shared storage system in which all portions of main and auxiliary storage are addressed as though they are within a single area (or level). The system uses the object name to determine where the object exists in the system. This means that the user can find objects by name, rather than by storage locations. Because operations cannot be performed on an object that is not in main storage, the system moves a part or all of the object into main storage as it is needed and moves it back into auxiliary storage when it is not needed. This transfer is controlled by the system and does not require control by the user or programmer. Figure 1-9 on page 1-15 shows how objects are moved into the single-level storage area.

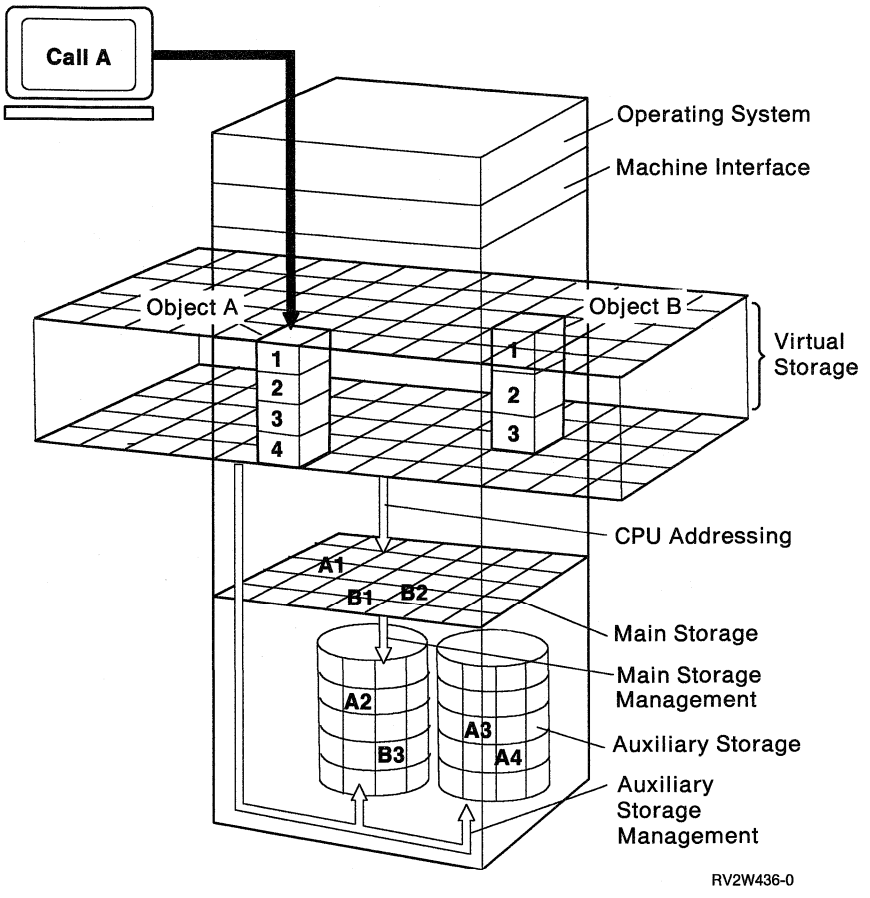


Figure 1-9. Objects are Moved into the Single-Level Storage Area

Storage Pools

Because the AS/400 system is also a system running more than one job at a time, main storage must be available for all the jobs that are running at the same time in the system. To reduce the amount of interference among jobs that are competing for main storage and to prevent a very large job from using too much main storage, main storage can be subdivided for use by different groups of jobs. Main storage is divided into storage pools, which are logical segments of main storage. When the system retrieves an object from auxiliary storage for a job, the object (or the part of the object that is needed) is moved into the storage pool in main storage that is assigned to the job that is running.

Main Storage: A storage pool provides a restricted quantity of main storage to the jobs that run within that storage pool. A storage pool is not necessarily a contiguous partition of main storage. Rather, it is 1KB (1KB is 1024 bytes) increments of main storage that are available to the jobs running in it. These increments can be located anywhere in main storage.

The AS/400 system reserves some of the main storage for system control objects that are always present in the system. These objects are not paged in and out during system operation. This storage is allocated to the system when the system is started. Other system functions, not directly related to system control, are paged in and out and use a storage pool that is assigned to the system itself (machine pool). The OS/400 program defines another storage pool that automatically includes all the

main storage that is not assigned to some other storage pool. A total of up to 16 different storage pools can be used at the same time.

Shared Objects: The sharing of objects by individual users simultaneously using the system provides efficient use of main storage. When an object (such as a program or a database file) is used at the same time by more than one system user, only one copy of that object is placed in main storage, even though the users might be running jobs in different storage pools. Any number of users can be using the object. The system synchronizes the requests between users as necessary. This object sharing reduces the amount of paging performed by the system and also reduces the need for large storage pools when users are sharing an object.

Storage Management: Most of the storage management functions are performed and controlled by the operating system. The OS/400 program provides the commands necessary for a programmer to establish the storage pools and assign jobs to them to ensure that jobs run efficiently.

In Figure 1-10, Program A calls another program. The system retrieves Program B from auxiliary storage and moves it into main storage. Program B requires a record from the data file. The system also moves the data file into main storage. Program D also uses Program B and Data File C. Because they are already in main storage, the system does not need to retrieve them. Such conservation of resources saves the system two read/write operations and improves the overall performance of the system.

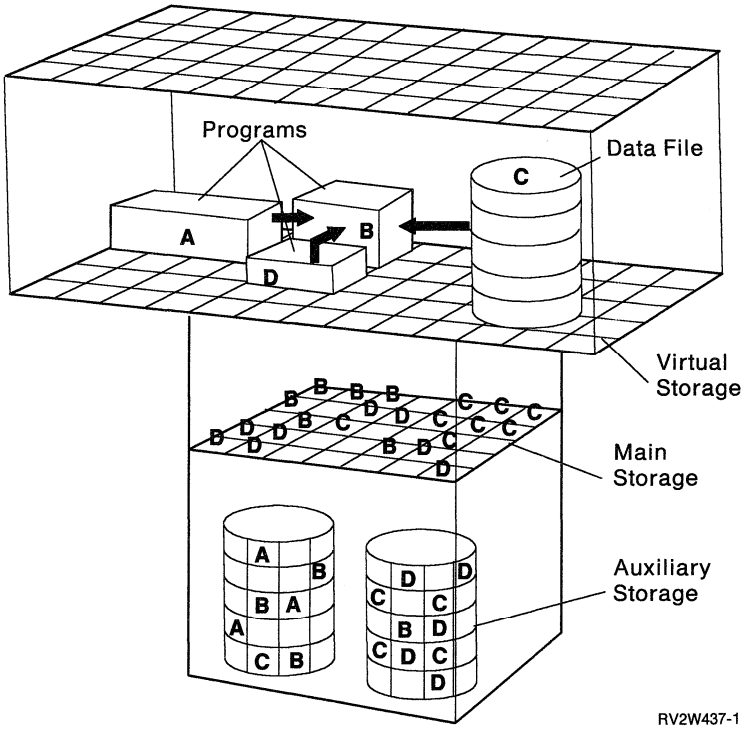


Figure 1-10. Virtual Address Space

Character Sets

A collection of characters that are recognizable by a system is called a character set. Character sets are encoded as bit patterns so that each character is assigned a predefined number or code point. Depending on the encoding scheme used, an individual character can occupy one or more bytes of storage.

Single-Byte: The most commonly used is the single-byte character set. It is so called because it requires only 1 byte to represent each of the characters. Using this character set, 256 characters can be represented. It is used for numbers as well as languages such as English and western European languages.

Double-Byte: This set of characters requires 2 bytes to represent each character. Languages that contain more symbols than can be represented by 256 characters require double-byte character sets (DBCS). These languages include Japanese, Chinese, and Korean. The AS/400 system provides support for reading and writing the characters. The AS/400 advanced DBCS printer utility supports printing these characters. Additionally, the character generation utility (CGU) allows the user to design new DBCS characters and change the existing DBCS characters available on the AS/400 system.

National Languages

It is most convenient for users to communicate in their national language, such as English, French, and Japanese. The AS/400 system supports 22 national languages. The system can support having different work stations responding in different national languages, depending on the configuration options chosen. Multinational businesses can produce documents for each of the countries they support and let the system check the spelling on each of them.

All the AS/400 licensed programs have two parts: program code needed to make the programs work and textual data, consisting of messages, prompts, displays, and online information. The textual data is translated into the different national language versions, the program code is not. The exception is the IBM Language Dictionaries licensed program, which provides spelling dictionaries for OfficeVision/400. The language dictionaries for the different national languages are not considered textual data. They are not translated from English, and there is no other textual data associated with this product.

High-Level Machine Interface

Access to the system function is provided by a powerful, consistent, high-level machine interface. The level of the machine language is closer to the functions a programmer or other user normally performs. For example, machine instructions can be used to get a database record, perform multiple programming tasks, handle storage management, and even query a database file. In traditional systems, such functions are handled by many programs. A high-level machine improves the integrity and reliability of the system. A programmer writes fewer instructions to accomplish a task on the AS/400 system. The fewer the instructions, the fewer the number of potential errors. Additionally, due to the wide range of functions available in the interface, a high-level machine reduces the development costs for operating systems, languages, and utilities. Functions available include:

Programming Language Functions: This includes data type conversions, storage allocations, procedure management, and embedded programming primitives.

Symbolic Program Debugging: The programmer can incorporate breakpoints into the source. The program can be run in debug mode stopping at the breakpoints to let the programmer check variables and field values. This operation can be done independently of other users on the system, even those who may be using the files or programs at the time.

Supervisory and Control Functions: This allows many high-level languages to be used to produce an application. For example, the data entry routines could be written in COBOL and the data manipulation in Pascal. The flow between the control by the application programs and the control by system functions could be managed by the OS/400 control language.

Data Management Functions: The functions used by most programs to access and manipulate data include declare, update, delete, retrieve, group, and recover, as well as data dictionary support. The data dictionary, which can be accessed by all of the programs, contains information about data such as meaning, relationships to other data, origin, usage, and format.

Chapter 2. Object Management

Everything on the system that can be stored or retrieved is represented as an *object*. An object is made up of a set of attributes that describe the object and a value. The attributes of an object include its name, type, size, the date it was created, a short description provided by the person who created the object, and the name of the library in which it is stored. Each object has an owner. When the system is secured, authority must be granted to use or change the object.

The value of an object is the collection of information that is stored in the object. For example, the value of a program is the code that makes up the program. The value of a file is the collection of records that makes up the file. The concept of an object simply provides a term that can be used to refer to a number of different items that can be stored in the system regardless of what the items are. Some simple objects exist below the machine interface. These simple objects are combined to form composite objects, providing the programmer with access to the combined functions, and are accessible through a single CL command.

Object management provides the functions necessary to place objects in storage and to locate objects needed for processing.

This chapter briefly discusses the topics listed in the following table. If you would like more information on the topic, and there is more information available in an IBM manual, the manual listed in the right column is a good first reference.

Topics	First Reference
Object types • Libraries • Folders • Files • Programs • Command Definitions • Queues • User profiles	*Programming: Control Language Programmer's Guide*, SC41-8077
Object management	
Security	*Security Concepts and Planning*, SC41-8083

Other more specialized objects exist that relate to specific system tasks such as work management (jobs, subsystems, and device descriptions) and recovery strategies (journals and journal receivers). These are discussed in Chapter 10, "Work Management" and Chapter 11, "System Management and System Recovery."

Object Types

Different object types have different operational characteristics. These differences make each object type unique. For example, because a file is an object that contains data, its operational characteristics differ from those of a program, which contains instructions.

Objects are arranged with a common object header and a type-dependent functional portion. For example, a program object header would contain a description of the object including the type, (program), and the owner, (the user who created the program). The functional part of the program object contains information that governs the way the object can be used. This allows the system to perform operations collectively on all objects, as well as permitting each object to be tailored for its own purpose. Figure 2-1 shows the structure of an object.

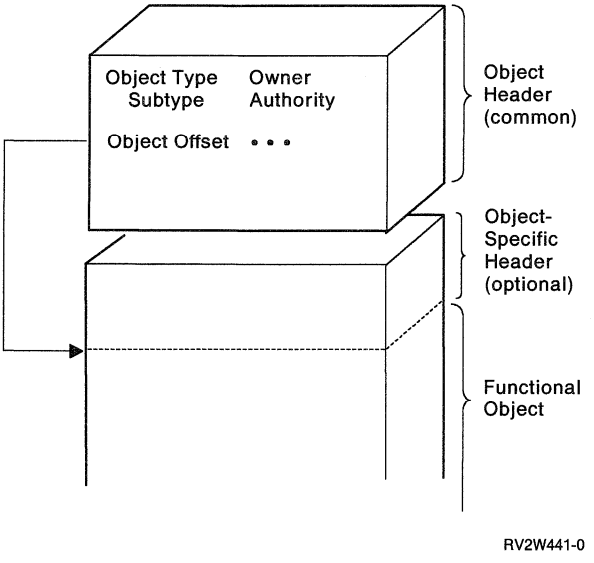


Figure 2-1. Structure of an Object

Each object has a name. The object name and the object type are used to identify an object. The object name is explicitly assigned by the system for system-supplied objects or by a user when creating an object. The object type is determined by the command used to create the object. For example, if a program were created and given the name OEUPDT (order entry update), the program could always be referred to by that name. The OS/400 program uses the name (OEUPDT) and object type (program) to locate the object and perform operations on it (such as copy, move, and delete). Several objects can have the same name as long as their object types differ, or as long as they exist in different libraries.

There are several types of names supported on the AS/400 system. The rules for each type are summarized in Figure 2-2. In CL command definitions, the name requirements are used to determine the types of names allowed for a given command.

*Figure 2-2. Allowable Characters for *NAME, *SNAME, and *CNAME*

Type of Name	First Character	Other Characters	Min. Length	Max. Length
*NAME[1]	A-Z, $, #, @	A-Z, 0-9, $, #, @, _, .	1	10
*SNAME[1]	A-Z, $, #, @	A-Z, 0-9, $, #, @, _	1	10
*CNAME[1]	A-Z, $, #, @	A-Z, 0-9, $, #, @	1	10
Quoted name[2]	"[3]	Any except blank, *, ?, ', ", X'00'-X'3F', and X'FF'	3	10

Notes:

1 The system converts lowercase letters to uppercase.
2 Quotation marks (") can only be used for basic names (*NAME).
3 Both the first and last characters must be quotation marks (").

Reprinted with the permission of *News 3X/400*.

Libraries

A *library* is an object that is used to group related objects and to find objects by name. Thus, a library is a directory to a group of objects.

Libraries can be used to group the objects into any meaningful collection. Objects can be grouped according to security, backup, or processing requirements. For example, all personnel data files could reside in a single library with access to the files limited by authorization to the programs and to the library. Backup and recovery procedures are simplified by saving the related files as a group.

The number of objects contained in a library and the number of libraries on the system are limited only by the amount of storage available. Thus, the number of libraries on a system can be tailored to the way the objects are used.

Object Grouping: The object grouping is a logical grouping and does not affect the location of the object in storage. Thus, objects in a library are not necessarily adjacent to each other. The size of a library, or of any other object, is not restricted by the amount of adjacent space available in storage. The system finds the necessary storage area for objects as they are stored in the system. If an object increases in size, the system automatically allocates additional storage to the object.

Objects are placed in a library when they are created. An object can be moved from one library to another, but a single object cannot be in more than one library at the same time. When an object is moved to a different library, the object is not really moved in storage, but it is located through a pointer associated with the new library.

Naming: A library name provides another level of identification to the name of an object. As described earlier, an object is identified by its name and its type. The name of the library further qualifies the object name. The combination of an object name and the library name is called the *qualified name* of the object. An *unqualified name* contains only the name of the object, and is useful only when the object and type are unique on the system or within the program's search path (library list). An unqualified name is always valid. However, if the name and object type are not unique within the library search list, then the first occurrence will be used.

Two different objects with the same name can exist in the same library, only if their object types differ. However, two objects with the same name and type can exist in

different libraries. Because of this, a program that refers to objects by name can be used to process different objects (residing in different libraries) without any changes to the program. A user who is creating a new object does not need to be concerned about names used for objects in other libraries.

There are three types of libraries: system, user, and product.

System Libraries: The system libraries are used to organize information needed by the system to perform tasks and include:

QSYS — Contains objects that are provided as part of the operating system

QHLPSYS — Contains help information such as the index search and command help

QRECOVERY — Contains objects gathered from main storage during a recovery procedure

User Libraries: IBM-supplied libraries are used to manage jobs and objects. Additionally, users can create their own libraries to organize work on the system for specific applications.

QGPL — Shipped with the system to be used for user objects provided by the OS/400 program.

QTEMP — Created by the system to contain temporary objects for each job and deleted when the job ends.

Current — A library name specified in the user profile for objects created by that user. Typically, it would be a user-created library. The default current library is QGPL. It has nothing to do with the library associated with the program being run. Objects created, but not qualified with a library name, go into the current library.

User-created — The user can create libraries to store objects pertaining to different tasks. For example, the INVEN library could contain all inventory information, programs, and data files.

QUSRTOOL — Shipped with the system, this library contains tools and helpful programs. Programs include system management utilities and tutorials.

Product Libraries: These libraries contain the licensed programs and related objects. For example, QRPG contains objects related to the RPG/400, such as messages, programs that make up the compiler, and related commands supplied by IBM. A product library is shipped with each licensed program.

Library Lists

An object name can be specified as a qualified name (where both the object name and library name are specified) or as an unqualified or simple name (where the library name is not specified). If a qualified name is specified, the operating system attempts to find the object in the specified library. If a simple name is specified, the OS/400 program searches a list of libraries until it finds the first occurrence of the object of that type and name or until it has searched all the libraries on the list without finding the object. The libraries that are searched, and the order in which they are searched, is determined by a search list called the *library list*. The OS/400 program automatically creates a default library list including the system library, product library, and the user's current library, for each job when the job is started.

Separate libraries and library lists can be created for each user or group of users. Library lists include the system, product, current, and user libraries.

System

Libraries in the system part of the library list are searched before libraries in the rest of the list. This specifies the libraries necessary to run system functions. When the system is installed, the system part of the library list consists of the system library (QSYS), the system file for user data (QUSRSYS), and the system help information (QHLPSYS). The list can be changed based on user needs.

Product Library

The product library part is changed by the system as users run commands or menus to include the library where the program is stored. The product library will vary while a job is running, based on the function being performed. For example, if a CL program also includes commands to COBOL and Pascal compilers, the product libraries for those compilers are accessed when those jobs are called.

Current Library

The current library is the default library used for the creation of objects. For example, users can specify the current library for a job using CL commands, in the user profile, in a CL program that calls the job, or from the sign-on screen. The current library can be one of the libraries from the user library list or can be another library to which the user has authority.

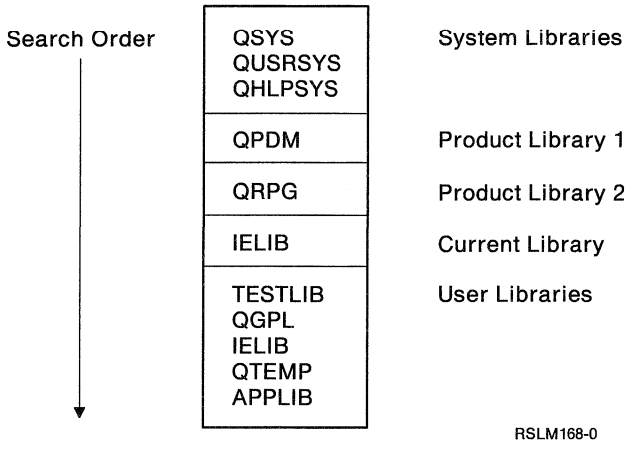
User

The user part contains the libraries used by application programs to perform their functions. When the system is installed, this part contains the general-purpose library (QGPL) and the job's temporary library (QTEMP). When a system has a number of user-defined libraries, the user part of the library list can vary between different jobs. For example, for an order entry job, the user part of the library list might include:

- OELIB (order entry library)
- QGPL (general-purpose library)
- QTEMP (job's temporary library)

During a job, the user can specify a different current library and user part of the library list. This would change the libraries used and the order in which the libraries are searched.

The following diagram shows an example of a library list and the search order:



Strategies for Management: Specifying only the object name and using the library list to search for the object can make the AS/400 system easier to use and more flexible. A library list can be designed for each job to ensure that the correct objects will be located without using the qualified names. This approach provides advantages such as:

- *Easier testing of application programs.* Libraries can be created to contain sample data when programs are tested. In the example above, the application programmer has created a test library, named TESTLB, which contains the new program and the data used to debug the programs. The actual data is contained in APPLIB. Because a library containing test objects is placed before the normal production library in the library list, those files will be used by the application and the integrity of the real data is preserved. The object names used in the test library are the same as those used in the normal production library. After the program is fully tested, that library can be removed from the user's library list and the program is moved to the application library. The program then processes objects contained in the production libraries, and the object names do not need to be changed in the program.
- *Flexible use of the libraries on the system.* As processing needs change, existing libraries may need to be divided into more than one library to help simplify the organization of objects on the system. This change does not require the names of objects in the programs to be changed. Only the library lists used by the jobs need to be changed.
- *Access to multiple objects from a single program.* Different system users may operate on different objects using the same application program. The library list for each user's job ensures that the correct objects are used by the program for each user.

Because of these advantages, qualified names are not usually specified in the program when existing objects are used. However, the qualified name can be specified in situations where it is more efficient than changing the library list, or where several objects exist with the same name and type to ensure that the correct object is used.

Folders

A *folder* is a named object that is used as a directory for documents and other folders. For example, users from a personal computer network could store personal computer programs, files, and documents created with their word processing program in folders on the AS/400 system.

Folders can be filed within another folder. Folders within folders are similar to a filing cabinet. The first folder, or root directory, as it is called on the personal computer, would be the file cabinet itself. Folders within folders can be compared to the drawers within the file cabinet. The actual files within the drawers are folders within folders within folders, and so on. The structure begins at the first directory and proceeds to related folders. This structure was chosen for folders so that personal computer users can share files within the network (from PC Support/400 through the AS/400 system).

A *folder path* is a list of the folders within folders needed to find a document or object within a folder. Figure 2-3 shows how folders and folder paths are used to find information. Notice the organization within the folders. The objects are grouped according to categories. This makes it easier for the user to locate the documents when editing. In the example, the user organized the documents by reports and memos. The report category is further divided by the month of the report. The path to the document APRIL begins in MYTEXT, the root directory, through the REPORTS folder to the MONTHLY folder.

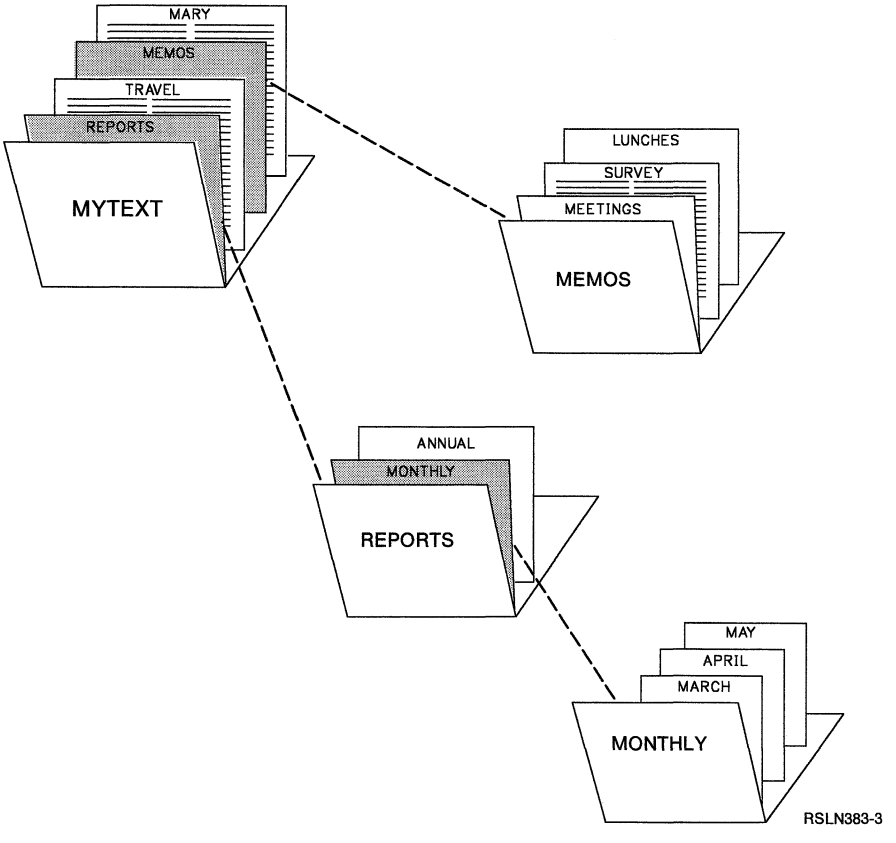


Figure 2-3. Folder Paths to Find Information

In addition to the functions provided by the AS/400 operating system, PC Support/400 allows users of personal computers connected to an AS/400 system to share folders and documents with AS/400 system users. Information is stored on

the AS/400 system but is accessed by the PC Support/400 user as if the data resided on a personal computer disk. The commands work the same when a drive is assigned to a folder as they do when the drive is physically located on the personal computer. (Without PC Support/400, the personal computer connected to the AS/400 system functions as a dependent work station.)

Files

A file is an object that contains either a set of related records handled as a group or a *stream* of data. One of the most common types of files that contains records is the database file. A document is a type of file that contains only a stream of data.

Programs perform read and write operations on these files. For example, a file could contain all of the payroll information including employee numbers, salary, benefits, and withholding information. To create a report, a program would need to open the file, read information from the file, process the information, close the file, and write the results to a device.

Device files contain a description of how data is to be presented to a program from a device or to a device from a program. For example, when sending a request to a printer, an application must format the data according to how the printer can process it, for example, the placement of the data on the page. On the AS/400 system, the device file can be used to format the data for multiple application programs so that the individual programs do not have to include the information.

Programs

There are two general types of programming languages on the AS/400 system:

- Interpreted
- Compiled

For interpreted languages, such as REXX, the program only exists in source form. As a result, such programs are contained in a file object only. For compiled languages, the source code is processed by the compiler which generates a program. The source code exists in a file object and the compiled program exists in a program object.

A program is an object containing a set of instructions that tell the system where to get information (input), how to process it, and where to put the results (output). When the system compiles the program description, the object *type* identifies it as a program. Because it is a program object, the system begins to read the lines of code and to process the commands. A program object is compiled from a set of instructions contained in a source file.

Command Definitions

Command definitions are objects that contain the description of the command (including the command name, parameter definitions and syntax checking information) and identify the program that performs the function requested by the command. The system provides a large set of commands, but the user can also create and customize commands. The command definition statements, used to define commands, contain the information necessary to prompt the work station user for input, to check the accuracy of that input (for example, range of values), and to define the values to be passed to the program that is called when the command is run. Existing commands can be changed by the user. For example, the command could be changed to restrict possible parameter values, change the defaults, or even change the name to reduce the number of keystrokes needed.

Queues

A queue is a list of objects that are waiting to be processed. These range from user requests from interactive work stations to batch jobs waiting to be processed. There are four types of queues: job, output, message, and data. Even though they have similar functions, they are unique object types.

Job

The system handles multiple operations at the same time and supervises the sharing of system resources. As opposed to the time it takes to process the data, interactive programs spend a relatively large amount of time waiting for users to enter data. The system recognizes the wait times and uses them to process other programs. Batch programs seldom require any display station input, however, they often require longer times for calculating and processing. Even though the system can easily recognize batch programs, it can do little to improve their performance. What the system can do is minimize the effect of batch programs on the response time for interactive programs.

The job queue manages the batch requests submitted by the users. A user can then continue to work at the work station on other tasks while the system processes the request. For example, the data for the payroll is entered interactively, but the processing of the data, including calculating tax deductions, is typically processed as a batch job. Management of the queue allows the user to arrange jobs by priority which, in turn, can affect system performance. For example, jobs requiring extensive processing can be set to run overnight and would not affect work station users on the system.

Output

As a program processes a request to print data, it gets the data from a database file and uses the print device files to format the data. The formatted print files are placed on an output queue until a writer is ready to send the information to a printer. Like the job queue, requests in the output queue can be arranged by priority depending on user needs.

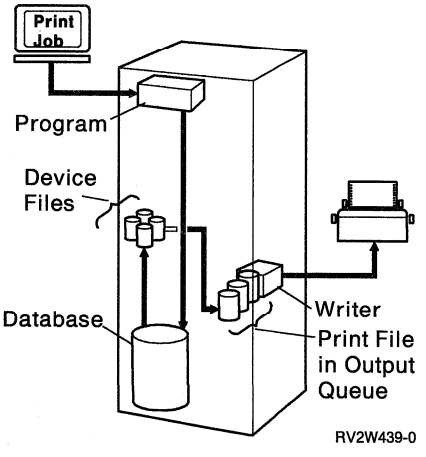


Figure 2-4. Objects Involved with Output

Message

Communication between programs, between jobs, between users, between users and programs and between users and the system occurs through messages. When a message is sent to a program or to a system user, it is placed on a queue associated with that program or user. The program or user sees the message by receiving it from the queue. The OS/400 program automatically provides message queues for:

- Work stations on the system
- Users enrolled on the system
- Users responsible for system operation
- System history log

Additional message queues can be created by the user to meet any special application program requirements. For example, a message queue called PRGQ could be created for an application to receive messages from the system regarding the status of objects the application requires. If the system detects that an object is damaged, a message is sent to PRGQ which, in turn, notifies the user that the object may need to be restored.

Depending on the severity of the message, messages sent to message queues can be held until the program or user decides to receive them. In the example above, a break message could be sent, which would attempt to display the message to the user at the time the message was received and give the user the opportunity to respond immediately. If the user is not signed on the system, the message is placed in the message queue.

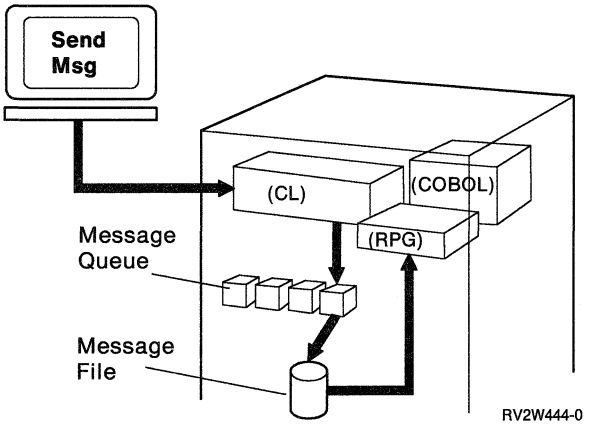


Figure 2-5. Messages in a Queue

Data

When running an application consisting of several programs, it is often necessary to pass data and variables to other programs. Programs can set up data queues to be used by the entire application so that all programs can refer to a single set of data and variables passed to the programs through the queue. Other applications can also refer to the information in the queue. Like other queues, the values can be used immediately or be held for processing at a later time.

User Profiles

A user profile is an object that identifies a particular user or group of users to the AS/400 system. The user is known in the system by a user profile name. When a work station user signs on, the user ID is used to find the user profile. The password is defined in the user profile. All AS/400 system security functions rely on a user profile to describe each user. The user profile identifies which objects and functions the user is authorized to use, identifies the first menu the user sees, and also identifies a program that is to be run when the user signs on the system. For example, a data entry operator requires the data entry menu and a user responsible for system operations might require the system menu.

A group profile is used to provide the same profile for a group of users. For example, a group profile could be assigned to a group of data entry operators with update authority for the data files. This eliminates the need to assign the authority to each user individually.

Object Management Operations

Programs, files, and queues all have many similar system requirements, such as storage space or security. The functions performed by most of the control language commands are applied to objects. Some commands can be used on any type of object and others apply only to a specific type of object. Some are performed by the user, others are performed automatically by the system.

General Operations

The operational characteristics of objects vary depending on the type of object involved. For example, some operations that apply to files do not apply to programs and vice versa. Additionally, *Work with* operations differ from *Display* operations in that the object can be changed. The following set of operations apply to most object types:

Allocate	Reserves an object for exclusive or shared use.
Authority, Display	Displays the user's ability to access an object.
Authority, Grant	Provides functions that control user access to an object and protects the object owner's rights, such as granting the right to change an object.
Authority, Revoke	Revokes user access to an object and protects the object owner's rights.
Change	Provides the functions necessary to change the attributes of user-created objects. For example, users can change the text description of a particular object.
Clear	Deletes the contents of an object but does not delete the object.
Copy	Provides the functions necessary to copy objects.
Create	Provides the functions necessary to create user-defined objects. For example, once a library has been created, objects can be created in it or moved into it.
Delete	Provides the functions necessary to remove user-defined objects from the system. When a library is deleted, any objects in it are also deleted.

Display Contents	Displays the contents of an object or a specified group of objects. When used in reference to a library, a list of all the objects in the library or libraries is displayed. The information can be printed from this operation.
Display Description	Displays the descriptions of a group of objects by object type (all data files), by a generic name (all objects whose names begin with *EMP*), or by generic name and object type (all data files whose names begin with *EMP*). The information can also be printed from this operation.
Dump	Copies to main storage or to an external media the contents of any object stored in a library. The user must have authority for both the object and the library.
Move	Moves an object out of its current library and into a different library. After the operation is complete, the object is no longer available through the original library. This operation is not valid for all object types.
Rename	Changes the name of an existing object. The object contents does not change, nor does the object change libraries.
Save and Restore	Pertains to an object, a group of objects, or an entire library. These operations provide the functions to save copies of objects to an external media or save file and restore them to the system. These functions can be used to provide backup copies of objects that can be used for recovery procedures. For example, applying this operation to a library saves and restores all objects contained in the library.
Transfer Ownership	Allows the ownership of an object to be transferred to another user.

System Operations

System functions are performed automatically by the system to ensure that objects are processed in a consistent, secure and proper way. For example, if an updated version of a data file is to be copied over an existing file, the system ensures that the copy operation is successfully completed before deleting the original version of the file. The following is a list of system operations:

Authority Verification	Checks the object, the function, and the user to verify that the user has the authority to perform that function on that object.
Commitment Control	Ensures that a function performed on an object is ended, whether successful or not. If the function cannot be successfully completed, the object is returned to the state it was in before the function was requested.
Detect Damaged Objects	Monitors for errors during the processing of objects and communicates to the user failures that result from the unreadable contents of objects. The system is designed for monitoring and communicating these failures quickly to maintain integrity.

Enforce Locks	Ensures that the integrity of the contents of objects is preserved when two or more users try to use an object at the same time.
Verify State	Various checks are available such as existence and lock state (whether the object is available for updating or is in use by another user).
Verify Types	Checks the type of object and the type of function being performed on the object to verify that the function can be performed on that type of object.

Damaged Objects

If for some reason the system can no longer process an object correctly, that object is considered *damaged*. Object damage is the general term used to refer to a class of infrequent failures that occur for a variety of reasons. Such failures include:

- Logic failures internal to the system, causing a portion of the object to be updated incorrectly.
- Hardware failures causing some portion of an object to be inaccessible from auxiliary storage.
- System failures resulting from external environmental influences, such as power failures.

If the system is operating correctly when a damaged object is encountered, the system informs the user by sending a message to the program encountering the damage. If damage to a particular object is encountered for the first time, a message is also sent to the user designated in their user profile as the *system operator.*

Object damage encountered while the system is being started is communicated through the lights on the control panel, if the damage is extensive enough that the system cannot be started. These provide the system operator with the information necessary to start problem analysis.

When IBM-supplied OS/400 objects are damaged, some are automatically deleted and re-created by the operating system, while others are deleted and restored during the reinstallation of the OS/400 program. This ensures that, in the case of damage to an object necessary for the operation of the OS/400 program, the system can still operate. For example, user-modified objects, such as QGPL and QUSRSYS, are restored to their original state (as they were at installation or at the last backup). The system operator message queue is re-created and set back to an empty queue, as it was when the system was installed.

The operating system does not automatically replace user-defined objects if they are damaged. If users have changed system objects or created new ones, the system has no way of determining what changes were made by the user. When the OS/400 program encounters a damaged object, it sends a message providing the necessary recovery procedures to the user. The user can delete these objects, using the delete object commands, and restore them from a backup copy. This allows the user to return objects to a known condition. This process takes time as the system must be completely shut down while the OS/400 program pages through main and auxiliary storage collecting fragments of objects. The user issues the command to start the process and is in total control of the recovery process. Fragments of objects and damaged objects that are no longer usable can be deleted by reclaiming auxiliary storage. The remaining fragments are placed into a special

library (QRECOVERY). Those that can be restored, are; the rest should be deleted by the user.

Security

The system is shipped with minimal security active. This means that any user (including remote users starting communications jobs) can use any resource.

Selection of the level of security for the system is done by changing the system value QSECURITY and then doing an initial program load (IPL) of the system. The level of security selected determines what authority users are given by default and the level of enforcement performed by the system.

The levels of security are:

Level	Description
10	Requires only a user ID to sign on. After signing on, the user has access to all objects on the system. (This is the way the system is shipped.)
20	Requires a user name and password to sign on. After signing on, the user has access to all objects on the system.
30	Requires a user name and password to sign on. After signing on, the user must be given authority to objects before they can be used.
40	Requires a user name and password to sign on. After signing on, the user must be given authority to objects before they can be used. System integrity checking is active. All attempts to use undocumented and unsupported interfaces fail.

At all levels of security, the system-supplied defaults in the user profile can be changed and authority can be specifically given or taken away from the users.

User Profiles: User profiles are an important part of security and are used to identify users on the system. They tell the system who can sign on and what functions the user can perform on the system resources after signing on. The term *user* also includes remote users starting communications jobs. Therefore, user profiles should also be created for remote users.

Resource Security: Resource security is the security measures used to authorize users to specific objects. When the system value for security is set to level 30, the users must be given authority to each object they need to access.

Types of Authority

The authority provided by the system allows tailoring a user's environment on the system. There are two major types of system authority:

- Special
- Specific

Special Authority: Allows a user to perform system control operations such as saving the system, controlling other users' jobs, using the system service tools, controlling spooled files, and creating user profiles. Special authority is defined in the user profile by specifying the class of user or by tailoring the special authorities.

The class of user is the category which best identifies the authorities the user will be given. For example, programmers are typically given the authority to create,

change, and delete program objects. Data entry operators are typically given the authority to enter and update data records but not to change the actual program.

The special authority you specify by defining the user's class determines what system control operations and what menu options the user can use. If a specific user class does not provide the required special authorities, the special authorities can be tailored to fit the user's need.

Specific Authority: Specific authority determines how a user can use a specific object. This includes object authority and data authority. *Object authority* allows a user to perform operations on an object (object authority) such as move or rename the object, control the object's existence, and use the data (data authority) contained in the object. Object authority is given to users with the Grant Object Authority command. *Data authority* allows users to access the contents of an object. For example, users can read, add, update, and delete entries from an object.

Authorization Methods

The OS/400 operating system provides the following methods for assigning authority:

- Private authority
- Public authority
- Authorization lists
- Group profiles
- Adopted authority
- Authority holders

Private Authority: Private authority is the authority specifically given to a user for an object. The objects that a user has authority to are recorded and listed in the user profile. A user's private authorities override any other authorities.

Public Authority: Public authority is specified for the object when the object is created or can be defined in the authorization list that secures the object. This is the authority that applies to all users on the system who have not been granted private authority.

Authorization Lists: An authorization list contains a list of users and the authority that each user has to the objects that the list secures. One user's authority may differ from the authority of another user on the list. If a user on the authorization list also has private authority to the object secured by the list, the user's private authority overrides the authority specified for him on the authorization list.

Group Profiles: A group profile specifies the same authorities to a group of users. A group profile provides a way of specifying the same set of authorities to multiple users by allowing each member to use the authority assigned to the group profile. Private authority to an object overrides the authority of the group profile.

For example, a group of data entry personnel can be assigned to a single group profile. The group profile only allows the users to enter and update entries. This eliminates the need to assign each user profile authority individually. The coordinator of the group may require additional access to the files and can be given private authority, which will override that allowed by the group profile.

Adopted Authority: When a program is created, it can specify that the program always runs under the program owner's user profile. A user does not need authority specifically given to him for the objects used by the program, but uses (adopts) the program owner's authority. The user has authority for the objects used

by the program only when he is running the program and other programs called by the program. The authority given to a user by the program adopt function is in addition to any private authorities the user has to objects.

For example, a user has the authority to run the account data update program. Through this program, the user has read and update authority to the account address information and only read authority to the account balance information. While the users are running this program, they have the same access to the data. Once they exit the program, their authority to the same data is limited to what is specifically assigned through private authorities and related group profiles.

Authority Holders: An authority holder allows security information to be specified for program-described database files before the files actually exist on the system. Then, when a file is created with the same name as the authority holder, the security information specified in the authority holder is linked to the file. This allows the system to keep authorities for System/36 environment applications that often delete files and then create them again. If the file is deleted, the authority information is kept with the authority holder to be used again when the file is created again or restored from tape or diskette.

Security Auditing

The AS/400 system allows for auditing of security related information. Auditing is enabled by the creation of the QAUDJRN journal and setting the QAUDLVL system value to dictate the types of access to be recorded. Journal entries are added to QAUDJRN by the system for each occurrence of the requested access type. Accesses recorded include authority failures, integrity violations, and so on.

Chapter 3. User Interface

The AS/400 user interface spans the needs of all types of users operating in a single or multiple-user environment, whether they are new users, data-processing professionals, or programmers.

The AS/400 system provides several ways for users to work with the system. It provides inexperienced users with a simple method of starting system functions. More experienced users can interact with the system using menus and prompts that emphasize efficiency. Advanced users can start system functions with a minimum of supporting information online.

The primary interface is a series of menus that allows users to select desired tasks. As users become more experienced, they can specify multiple actions from the *work with* lists, take a direct path to any menu, or enter commands directly. All of the methods can be used interchangeably. This allows users to use the more advanced interface methods without requiring them to abandon the ease-of-use associated with the menu interface.

If users do not understand a display or how to start a task, help is available from any screen through the AS/400 help function. Help ranges from information about a complete display to information about an individual item on the display.

Online tutorials are provided with the operating system for those users who need basic training using system functions. They are available to any user from the User Support menu and include topics such as object and work management, basic communications, and database concepts. Each online tutorial module takes approximately 25 to 45 minutes to complete. Users learn at their own work stations, at their own speed.

The national language support provided by the AS/400 system allows users to interface with the system in the language of their choice. The AS/400 system can support many languages at the same time. All menus, help information, and messages are displayed in the language specified in each user's profile.

This chapter briefly discusses the topics listed in the following table. If you would like more information on the topic, and there is more information available in an IBM manual, the manual listed in the right column is a good first reference.

Topics	First Reference
Displays Assistance levels	*New User's Guide*, SC41-8211
Commands Messages	*Programming: Control Language Programmer's Guide*, SC41-8077

Displays

The AS/400 system provides a number of different displays as a part of its user interface. Each display plays an important role in providing the users with an efficient way to enter and retrieve data.

Types of Displays

The display types that make up the user interface are:

- Menu
- Entry
- List
- Information
- Help

Menu

The menus allow users to select the task they would like to perform without having to use system commands. As can be seen in Figure 3-1, the main menu allows the user to select from a broad category of objects and tasks. Some choices show specific product task menus, such as the office menu that appears when the office option is selected from the main menu. These task menus provide users with a more refined group of choices regarding tasks or objects available. Specific menus are provided for common groups of tasks, such as office, programming, and working with files.

As users become more familiar with the menu titles, a specific menu can be called by entering the GO command and the name of the menu on the command line found on most system menus. If users know the name of the menu they want to use, all they need to do is type GO and the menu name, and the requested menu will appear. With menus, users do not need to remember any commands, keywords, or option names to find the object they want to work with or the task they wish to complete.

The command line, found on most displays, allows the user to request functions with a command without using the menu options. For example, WRKDOC entered on any command line (as seen in Figure 3-1), calls the Work with Document function. The command line allows more experienced users to access the tasks or objects directly.

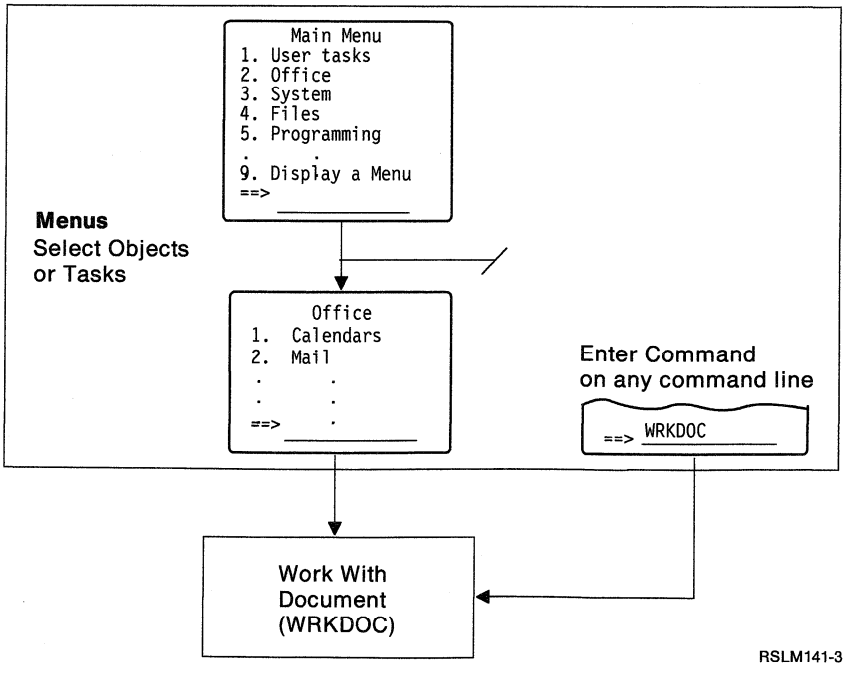


Figure 3-1. Menu and Command Interface

Entry

Entry displays are shown by the system when users select a task from a menu or when they enter a command on the command line. Entry displays are very easy to use, they consist of a single column of entry fields preceded by a descriptive phrase called a prompt. This prompt reminds the user which value needs to be specified in that particular field. A list of the acceptable values for that field may also be provided for the user to choose from. The user can then select from the list, as seen in Figure 3-2, instead of typing the needed information, thus requiring less interpretation by the program.

When an entry display asks the user for information, only information related to the user's request is asked for, and values assumed by the system (default values) are already entered in the fields. Infrequently used values are not asked for on the first display. If the system determines that more information is necessary based on the values typed on the first display, more fields will be shown. All other prompts can also be seen by the user, if needed, by pressing the Additional Options function key available on each entry display.

Through the use of conditional prompting and default values, the user does not have to analyze each of the many possible fields, but can do so if necessary.

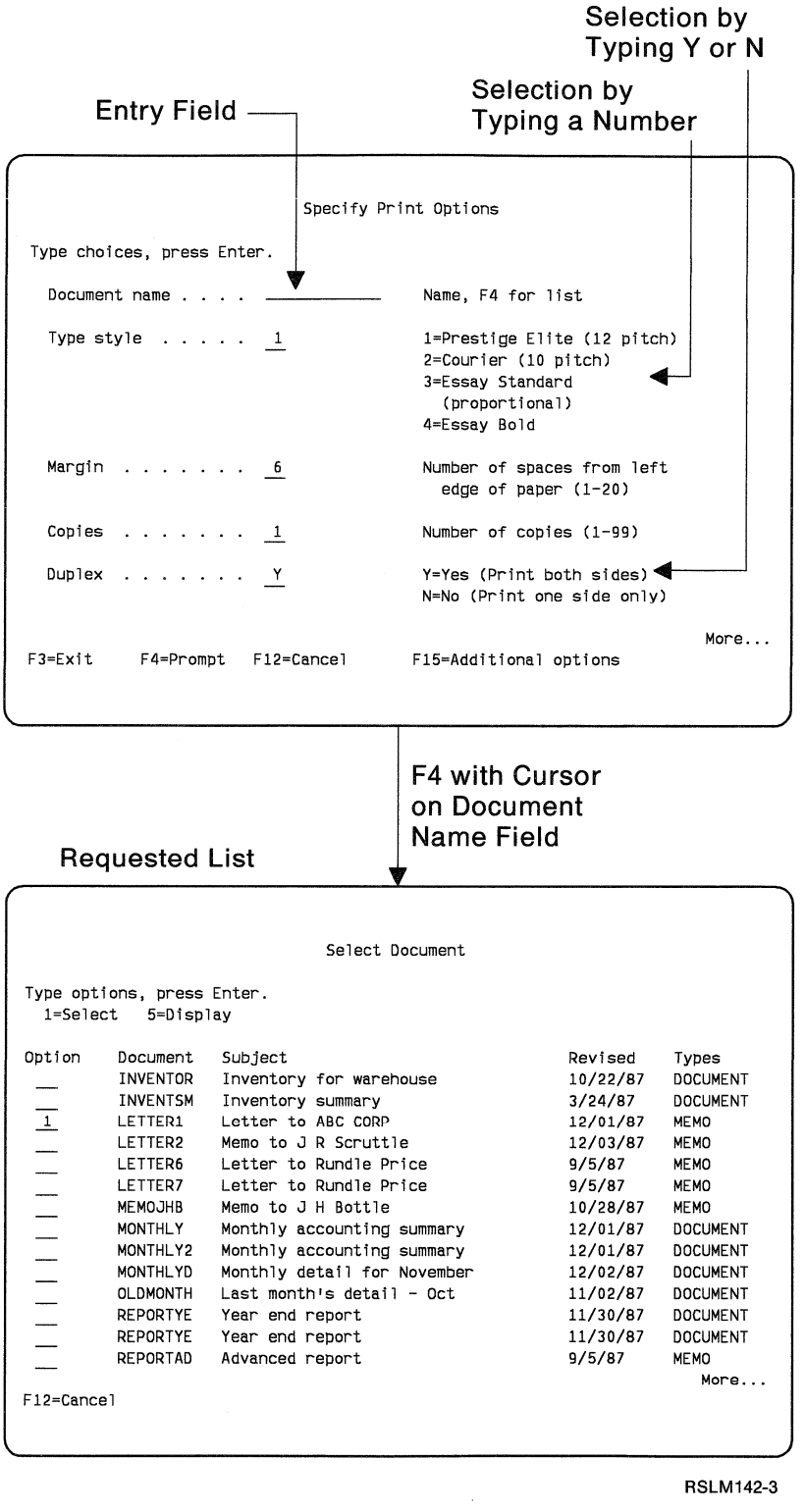


Figure 3-2. Entry Display and Associated List Display

List

When the user is prompted for the name of an object from either a menu or a command, a list of objects can be requested. One or more actions that can be performed on the displayed objects is included when the list is displayed.

When an object is requested, a list of related attributes, including type and other identifying information, is displayed. The list display provides a convenient means to perform actions (such as change, delete, and copy) directly on these objects without recalling and entering an object name for each action. Also, the user can specify actions to be performed on several different objects from this one screen.

An option field is next to each object name. The actions supported are displayed in the instruction area at the top of the screen. To perform an action on a listed object, the user enters the number corresponding to the action in the option field preceding the object, as shown in Figure 3-3. In the example, the user wants to display a memo and to print two documents.

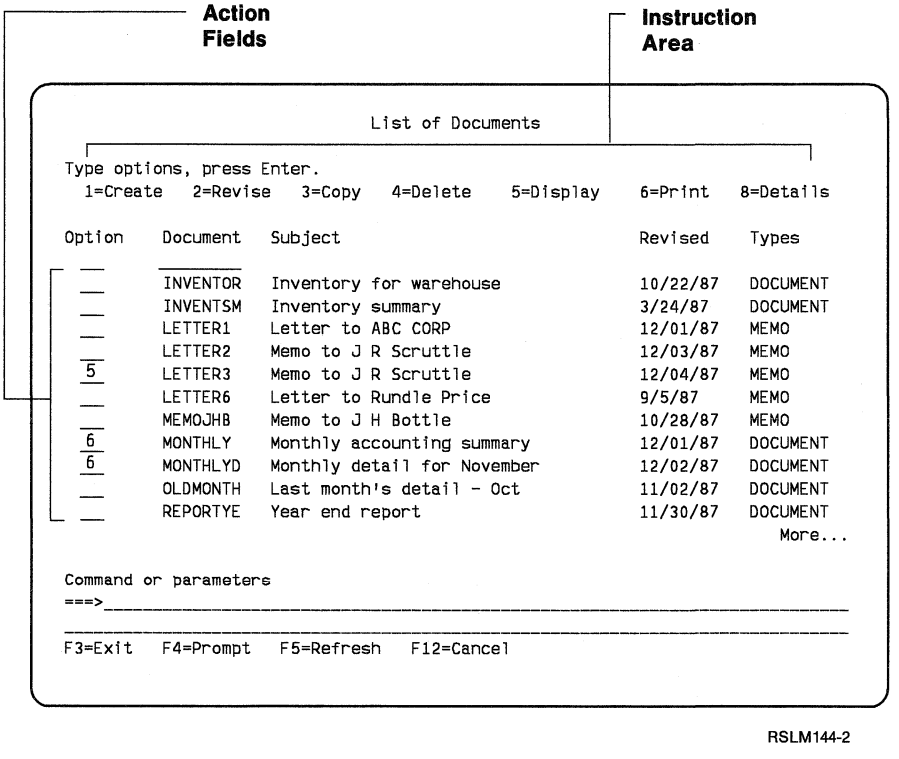


Figure 3-3. List Display

Information

An information display is like an entry or list display, except the user can only view the data in the fields, not change or enter the data.

Help

The help displays provide information about how to use a display or how to get started on a task. Through the use of AS/400 help, users can obtain several levels of help. Help can range from information about a whole screen to help about how the user can accomplish a task when the command is not known. Figure 3-4 on page 3-6 shows how the AS/400 help provides users with the information they need quickly, so they can complete their tasks.

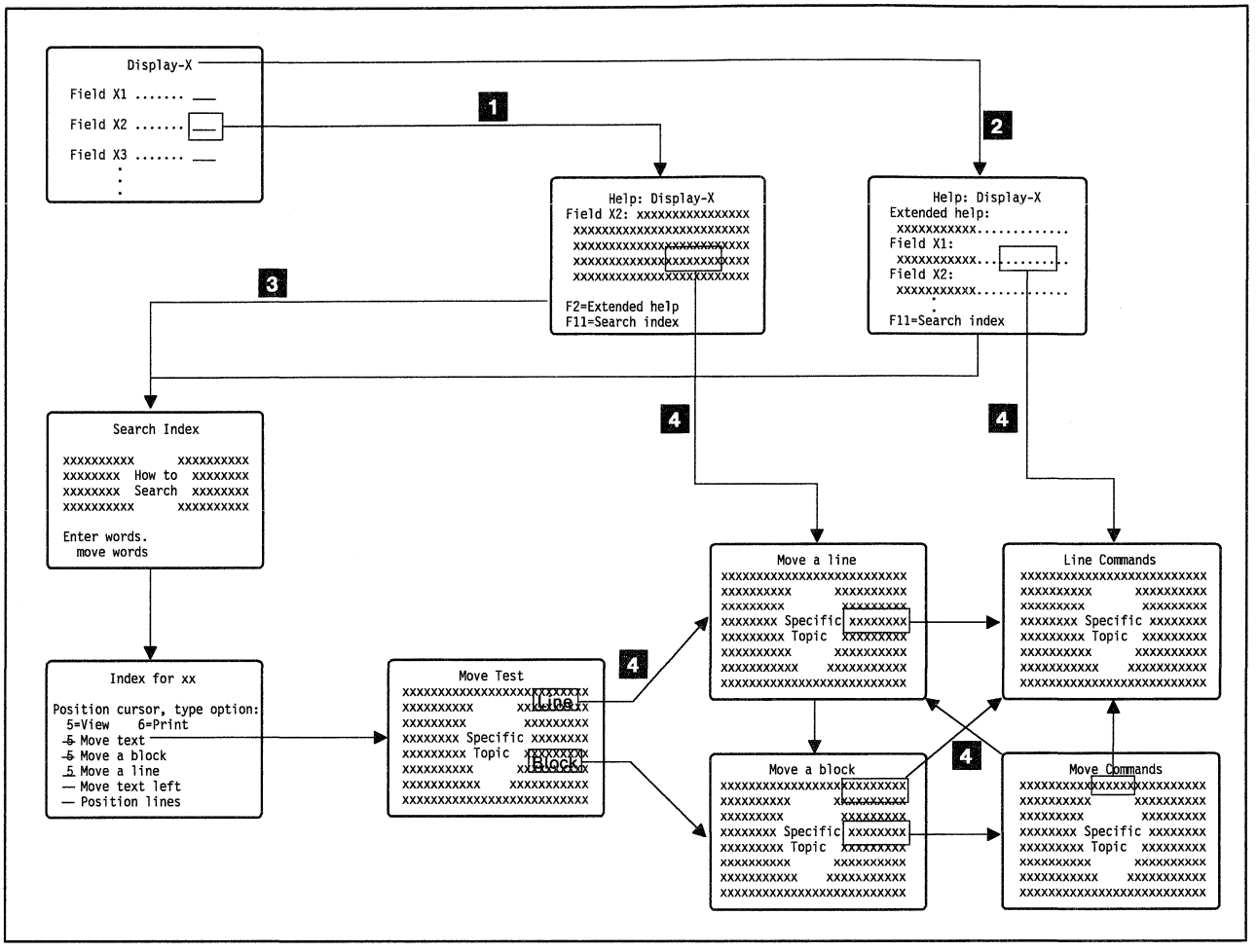


Figure 3-4. How a User Gets Help and Index Search

Contextual Help: The first level of help available is information about the prompts on a screen. By placing the cursor on a specific prompt or its entry field and pressing the Help key, users can find information relating to that specific field (see Figure 3-4 1). This ability to find specific information allows users to determine a solution to problems quickly, without having to look through a manual or read through many displays of online information.

Extended Help: The second kind of help display is more general than the contextual help. Extended help provides information about the task generally, about all the prompts on that screen shown at the same time, and about the function provided by the function keys. With extended help, users can page through all of the topics related to a screen (See Figure 3-4 2). This type of information is helpful to users who need general knowledge of the display.

Extended help can be accessed by pressing the Help key while the cursor is not on a specific field, or from the contextual help screen. By pressing a key assigned to the help function from one of these screens the user can also find extended help. Depending on the keyboard, this can either be a help key or a function key, which has been programmed as a help key, typically F1.

Index Search: The third type of help available to AS/400 users is the index search function. The previously described help was all related to a specific display, or to an individual area of a display. Index search allows the user to request more information that may or may not be related to the current display. The index search function can be used by pressing the index search function key (F11) from any help screen.

Index search allows users to enter, in their own words, any word or phrase for which they would like more information. After the user types a word or phrase, the system searches for this and similar entries in one of its indexes. The index that is searched depends on the system function being used at the time the search is requested. If the user is using an OS/400 function, the index of the help information relating to that function is searched for a match to the word or words the user entered. When the request is made, each of the search words entered by the user (except for words used as simple connectors such as *the* or *of*) is matched against tables of keywords and synonyms. When the search is complete, a list display of the index items containing the search words or synonyms that most closely matched is produced. From this list display, users can select which index item they would like to see (See Figure 3-4 on page 3-6 **3**).

A major advantage to this type of search is the ability for users to get help by entering a word or phrase in their own words. For example, a user could enter "How do I send a message to another user?". The index search would display a list of all of the help displays related to the sending messages to other users. The user could then select the topics to be displayed or printed. Index search topics do not have to be directly related to the function in which the user is currently operating.

Hypertext: Another way for users to get to online help is through the use of hypertext. This is a weblike organization of nonlinear information nodes linked together to allow users to move to other information that is provided online. A node is a unit of information that contains a single topic in a particular context. One node can be linked to one or more other nodes.

Hypertext can link information in a hierarchy, such as information that relates a task to subtasks. Hypertext can also link information together in clusters, such as between procedural and reference information or between two similar procedures. In either case, the link is identified to users with a high-intensity, underlined, link marker. This tells the user that there is more information at the other end of the link. See Figure 3-4 on page 3-6 **4** for a view of how hypertext can connect online information nodes together.

A link is defined by the programmer in one direction only, from node A to node B. However, users can return from node B to node A by pressing F12. Users can also press F6 to display a list of the titles of nodes previously displayed, then position the cursor next to a title on that list and press the Enter key to display the selected node again.

```
                        Work with Jobs - Help

The Work with Jobs display shows you the status of your   jobs.
     .
     .
     .
```

The above display shows how users see a hypertext link marker in a help node. The link marker is high intensity, underlined, and preceded by three blanks for an attribute byte.

Assistance Levels

The OS/400 program allows the user to choose the amount of assistance requested when interacting with the system. There are three assistance levels:

- Basic
- Intermediate
- Advanced

When the system is shipped, the assistance level for all users is defaulted to the basic assistance level set by the QASTLVL system value. A user can change the assistance level for different users either by:

- Specifying a different assistance level in the user profile [change the ASTLVL parameter on the Change User Profile (CHGUSRPRF) command]

  This assistance level is used unless a different assistance level is selected for a specific command.

- Specifying a different assistance level when entering a CL command that supports the ASTLVL parameter

A function key (F21 = Select assistance level) is also provided on the AS/400 system displays that support different assistance levels so that a user can go from one assistance level to another. The system remembers the last assistance level used for a display.

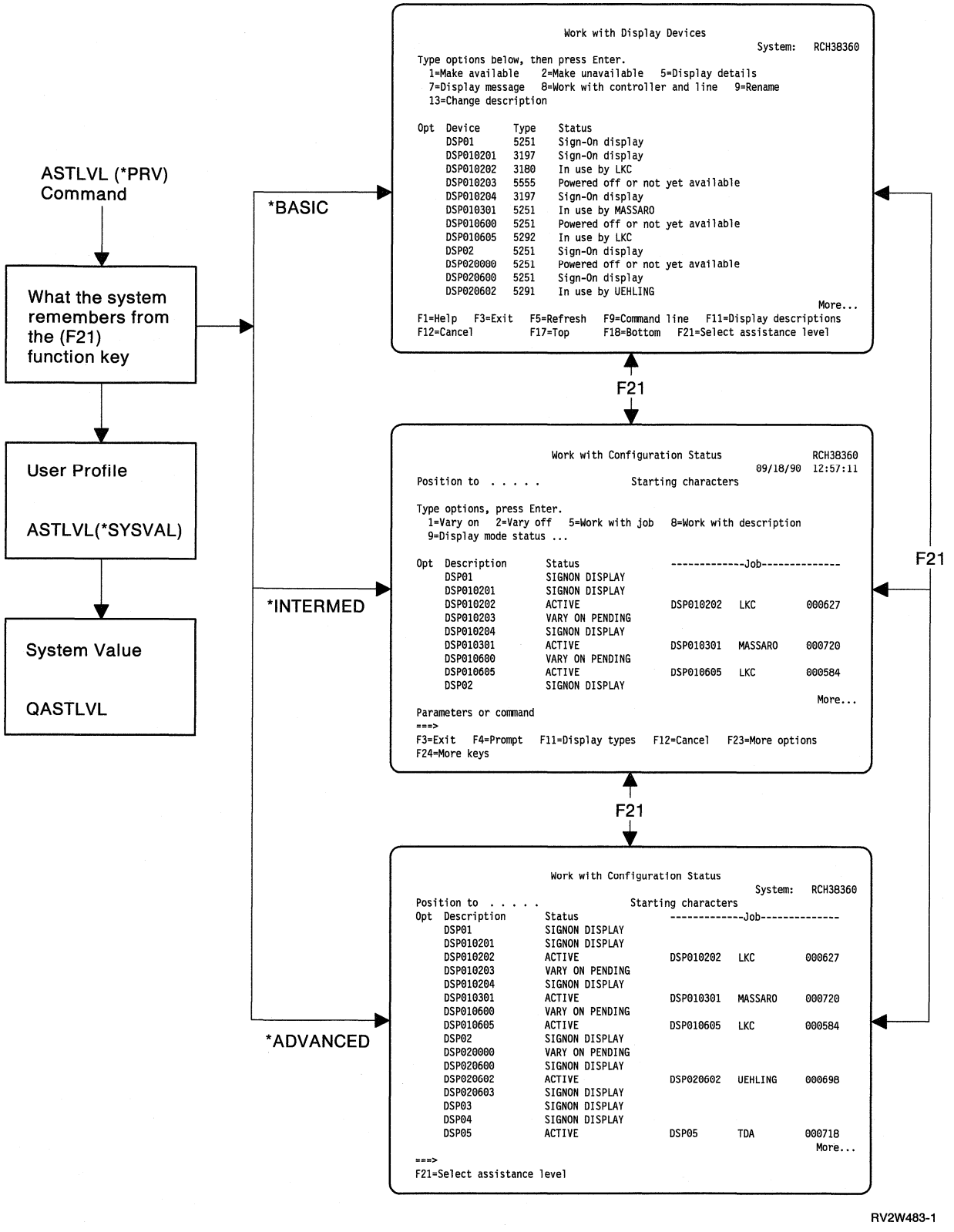


Figure 3-5. Assistance Level from System Value

Basic

Basic assistance level provides the most assistance. Basic assistance level supports the more common user and operator tasks through the Operational Assistant menu and does not use computer terminology. This method of accessing is easy even for the person who is not a data processing professional. It does not simplify the entire AS/400 system, but focuses on some common functions such as printing, working with batch jobs, handling messages, automatic housekeeping, and problem handling. Basic assistance, called Operational Assistant, does the following:

- Uses terminology that is readily understandable by users who are not data processing professionals. For example, the term "printer output" is used instead of "spooled output file."
- Limits the choices allowed for the commonly done tasks. For example, in working with printing, there are only a few things a typical user may want to change, such as the printer selected or the number of copies. With the *BASIC assistance level, the user only sees the most commonly changed options, and not the entire list of options that are available through the Change Spooled File Attribute (CHGSPLFA) command.
- Cleans up the system by doing some housekeeping automatically that the AS/400 system otherwise requires the user to do manually. Job logs, history logs, and system journals are good examples of objects that are automatically cleaned up.
- Groups commonly done tasks into a single set of menus and displays. The *BASIC assistance level can be set up as an attention program. If the application being used already has an attention program, Operational Assistant can be added as an option on the existing menu. The Operational Assistant main menu can be accessed by entering GO ASSIST from any command line.

The AS/400 Operational Assistant menu can be set up as the first menu users see when they sign on the system by specifying *ASSIST for the initial program (INLPGM) parameter on the Change User Profile (CHGUSRPRF) command or by specifying *ASSIST for the first menu field when adding a user to the system through the Work with User Enrollment option on the Manage Your System display.

Intermediate

Intermediate assistance level supports all system tasks and uses computer terminology. Complicated tasks can be done using this level of assistance.

Advanced

Advanced assistance level supports the same functions as the intermediate assistance level. However, the displays contain as much information as possible by not displaying the allowed function keys and options.

Commands

As was discussed earlier, commands can be used in place of the menus. To simplify the use of control language (CL) commands, all the commands use a consistent syntax. In addition, the operating system provides prompting support for all commands, default values for most command parameters, and syntax checking to ensure that a command is entered correctly before the function is performed.

Command Syntax

Each command has a command name and parameters. The IBM-supplied commands have a consistent naming convention. Generally, three letters from each word in the descriptive command name are used to form the abbreviated command name that is recognized by the system. A command name usually consists of a verb, or action, followed by a noun or phrase that identifies the receiver of the action. The command name identifies the function performed by the program when the command is run. For example, SNDMSG is the CL command that sends a message from one user to another.

Most CL commands have one or more parameters that specify the objects and values used to run the commands. When a command is entered, the user supplies the object names and the values for these parameters. The number of parameters specified depends on the command. Some commands have no parameters, and others have one or more.

The parameters used in CL commands are associated with keywords. The keyword, usually abbreviated in much the same way as commands, identifies the purpose of the parameter. For example, one of the parameter keywords for the Send Message command is TOUSR, which stands for To User. The value for this keyword parameter tells the Send Message command to whom to send the message.

Figure 3-6 shows the SNDMSG command, parameters, and values entered on the Command Entry display and identifies the parts of the command.

```
                              Command Entry                    ABC91802
                                                    Request level:   3
Previous commands and messages:
    > /* Programmer menu started */
    > strcpyscn
      Parameter SRCDEV required.
      Parameter OUTDEV required.
      Error found on STRCPYSCN command.
    > STRCPYSCN SRCDEV(*REQUESTER) OUTDEV(*NONE) OUTFILE(TEST/SCREENS)
      Format of output file SCREENS in library TEST not valid.
    > SNDMSG    MSG('Good morning')  TOUSR(BOB2)
        1       2       3              4


                                                                  Bottom
Type command, press Enter.
 ===> ____________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________

F3=Exit   F4=Prompt   F9=Retrieve   F10=Include detailed messages
F11=Display full      F12=Cancel     F13=User support   F16=System main menu
```

Figure 3-6. Example of the Command Entry Display

1 Command name. The SNDMSG command is constructed like most CL commands where the first part tells what action is taken; the second part tells what object receives the action. The Send Message command is used to send a message from one user to another user. Here, SND is the abbreviation used for *send*; MSG is used to abbreviate *message*, the thing that is sent.

At least one blank must separate the command name and parameters from the following parameter keyword or value.

2 Keyword. The MSG keyword states that the MSG (message) parameter is being specified.

3 Parameter value. The parameter value specified for the MSG parameter is 'Good morning'. No blanks can be used between the keyword (or parameter name) and the left parenthesis of the parameter value.

4 The last specified keyword, TOUSR, instructs the system to send the message (Good morning) to a user profile named BOB2. BOB2 is specified as a value for the TOUSR (To user) parameter.

Command Prompting

The operating system provides interactive command prompting for any command supplied with the system or created by the user. The user can type the command name only, then press a function key (F4) to see the prompt display for the command. The prompt display provides a list of required or often-used parameters. The prompt display can show either a list of possible values or keyword names. A function key can be used to switch between the two display options. For example, when creating a display description, only those parameters required for that device are presented.

Based on the values the user specifies for the parameters displayed, the user may be prompted for additional information. This *conditional prompting* limits the information requested to those parameters required to perform the function.

Figure 3-7 shows the prompts for the parameters on the SNDMSG command.

```
                                   Send Message (SNDMSG)  1

Type choices, press Enter.

Message text . . . . . . . . . .  2 _____________________________________
_____________________________________________________________________
_____________________________________________________________________
_____________________________________________________________________
_____________________________________________________________________
_____________________________________________________________________
To user profile . . .  2  _________    Name, *SYSOPR, *ALLACT...  3

                                                                  Bottom
F3=Exit    F4=Prompt    F5=Refresh    F10=Additional parameters    F12=Cancel
F13=How to use this display      F24=More keys  4
```

Figure 3-7. Example of Command Prompting

1 Command name

2 Entry fields (allow the user to specify parameter values)

3 Possible values for the To User (TOUSR) parameter

4 Function keys

If some command parameters are specified before the prompt display is requested, those values already specified are shown on the prompt display.

Some of the parameters used on commands have default values that appear in the entry fields and are used if no other value is specified. The default value can be explicitly entered if the user wishes.

Online help information is available for IBM-supplied commands by pressing the Help or F1 key. This help can include a list of values for a command parameter as a reminder to the user of the type of information or action required.

Messages

A message is a communication sent to one system user or program by another. A large set of predefined messages is supplied with the system. These messages can be used by application programmers to identify status or error conditions in the application program, for communications between programs within the system, or between the system and the users of the system. For example, the system response to the command to start the batch subsystem is in the form of a predefined message:

```
Start of subsystem QBATCH in library QGPL in progress.
```

In addition, a user can communicate with other users of the system through messages that are created at the same time they are sent. An example of a message a user might send was described with the SNDMSG command.

System users receive messages in response to commands entered at display stations. These messages are placed in message queues, which act as mail boxes on the system, which store messages for users. Each display station has a message queue named the same as the display station ID. Each user has a message queue named the same as the user ID. The system operator has a message queue named QSYSOPR.

These message queues are created automatically by the system.

- When the user profile has been created and the user signs onto the system for the first time, the system automatically creates a user message queue.
- When a display station is connected to the system for the first time, the system automatically creates a display station message queue.
- When the system is configured for the first time, the system automatically creates a system operator queue.

Each message has a severity code that indicates how severe the conditions are that caused the message to be sent. For example, an informational message has a severity code of 00. A system integrity message has a severity code of 90, and the most severe of all, the action message, has a severity code of 99. Help is available for all OS/400 messages.

Chapter 4. Data Management Support

The data management function of the operating system provides the operations that an application program uses to create, store, and access data on the AS/400 system, to communicate with external devices, and to ensure the integrity of the data according to the definitions of the application program.

The data can be on auxiliary storage (disk), external media (diskette or tape), on another system connected by a communications line, or can be sent to a printer, or sent to and received from a display device.

For any program to work with the database, documents, a device, or another system, the program must use files. Like other objects on the system, support for files is inherent in the operating system. There are several different types of files supported by the operating system. Each type has associated control language (CL) commands to create, work with, override, and delete it. The application program refers to the files for things other than just programs and data entered by the end user. For example, the printer and other devices are accessed through device files. This is one of the key differences between the AS/400 system and traditional data management. On traditional systems, files do not exist as a common interface for devices and programs refer directly to the devices.

On the AS/400 system, most files have a file description. The file description for database files describes the characteristics of the data associated with the file and how the data is organized into records and the records into fields. The file description for device files describes the characteristics and capabilities of the external devices such as display stations, printers, tape units, or diskette units. However, documents do not have a file description.

Data management uses the file description to process the data and to control the devices for processing the data. For efficient use of the printers and diskette devices, data management also provides the spooling function for input and output.

This chapter briefly discusses the topics listed in the following table. If you would like more information on the topic, and there is more information available in an IBM manual, the manual listed in the right column is a good first reference.

Topics	First Reference
Database files	See Chapter 5, "Integrated Database"
Device files (printer)	*Guide to Programming for Printing*, SC41-8194
Device files (display)	*Guide to Programming Application and Help Displays*, SC41-0011
Device files (tape and diskette)	*Guide to Programming for Tape and Diskette*, SC41-0012
Device files (ICF)	See "Accessing Remote Data (Record Level)" on page 7-1
Distributed Data Management (DDM) files	*Distributed Data Management Guide*, SC41-9600
DDM restrictions	*Distributed Data Management Guide*, SC41-9600
Save files	*Backup and Recovery Guide*, SC41-8079
Documents	*Office Services Concepts and Programmer's Guide*, SC41-9758

Topics	First Reference
File descriptions	No additional manual
File operations	*Data Management Guide*, SC41-9658
Deciding to use distributed file management or distributed relational database	No additional manual
Distributed file management	*Distributed Data Management Guide*, SC41-9600

File Types

The different types of files supported by the operating system can be classified into these general categories:

- Database
- Documents
- Diskette
- Display
- Intersystem Communications Function (ICF)
- Printer
- Spool
- Distributed Data Management (DDM)
- Save

Although there is a functional similarity among the file types, and the overall structure is the same, there are two major differences.

Visibility of the Device: With database, DDM, and save files, the program need not be aware that the data is coming from or going to a device. Data management automatically controls all device-related operations for those three types of files; therefore, the application program can use the data without knowing where the data is stored.

Data Storage: In a database, document, or save file, there is actual data associated with the file. A database or save file uniquely identifies the data. In a DDM or device file, no data is associated with the file.

Regardless of which type of file an application program uses, the overall processing of the file is the same.

Database Files

The data associated with a database file is always organized and formatted according to the data description that was specified when the file was created. The database file contains the data and the descriptions of the data. This description, which is an integral part of the file, is used to control the flow of data between the program and the database file. For example, if the description specifies a key field, the program can retrieve records in ascending or descending order by the value of the key field. If the description does not specify a key field, then read-by-key operations are not valid. The data is accessed in the order in which it was entered in the file (arrival sequence). (These functions are discussed in greater detail in Chapter 5, "Integrated Database.")

Device Files

Before an application program can read or write to the devices attached to the system, a device description that identifies the hardware capabilities of the device to the operating system must be created when the device is configured. The device descriptions describe externally attached devices to the system, such as display stations, printers, diskette units, tape units, and other systems. These are created automatically by the system if the system is set for automatic configuration. They can also be manually created.

The actual device is externally attached hardware, such as a display station, printer, tape unit, or diskette unit. The externally attached hardware can also be another system that is connected by a communication line to the system on which the program is running. If the device is another system, the program that is using the device file can be connected to another program rather than to an actual hardware unit.

If the device is connected to the system by a communications line, a line description and controller description must also be created for that device when it is configured. The three descriptions—line, controller, and device—represent the hardware capabilities of the communications device.

A device file specifies how a device can be used. By referring to a specific device file, the application program uses the device the way it is described. The information in the device file must be consistent with the hardware capabilities of the device. Generally, the device description defines the device to the system, while the device file formats output data from the program for presentation to the device, and formats input data from the device for presentation to the program.

A device file does not have actual data associated with it. The relationship between the data and a device file is temporary, and is established when the device file is opened. When the file is closed, the relationship between the data and the device file is ended.

For diskette and printer files, spooling can be specified. Data management supports both input and output spooling. When spooling is specified, data management stores the data in an object, called a *spooled file*, until the program or the device is available to process the data. The application program uses the spooled file as if it were reading from or writing to the actual device, and is not aware that the spooled file exists. Spooling can usually shorten the run-time of the job and increase the number of jobs that can be run sequentially because spooling allows the job to continue independently of the speed or availability of the device. Spooling is especially important in a multiple-user environment where the number of jobs running often exceeds the number of available output devices. With output spooling, the output can be redirected from one device to another.

Output spooling functions are performed by data management without requiring any instructions in the program that produces the output. When a printer or diskette file specifying spooling is opened, the spooled file containing the output of the program is automatically placed on the correct output queue.

A spooled file can be scheduled for selection from an output queue for printing when the printer file is opened, when the printer file is closed, or at the end of the job. An IBM-supplied program called a writer is started in the spooling subsystem and selects spooled files from the output queue and writes the records in the spooled output file to the printer.

Input spooling functions are performed by data management for database and diskette files. An IBM-supplied program, called a *reader*, is started in the spooling subsystem and reads the batch job streams from the device (database or diskette) and places the jobs on a job queue.

Intersystem Communications Function (ICF) Files

An Intersystem Communications Function (ICF) file provides a common file interface across several communications methods. The file allows an application on one system to establish a communications session on a remote system, start a process on that remote system, and send data to or receive data from the remote system.

Distributed Data Management (DDM) Files

DDM files describe database files stored on a remote system. A DDM file allows an application program to use data stored on another system in the network. Programs and commands using DDM files run as though the remote database files were local database files. The DDM file description associates the program on the local system with a remote database file. The DDM file description refers to the communications path and the remote database file, and links the remote system to the local system when the file is opened. For more information on how DDM files work, see "Distributed Data Access" on page 4-11.

Save Files

Save files are used to store data on disk in a format designed for backup and recovery operations or for sending data to or receiving data from another system. It is also possible, however, to use save files in an application program.

Because the data is specially formatted, the program cannot access the data with the same flexibility as other types of files.

It is not necessary to specify a file description for a save file. When the file is created, the operating system automatically associates a file description with the save file.

Documents

The data associated with a document is formatted and organized by the application. Data descriptions do not exist for these objects. The data in a document is accessed as a *stream* of bytes and is completely under application control. Documents are stored in folders and are managed by the folder management services function. See Chapter 6, "Office Enablers" for more information on documents.

File Descriptions

A file description is information that describes the characteristics of the associated file. It is an integral part of the file, and remains with the file until the file is deleted. Changes can be made after the file is created and the records are updated immediately.

The information required in a file description is determined by the file type because different file types have different capabilities. A file description is a declaration of the valid operations for a particular file, and determines how a program can use the devices or data. If the program attempts an operation that is not consistent with the

file description, the system does not allow the operation to continue, and will return an error code to the program.

A file description can specify:

- Device controls
- Linking information for communications devices
- Spooling attributes
- Data organization
- Data presentation formats
- Data storage formats

A file description is constructed based on information provided when the file is created. This information may be provided through:

- Parameters on CL commands used to create the files
- Data description specifications (DDS) for database, printer, display, and communications device files
- Interactive data description utility (IDDU) for database files only
- Screen design aid (SDA) for display files only

Because data is organized by fields, records, and files, file descriptions reflect this organization. A field is the smallest unit of data that is recognized and processed by the data management support of the system. A record is the arrangement of one or more related fields. A file is an arrangement of related records. A file may need to be described at three different levels, depending on the type of file and what purpose the file is to serve. These three levels are:

- Field-level descriptions
- Record-level descriptions
- File-level descriptions

Field-Level Descriptions

Field-level descriptions allow the user to define the detailed characteristics of the smallest unit of data that can be processed by data management. Field-level descriptions can be specified for database, display, printer, and communications device files. They cannot be specified for tape, diskette, save, or DDM files. Tape and diskette files are not processed on a field level; therefore, processing operations are at the record level only. Although field-level descriptions are not valid for the DDM file itself, the field-level descriptions of the database file on the target system are associated with the DDM file on the source system.

Field-level descriptions for a database file can define the length of each field, and the type of data (character, numeric, or binary). A field-level description for a database file specifies how the data is stored in the file, the format of the data when it is presented to a program because of a read operation, and the format of the data when it is presented by a program during a write operation. The presentation format need not be the same as the storage format.

Data can be described either externally or within the program source code. For externally-described data, all programs that use the data can refer to a single description which is automatically incorporated into the program code when the program is created. For program-described data, each time the file is changed, every program which refers to the file must change the lines of source code which refer to the file.

Data management uses the field-level descriptions to move data in and out of a database file, performing mapping whenever necessary, and ensuring that data is valid according to the description.

A field-level description for a display file specifies how the data is to be presented at the display device, and how the program receives the data on input and presents the data on output. The field-level description specifies such things as whether a field is input- or output-capable, or both, the type of data that is valid for the field, highlighting characteristics, and the location of the fields on the display screen.

When record formats for display files are specified at the field level, detailed presentation characteristics can be defined for each field. Individual fields can be designated as input or output, locations on the screen can be controlled, and individual highlighting characteristics can be defined.

Communications device files can be defined at the field level. However, data management processes the data only at the record level. Data management allows the communications device files to be defined at the field level to standardize the fields within a file. The standardized field-level descriptions can be used by an application program that will work with the file.

If the file is not described at the field level, data management cannot refer to specific fields and cannot check if the data type is valid; data management can process data only at the record level.

Record-Level Descriptions

A record is the unit of transfer between data management and the application program. On a read operation, the program receives a record from the file. On a write operation, the program sends a record to the file. Therefore, a record-level description is required for all file types to specify how much data to transfer on a read or write operation.

A record-level description, called a *record format* provides the arrangement of the fields in a file. If field-level descriptions are used, the record format is specified by one or more field-level descriptions. The record format uses the field names, and arranges the fields within the record. Specifying record formats with field-level descriptions allows much more flexibility for the program to manipulate the data. For example, with database files, it is possible to specify a record format that describes how data is to be stored in the file. It is also possible to define a different format for the same data that describes how a program is to read and write the data. Because the relationship between the storage format and the presentation format are defined, data management can map the data between the two.

If field-level descriptions are not used, the record format is specified by the record length (number of characters). The record length for save files is determined by the system. When the record format is specified as a length, the program must process the entire record as a unit. The program cannot manipulate one part of the record one way and another part of the record a different way. Mapping fields between the presentation format and the storage format is also not possible if the record is specified as a string of characters that specify only the length of the record.

File-Level Descriptions

File-level descriptions apply to the file as a whole. The specifications at this level vary depending on the type of file being described. For example, a database file can specify what record formats are valid for the file, how data is to be organized (sequentially or by key), and if by key, what fields are designated as the key fields. A display file can specify the record formats for the file, the devices the file is usable with, the graphic character set, and other identifying information. A tape file can specify such things as block size, label information, and recording density. A printer file can specify such things as the size of the page, font identifier (for printers that support fonts), characters per inch, lines per inch, and other printing characteristics.

Methods of Describing Files

There are several methods available on the system for specifying file descriptions. The following summary briefly describes each of these methods.

CL Commands: A specific control language (CL) command supports each type of file. On each of those commands are parameters that define file-level descriptions.

For example, to create a tape file, use the Create Tape File (CRTTAPF) command. On this command, there are parameters that specify such file-level descriptions as block length, recording density, label information, and record length.

For tape, diskette, save, and DDM files, using the CL commands is the only method available for specifying file descriptions.

For database, display, printer, and communications device files the CL commands work with data description specifications (DDS).

DDS: DDS is an OS/400 language that can be used to define file-level, record-level, and field-level descriptions for a database, printer, display, or communications device. DDS source statements that are entered into a source file are used to create the related file. The name of the source file is one of the parameters on the CL command that creates the file. The parameters on the CL command and the DDS source statements are compiled to create the file description for the file.

DDS is functionally rich and provides specification statements and keywords to enter all record- and field-level characteristics for a file description. However, use only those specification statements that are valid for the type of file being created. If the statements are not valid for the type of file, the system sends a message. The system creates the new file only if there are no errors detected that would make the file not valid.

The DDS source statements can be entered into the source file by several methods. One method is to use the source entry utility (SEU) that is part of the Application Development Tools licensed program. SEU provides the online specification sheet and checks the syntax of DDS source statements as they are entered to identify errors before the statements are compiled.

IDDU: IDDU is an interactive function of the operating system that can be used to create file descriptions for database files only. The specifications for IDDU are referred to as definitions because they exist in the data dictionary. When a file is created, the definitions in the data dictionary are used as input to the file description. The file description is stored as part of the file, and is separate from the definitions in the data dictionary.

The IDDU specifications are similar to the DDS specifications to a great extent, although IDDU does not support all the processing capability that is available through DDS. Some advantages of using IDDU instead of DDS are:

- IDDU is interactive. Instead of entering DDS source statements to specify a file description, the user responds to interactive screens, which might be easier for some users.
- Definitions kept in the data dictionary can be used over and over in different combinations. In contrast, a DDS source file represents a single file description. However, more than one file can be created with the same DDS source file. The files have the same file description unless the DDS source file is explicitly changed. With IDDU, definitions in the data dictionary can be used in a different way to create a new file description.

To create a file using the existing specifications stored in an IDDU data dictionary, IDDU must be used again. IDDU specifications cannot be used to create a file with CL commands.

SDA: SDA is an interactive function of the Application Development Tools licensed program to design screens for user application programs. On the SDA displays, each field for the application program can be dynamically arranged and the characteristics described interactively. SDA converts the visual display into DDS source statements, which are compiled to create the display file.

Methods for Changing File Descriptions

After a file is created, it can be changed. If the file-level description that was specified on the CL command used to create the file needs changing, there are corresponding CL commands for working with each file type.

If the file-level, record-level, or field-level specifications contained in DDS need changing, the DDS source file must be changed first. The DDS statements can be changed using the same method used for entering the original source. After the DDS is updated, the appropriate CL create command must be used again to create the file with the updated DDS. If both the CL command and the DDS must be changed, the new values must be specified on the CL create command used to create the new file.

Whether programs that used the changed file will use the new file description the next time the program is run depends on what was changed in the file. If the file-level description was changed with a CL command, any program that uses the file will automatically use those new descriptions. If the DDS descriptions were changed and the program uses the file as a program-described file, then the system will use the new file description, but the program view of it may not be correct anymore, which could give incorrect results. If the DDS descriptions were changed and the program uses the file as an externally described file, then the record-level and field-level descriptions used when the program was compiled may not match the changed file. The system may detect such a mismatch when the program opens the file and return an error condition to the program.

The system also supports a way to temporarily change the file-level descriptions when a file is opened to affect only the program opening the file. Temporary changes can provide greater flexibility to the application. Temporary changes are made when the program first opens the file.

Temporary changes can be made in one of two ways:

- By information that is specified within the program, and passed as parameters on the open operation.
- By using CL override commands to set up the run-time environment for the program.

The override commands can be used regardless of the programming language that is used. CL override commands are provided for each file type. By including override commands with the application program, temporary changes can be made to the file description in a file that the program uses.

Both methods can be used together. Some parameters can be changed by information contained in the high-level language statements, while others can be changed by using a CL override command. The same parameter may be changed from both places. The operating system uses the following order when making temporary changes to a file:

1. The file description provides a base of information.
2. Changed information received from the application program during the open operation is applied first to the base information.
3. Changed information found in the override command is applied last. If the same information is changed from both places, the override has precedence.

Temporary changes are seen only by the application program that processes the change. The file, as seen by another application, remains unchanged. In fact, two programs can use the same file at the same time, and each can change it temporarily according to its needs. Neither program is aware the other program made a temporary change.

File Operations

Data management supports many operations that high-level language programs can use to process data. These operations are divided into those files that contain records and files that contain stream data.

Files That Contain Records

Files that contain records use these operations:

- File Preparation

OPEN	Attaches a file to a program, starting I/O operations. A file can be opened for any combination of read, write, update, or delete operations.
ACQUIRE	Attaches a device or establishes a communications session for an open file in preparation for I/O operations.

- Input/Output

READ	Transfers a record from the file to the program. The data is made available to the program after the read is successfully completed.
WRITE	Transfers a record from the program to a file.
WRITE-READ	Combines the WRITE and READ operations as one operation.

UPDATE — Updates a record with changed data. The record must be successfully read before the update operation.

DELETE — Deletes a record in a file. The record must be successfully read before the delete operation.

- Commitment Control

COMMIT — Guarantees a group of changes are made as a complete transaction across multiple records or multiple files.

DECOMMIT — Rolls back a group of changes to the last commitment boundary.

- Completion

FEOD — Positions the file at the last volume or at the end of data. For those programs processing files for output, the last buffer of data is written. For those programs processing files for input, an end-of-file condition is forced for the next input operation.

RELEASE — Detaches a device or a communications session from an open file. I/O operations can no longer be performed for this device or session.

CLOSE — Detaches a file from a program, ending I/O operations. Any remaining data in the output buffer that was not written is written before the completion of the close.

The operations listed above have certain restrictions based on file type and language support. For example, a program may not write to a file that is opened for read only. Similarly, a read-by-key operation cannot be issued for a communications device file. Since file overrides can occur during processing, an operation may not be allowed for the type of file that is ultimately being processed.

Files That Contain Stream Data

Files that contain stream data use these operations:

- File Preparation

QHFOPNSF — Attaches a file to a program, starting I/O operations. A file can be opened for read, write, or update operations. The file can also be optionally created.

- Input/Output

QHFRDSF — Transfers a stream of bytes from the file to the program. The data is made available to the program after the read is successfully completed.

QHFWRTSF — Transfers a stream of bytes from the program to a file.

QHFLULSF — Locks or unlocks a range of bytes in a file.

QHFCHGFP — Changes the file pointer with a file that is open.

QHFFRCSF — Forces internal buffers to nonvolatile storage.

- Completion

QHFCLOSF — Detaches a file from a program, ending I/O operations. Any remaining data in the output buffer that was not written is written before the completion of the close.

Distributed Data Access

In an environment with more than one computer system, the AS/400 distributed data access functions are supported for developing and running application programs that require access to data that is stored on another (remote) system. In general, an application program can be designed to access remote data in either of two ways:

- Copy the data from the remote system to the local AS/400 system
- Directly access the data on the remote system

Both techniques work and are appropriate for different types of application programs and system environments.

Copy Data from Remote System: When the application program runs, it accesses the local copy of the remote data. This technique is very useful when the amount of data that has to be copied is small, the data is not updated very often, and the application program does not need to update the data on the remote system. The major advantage is that after the local copy of the data has been made, the application program performs faster. However, this technique is less practical if:

- Large amounts of data must be copied
- One application program needs to access the data and it only uses the data once
- The data is updated very frequently on the remote system and having the most current data is important to the design of the application program
- The application program updates the data and the changes must be reflected in the data at the remote system

Access Data on Remote System: This is very useful when the amount of data to be accessed is small compared to the total size of the remote data. The major advantage is that the application program has access to the actual, current data; any updates made by the application are made directly to the remote system. However, if the amount of data to be accessed or updated is very large, the time it takes to run the application program is affected to some degree.

Using Distributed Data Management or Distributed Relational Database

The AS/400 system provides distributed data management (DDM) for accessing file data stored on remote systems and provides distributed relational database for accessing data stored in remote relational databases. For more information on how DDM works, see "Distributed Data Management" on page 4-12. For more information on how distributed relational database works, see "Distributed Relational Database" on page 5-14. To select whether an application program should use DDM or distributed relational database, consider the following possible situations:

- If the data resides in a file on the remote system, the application program should use the DDM functions. Both the copy and direct access techniques are supported by DDM.
- If the data resides in a relational database on the remote system, the application program should use the distributed relational database functions. The direct access technique is supported by distributed relational database.
- If the data resides on a remote AS/400 system, either DDM or distributed relational database can be used. The choice should be based on whether the application program should use the Structured Query Language (SQL) as the interface for accessing the data in a database or the normal programming language file statements for accessing the data in flat files.

Distributed Data Management

Distributed data management (DDM) allows an application program to access data that is stored in a file on a remote system. The AS/400 system uses the Distributed Data Management (DDM) Architecture as the means of providing the DDM functions.

With DDM, an application program can access file data stored on remote System/36, System/38, AS/400, or CICS systems. Application programs running on System/36, System/38, and DOS systems can access database files on the AS/400 system by using the DDM architecture implemented on each system.

The DDM functions allow an AS/400 application program to:

- Copy a file from the remote system to the AS/400 system
- Copy a file from the AS/400 system to the remote system
- Manage (create, delete, clear, rename) files on a remote system
- Access (read, update, delete, insert) individual records in the remote file
- Share access to a file's data with other application programs
- Obtain a description (attributes) of the remote file

When using DDM, the system on which the application program runs is called the *source system* and the system on which the remote file resides is the *target system*. When an application program that resides on the source system needs to use data stored on the target system, a DDM file is required to identify the remote location name, the name of the file defined in the application program on the source system, and the name of the file associated with the data on the target system. The DDM file on the source system allows the application program to access the target system first, and then to read and write to a database file on the target system. Although the program is working with the DDM file, it appears to the program as if it were working directly with the database file.

The data is sent from the target system to the source system on a read operation, where it is made available to the program. On write operations, the data received from the program is sent from the source system to the target system into the database file being used by the program.

The DDM file is only a temporary connection between the application program and the target system. When the DDM file is closed, the connection is ended.
Figure 4-1 shows an example of DDM files working with a source and target system.

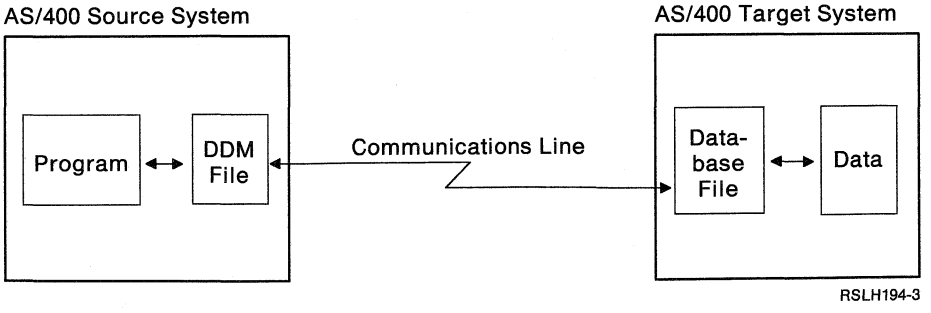


Figure 4-1. Distributed File Management Source and Target Systems

Using DDM Functions

The first step in using DDM functions is to define remote files to the local AS/400 system. This is accomplished by creating a separate DDM file on the local AS/400 system for each remote file to be accessed. Each DDM file contains:

- An AS/400 system file name that can be used by the application program as the name of the remote file.
- Information about the remote file including its actual name on the remote system and how the file should be accessed; for example, sequential, random, or by key.
- Information about where the remote file is located so that a communications path can be established to the remote system.

Note: A DDM file is only necessary on an AS/400 source system when an application program wants to access a file stored on a remote target system. A DDM file is not required on an AS/400 target system for application programs on other source systems to have access to local AS/400 database files.

The development of an application program that uses DDM functions is the same as for an application program that accesses local AS/400 database files. However, the application program uses the name of the DDM file instead of a local database file when it needs to access the remote file data. This provides application programmers with a high degree of file location transparency and minimizes the training needed by programmers to write programs that access remote file data.

Using DDM to Make Local Copies of Remote File Data

For application programs that need to use the copy technique for accessing remote data, DDM can be used to make the copy of the remote file data on the local AS/400 system. This is accomplished with a simple two-step process:

1. Create a DDM file with the Create DDM File (CRTDDMF) command for the remote file.
2. Issue a Copy File (CPYF) command with the DDM file specified for the from-file (FROMFILE) parameter and a local AS/400 database file specified for the to-file (TOFILE) parameter.

At the successful completion of this process, a copy of the remote file data exists on the local AS/400 system and application programs can be run that access the local copy of the remote data.

Using DDM to Make Direct Access to Remote File Data

For application programs that need to directly access remote file data when running, DDM supports this capability through the normal AS/400 database file interfaces. This is accomplished by a simple two-step process:

1. Create a DDM file with the Create DDM File (CRTDDMF) command for the remote file.
2. Code the application program as if all of the file data is stored locally on the AS/400 system. However, for remote file data, specify the name of the DDM file instead of an AS/400 database file. (Alternatively, the Override Database File (OVRDBF) command could be used to override the application program's use of an AS/400 database file to a DDM file.)

When the application program runs, all references made to the DDM file are automatically transformed by DDM into remote access requests to the remote file.

Considerations for Using DDM

Before deciding to use DDM, you should consider the following:

Performance: The performance effect of DDM is usually noticeable and, therefore, should be given careful consideration. Access to remote file data through communications is generally slower than to local AS/400 storage. The need for application speed must be balanced against the need for access to the most current, most accurate data.

Copy Management: The copy technique to distributed data access can cause some management problems. The size of the remote file verses the amount of data in the file that an application program will actually access should be evaluated. If the amount of data to be accessed is small and the file is large, the direct access technique may be better. Managing the local copies so that the local copies can be deleted when they are no longer needed and ensuring that new copies are made when the local copies are sufficiently out-of-date is an additional administrative task.

Security: File data security is maintained on the system storing the data. The AS/400 system user has to have proper access authority to the remote system that contains the remote file. This may mean that the application program user must have a valid user identifier (user profile) defined on the remote system.

DDM Restrictions: Some restrictions on the DDM functions depend on which remote system the remote file is stored.

Chapter 5. Integrated Database

Chapter 4 described the OS/400 data management function, the file types supported on the AS/400 system, and the file descriptions that support those file types. This chapter discusses one of the file types, the AS/400 database file. It discusses in particular the topics listed in the following table. If you would like more information on the topic, and there is more information available in an IBM manual, the manual listed in the right column is a good first reference.

Topics	First Reference
Attributes and advantages	No additional manual
Database file types	*Database Guide*, SC41-9659
Methods of creating a database file description	• *Data Description Specifications Reference*, SC41-9620 • *Utilities: Interactive Data Definition Utility User's Guide*, SC41-9657 • *Systems Application Architecture* Structured Query Language/400 Programmer's Guide*, SC41-9609
Methods of processing data	• *Query/400 User's Guide*, SC41-9614 • *Programming: Query Management/400 Programmer's Guide*, SC41-8192 • *Programming: Cross System Product/Application Execution User's Guide and Reference*, SH23-0516 • *Systems Application Architecture* Structured Query Language/400 Programmer's Guide*, SC41-9609 • Appropriate high-level language manual
Distributed relational database	*Distributed Relational Database Guide*, SC41-0025

Attributes and Advantages

A *database* is simply a place to store data. The files on the system can be viewed as a database.

Most programs on systems without a database are tied very closely to the way data is physically stored on the system. That is, if you have defined a file as containing fields *A*, *B*, and *C*, each program that uses that file is coded with the knowledge of that record layout. If you decide later to add field *D*, all your programs that use that file must be changed and recompiled.

A database management system, on the other hand, can hide the physical record format of a file. In the previous example, the database management system could present data to your program as if the record layout were still *A*, *B*, *C* (even though the new physical record format was *A*, *B*, *C*, *D*). Your programs would not have to change or be recompiled (unless, of course, the programs needed to use the new field).

What sets the AS/400 system apart from most other traditional systems is that the database is integrated into the operating system and licensed internal code of the AS/400 system. There is no requirement to install or maintain a separate database management product. The AS/400 integrated database has advantages in productivity and automatic processing over the functions of a traditional database system. Figure 5-1 on page 5-2 compares the integrated database and the data manage-

ment structure of the AS/400 system to the segmented structure of the traditional system.

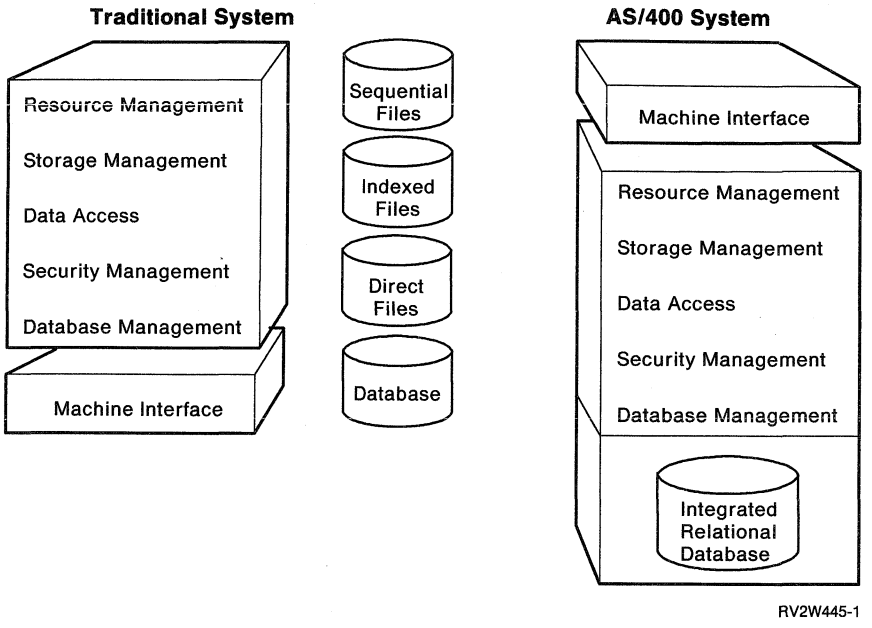


Figure 5-1. Comparison of Database Management Systems

Any program that wants to store or use data on the AS/400 system uses the integrated database. The data management function of the operating system supports the integrated database to achieve a level of productivity, ease of use, and consistency that many other systems with separate database management programs cannot achieve.

The consistency of the data management function enhances many parts of program design, from the method for describing data to the actual file operations without requiring individual programming statements for each task.

For example, if you need to read data for customer number 100 and that customer has information in both a master file record and an address file record, a program would normally do two read operations in the program (one read for each file). However, using a database, the program could do one read operation for customer number 100, and the data management function could get the data from the two files and present it to the program as if it were coming from a single file. The program does not need to know how many files, nor from which files, the data is being read.

Data and Program Independence

The AS/400 system can separate the program's view of the data from the way the data is physically stored on the system. The ability to logically view data independently of its physical structure in storage allows the same data to be used for many applications by defining only the fields appropriate to the program. Fields can be added to a file, removed from a file, or the attributes changed in a file without affecting programs that do not use those fields.

Externally Described Data

Historically, how records in a file should appear was not recorded anywhere outside of the programs that used those files. That is, what fields made up the record, how long each field was, and what type of data was in each field was specified only in the high-level language statements that were used to define the files within the program. Because there was no easy method to standardize a description of a file, all programs had to define the data.

On the AS/400 system there is a way to standardize descriptions for database, display, printer, communications device files, and indirectly for DDM files, by using a file description that is associated with any file of these types.

The term *externally described data* refers to the detailed description of the file that exists outside the program, and is associated with the file itself. The data format is always available in the file description.

Because the description is centralized, the file description can be incorporated in the using program. Most high-level programming languages supported by the system provide this capability.

The fields and records do not need to be defined in the program. The language compiler or interpreter extracts the information from the file description and incorporates it into the program just as if the files were specified in the source program. When the file description changes, for example, a new field is added, the programs that refer to that field should be recompiled to include the new field. Otherwise, logical files can be used to present a *view* of the new format without requiring existing programs to be recompiled. No change needs to be made to the program source code. The way the file description are incorporated into the program varies by the programming language.

There are several advantages to using externally described data:

- *Increased programmer productivity.* The language automatically describes the record formats without additional coding.

  The records and fields need to be described only when the file is created, and can then be referred to from any program.

- *Ease of file and program maintenance.* When fields are added, deleted, or changed, they can be specified in the file description instead of maintaining the record format in each program that uses the file.

- *Increased data integrity.* Because the fields and records are described in one place, the system programs and the application programs using the file will use the same data.

- *Automatic level checking.* The operating system automatically determines, when the program is run, if the file description was changed since the program was last compiled.

Program-Described Data

Diskette, tape, save, and DDM file types do not allow a detailed description to be used because field-level descriptions are not supported.

For those file types that do support field-level descriptions, the program is not required to use the external descriptions. If externally described data is not allowed because of the file type or if you choose not to, then the program must declare variables in the source program that define to the compiler or interpreter what the program thinks the data looks like. Such declarations are referred to as *program-*

described data. That is, the description of the data related to the file is represented by the variable declarations coded in the program.

The data definitions or declare statements specified within the source program are called *program-described data.* The AS/400 system allows the data definitions declared within the program to override any that exist as externally described data at the file level.

For tape, diskette, and save files, program-described data is the only option supported.

When program-described data is used, the application program and the system programs may not have the same definitions of the data. The system programs cannot read the data defined within the application program; the data defined with the application program is available only to that program.

- If the file does not have any field-level descriptions associated with it, the system must operate at the record level. The only concern, in this case, is that the record length the program is using is the same as what the system is using. It need not be, but the system always operates with the record length that is described in the file description. If different from what the program is using, the system truncates or pads as appropriate. The exception to this is save files. For save files, the application program must operate with the same record length defined by the system or a severe error condition will result when the file is first opened.
- If the file has field-level descriptions, but the program has elected not to use them, the system still uses the field-level descriptions. The system still expects the program to present data according to the file description and, conversely, will provide data to the program according to the file description. If the program description is different from the file description, an error could result.

Selection and Arrangement of Data

The AS/400 database file descriptions can improve productivity by doing some of the operations normally coded in a program.

- The data management function of the operating system can select or omit records based on selection values specified in the file description. The program only sees the records selected. For example, in a payroll database, the salary for each employee is included. The data management function could be used to select employees based on a salary range. The programmer's coding effort is reduced due to the record selection based directly on DDS keywords. The program runs more quickly because it now has a subset of the original data to process.
- The data management function presents data in the sequence and groups specified in the file description, without the need for the program to sort or duplicate the data. Using the same example from the previous discussion, a data management function could be used to organize the employee information by department, by job category, or by date of employment. Because the data management function presents this view of the data to the program based on the specifications in the file description, the programming code to do the sort is not required in the program.
- The data management function can be used to determine, among other things, the standard deviation, variance, square root, and sum of a series of values in the database.

Data Joined from More Than One File

The AS/400 database can combine two or more files defined and stored independently as if they were one file. The data management function of the operating system does all the necessary read operations to get the data from the separate files and presents the data to the program to make it appear as if it were coming from one file.

For example, assume there is a customer payment master file and an accounts receivable payments file on your system. The master file has a record for each customer. The master file record format includes the customer number, customer name, and customer address fields. The payment file contains a record for each payment received. Each payment record includes the customer number, payment due date, invoice number, and payment amount fields. There is a relationship between the master file data and the payment file data. That is, the record in the master file with customer number 15 may have one or more corresponding payment records in the payment file. Using a traditional file system to get the master record information of the customer along with the payment record, the application program would have to:

1. Read the record in the master file.
2. Read the corresponding record or records in the payment file and compare the dates.

Once the relationship between the two files is defined, the data management function presents the data to the program as if all the data resided in one file. That is, your program would do one read to retrieve both the customer data and the payment records from the two files.

Single Source of Data

Traditional systems, which do not have an integrated database, usually have many ways of storing and retrieving data. The programmer frequently must use one language to access file data (a file interface), and another language to access data (a database interface). In addition, the data stored by the file interface often cannot be directly accessed by the database management interface program, and the data stored by the database management system cannot be directly accessed by the file system.

The AS/400 database serves as the single source of data on the system, and the AS/400 data management function of the operating system serve as the single manager for that data. Because there is only one way to store and retrieve data on the system, programs can access all the data on the system (provided, of course, the user running the program has the proper authority to use the data).

Several ways to access and manage data are supported by the operating system:

- Traditional program-described files and high-level language operations
- Structured Query Language (SQL/400*)
- OS/400 IDDU
- OS/400 control language
- Data description specifications (DDS)
- System/36 and System/38 environments
  - System menus
  - Disk data management functions

In addition, licensed programs such as AS/400 Query, OfficeVision/400, PC Support/400, and the AS/400 Application Development Tools use data stored in the database in the same way as the operating system. The data management function

understands all operations necessary for these interfaces and coordinates their interaction.

Because a file interface is available for the database, many programs originally written for traditional file systems, can be used with little or no change. These traditional programs, when run on the AS/400 system, immediately use the integrated database. Over time, traditional file programs can be changed to take full advantage of the capabilities of the AS/400 data management function. Figure 5-2 shows the various database tools interfacing to the AS/400 database.

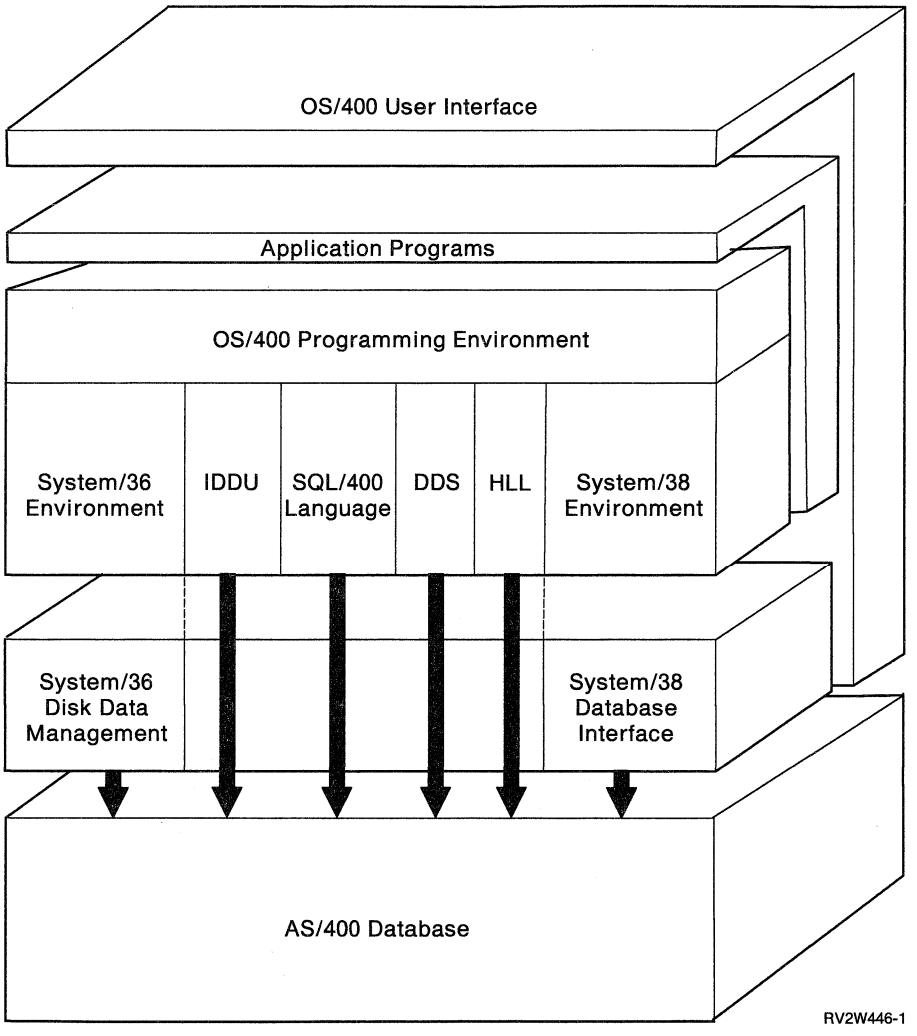


Figure 5-2. Operating System Interfaces to AS/400 Database

Shared Data

The AS/400 data management functions of the operating system allow many users and programs to share the data in the database files at the same time. The data management function maintains the integrity of the data regardless of the number of programs sharing the data or how those programs use the data.

Record locks ensure data integrity and the ability to safely share data in the database. The operating system automatically protects a record from being updated by more than one user at the same time, and can also ensure that a transaction is not seen by other users until the transaction is complete. After the transaction is com-

plete, the system ensures that the changes are reflected in the database; therefore, programs use the latest information. New application programs can be added to the system and can immediately share the data, without affecting existing programs.

Because the database allows greater data sharing, the amount of data that must be defined and stored is reduced. Traditionally, each application program has its own files, usually defined within the program. For example, both the inventory application and the accounts payable application might each have a version of the inventory file. The inventory application file might contain an item number field, an item description field, and a warehouse location field. The accounts payable file might have an item number field, an item description field, and a supplier number field. One file containing the fields necessary for both files could be created, and the file description could specifically identify the fields needed by the accounts payable and inventory application programs.

The database on the AS/400 system can store data from one program, then use that data in another, completely different application. For example, AS/400 Query can select records from a production database file and write the results to a new database file. Then, the data could be used by AS/400 Business Graphics Utility (BGU) to produce a chart of the data in that new file. A high-level language program then might be used to change the data in the database file and, finally, the OfficeVision/400 might merge the changed data into documents or form letters.

Database File Types

The integrated database supplies a common, consistent way of describing data. When a database file is created, the fields that are contained in that file are described. All programs that use that file can then use the common definition of the fields in that file. This helps reduce coding errors, makes programs easier to maintain, and helps ensure the consistency of field names and field characteristics.

There are two kinds of database files: physical files and logical files. Both physical files and logical files can be used by the program to access data in the database. In most cases, a programmer does not need to specify whether the data is accessed through a physical file or through a logical view of one or more physical files.

Physical Files

Physical files, or tables as they are called in SQL, contain the actual data stored on the system. They are similar to traditional files. Each physical file has only one, fixed-length record format. A physical file can have a keyed sequence access path to present the data to the program in a sequence other than the order in which the records were added to the file.

For example, employee name, address, date of employment, and other basic information could be part of a single physical file. The keyed sequence access path could be defined to present the records alphabetically by name, by date of employment, by town, or by using any combination of fields.

Logical Files

Logical files, or views as they are called in SQL, do not actually contain data, but describe how records contained in one or more physical files are to be presented to the program, in other words, define the record format for the file.

For example, data about employees may reside in several physical files. The program used to create payroll checks may require information from the file which contains the salary information, the employee file containing names and addresses, the tax file used to calculate withholdings, and the time-card file containing hours recorded for a week's work. The program uses the logical file to create a view of the data from these various physical files that provides all of the required information.

Some of the things the user can control with a logical file are:

- Change the attributes of fields defined in physical files (for example, field name and field order)
- Provide logical sequences of records
- Protect one or more fields in physical files from being read or changed
- Select or omit records based on the value of a field
- Derive new fields from physical file fields (for example, using the number of widgets on inventory and the cost per widget to calculate the value of the widget inventory)
- Join two or more physical files to appear as a single file

There are four categories of logical files:

- A *simple logical file* uses data from one physical file. A simple logical file is the most commonly used category of logical file. It is used to select fields or records from the physical file it is based on. It is also used to arrange the physical file data through a keyed sequence access path. Read, update, add, and delete operations are permitted with a simple logical file.
- A *join logical file* combines fields from two or more physical files into one record format. A join logical file is read-only.
- A *multiple format logical file* uses fields from two or more physical files. A multiple format logical file is either:
  - One record format based on multiple physical files or
  - More than one record format, each based on one or more physical files

  Read, update, add, or delete operations are permitted with a multiple format logical file. A record format for each physical file used must be described in the logical file. The records from each of the physical files is presented in a hierarchical fashion, by key field.
- A *view* is created using SQL/400. A view can be created over both physical and other view files. Read, update, add, or delete operations are permitted with a view. SQL views can be joined views. These have the same restrictions as the join logical file.

File Organization

The AS/400 system does not require the file organization method to be specified when the file is created. The system will store or read the data in the files by building an access path based on the information specified in the file description. The program does not have to contain any programming code to identify the sequence of the records for either read or write operations.

Access Paths

In many systems, records are stored on the disk according to the organization specified by the access method. In the AS/400 system, records are stored independently of their retrieval organization. When records are added to a physical file, new records are normally stored in the order that they arrive in the file. That is, records are stored at the end of the file. Records can be stored in the physical file by the value of a key field, if the key field is specified in the file description.

Records can be read from the file either by arrival sequence or by keyed sequence. The system automatically creates an access path based on the information specified in the physical or logical file descriptions.

Processing files by an arrival sequence access path is similar to processing sequential, indexed, or direct files consecutively on traditional systems, such as the System/36.

Processing files by a keyed sequence access path is similar to processing indexed files on traditional systems. With a keyed sequence access path, the system can present data to a program arranged by the value of the key field or key fields, if more than one is specified.

A keyed sequence access path changes whenever records are added to or deleted from a file, or whenever a record is changed that changes the contents of the key field. The access path is automatically maintained by the system and the programmer does not need to change the application program.

The system uses the same access path unless the file is changed, or unless there is a system failure. Access paths can be recorded in a journal and recovered following a system failure, but the programmer or the person who operates the system must start the journal operation. If an access path is lost, the system creates it again the next time the file is used by a program. However, it takes extra time to rebuild the access path the first time the file is opened.

Members

Data records in a database file can be grouped into members. All the records in a file can be in one member or they can be grouped into different members. To process data in a file, the file must have at least one member. Normally, database files have at least one member, which is added automatically by the system when the file is created. Although most files are created with only one member, the file can contain up to 32,767 members. The system automatically supplies the member name for most database operations when only one member name is specified in the file.

Depending on how the data in the file is used, the file could be divided into smaller groups of records. The smaller groups can be managed as subsets of the data by assigning a member name to each group. The system can then read only records from that member when the member name is specified.

For example, you define an accounts receivable file. You decide to keep one year's data in that file, but you usually process just one month's data at a time. In this case, you can create a physical file with 12 members, one named for each month. The January member only contains data for January, the February member only contains data for February, and so on. You can then process each month's data separately, by processing a member at a time. At year end, you can process

several (or all) of the members together. Figure 5-3 illustrates the concept of members.

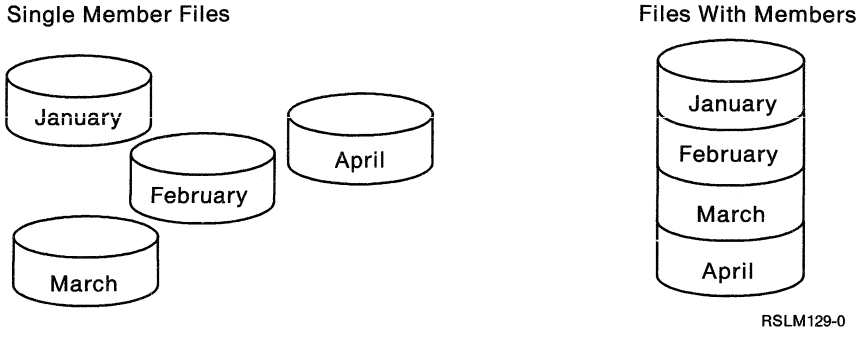


Figure 5-3. File Members

Each member has its associated data and its own access path for that data. The system creates and maintains an access path for each member from the specifications in the file description in the same way as it does for a file with only one member.

Members are especially useful when storing programming statements (called source) in a database file. Members make it easier to organize the source statements.

For example, if you had one file that contained the source statements for all your programs for an application, you could divide that file into members, with each member containing one program's source statements. The PGMA member would contain the source statements for a COBOL/400 program A, the PGMB member would contain the Pascal source statements for program B, and so on. You could manage each program's source separately with member names without creating multiple files. You might also want to create a source file for all the DDS statements, and divide the file into members by the type of file, or by the file name.

Methods of Describing Data

If files are described to the file or record level, the create commands in the control language for a particular type of file contain the only information required to create the file.

If files are described to the field level, there are several methods of describing data for database files: DDS, IDDU, or SQL/400 commands.

Data Description Specifications (DDS)

Externally described data files can be described using DDS. On the AS/400 system, DDS provides the most detailed specifications for the programmer to describe data in the database. In addition to file-, record-, and field-level descriptions available for all methods of describing data, DDS can be used to describe data at the join level, key-field level and select/omit level.

- File-level specifications provide information about the entire file.
- Record-format (or record) level specifications provide information about a specific record format in the file.

- Join-level specifications provide information about which fields to use to join one record format to another record format. Join specifications apply only to join logical files.
- Field-level specifications provide information about the characteristics specified for individual fields in the record format.
- Key-field level specifications provide one or more key fields for the file and describe the order (ascending or descending) for the key.
- Select/omit level specifications provide the comparison values for identifying which records are to be returned to the program during processing. Select/omit specifications apply only to logical files.

Figure 5-4 provides an example using four levels of DDS specifications of a physical file.

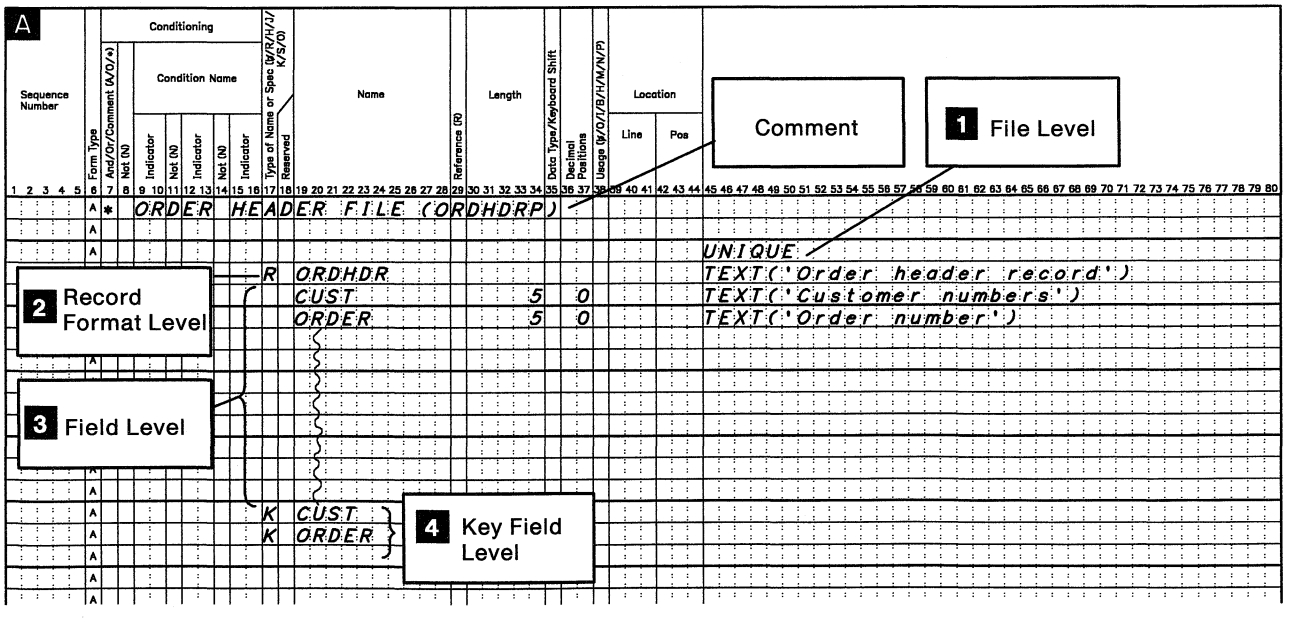
1 File-level specification (optional). The UNIQUE keyword is used to indicate that the value of the key field in each record in the file must be unique. For example, customer identification numbers are usually unique. Duplicate records with the same key value are not allowed in this file.

2 Record-format level specification. The record format name is specified, along with an optional text description.

3 Field-level specification. The field names, field lengths, and number of decimal positions are specified, along with an optional text description for each field.

4 Key-field level information (optional). The field names used as key fields are specified.

Data Description Specifications



RSLM172-1

Figure 5-4. DDS for a Physical File

Interactive Data Definition Utility (IDDU)

IDDU is a menu-driven, interactive method of describing data. In addition to externally described data files, IDDU can be used to describe multiple-format physical files for use with Query, PC Support/400, OfficeVision/400, and DFU. Field, record format, and file definitions can be created and managed independently, and database files can be created from these definitions. These definitions are stored in a system object called a data dictionary. Programmers can use the data dictionary in planning, controlling, and evaluating the collection, storage, and use of data.

IDDU data dictionaries are made up of a set of related database files that contain the definitions. Users can query the data definitions in a dictionary or access them from a program. However, the data dictionary is protected from direct changes by users. The AS/400 IDDU data dictionaries are always active and the system keeps the definitions synchronized with the files they describe. Files are described in a data dictionary if:

- The file is created using IDDU
- The definition of an externally-described file, created by another method such as DDS, is added to the data dictionary using the Add Data Definition (ADDDTADFN) CL command, or
- The file is created in an SQL collection

The data management function determines whether a file is linked to a data dictionary, and therefore can prevent any changes to the definition while it is linked to the file.

Structured Query Language (SQL)

Structured query language (SQL) is the IBM SAA database interface. The SQL/400 licensed program uses a relational model of data; that is, all data is perceived as existing in tables. On the AS/400 system, SQL/400 objects are created and maintained as AS/400 objects. The following table shows the relationship between AS/400 terms and SQL/400 relational database terms.

AS/400 Term	SQL/400 Term
Library	Collection. Consists of a library, a journal, a journal receiver (journal and journal receiver are used to record changes to tables and views), a data dictionary (a set of tables containing object definitions) and an SQL catalog (set of views and logical files).
Physical file	Table. A collection of columns and rows.
Record	Row. The horizontal part of a table containing a serial collection of columns.
Field	Column. The vertical part of a table of one data type.
Logical file	View. A subset of columns and rows of one or more tables.
Keyed sequence logical file	Index. A collection of the data in the columns of a table that are logically arranged in either ascending or descending order. Each index contains a separated arrangement.
No comparable term	Catalog. A set of views and logical files.

SQL supports powerful data definition statements. For example, the SQL CREATE VIEW statement can create an alternative view of a salary table for sales representatives that presents average salaries by department. Or, a single SQL UPDATE statement can add 10% to the salaries of sales representatives who exceed their

quota by 50%. On the System/38 or System/36, the same functions require programming statements.

Methods of Processing Data

Several methods can be used to process database files on the AS/400 system.

AS/400 Query

AS/400 Query is an IBM-licensed program that can be used to obtain information from the database. It can obtain information from any database files on the system that were defined using OS/400 data description specifications (DDS), the OS/400 interactive data definition utility (IDDU), or the Structured Query Language/400 (SQL).

Query can select, arrange, and analyze data stored in one or more database files to produce reports and other data files. Users can create query definitions and run them, use existing queries that were created by someone else, or can run a "default" query (using an unnamed query) against any database file. The user determines what data the query is to retrieve, the format of the report to be created, and whether the report should be displayed, printed, or written to another database file.

Query can obtain information from a single file or a combined set of up to 32 files. Query can select all the fields or a few of the fields, and organize and present the data as specified in the query definition.

Cross-System Product

The Cross-System Product licensed programs provide power and flexibility across systems. Application programmers use the Cross-System Product/Application Development (CSP/AD) licensed program to define, create, and test application programs on a host system. The programs are then sent from the System/370* host system or a programmable work station to the AS/400 system and run under one of the Cross-System Product/Application Environment (CSP/AE) functions of the AS/400 system.

Structured Query Language

Using SQL to access data in a relational database is unlike many programming and data languages. Application programmers do not have to code a sequence of instructions explaining how to get to the data. SQL allows selection of data using a single statement directed to the data management function. The data management function is designed to access and to maintain the data. SQL can be used to retrieve, insert, update, and delete data and control access to data.

SQL statements can be issued interactively or embedded in application programs. Interactive SQL allows statements to be entered directly from the keyboard. Either a completion message or an error message is displayed after each statement is processed. Status messages are normally displayed during long-running statements. Depending on the type of statement, interactive or embedded, either can result in a call to the data management function to run it or to create an intermediate representation of the statement for storing with the program for later processing.

The SQL statements can be either *static* or *dynamic*. Static statements are embedded inside application programs written in other programming languages and are present in the program at the time it is precompiled. This allows the user to

incorporate the processing functions of SQL into application programs as opposed to coding each operation individually in the high-level language. Dynamic statements are typed in from a keyboard or created by a program and are not provided to the data management function until the program runs. The SQL functions can be used for spontaneous processing of information that is outside of the regular application program. For example, if a user wants to create a one-time listing of department numbers, managers, and projects, they need not create a new program but can use SQL to quickly produce the list.

High-Level Language Programs

High-level language (HLL) programs can be used to process database files. The logic for the processing is coded into the program. High-level language programs can also include embedded AS/400 Query definitions, CL commands, embedded SQL statements, and other statements to process the data.

For example, the database file is opened with statements in your HLL program or with the CL open commands: Open Database File (OPNDBF) and Open Query File (OPNQRYF). The OPNDBF command is useful in an initial program for opening shared files. The OPNQRYF command is very effective in selecting and arranging records outside of your program. Then, your program can use the information supplied by the OPNQRYF command to process only the data it needs.

Utilities

The AS/400 Application Development Tools provides two utilities that can be used to process data. The data file utility (DFU) is used to add, change, and delete data in a database file. DFU can be used for files described by RPG/400, DDS, and IDDU.

The source entry utility (SEU) can be used to enter and change source members. SEU can also be used to change, list, print, or delete the existing members in a source physical file, and to create new members in a source physical file.

Distributed Relational Database

Distributed Relational Database allows an application program to access data that is stored in a remote relational database. The AS/400 system starts the Distributed Relational Database Architecture* (DRDA*) as a means of providing the distributed relational database functions.

With distributed relational database (also called distributed database), an application program can access data stored in a remote DATABASE 2* (DB2*), Structured Query Language/Data System (SQL/DS), or AS/400 database. Remote database application programs that use DB2, SQL/DS, AS/400, or Operating System/2* (OS/2*) implementations of DRDA can access the database file data stored on the AS/400 system.

The distributed database functions allow an AS/400 application program to:

- Manage (bind, drop) application program database access plans (called *packages*)
- Manage (create, drop) objects in a remote database including tables, views, and indexes
- Manipulate (select, delete, insert, update) the data in a remote database
- Manage (grant, revoke) authorizations to database objects stored in a remote database

- Obtain (describe) descriptive information about the columns of a remote database table or view
- Run multiple database requests to a single remote database within one unit of work

When using distributed relational database, the system on which the application program is run is called the *application requester*, and the system on which the remote database resides is called the *application server*. This is illustrated in Figure 5-5.

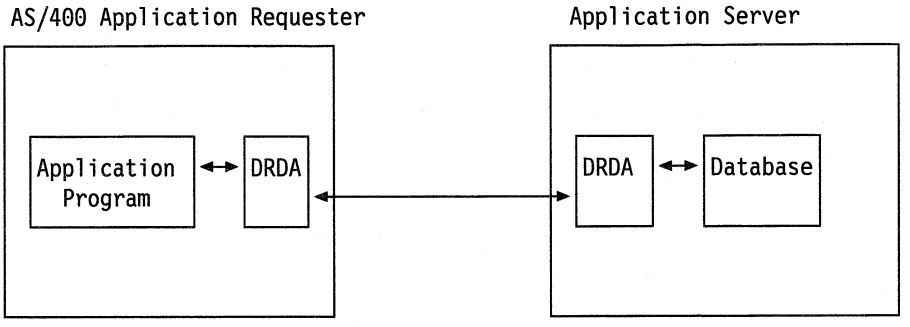


Figure 5-5. Distributed Relational Database Application Requester and Application Server

A significant feature of distributed relational database is that the application program's interface to distributed database functions is through the Structured Query Language (SQL). The AS/400 database interfaces cannot be used to access remote database data.

Using Distributed Database Functions

The first step in using distributed relational database functions is to define remote relational databases to the local AS/400 system. This is accomplished by adding an entry to the relational database directory for each remote database. Each relational database directory entry contains:

- The unique name for the remote relational database. The name can only appear once in the directory. It is this name that is used within the local AS/400 system to identify the remote database. This name should be obtained from the database administrator of the remote database.
- Information about where the remote relational database is located so that a communications path can be established to the remote system.

In addition to the remote databases, the relational database directory should have an entry for the local AS/400 system database. This entry allows the local AS/400 system database to be viewed as an application server by local AS/400 system applications. This entry is also used by SQL in connection with the current server special register, the SQL CONNECT statement, and as the high order part of database object names in SQL statements.

The development of an application program that uses distributed relational database functions is very similar to the development of an application program that uses SQL to access local AS/400 database data. The main differences are described in the following sections.

Database Object Names: Database object names can have three parts:

- Relational database name (optional)
- Collection name (optional)
- Object (table, view) name (required)

Precompilation: When the SQL application program is precompiled (CRTSQLxxx commands, where xxx = RPG, CBL, PLI, and so on) the remote database name must be specified for the relational database name parameter. This name must match an existing relational database directory entry. Specifying this name causes the precompiler to create a package at the remote database for this application program. The specified remote database is the one connected to when the application program runs.

Accessing Multiple Databases: If the application program is going to access more that one remote database (but only one for a single unit of work), the Create SQL Package (CRTSQLPKG) command must be run for each additional remote database. This command creates a package for the application program at the remote database in the same manner as the precompiler. An application program can switch which remote database it is accessing by:

- Committing or rolling back its current unit of work by running an SQL COMMIT or ROLLBACK statement
- Running an SQL CONNECT statement that specifies the name of the new remote database to which the application program wants to be connected

Non-AS/400 SQL Statements: Those SQL statements not supported by the AS/400 system can be used by the application program provided the remote database supports the SQL statements. Errors associated with these statements are generated when the package is created (by the CRTSQLxxx or CRTSQLPKG command) at the remote database. Using this capability makes the application program dependent on a specific type of remote database which means the application program may lose some of its database location transparency. It may also mean that additional programmer training may be required so that the programmer can understand the specific remote database functions and capabilities.

SQL Errors: When the application program checks for SQL errors, the SQLSTATE field of the SQL Communications Area (SQLCA) should be used instead of the SQLCODE field. The SQLSTATE field returns the same error code values for all IBM relational databases. This is not true for the SQLCODE field; that is, the same SQL error may be reported with different SQLCODE values by different IBM relational databases.

Considerations for Using Distributed Relational Database

Before deciding to use distributed database, you should consider the following:

Performance: The performance effect of distributed relational database is usually noticeable and, therefore, should be given careful consideration. Access to remote databases through communications is generally slower than to the local AS/400 database. The application program package that is created and stored at the remote database helps improve the performance of static SQL statements but does not help with the performance of dynamic SQL statements. Whenever possible, use static SQL statements for the best performance.

Transparency: For most SQL functions, the location and type of remote database is transparent to the application program. This reduces the need for programmer training and makes the application programs more portable and database independent. However, the transparency offered by distributed database is not complete. For best transparency use:

- SAA SQL statements
- SAA date and time formats

Security: File data security is maintained on the system storing the data. Therefore, the AS/400 system user has to be properly authorized to have access to the remote database and have proper authorization to the objects contained in the database. This may mean that the application program user has to have a valid user profile defined on the remote system.

Chapter 6. Office Enablers

The AS/400 system provides enablers built into the operating system to support OfficeVision/400 and other similar tasks. In addition to the basic display functions, the AS/400 work station controllers provide word processing functions, including the following:

- Word wrap and continuous insert (gives the user the appearance of an infinitely-long sheet of typing paper)
- Scale line (shows tab stops and margin positions)
- Copy, move, and delete (on a block, line, or word basis)
- Text centering
- Word underline
- Split-screen capabilities

The distribution of functions between the operating system and the work station controller is tightly coupled to offer the best performance.

This chapter briefly discusses the topics listed in the following table. If you would like more information on the topic, and there is more information available in an IBM manual, the manual listed in the right column is a good first reference.

Topics	First Reference
Folders and files Documents Dictionaries Calendar Mail	*Office Services Concepts and Programmer's Guide*, SC41-9758

Folders and Files

Although folders and files are AS/400 objects, and not associated only with OfficeVision/400, they are used for many of the office functions. Folders are the storage and directory for all documents prepared on the system. The folder object works directly with document library services to organize, store, and retrieve documents.

Database and device files are used by OfficeVision/400 the same way they are used by any application program. OfficeVision/400 can use data from the database file to create reports or to merge data within a document when it is printed.

Folder Management Services (FMS): The folder management services function allows the user to work with folders on the AS/400 system. While some objects that a user would store on a system are unique and can be stored as individual documents, most of them will be related to some project or assignment the user is working on. Through the use of folders, the AS/400 system allows the user to store a group of related documents under a common subject. This concept is the same as having a series of folders in a filing cabinet in an office. On the AS/400 system, the document library serves as the filing cabinet.

Each folder can contain many individual documents, and in some cases, a folder can even contain other folders. FMS also allows users to search through folders for

documents by comparing search values entered by the user. After the search, a list of documents matching the search values is produced. This list can be saved as a document and used for future reference. Figure 6-1 shows how the document library, folders, and documents are used to organize data.

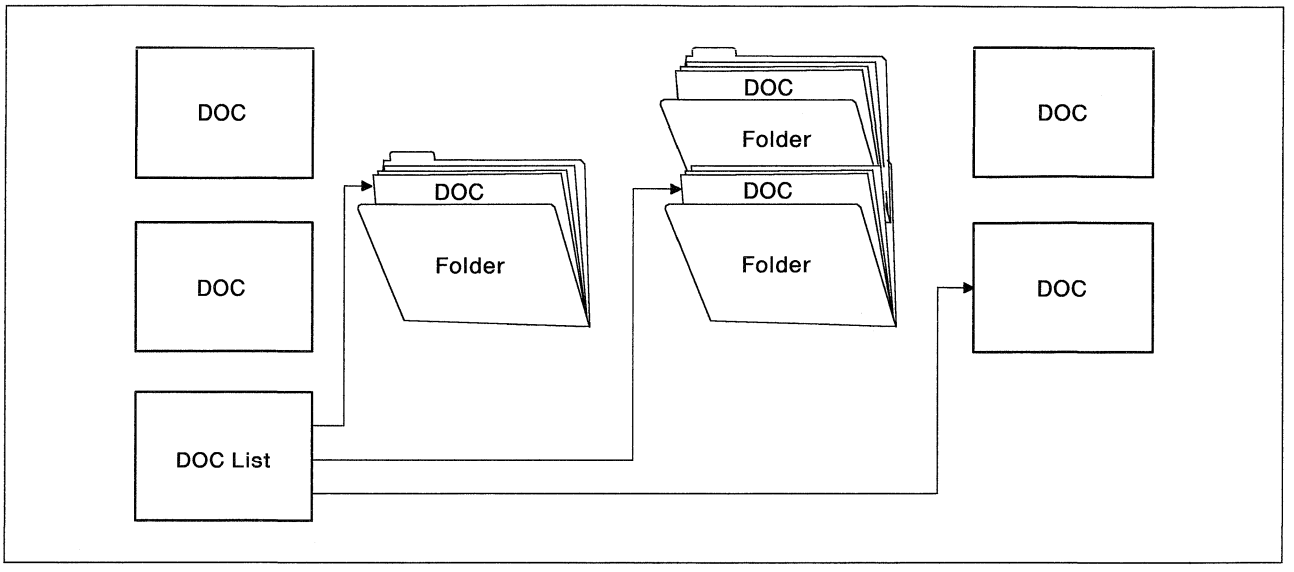


Figure 6-1. Folders and Documents in the Document Library

Security: Using the built-in security functions in the operating system, security for a folder on the AS/400 system is separate from security for the documents or other objects stored in the system. A user who has authority to a folder, does not necessarily have authority to a document in a folder. Similarly, a user who has authority to a document in a folder, may not have authority to the entire folder. If a user lists the contents of a folder, the list includes only those objects the user has authority to.

When a user secures a document, it is the document content, document description, and the authority to change the information about the document and the access codes that is secured. When a user secures a folder, they are securing the folder table of contents, the folder description, and the authority to change the security of the folder.

Documents

The word processing function of OfficeVision/400 allows an AS/400 user to create and edit documents and to store them in folders on the AS/400 system. A document can be as simple as a written report or as complicated as a letter in which the address and graphics are pulled from other documents and inserted in the specified location. Documents can be created through the use of menus or by using commands.

The PC Support/400 licensed program also allows a personal computer user to create and work with documents and to store them in folders on the system. A document can contain personal data or programs. These documents can also be used by OfficeVision/400. See Chapter 8, "Cooperative Processing Enablers" for more information.

When a document is created, it is possible for the user to assign a number of attributes to the document, such as author, subject,and keywords that refer to the document. The purpose of assigning a document this kind of information is to allow the system to be able to search for individual documents based on search values provided by the user. The owner can also assign authority to other users to allow them to read or change the document.

Creating documents with OfficeVision/400 allows the user to take advantage of many advanced editing techniques. The word processing function allows the user to manage text through the use of such operations as document formatting, text moving and highlighting, statistical data support, and the support of text and graphics merging in the same document. But perhaps the most advantageous feature of the OfficeVision/400 word processing function is the ability to create tailored documents.

Tailored Documents

Tailored documents can be created by OfficeVision/400 from two types of data specified by the user:

- *Constant data* is data that will be the same in each version of the document
- *Variable data* is the data in a document that will change in each version of a document

The data can be stored in several different places, and inserted into the document as each document is processed. The word processing function of OfficeVision/400 provides paragraph documents and fill-in documents to store the variable information for tailored documents. The data text merge function also provides a method of coding a document so variable data from files or queries is merged into the documents. The constant data is stored in a shell document, which contains the constant data, format, and codes to insert the variable data.

Several different operations allow the shell, paragraph, and fill-in documents to be combined in many different ways to form the tailored document.

Shell Documents: A shell document is the constant data prepared by the user for a tailored document. The shell document also specifies where the variable information will be included in the tailored document.

Paragraphs Documents: A paragraphs document is a document that has each paragraph placed on a separate page when the document is created. By specifying instruction codes in these shell documents, these paragraphs can be merged with the constant data in any order. In Figure 6-2 on page 6-4, any combination of the individual paragraphs in the paragraphs document could be used to complete the body of the letter. As is shown also in Figure 6-2 on page 6-4, it is possible to incorporate several techniques to merge variable and constant data in one tailored document. Here the shell document is using data field instructions in the paragraphs document to insert the name and address of the person to whom the letter is to be sent. A data field instruction reserves a place in the document for some piece of variable information. The paragraphs document inserts information with data field instructions in several of the paragraphs before they are included in the final document.

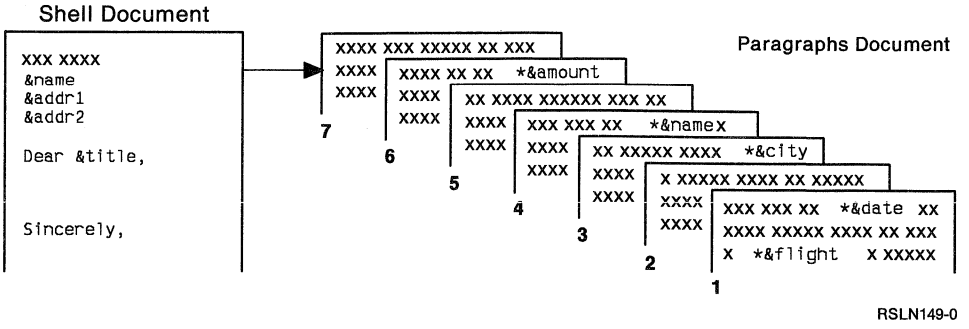


Figure 6-2. Paragraphs Document in Tailored Documents

Fill-In Document: A fill-in document is a document that stores data for use in other documents. Included in this document are the names of the data fields that are used in the shell document and the value associated with the data field that allows the shell document to be tailored. As can be seen in the sample of a fill-in document (Figure 6-3), the records are read one at a time until the end of the file is reached.

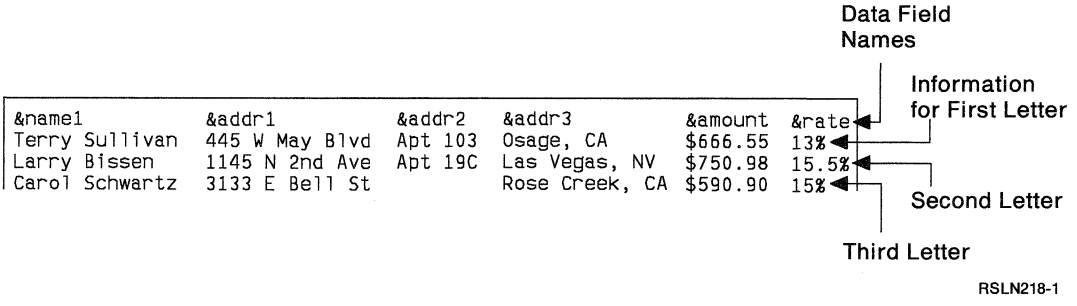


Figure 6-3. Fill-In Document

Stop Codes: Another method of inserting variable data into a shell document is the use of stop codes. Stop codes are control characters that indicate the places in a shell document that require variable data to be inserted as shown in Figure 6-4 on page 6-5. When editing the document in the example, the cursor can be positioned to the next stop code by using a keyboard key. The user can enter the missing data. The stop code method is a quick and easy way to insert the information when a user needs a small number of copies of a document, or the variable data is not already stored in another document.

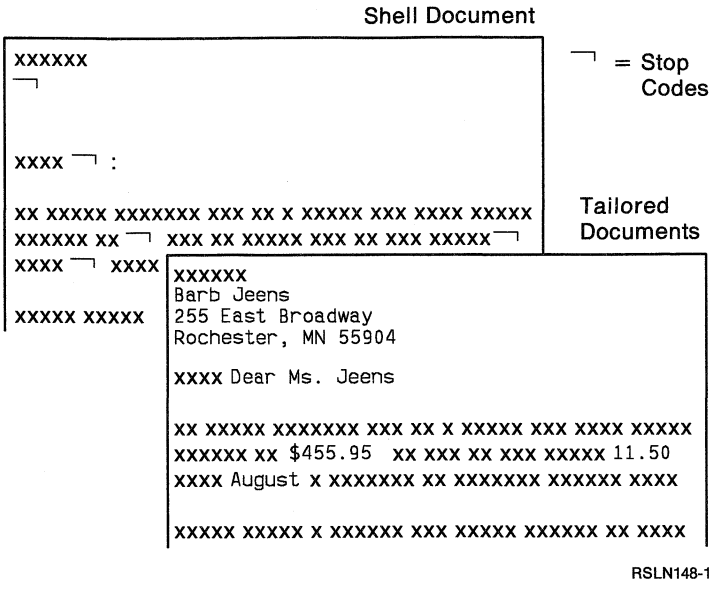


Figure 6-4. Stop Codes in Tailored Documents

Data/Text Merge: Using the data/text merge function, OfficeVision/400 creates tailored documents from data stored in a fill-in document, a file, or a query. When data/text merge is used to create a tailored document, the actual document with all of its text is not created until the document is printed. When the users want to print their document, they select the shell document they wish to use, and the system pulls in the needed data based on the instructions found in the shell document. In order to create tailored documents using the data directly from a file, a query, or a fill-in document, the shell document that the user is using must include data field instructions, as is shown in Figure 6-5. A paragraph document, on the other hand, can include the field names instead of coding the data field instructions in the shell.

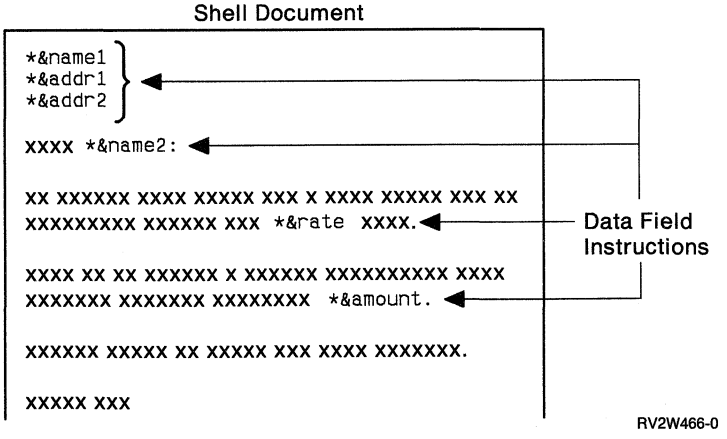


Figure 6-5. Data Field Instructions in Shell Documents

When the data field instructions are created, the user specifies within the shell document where to look for the variable data: in the fill-in document, file, or query. The user also specifies the type of merge to use: multiple letters or column list.

Just as with other objects on the system, document owners can control who has the authority to read or change the documents. An *access code* is assigned to a document when filing a document or changing a document description. If a document is

given an access code, any user with the access code can read and change the document.

Library Services

Using the OS/400 library object, the filing system consists of a single container for all objects that contain data for any of the office products. Various types of objects, including mail, text, programs, and folders, can be stored in this library.

Document library services (DLS) is made up of three different components: library services, remote library services, and distribution services.

- *Library services* allow a user to search, store, and retrieve documents in the document library of the system.
- *Remote library services* allow users to store and retrieve documents on an AS/400 system other than their own. With the proper authorization, users can get the document from the remote system, make the changes to the document, and then return the document to the remote system. While a document is in use, only the user who has control of the document can change the contents of the document. This prevents several users from making changes to a document at the same time.
- *Distribution services* allow the user to list, receive, and cancel mail from the mail log and distribute documents and messages to other users.

Dictionaries

The inclusion of dictionaries in OfficeVision/400 allows the user to proof documents for both spelling mistakes and reading level. The user can proof a single word or an entire document. When a misspelled word is found, a list of possible words can be displayed to the user with a function key. If the user wishes to see a list of synonyms for a word, a list of synonyms can also be displayed.

AS/400 language dictionaries include dictionaries in several languages, a dictionary of United States medical terms, and a dictionary of United States legal terms. The user can build a dictionary that includes words that are not in the AS/400 language dictionaries. Terms common to the user business can be added. Words can be added individually (for example, as they are encountered in the document) or as a group. After the user's dictionary is created, it must be added to the dictionary list for the text profile or document. When a user checks a document for spelling, it can be checked against any of the available dictionaries.

The following table lists the AS/400 language dictionaries:

Language	Dictionary Name
Brazilian Portuguese	BRASIL
Catalonian Spanish	CATALA
Danish	DANSK
Dutch Preferred	NEDERLND
Dutch Modern	ACTUEEL
Finnish (hyphenation only)	SUOMI
French	FRANCAIS
French Canadian	FRA2
German	DEUTSCH
Greek	GREEK
Icelandic	ISLENSK

Language	Dictionary Name
Italian	ITALIANO
Legal	LEGAL
Medical	MEDICAL
National Portuguese	PORTUGAL
Norwegian-Bokmal	NORBOK
Norwegian-Nynorsk	NORNYN
Spanish	ESPANA
South African	AFRIKAAN
Swedish	SVENSK
Swiss-German	DSCHWEIZ
UK English	UK
US English	US

Calendar

The calendar function gives the AS/400 user an efficient and easy way to organize the constant scheduling, canceling, and rescheduling of meetings and appointments that is present in every business environment. The OfficeVision/400 calendar allows the user to create several different types of calendars.

- Users can create calendars for themselves or for another person who has given them authority to do so.
- Calendars can be created for special functions such as keeping a schedule for a meeting room or scheduling the use of equipment such as an overhead projector.
- A *calendar group* can be created and used for several different people. For example, if a user were given the position of team leader for a project, the leader could create a calendar group consisting of all the team members. With this calendar group, the team leader could check each team member's calendar to find when the whole team was available for a meeting. When an open time was found, the meeting could be scheduled with all the team members at the same time.

Calendars also have object level security features supported by the OS/400 program. For example:

- Users can specify who is allowed access to any calendar they create. This access can be either view authority or change authority. If a user gives someone change authority to a calendar, that person can add, delete, and change entries on that calendar.
- Users can classify the status of each entry they make to a calendar. If the user classifies an entry as tentative, people who are authorized to the calendar can see the entry and the classification. Along with the entry, however, there will be a mark that will allow others to know that this entry may change. A personal classification to an entry allows authorized users to see that an appointment is scheduled, but the details of the entry are not displayed. If no classification is assigned to the entry, the entry is considered to be confirmed and other users know that the information will not change.

Mail

The mail function of OfficeVision/400 provides an efficient method of communication between AS/400 system users and users of the same or other systems. OfficeVision/400 treats any message, note, or document sent from one user to another as a mail item. Mail can be received or sent either by printed copy or displayed at a work station.

Classifications: When a user sends any type of mail to another user, one of two different types of classifications can be assigned. These classifications limit who can read the mail item.

1. *Personal classification.* If a mail item is given a personal classification, only the person to whom the item is addressed can view it.
2. *Priority classification.* The sender indicates priority level of the mail. If a mail item is a given high priority, the receiver of the item is notified by a message sent to the work station that a high priority item was received.

Security: A user can give other users authority to work with the user's mail by assigning them authority to *work on behalf* of others. If a piece of mail classified as *personal* is received, only the user to whom it was sent has the authority to view or change it.

Distribution Methods: OfficeVision/400 allows the user the ability to send mail to one user or to a group of users. Sending data to other users or groups of users is accomplished through the use of directories and distribution lists.

- The *system distribution directory* contains the information that is used by the system to distribute electronic mail to system users. Included in this directory are the user ID and the user address of each of the system users. The user ID tells the system who a user is, and the system address tells the system where to find the person. From this and other information, the system can determine if the user is a local, remote, or indirect user. An indirect user is someone who receives mail in printed form from the printer at that location rather than at a display station. Other items such as the user's street address and telephone number can also be stored in this directory. Anybody who is given a user ID is included in the system directory.

  Special authorization is required to update the system distribution directory. The user(s) with this authority is identified to OfficeVision/400 as the system administrator. Therefore, if a new user is added to the system or a current user leaves the system, the system administrator is the one who must change the directories. However, users can change some information pertaining to their own directory entries. The system distribution directory can be viewed by any of the system users, but users can only change information relating to themselves.

- A *distribution list* is a collection of system distribution directory entries. While sending a single mail item from one user to another, user IDs are entered. However, if a user wants to send the same mail item to several users, OfficeVision/400 allows the use of distribution lists. If a user frequently sends mail items to the same group of people, the user IDs and system addresses can be used to create a distribution list. To send an item to the people on the distribution list, type the distribution list name in the user ID space. The item is sent to all of the users on the list at the same time. A user can create, change, or delete a distribution list at any time.

Chapter 7. Communications Concepts for Application Programming

Communications is an integral part of the AS/400 system. Much of communications support is included as part of the OS/400 program, including:

- Systems Network Architecture (SNA)
- Asynchronous
- Binary Synchronous Communications (BSC)

Other communications types are included as separate licensed programs. These include:

- Transmission Control Protocol/Internet Protocol (TCP/IP)
- Open systems interconnection (OSI)

Application programs work with communications functions similarly to how they work with local devices and files. Many applications required for day-to-day use are supplied by IBM as part of the OS/400 program, or part of a licensed program.

This chapter provides an overview of the communications support provided for the AS/400 system. It is organized into three major sections:

- Application enablers
- Communications protocol support
- Link-level connectivity

Programmers may only care about the application enablers; this is where the work is actually accomplished. However, certain application enablers or functions are only available with specific communications protocols and certain link connections. These affect network design and operation.

Abbreviations and More Information

Many of the topics covered in this chapter are features, products, or functions better known by their abbreviation than by the long version of their names. All abbreviations used in this chapter are listed alphabetically in the table "Communications Abbreviations and More Information" on page 7-17. This same table also lists the best first reference if more information on the topic is available in another IBM manual.

Application Enablers

The AS/400 system provides several application program interfaces that allow you to take advantage of the services offered by the system. Some have IBM origins, some are supported by general vendor agreements, and some support international standards. This allows applications to be customized to meet user requirements.

Accessing Remote Data (Record Level)

There are many methods to access files, specifically record-level access on the AS/400 system. The programming interfaces described in this section allow record-level access over asynchronous, BSC, SNA, TCP/IP, OSI, and user-defined communications as shown in Figure 7-1 on page 7-2.

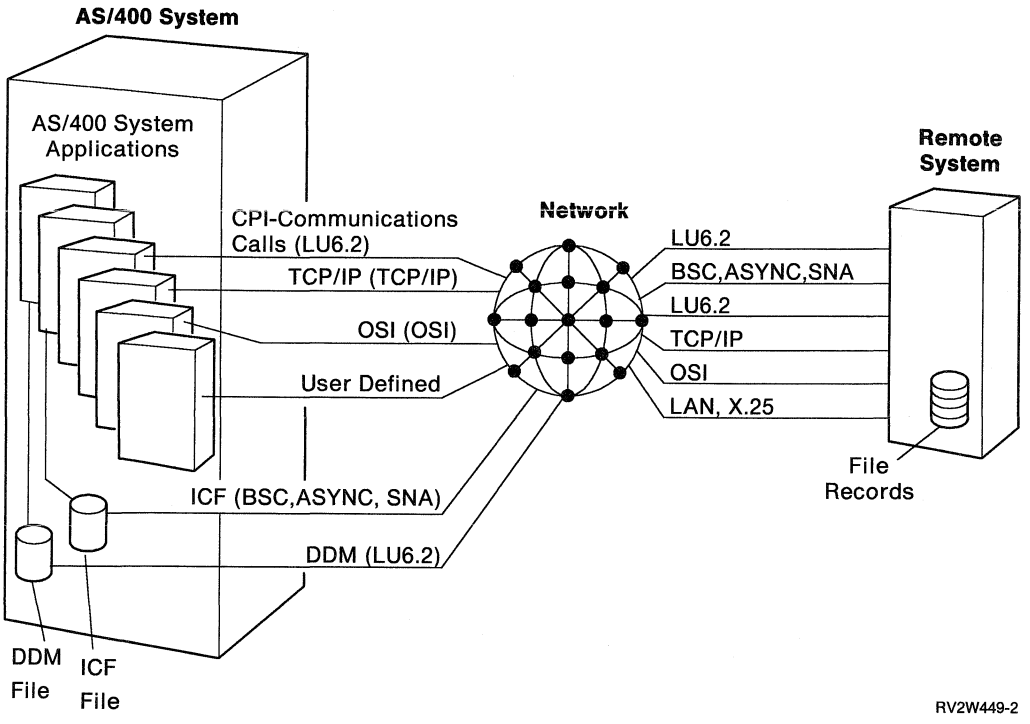


Figure 7-1. Accessing Remote Data (Record Level)

DDM: DDM support on the AS/400 system allows application programs to access data files that reside on a remote system. Any system that supports Level 1.0 of the DDM architecture as a source system can access data (if authorized to do so) on any other system to which it is attached. The system must support DDM as a target system and the systems must support compatible subsets of the DDM architecture.

The OS/400 program DDM, as a source system, supports Level 1.0 of the DDM architecture. DDM as a target system supports Level 1.0 of the DDM architecture for record file types and Level 2.0 for stream files (documents) and directories (folders).

Systems that use DDM communicate with each other using APPC support and can use the networking support provided by APPN.

Using DDM, an application program can get, add, change, and delete data records in a file that exists on a target system. It can also perform file-related operations, such as creating, deleting, renaming, or copying a file from the target system to the source system.

When DDM is in use, neither the application program nor the program user needs to know if the file that is needed exists locally or on a remote system. DDM handles remote file processing in essentially the same way that local file processing is handled on the local system, and the application program normally does not receive any indication of where the requested file is located.

ICF Interface: ICF provides a common file interface across many communications methods, such as APPC, BSCEL, and SNUF as shown in Figure 7-2. This interface includes functions such as establishing a communications session between a local and remote system, starting a process on a remote system, and sending and receiving data.

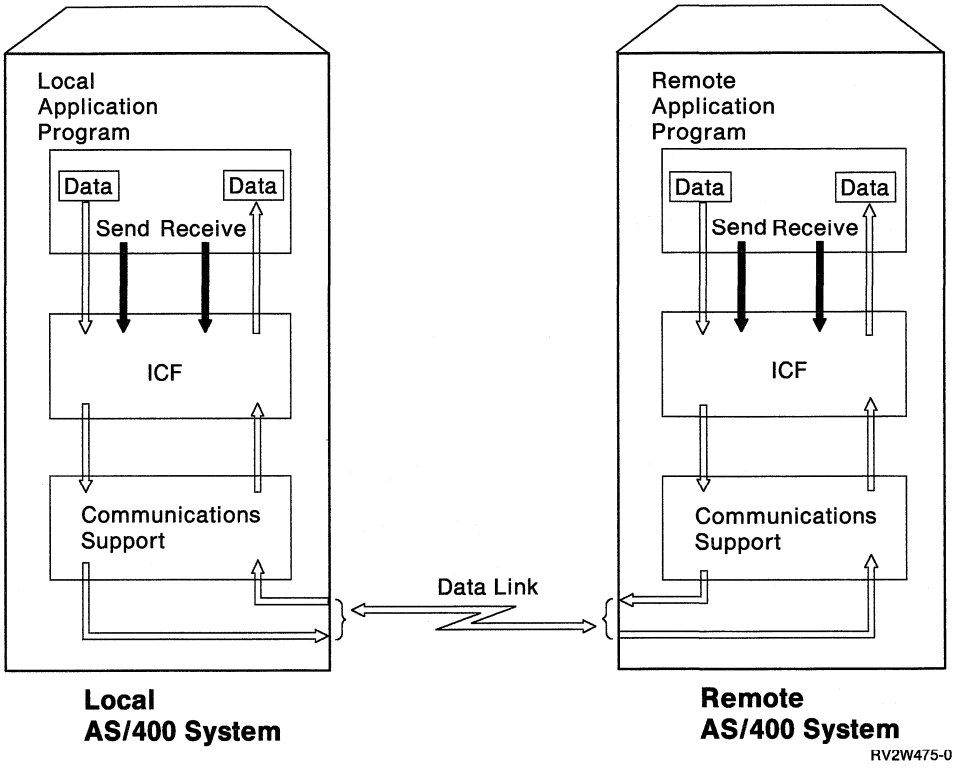


Figure 7-2. Overview of ICF

CPI Communications: CPI communications is the communications element of the SAA common programming interface. It allows program-to-program communications using APPC. The application interface provided by CPI Communications is a call interface that provides functions such as establishing a conversation, sending and receiving data, and exchanging control information.

TCP and UDP (TCP/IP Programming Interfaces): TCP/IP includes interfaces that allow applications to communicate with other systems. These interfaces include both TCP routines that perform end-to-end connectivity and recovery and UDP routines that require recovery to be built into the application.

ACSE Presentation and Session Programming Interface: The OSI Communications Subsystem/400 licensed program provides several APIs that allow the user to take advantage of the open system environment. OSI protocols allow an AS/400 system to communicate with IBM or non-IBM systems running a compatible set of OSI protocols. The program operates with non-IBM systems, recognizing the regional differences between North America, Europe, and Japan. Popular profiles supported include:

- US GOSIP
- UK GOSIP
- CEN/CENELEC
- CEPT
- INTAP

User-Defined Protocols API: The base OS/400 operating system includes application interfaces that allow user communications protocols to communicate over X.25, token-ring, or Ethernet lines. These callable routines allow the user to allocate lines, set protocol filters, and send and receive data.

Accessing Remote Files

There are many methods to transfer files on the AS/400 system. This section outlines a few different methods that run over asynchronous, BSC, SNA, TCP/IP, and OSI communications as shown in Figure 7-3.

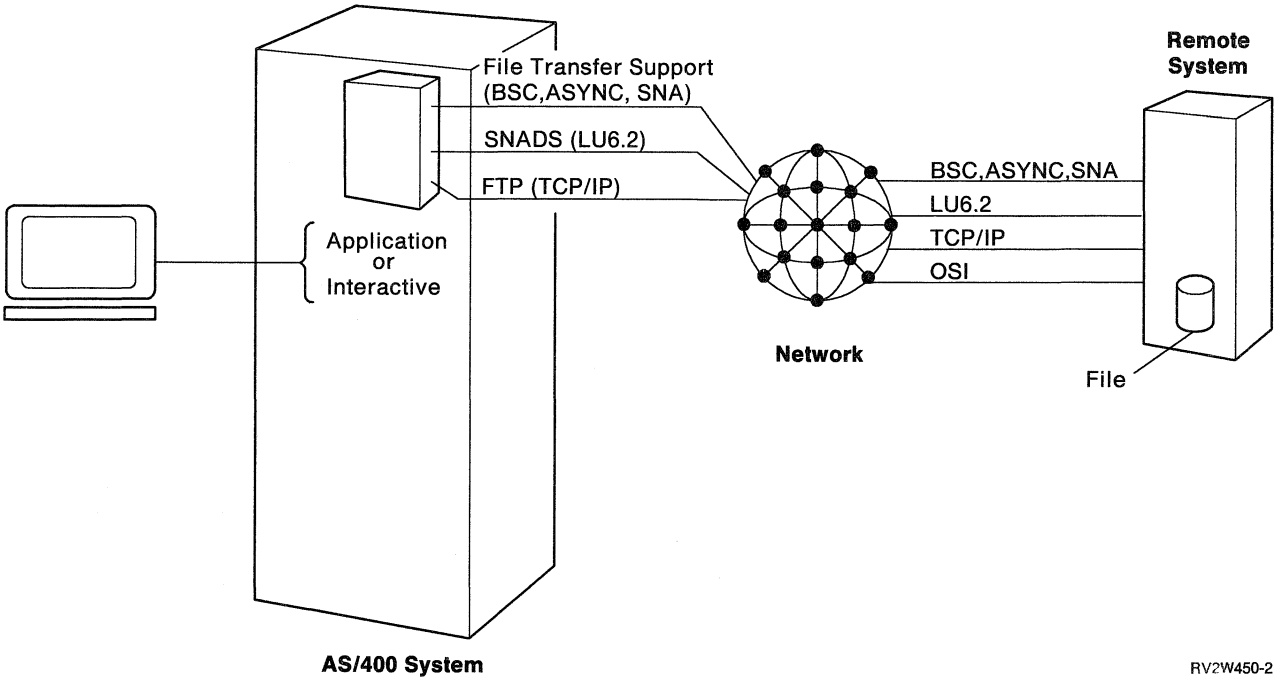


Figure 7-3. Accessing Remote Files

File Transfer Support: File transfer support is a callable function that can be used to transfer files using asynchronous, BSC, or APPC communications. It allows AS/400 systems to transfer files to other AS/400 systems within a simple application environment.

SNADS (Office): SNADS is the AS/400 implementation that provides an asynchronous distribution service for the distribution of objects such as documents, files, job streams, and messages to other systems in the network.

FTP: FTP is an application that comes as part of the AS/400 TCP/IP Connectivity Utilities/400 licensed program. The user can send and receive many types of files, depending on the capabilities of the remote system.

Accessing Remote Objects and Sending and Receiving Messages

The AS/400 system uses a variety of ways to send and receive messages and receive and distribute objects and messages within a computer network as shown in Figure 7-4. This section describes some of the processes, which vary depending on whether the network is a TCP/IP network or an SNA network. Within an SNA network choose an option depending on whether it consists of one or more of the following:

- Hosts
- AS/400 system, System/36, or System/38
- Personal computers
- A combination of all of these

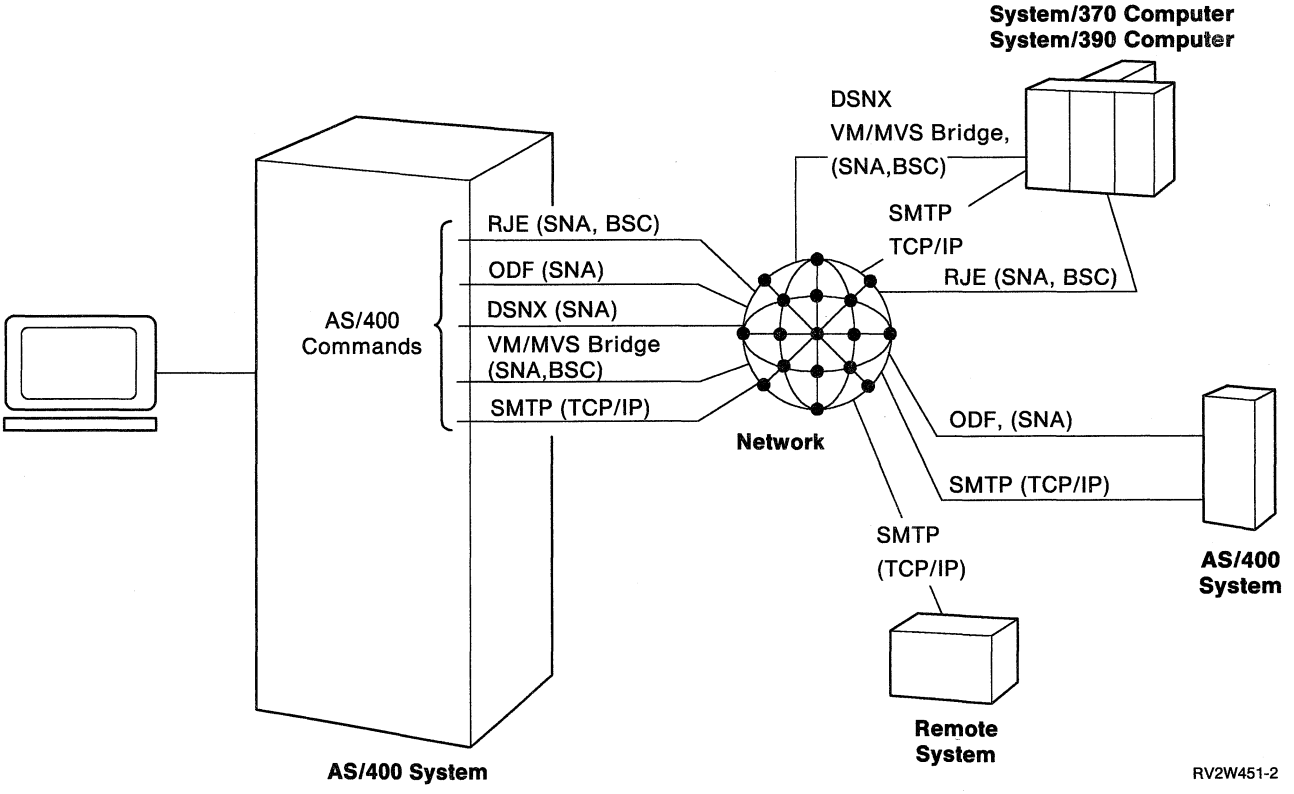


Figure 7-4. Accessing Remote Objects

Object Distribution: AS/400 object distribution uses SNADS to distribute objects other than documents between System/36, System/38, and AS/400 systems. The objects can be data files, save files containing AS/400 objects, spool files, job streams, and messages.

DSNX: AS/400 DSNX support allows the AS/400 system to participate in a NetView* DM network. It can be used to distribute files and job streams in a System/370* controlled network. It can also be used to do central site programming and maintenance and distribution of AS/400 objects. DSNX allows an AS/400 system to function as an end node to the host system and as an intermediate node between the host system, other AS/400 and System/36 systems, and personal computers for distribution of objects.

VM/MVS Bridge: AS/400 VM/MVS bridge is an application program that allows users on an AS/400 system and users on a System/370 host to exchange distributions, including DIA documents, OfficeVision/VM documents and notes, data files, and print files.

SMTP: SMTP is an application provided with TCP/IP Connectivity Utilities/400 licensed program. It allows OfficeVision/400 and other SNADS users to send and receive mail from TCP/IP network users. Documents, notes, and messages sent from an AS/400 system are formatted as notes. In addition, documents must be formatted as FFT before they are transmitted.

RJE: The AS/400 system can also operate as an RJE work station to a host system in either a BSC or SNA environment by using the remote job entry function, which is a part of the Communications Utilities licensed program.

With RJE, the AS/400 system functions as a remote job entry work station that submits batch jobs to a System/370 or System/390* host for processing under one or more of the following:

- OS/VS1 RES
- MVS/SP* operating system, JES2, or JES3
- VM/SP RSCS
- DOS/VSE, VSE/POWER, and Input Readers/Virtual Storage Extended

Accessing Remote Systems

There are many methods by which you can access other AS/400 system applications through remote log-on procedures (see Figure 7-5 on page 7-7). This section outlines a few different methods that run over asynchronous, SNA, TCP/IP, BSC, and user-defined communications.

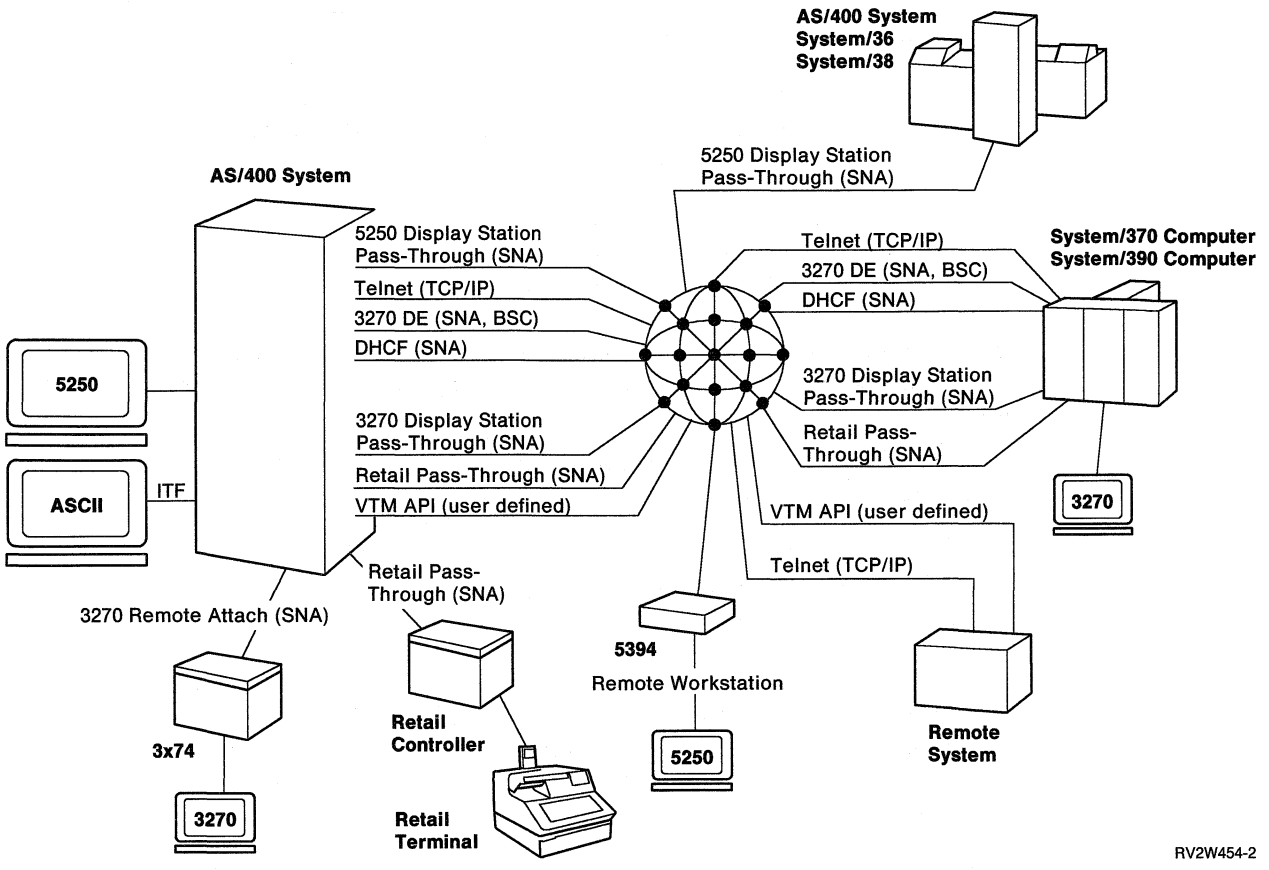


Figure 7-5. Accessing Remote Systems

5250 Display Station Pass-Through: 5250 display station pass-through allows an AS/400 work station user to sign on to another system in the SNA peer network and to run any application or system task on that remote system. Tasks that may be run include change management, problem analysis, or any activity a local user may perform.

TELNET: TELNET is an application that comes as part of the AS/400 TCP/IP Connectivity Utilities/400 licensed program. It allows IBM or non-IBM systems to log on and access applications as if locally attached. It also allows the AS/400 system to log on to remote systems and access applications as if locally attached to the remote system. This is accomplished without any special customizing of applications.

ITF: ITF is included as a part of the OS/400 program asynchronous communications support. It allows the user to send and receive data through application programs, such as electronic message services for asynchronous display stations. Through ITF, these application programs can be used to send messages, such as interoffice memos. In addition, ITF lets the user send and receive file members and send OfficeVision/400 documents.

3270 DE: 3270 DE allows any 5250-family display or printer to emulate a 3270-type device and access a System/370 or System/390 host. It allows use of a 5250-type display station with a host system application as though it is a 3278 Model 2 display station. It also allows use of any printer attached to the AS/400 system to print information received from the host system, as though it is a 328x printer. In the SNA

3270 network, the AS/400 system appears to the host system to be a 3174 or 3274 controller with attached display stations and printers.

Host applications supported for 3270 DE are CICS/VS, IMS/VS, and TSO/VS. Access methods supported are VTAM* and TCAM. Communications protocols supported are SDLC, X.25, and token-ring.

The AS/400 system and 3270 devices can share the same SDLC, X.25, or token-ring link. Multiple sessions of LU1, LU2, and LU3 3270 emulation, SRJE, APPC, SNUF, and DHCF are supported on the same link.

BSC 3270 DE: BSC 3270 DE provides the same support as SNA 3270 DE, including the same host applications and emulated display stations and printers, with the following differences:

- VM/SP CMS, RPS/CM2, and EDX/CF (Series/1*) host applications are supported in addition to CICS/VS, IMS/VS, TSO/VS.
- The BTAM access method is supported in addition to VTAM and TCAM.
- The communication protocol supported is BSC, multipoint tributary, nonswitched line, EBCDIC or ASCII code.

DHCF: DHCF on the AS/400 system allows a user at the host system to remotely operate the AS/400 system as though the user is at an AS/400 system station. 3270 display stations on the HCF host system are used as if they were remote 5250 display stations attached to the AS/400 system. DHCF is part of the OS/400 program.

3270 Display Station Pass-Through: Sometimes called 3270 pipeline, the 3270 display station pass-through support passes remote 3270 data streams through the AS/400 system to an upstream System/370 host system. It permits the user of a 3270 display station attached to an AS/400 system to access a System/370 host. It does this by using SNA 3270 device emulation without data stream translation.

BSC 3270 DE cannot be used with 3270 display station pass-through.

3270 Remote Attach (RA): 3270 RA permits 3270-type display stations and printers to be remotely attached to an AS/400 system. 3277, 3278, and 3279 display stations are seen by the AS/400 system as 5250-type displays. 3270 keyboards are mapped to logical 5250 keyboards. The 3287 printer is seen as a 5250-type printer. In all cases, an attached 3174 or appears as a remote 5251 display station to the AS/400 system and application programs. All emulators that conform to 3274 Model 31C protocol are also supported.

Communication types supported are SDLC, X.25, and token-ring.

Remote Work Station: Remote work station is a function that allows you to attach a work station to an AS/400 system using a communication line. This allows connections like display station pass-through, DHCF, NRF, and 5394 on an SNA backbone.

Retail Pass-Through: The AS/400 system retail pass-through support is used when a host system, such as a System/370 host, is connected to the AS/400 system and the AS/400 system is connected to various retail controllers. The retail pass-through support provides the ability for the AS/400 system to pass the data from the retail controllers through the AS/400 system to the host system. This support uses

two SNA LU type 0 sessions: one session between the AS/400 system and the host system and the other session between the AS/400 system and the retail controller.

Virtual Terminal API: The virtual terminal allows an AS/400 system program (called a *server program*) to work with an AS/400 application that is performing work station input/output as shown in Figure 7-6.

Using the virtual terminal API, the server program handles data transmission and translation between the virtual terminal and a work station system program (called a *client program*). The client program handles data transmission and translation between the physical work station, such as a PC, and the AS/400 server program. Together the server and client programs allow the virtual terminal to be supported as if it is locally connected.

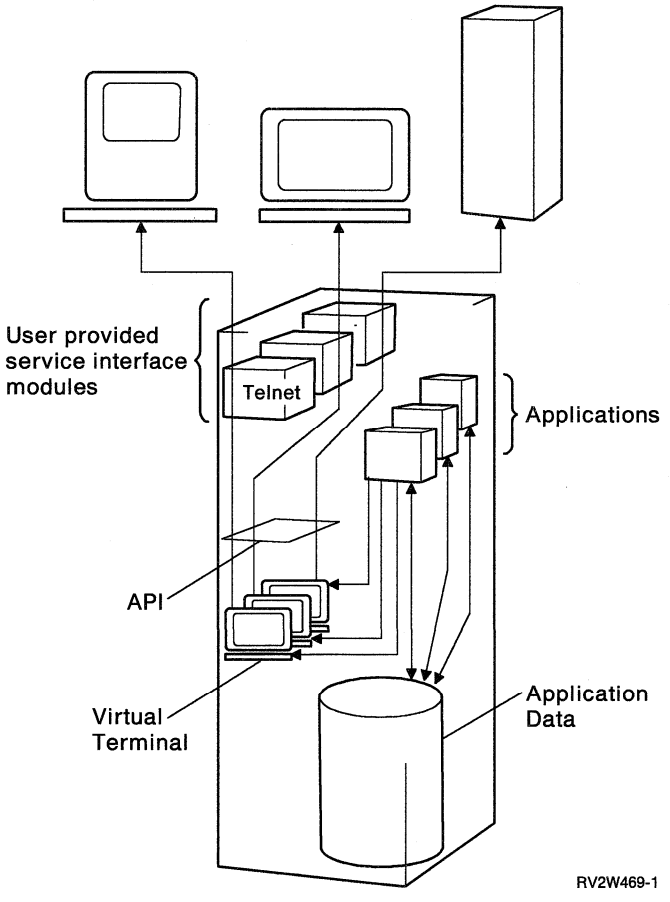


Figure 7-6. Virtual Terminal Model

Communications Protocol Support

The AS/400 system allows communications using a variety of data link protocols to a wide array of products selected from a set of multiple protocols. The support allows communication with IBM host products, personal computers, and a wide range of non-IBM equipment. The various protocols available are asynchronous communications, BSCEL, SNA, OSI, and TCP/IP.

BSCEL

BSCEL provides distributed data processing support to users who want to communicate with another system or device at a remote location using BSC. It supports interactive and batch communications between application programs on different systems using BSC. On the AS/400 system, the BSCEL support can be used to write application programs to communicate with another system that also has BSCEL support. BSCEL support on the AS/400 system can be used for BSC with any of the following IBM host systems:

- System/370 and System/390 hosts
- 30xx
- 43xx
- 9370

OS/400 BSCEL support can communicate with a host system using BTAM and any host operating system that supports BTAM. For example,

- DOS/VS
- CICS/VS
- IMS/VS
- VM/370

SNA

Systems Network Architecture (SNA) is an architecture made up of several logical unit (LU) types. These logical units are architectural definitions of how to communicate with systems, controllers, and terminals that also support the same LU types. Most of the SNA support on the AS/400 system is part of the base OS/400 licensed program. See Figure 7-7.

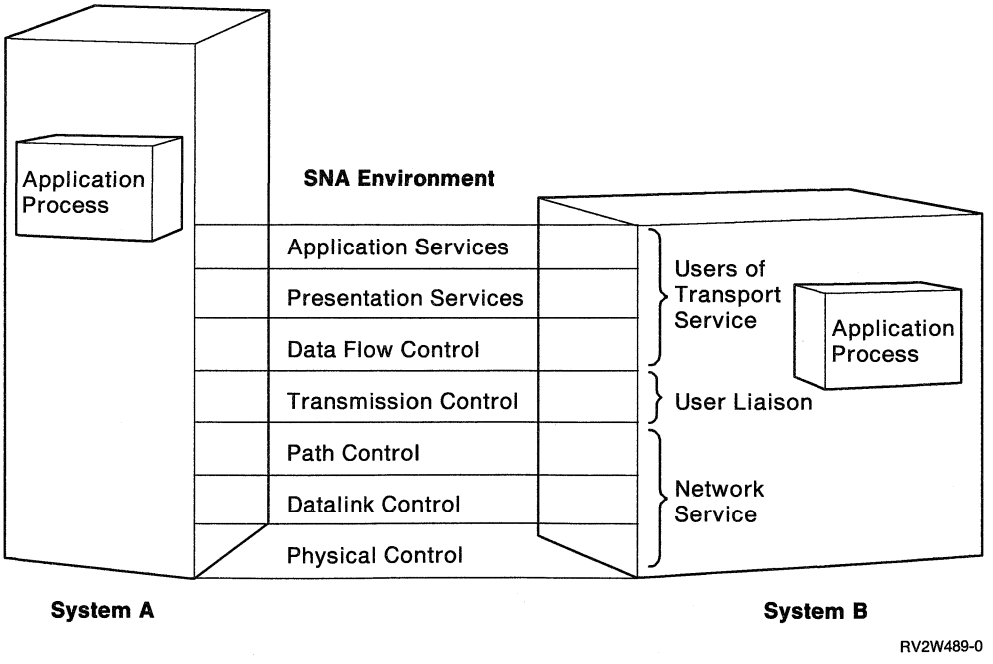


Figure 7-7. SNA Interconnection Overview

APPC and APPN: APPC is the AS/400 system implementation of SNA LU 6.2 and PU T2.1 architectures. It allows applications that reside on different processors to communicate and exchange data in a peer relationship with one another. This connectivity extends to any product, IBM or non-IBM, that uses the LU 6.2 base and

equivalent set of optional functions. It also allows two application programs that are running in two different jobs on the same system to communicate with each other.

APPN support is an enhancement to the PU T2.1 architecture which provides networking functions such as:

- Dynamically locating LUs in the network through the use of searching distributed directories
- Dynamically selecting routes to LUs based on selection characteristics when an application requests a session
- Intermediate routing of LU 6.2 session traffic through the node for sessions between other LU 6.2 partners
- Routing session data based on transmission priorities
- Dynamically creating and starting remote location partner definitions

Finance Communications: AS/400 finance communications uses high-level language operations and communications functions to allow communication between an AS/400 system and finance control units, providing a banking environment communications system.

AS/400 finance communications provides specific SNA LU type 0 primary support. LU type 0 support allows the attachment of finance control units and allows programs in the supported high-level languages on an AS/400 system to communicate with the attached finance control units. In addition, finance communications provides support for ICF files and it provides a non-ICF file interface to support migration of those System/38 finance applications that used this interface.

Some finance controllers also communicate using asynchronous, BSC, or LU 6.2 protocols. The asynchronous, BSC, or APPC communications support is configured and used for those protocols.

Retail Communications: AS/400 retail communications uses high-level language operations and communications functions to allow communication between an AS/400 system and retail controllers, thus providing a retail store point-of-sale environment communications system.

AS/400 retail communications refers to the retail industry-specific SNA LU type 0 primary support provided on the AS/400 system. This support allows the attachment of retail control units that use SNA LU 0 protocol. Retail communications allows programs in the AS/400 system supported high-level languages to communicate with the attached retail control units. Retail communications can also use ICF files. Some retail controllers also communicate using asynchronous, BSC, or LU 6.2 protocols. The asynchronous, BSC, or APPC communications support is configured and used for these protocols.

SNUF: SNUF (LU 0) allows the user to write programs that can communicate with either CICS/VS or IMS/VS on a remote host system through SNA without being concerned with the unique communications requirements of the host system. SNUF is part of the OS/400 program and can be used for both interactive or batch communications between an AS/400 system and the host system. The host system can be an SNA System/370, 30xx, or 43xx processor using either CICS/VS or IMS/VS. The communication line can be SDLC, X.25, or token-ring.

SNUF has the following capabilities:

- AS/400 programs can start system tasks or user programs on remote host systems with CICS/VS or IMS/VS.
- CICS/VS and IMS/VS tasks on a remote host system can start programs on the AS/400 system.
- More than one program on an AS/400 system can communicate at the same time with CICS/VS or IMS/VS programs on a host system.
- SNUF can share a communications line with other SNA-based facilities on the AS/400 system.

Intrasystem Communications

Intrasystem communications allows two application programs, which are running in two different jobs on the same system, to communicate with each other. It supports an ICF interface for sending and receiving data between two programs.

OSI Communications Subsystem/400 Support

The OSI Communications Subsystem/400 Support licensed program supports the OSI protocols in the AS/400 system environment. The protocols are international standards for systems interconnection that have been established by:

- ISO
- CCITT

OSI is made up of a local system environment and an OSI environment. The local system environment is a closed system and is complete by itself. When an application process on one system needs to interact with an application process on another system, both systems must be open to the OSI environment. This last environment is made up of seven layers of OSI protocols:

Application Provides the distributed information service to support an application process and manage the communication.

Presentation Translates and formats the information to allow the application process to interpret what is communicated.

Session Supports the dialog between cooperating application processes, binding and unbinding them into a communicating relationship.

Transport Provides end-to-end protocol and information exchange with the level of reliability needed for the application process. The services provided to the upper layers are independent of the underlying network. This layer is the liaison, acting as the go-between for the user and the network.

Network Provides the means to establish, maintain, and end the switched connections between systems. Addressing and routing functions are included. An additional global sub-layer may also be provided to ensure a consistent quality of service on connections over more than one network. The interface between this layer and the transport layer provides services that are independent of the underlying media.

Data link Provides synchronization and error control for the information transmitted over the physical link.

Physical Provides the functional and procedural characteristics to activate, maintain, and deactivate the physical connection. The electrical and mechanical characteristics provide the physical interface to the transmission media.

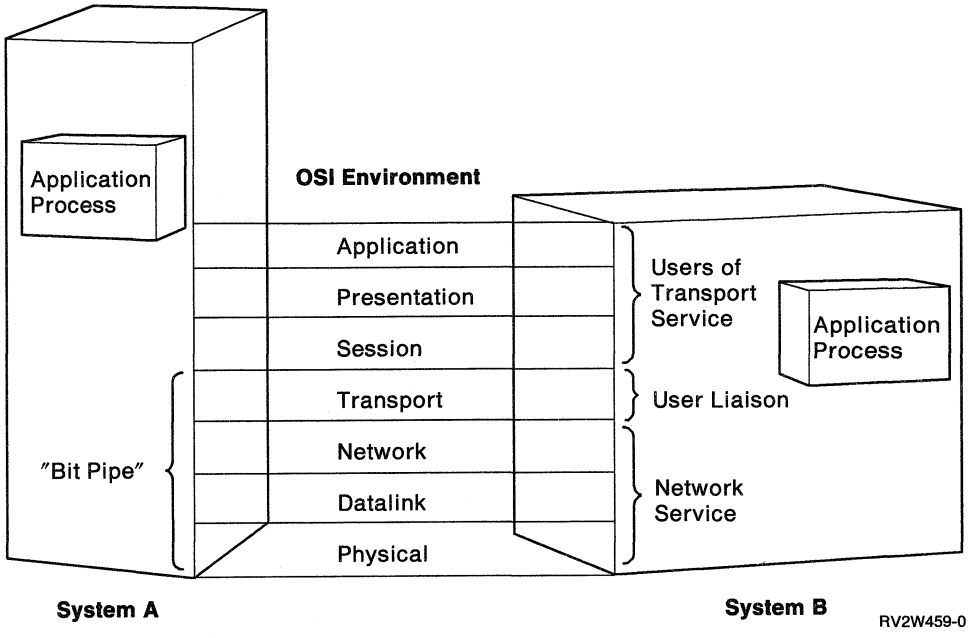


Figure 7-8. Open Systems Interconnection Overview

Collectively, the four lower layers are called the *bit pipe*. It is the bit pipe that transfers the information between the communicating end systems. The AS/400 system OSI support consists of the following protocols:

- ACSE (ISO 8650) (CCITT X.227)
- Presentation layer
  - Kernel and ASN.1 (ISO 8823, 8824, and 8825) (CCITT x.226, X.208, and X.209)
  - DBCS (ISO 2022 and T.61)
- Session layer—all functional units for Session Versions 1 and 2 (ISO 8327) (CCITT X.225)
- Transport layer—classes 0,2, and 4 (ISO 8073) (CCITT X.224)
- Network layer:
  - CLNS (ISO 8473) (Internet Protocol)
  - CONS as follows:
    - 1980 X.25 protocols over 1980 and 1984 X.25 networks
    - 1984 X.25 protocols over 1984 X.25 networks (ISO 8878)
- Directory services—supports a subset of (ISO 9594) (CCITT X.500 - CCITT X.521 subset)

TCP/IP

TCP/IP is an industry standard for communications between heterogeneous systems. It is not just the transport device, it also includes standard applications such as:

- FTP
- SMTP
- TELNET

TCP/IP Connectivity Utilities/400 licensed program runs on the OS/400 program and communicates between systems over X.25, Token-Ring, and Ethernet links.

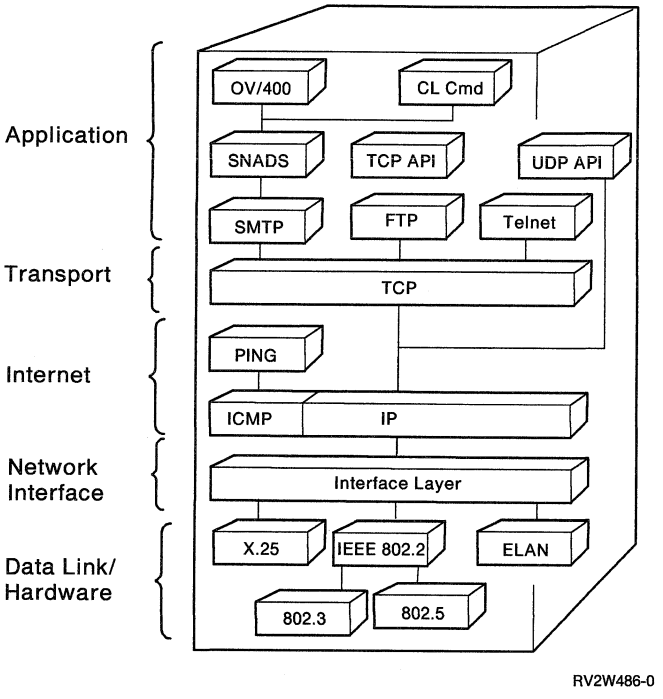


Figure 7-9. Overview of TCP/IP on the AS/400 System

Link Level Connectivity

Link level connectivity ties the communications protocols to the physical layer or hardware. The different communications protocols on the AS/400 system each support a set of data links. Refer to "Communications Protocol Support" on page 7-9 for specific information about each protocol.

Asynchronous: The AS/400 system provides support for a large number of devices, application programs, and services that use the asynchronous communications protocol. The performance of this support depends heavily on the application program or service it is used with and the speed of the line or network used. Asynchronous communications is not compatible with SNA. The following are performance considerations when using asynchronous communications:

- Logical record size
- Buffer size
- Asynchronous communications overhead

BSC: The AS/400 system provides support for the many devices that communicate using BSC. BSC is not compatible with SNA. Consider the following when using BSC:

- Block size

  The AS/400 support for BSC uses block sizes up to 8192 bytes. Usually, the larger the block size, the better the performance. Large block sizes may not work in an environment where the line has a high probability for errors (such as public switched networks). If the blocks need to be transmitted again, larger blocks take longer.

- Blank compression

  BSC data compression and blank truncation significantly reduce the amount of data transmitted on the line when the data contains large numbers of blanks. This reduction can be significant for text and forms data that usually contains many blanks.

- Duplex lines versus half-duplex lines
- BSC overhead

Ethernet: The IEEE used the Ethernet network as the model for the IEEE 802.3 standard. Ethernet (IEEE 802.3) is a local area network with a bus topology that uses the CSMA/CD access method. Most Ethernet networks conform to the following:

- Coaxial cable bus (fiber optic cable can be used)
- Data rate of 10 million bits-per-second
- Maximum length of 500 meters per segment
- Maximum network length of 2500 meters when segments are connected by repeater sets
- Maximum of 100 stations per segment

The AS/400 system can be connected to an Ethernet network through an optional communications subsystem or adapter as follows:

9406 System Unit	Ethernet Local Area Network Subsystem (Feature 6250)
9404 System Unit	Ethernet Network Adapter (Feature 2625)
9402 System Unit	Ethernet Network Adapter (Feature 2625)

Features 6250 and 2625 conform to IEEE 802.3 standards, and can connect to any Ethernet network that follow the standards.

The AS/400 system can be connected to an Ethernet Version 2 standard through the TCP/IP set of protocols.

Some factors that affect the performance of Ethernet networks are:

- Bus configuration
- Data volume
- Frame size

ISDN: ISDNs are high-speed all-digital networks that support circuit switching and packet-switched network technologies. ISDNs combine voice, data, and image communications into a single physical interface.

Two configuration objects that are new for ISDN have been added to the AS/400 system. The *network interface description* represents the physical connection to the ISDN. The *connection list* simplifies the physical interface to the ISDN. The con-

nection list simplifies the set-up and termination of the switched connections when an ISDN network is being used by the AS/400 system.

ISDN networks provide advanced functions that cannot be obtained through other technologies, including a program interface to ICF.

CPID and DNI are services available through ISDN networks presented to AS/400 applications through a programming interface to ICF applications. These call attributes provide capabilities for routing to unique applications and for improved security in the network.

ISDN uses a data link control protocol called IDLC that supports SNA on the AS/400 system. This protocol uses a duplex interface to the network allowing the system to use the duplex capability of the ISDN transmission facility for increased throughput characteristics. ISDN basic rate interface support is integrated into the OS/400 program.

SDLC: SDLC is a communications protocol that conforms to subsets of ANSI ADCCP and ISO HDLC. It is used for sending and receiving synchronous, code-transparent, serial-by-bit data. These transmissions may be duplex or half-duplex over switched or nonswitched lines. The configuration may be point-to-point, multipoint, or loop.

Performance considerations when using SDLC include:

- Polling
- Connection considerations
- Overhead
  - Zero-bit-insertion
  - Control characters
  - Data acknowledgment

Token-Ring Network: A token-ring network is a local area network that conforms to IEEE 802.5, which allows all members on the local area network to exchange data using a token-passing scheme. When a sending system on the network receives the token, it gets control of the line to send data.

Twinaxial Data Link Control: Twinaxial data link control is a function that enables personal computers attached to the work station controller by twinaxial cable to use APPC or APPN.

X.21: The AS/400 system supports SNA, OSI, and TCP/IP data communications through digital networks conforming to CCITT Recommendation X.21. Both non-switched and circuit switched operation is available. For circuit switched facilities, the short hold mode (SHM) of operation with multiple port sharing (MPS) is supported.

X.25: The AS/400 system supports SNA, OSI, and TCP/IP data communications over packet-switched networks conforming to CCITT Recommendation X.25. X.25 is capable of sending and receiving data at the same time, which provides a significant performance advantage for those application programs that can make use of this feature. The following are performance considerations when using X.25:

- Packet size
- Window size
- Overhead
  - 10 percent more than SDLC
  - Zero-bit-insertion and framing characters
  - Control characters to each frame

Communications Abbreviations and More Information

The table that follows lists and defines all the abbreviations used in this chapter. If there is an explanation of the abbreviation in this chapter, look for it on the page in the "This Chapter" column. If you would like more information on the topic, and there is more information available in an IBM manual, the manual listed in the right column is a good first reference.

Abbreviation	Long Form and Topics	This Chapter	First Reference
ACSE	association control service element	7-4	*OSI Communications Subsystem Programmer's Concepts and Guide*, SL23-0191
ANSI	American National Standards Institute		
API	application program interface		
APPC	advanced program-to-program communications	7-11	*Communications: Advanced Program-to-Program Communications Programmer's Guide*, SC41-8189
APPN	advanced peer-to-peer networking	7-10	*Communications: Advanced Peer-to-Peer Networking Guide*, SC41-8188
BTAM	basic telecommunications access method		*Communications: Operating System/400* Communications Configuration Reference*, SC41-0001
BSC	binary synchronous communications	7-15	• *Network Planning Guide*, GC41-9861 • *Communications: Operating System/400* Communications Configuration Reference*, SC41-0001
BSCEL	binary synchronous communications equivalence link	7-10	*Communications: BSC Equivalence Link Programmer's Guide*, SC41-9593
CCITT	International Telephone and Telegraph Consultative Committee		
CICS/VS	Customer Information Control System for Virtual Storage		*Communications: Operating System/400* Communications Configuration Reference*, SC41-0001
CLNS	connectionless-mode network service	7-12	
CONS	connection-mode network service	7-12	

Abbreviation	Long Form and Topics	This Chapter	First Reference
CPI-C	common programming interface - communications	7-3	• *Communications: Advanced Program-to-Program Communications Programmer's Guide*, SC41-8189 • *Systems Application Architecture: Common Programming Interface Communications Reference*, SC26-4399
CPID	calling party identifier		*Communications: X.25 Network Guide*, SC41-0005
CSMA/CD	Carrier Sense Multiple Access with Collision Detection		*Communications: Local Area Network Guide*, SC41-0004
DBCS	Double Byte Character Set		
DDM	Distributed Data Management	7-2	*Distributed Data Management Guide*, SC41-9600
DHCF	distributed host command facility	7-8	*Communications: Remote Work Station Guide*, SC41-0002
DIA	Document Interchange Architecture		
DNI	dialed number identification service		
DOS/VSE	Disk Operating System/Virtual Storage Extended		
DSNX	distributed systems node executive	7-5	*Communications and Systems Management Guide (Alerts and Distributed Systems Node Executive)*, SC41-9661
EDX/CF	Event Driven Executive/Communications Facility		
FFT	final-form text		
FTP	File Transfer Protocol	7-5	*Transmission Control Protocol/Internet Protocol Guide*, SC41-9875
HCF	Host Command Facility		*Communications: Remote Work Station Guide*, SC41-0002
HDLC	high-level data link control		
ICF	intersystem communications function	7-3	*Communications: Intersystem Communications Function Programmer's Guide*, SC41-9590
IDLC	ISDN data link control	7-15	*Communications: Integrated Services Digital Network Guide*, SC41-0003
IEEE	Institute of Electrical and Electronics Engineers		
IMS/VS	Information Management System for Virtual Storage		*Communications: Operating System/400* Communications Configuration Reference*, SC41-0001
IP	Internet Protocol	7-13	*Transmission Control Protocol/Internet Protocol Guide*, SC41-9875
ISO	International Organization for Standardization		
ISDN	integrated services digital network	7-15	*Communications: Integrated Services Digital Network Guide*, SC41-0003

Abbreviation	Long Form and Topics	This Chapter	First Reference
ITF	interactive terminal facility	7-7	*Communications: Asynchronous Communications Programmer's Guide*, SC41-9592
JES2	Job Entry Subsystem 2		
JES3	Job Entry Subsystem 3		
LU	logical unit		
MVS/SP	Multiple Virtual Storage/System Product		
NCP	Network Control Program		*Communications: Operating System/400* Communications Configuration Reference*, SC41-0001
NetView DM	NetView Distribution Manager		
NWS	nonprogrammable work station		
OSI	open systems interconnection	7-4 7-12	*OSI Communications Subsystem Programmer's Concepts and Guide*, SL23-0191
OSI-ACSE	OSI association control service element	7-4	*OSI Communications Subsystem Programmer's Concepts and Guide*, SL23-0191
OSI-CS	OSI Communications Subsystem/400	7-12	*OSI Communications Subsystem Programmer's Concepts and Guide*, SL23-0191
OS/VS1 RES	Operating System/Virtual Storage 1 remote entry services		
PWS	programmable work station		
PU	physical unit		
RFC	request for comments		*Transmission Control Protocol/Internet Protocol Guide*, SC41-9875
RJE	remote job entry	7-6	*Communications: Remote Job Entry User's Guide and Reference*, SC09-1168
RPS/CM2	Real time Programming System/Communications Manager 2		
SDLC	synchronous data link control	7-16	• *Network Planning Guide*, GC41-9861 • *Communications: Operating System/400* Communications Configuration Reference*, SC41-0001
SMTP	Simple Mail Transfer Protocol	7-6	*Transmission Control Protocol/Internet Protocol Guide*, SC41-9875
SNA	System Network Architecture		
SNADS	SNA distribution services	7-4	*Network Planning Guide*, GC41-9861
SNUF	SNA upline facility	7-11	*Communications: SNA Upline Facility Programmer's Guide*, SC41-9594
SRJE	remote job entry		
TCAM	telecommunications access method		
TCP	Transmission Control Protocol	7-3	*Transmission Control Protocol/Internet Protocol Guide*, SC41-9875

Abbreviation	Long Form and Topics	This Chapter	First Reference
TCP/IP	TCP/IP Connectivity Utilities/400	7-3	*Transmission Control Protocol/Internet Protocol Guide*, SC41-9875
TELNET	terminal emulation protocol	7-7	*Transmission Control Protocol/Internet Protocol Guide*, SC41-9875
TSO/VS	time sharing option for virtual storage		
UDP	User Datagram Protocol		*Transmission Control Protocol/Internet Protocol Guide*, SC41-9875
VM/MVS	Virtual Machine/Multiple Virtual Storage		
VM/SP CMS	Virtual Machine/System Product Conversation Monitor System		
VM/SP RSCS	Virtual Machine/System Product Remote Spooling Communications Subsystem		
VSE/POWER	Virtual Storage Extended Priority Output Writers Execution Processor		
VTAM	Virtual Telecommunication Access Method		
VT API	Virtual Terminal Application Program Interface	7-9	System Programming Interface References
WSF	work station function		
3270 DE	SNA 3270 Device Emulation	7-7	*Communications: 3270 Device Emulation Guide*, SC41-9602
3270 RA	SNA 3270 Remote Attach	7-8	*Communications: Remote Work Station Guide*, SC41-0002

Chapter 8. Cooperative Processing Enablers

Cooperative processing refers to a personal computer running either OS/2 Extended Edition or DOS operating system and a host (AS/400 system) working together (cooperating) to accomplish a single task. The work of the application that is performing the task is divided between the two systems to take advantage of the unique strengths of each, thus providing a more effective total systems solution than could be provided on a single system.

The AS/400 system and personal computers in an organization can complement each other to provide a productive and cost effective solution for the business. The AS/400 system enhances the personal computer environment with advanced functions and applications. It provides support for an increasingly wide range of work stations, and can provide many options for connecting personal computer networks. It offers high availability, ease of use, network management, and a broad base of applications.

The PC Support/400 licensed program that provides integration of personal systems and personal computers to the AS/400 system is integral to cooperative processing on the AS/400 system. It provides personal computer users with access to data, programs, and resources on any AS/400 system in the network. PC Support/400 can be used for applications such as data transfer, file serving, print serving, and communications management. It can also be used for as personal computer programs such as spreadsheets, image, and graphics. Additionally, a window capability, including mouse support, is available for DOS-based systems using AS/400 emulation sessions. OS/2 EE Presentation Manager* on Personal System/2* (PS/2*) computers provides mouse and windowing support within the operating system.

PC Support/400 users can connect a personal computer running DOS or the OS/2 EE program to the system as a work station. The connection can be a local or remote twinaxial connection, a connection from a token-ring network, a local or remote asynchronous connection (DOS operating system only), a synchronous data link control (SDLC), or an X.25 (OS/2 operating system only) communications line.

The PC Support/400 licensed program collects many cooperative processing functions into a single product. With this product you can:

- Manage communication between a personal computer and the AS/400 system
- Use the work station function to directly access one or more AS/400 systems from a personal computer to manage multiple work sessions, and to enable the personal computer printer to serve as an AS/400 system printer
- Store information on the AS/400 system, and share the information with other users connected to the system
- Transfer data from the AS/400 system to the personal computer, and from the personal computer to the AS/400 system
- Use printers attached to the AS/400 system as though they were connected to the personal computer

Another function provided by PC Support is an organizer that provides an integrated view of the whole system.

Most of the PC Support functions are independent of each other. One function or any combination of the functions provided by the program can be used to satisfy

individual data processing needs of the users. For example, PC Support/400 can be used to access AS/400 printers to print data generated by personal computer applications or to transfer data from the AS/400 system to the personal computer, process it with another personal computer application, and then print a report on an AS/400 printer.

This chapter briefly discusses the application program interfaces (APIs) available with the PC Support/400 licensed program. These are listed as topics in the table below. If you would like more information on the topic, the manual listed in the right column is a good first reference.

Topics	First Reference
Communications APIs	*PC Support/400: Application Program Interface Reference*, SC41-8254
Multiple sessions support	*PC Support/400: Application Program Interface Reference*, SC41-8254
PC organizer	Depending on the operating system: • *PC Support/400: DOS Installation and Administration Guide*, SC41-0006 • *PC Support/400: DOS Installation and Administration Guide (PS/55)*, SC41-0008 • *PC Support/400: OS/2 Installation and Administration Guide*, SC41-0007 • *PC Support/400: OS/2 Installation and Administration Guide (PS/55)*, SC41-0009
File Server Support	*PC Support/400: Application Program Interface Reference*, SC41-8254
DDM/PC and DDM/PC API	*Distributed Data Management Technical Reference for the Personal Computer*, SC21-9644
PC Support/400 update command interface	Depending on the operating system: • *PC Support/400: DOS Installation and Administration Guide*, SC41-0006 • *PC Support/400: DOS Installation and Administration Guide (PS/55)*, SC41-0008 • *PC Support/400: OS/2 Installation and Administration Guide*, SC41-0007 • *PC Support/400: OS/2 Installation and Administration Guide (PS/55)*, SC41-0009

Program-to-Program Communications

APPC APIs

The AS/400 PC Support router is the key component for establishing communications for the PC Support product. It controls communications between a personal computer and one or more AS/400 systems that are physically connected either locally or remotely by a twinaxial cable, by a token-ring network, by an asynchronous connection (DOS only), by an SDLC, or by an X.25 (OS/2 program only) connection. The router consists of two separate routines; one to start the router connection to systems (STARTRTR) and one to stop the router connection (STOPRTR). Each personal computer can direct requests to any AS/400 system in the network, including up to six AS/400 systems within a token-ring network. Each

personal computer can also concurrently access the resources of 32 AS/400 systems an APPN network.

Note: Under DOS, the communications support is provided by the PC Support/400 router; however, under OS/2 EE, it is provided by the OS/2 communications manager. Both use the router function to manage the communications but the support is slightly different because of the operating system differences.

Figure 8-1 shows the communication support provided for both personal computers running DOS and those running under OS/2 EE.

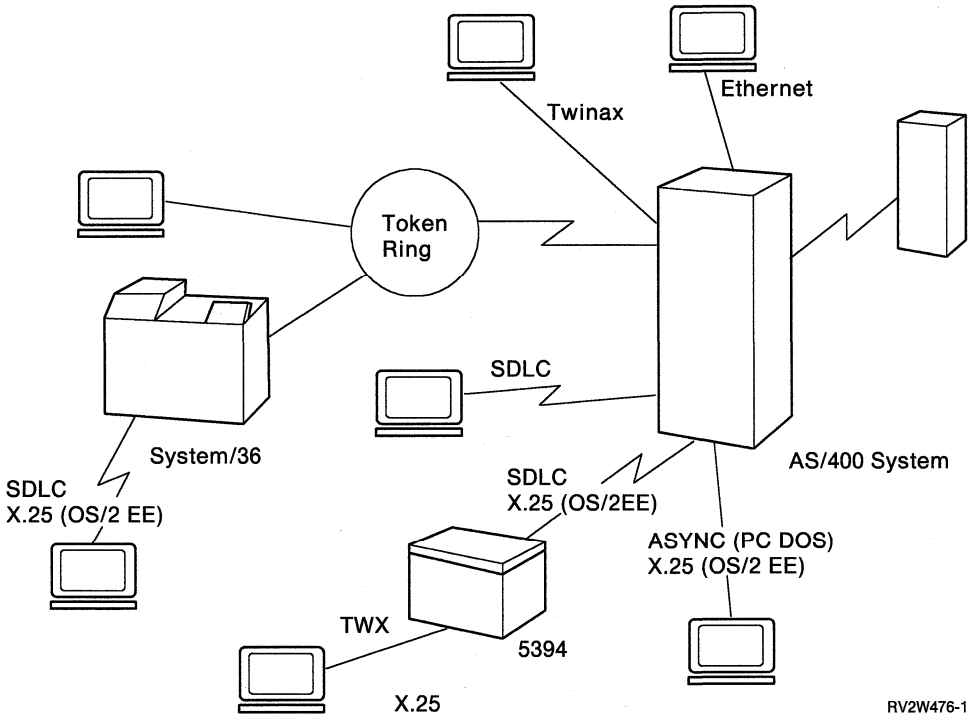


Figure 8-1. Communications Support for Personal Computers

A program using the router API can start the processing of a partner program on another system, but a program on another system cannot start a program on the PC.

When users wish to connect (or link) to an AS/400 system, the router looks for linking information in a personal computer *configuration file* in the form of identifiers. The default configuration file (CONFIG.PCS) is automatically created when PC Support/400 is first installed. It can be changed using the PC Support/400 configurator. Each *identifier* contains different information about how the router should set up the link. Some identifiers specify which system the personal computer is physically connected to, and others specify the type of hardware or communication method to be used. The router makes all the links specified in the configuration file.

The identifier that specifies the system link determines which system the personal computer is going to connect to. This includes both local and remote systems. The configuration file can contain one or several system link identifiers. By putting several system link identifiers in one configuration file, the personal computer user can quickly gain access to several AS/400 systems. If multiple system link identifiers are used in one file, the first one listed determines the default system. Being able to link to several systems at the same time allows users to access data stored

on different systems without having to end a connection each time they switch systems.

The router command can be added to a batch file on the personal computer, or controlled by the users when they wish to connect to a system. If the command is added to the autoexec batch file on the DOS version of the personal computer, the system (or systems) specified in the configuration file are automatically connected each time the user turns on the personal computer. Additionally, users can program a function key which can be used like a *hot key* to start or stop the router.

Submit Remote Command

Using the submit remote command, a PC Support/400 user can send single or multiple AS/400 control language (CL) commands to any AS/400 system in the network. The user does not need to be logged onto the AS/400 system. The submit remote command can also be used to run programs on the host system independently of what is running on the personal computer. No data is sent back from the AS/400 system on completion; only messages about the success or failure of the processing of the CL command or program are returned.

The Submit Remote command function is designed to easily submit noninteractive CL programs on a target AS/400 system. Because the commands are noninteractive, a display session is not required. One way to do this is to submit a single remote command (CL statement) from a PC command line prompt; another is to provide a batch process so multiple submissions of remote CL statements can be done with a single PC command.

Start PC Command

The Start PC command runs on the AS/400 system and is a complementary function to the Submit Remote command. It allows AS/400 applications to run PC commands and programs. The Start PC command is a function of the PC Support/400 organizer, and thus requires an active emulation session to perform its function. With the Start PC command, neither completion messages nor data are returned to the AS/400 application.

Message Commands

The message commands provide the user with the ability to send and receive messages from your PC without using a work station function display session. Messages can be sent and received between personal computer users and other display stations located within the network. These messages can be put on a message queue or, through the configuration parameters, can be specified to be displayed immediately.

Data Queue API

PC Support provides an API that allows OS/2 applications to work with AS/400 data queues. (This API is not supported in the DOS environment.) The data queue API allows high-level, connectionless, asynchronous interprocess communication between the personal computer and the AS/400 system. OS/2 applications can place data on an AS/400 data queue and receive records from the queue, thus allowing AS/400 and OS/2 applications to exchange data without having to be concerned about communications protocols.

Multiple Session Support

Work Station Function

The PC Support/400 work station function allows the personal computer to emulate the functions of an AS/400 system display station, printer, or graphics work station. Each emulated device is called a *session*. Each session requires a unique work station ID on the AS/400 system. The user can have all of the sessions running on one AS/400 system, or have each session running on separate AS/400 systems on the same network. The work station function allows the user to start up to five sessions at once. This could be multiple printer sessions, different display stations on multiple systems, or a combination of any of the session types allowed.

Note: As with the communications support, multiple sessions are provided by PC Support/400 for DOS; multiple sessions for the OS/2 program are part of the OS/2 Extended Edition operating system. Both use the work station function (called work station feature in the OS/2 EE program) to manage the sessions, but the support is slightly different because of the operating system differences.

Figure 8-2 shows an example of the multiple sessions available to the PC Support/400 personal computer user. In the example, the personal computer user could be running an OS/2 application on the PS/2 work station, checking electronic mail on the local AS/400 system, uploading data to the remote AS/400 system, and printing reports on both the personal computer and host system printers.

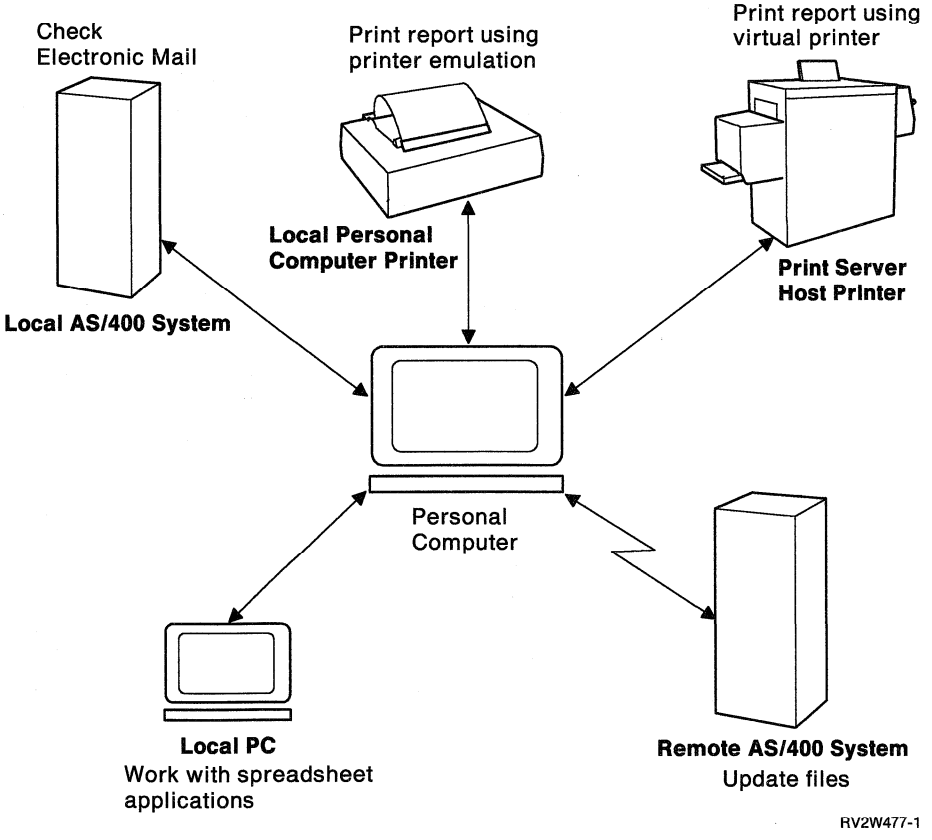


Figure 8-2. PC Support/400 User with Multiple Sessions

By using several sessions at the same time, a user can process several different jobs at the same time. For example, if the user has two reports to print and some interactive entry to do on the system, this can all be done at the same time. By setting up two printer sessions the user can print the documents, and by setting up a display session the user can do the entry work on the system. Moving between sessions is an easy key sequence, and changing sessions does not end the other sessions. If a program to process data and print a report is started on the local AS/400 system, the processing continues even though the user leaves the local session to begin entering data in a program running in another session on a remote AS/400 system. However, because of the single processing nature of DOS, leaving a stand-alone personal computer session will cause the personal computer application currently running to pause. When the user returns to the interactive personal computer session, the application resumes at the point of interruption. Applications running under the OS/2 EE program continue to process when the user leaves the session.

Not only does the work station function allow the personal computer display to emulate the work station display being used, but also allows the keyboard to function like the keyboard of the work station being emulated. This allows system users to use a personal computer as an AS/400 work station. Users can assign additional characters or functions to keys on the keyboard to customize the emulation.

Note: If more than one session is active, the same keyboard arrangement is used for all display and printer sessions.

Under DOS, sessions can be managed through the session manager windowing support, which can span multiple systems. Each session window can be dynamically sized, maximized, minimized, moved, or scrolled using either the mouse or keyboard. Text from any windowing session can be marked and copied to any other active session (called the *target session*), including to and from DOS sessions. As with other PC Support/400 functions, the windowing environment can be customized and incorporated into the initial start-up for each user.

PC Support/400 provides additional support for DOS personal computers that may be affected by main storage limitations. Using the PC Support/400 command interface, the user can control DOS main storage management. The user can remove the virtual printer, work station function, session management, message function, and some of the shared folder function support from DOS storage. The user can then run other applications that may require extensive amounts of main storage to run on the personal computer. Additionally, if expanded memory support is available in the personal computer, PC Support/400 can use the buffers for virtual printer, work station function, session manager, and shared folders functions. This leaves the remaining storage available for users' applications.

PC Support Organizer

The PC Support organizer menu is a tool for integrating applications on the personal computer. As part of the work station function, it provides easy access to AS/400 system tasks and applications, to PC Support/400 functions, to personal computer commands, and to OfficeVision/400 applications, all from a single menu. Figure 8-3 on page 8-7 shows a sample of a PC Support organizer menu.

```
PCCOMNU                AS/400 PC SUPPORT ORGANIZER

 Select one of the following:

   Perform Office Functions
     1. OfficeVision/400
     2. Work with documents in folders
     3. Select editor of choice

   Use PC Support/400
     4. PC Support/400
     5. Host system tasks for PC Support/400
     6. PC command prompt
     7. Start a PC command
    90. Sign off


Selection or command
===> _____________________________________________
_____________________________________________
F3=Exit  F4=Prompt  F9=Retrieve  F12=Cancel
F13=User support   F16=Main menu
```

Figure 8-3. Sample PC Support Organizer Menu

Menu options can be customized to include those functions most frequently used. By having all the needed options available on one menu, the user can access any function without using commands or remembering whether the function is available on the personal computer or on the AS/400 system.

File Server Support

As a file server, the AS/400 system provides personal computer users with increased storage and shared resources, as well as the AS/400 security functions, network capabilities, and change management functions. The PC Support/400 controls these options through shared folders, file transfer, and virtual print functions.

Shared Folders

The PC Support/400 *shared folders* function allows personal computer users to store personal computer data, programs, and documents in AS/400 folders. Access to personal computer programs from the shared folders is transparent to the user. PC Support/400 handles both the OS/2 program and DOS file commands, byte-level locking, upload and download capability, and full sharing between personal computer users and other users in the network. Users can create folders directly from the PC Support/400 organizer menu. Folders and files are named based on the personal computer naming conventions. (Names should conform to AS/400 naming conventions. Unusual characters should not be used.) An entire personal computer subdirectory can be copied to the AS/400 system as a folder.

Storing data, programs, or documents on the AS/400 system allows the personal computer user to take advantage of the large storage capacity available on the AS/400 system. Other AS/400 users can also access data from the folders. The data or programs can also be kept at the latest level because all users in the network with the necessary authority can access the same information.

Individual folders can be accessed by several users at the same time. To maintain the integrity of the information in the folders shared by several users, the shared folders function only allows the first user to access and make changes to the docu-

ment or data. For example, if two users were to access the spreadsheet folder they could both use the spreadsheet program, but only one person could work with an individual work sheet at a time. When the first user releases the work sheet to begin entering data into another work sheet or to run another program, the work sheet becomes available to other users.

To the personal computer user, an AS/400 folder is similar to using a hard drive on the personal computer. To use the folder, users either assign a drive letter to a specific folder on the AS/400 system, or assign one drive on the personal computer to all the folders that they have authority to use on a single AS/400 system. Up to eight personal computer drives can be assigned anywhere in the network.

Access to individual folders is accomplished through a command on the personal computer. Accessing a folder within a folder is the same as accessing a subdirectory on a personal computer hard disk. When the user has accessed the specified folder, the programs can be run as if that folder resided on the personal computer. The command can be included in the path command in the autoexec batch (AUTOEXEC.BAT) file on the personal computer.

Two new functions in PC Support assist users in the use of shared folders.

Check-Out and Check-In API

Check-out/check-in can help to maintain the integrity of the information in the folders shared by several users. It allows users (or PC applications) to:

- Check-out a file in a folder so no other user can change that file until it is checked-in
- Optionally copy a file to the personal computer or to another folder and back
- List files, showing who has them checked out

Hierarchical File System API

The hierarchical file system (HFS) API allows AS/400 applications to access AS/400 folders and their contents. AS/400 applications can use functions to share folder access and data with the PC Support shared folders function. The HFS API provides AS/400 applications with functions similar to functions found in the OS/2 file system. AS/400 applications, using the HFS API, PC applications, and shared folders function can combine processing capabilities to provide you with an interpreted application solution.

File Transfer API

While the shared folders function allows personal computer users to access personal computer data stored in AS/400 documents, there may be times when the users wish to transfer data from AS/400 database or source files to their personal computers. The PC Support/400 *transfer* function allows users to transfer files from the personal computer to the AS/400, or from the AS/400 to the personal computer. The transfer function is used as an application program interface (API) to upload data to be stored in AS/400 databases from the personal computer. (After the data is stored on the AS/400 system, this data can be accessed by all users on the system, not just those with PC Support/400.) Information from the host AS/400 system can be sent to a personal computer file, display, or printer. To complete these transfers, four programs are provided, two are interactive, two are automatic.

- Interactive transfer programs use displays and prompts to lead the user through creating, changing, running, or recalling a transfer request. One-time transfer requests, and creating or changing automatic transfer requests, are usually

carried out using these programs. One program transfers data to the personal computer and the other transfers data to the host system.

- Automatic transfer programs use the saved requests created with the interactive program as their source of information. As with the interactive programs, one transfers data to the personal computer and the other transfers data to the host system.

While the most common type of transfer is the transfer of an entire file, other transfers can also occur. Users can select records from a single file, individual fields from the records, or summary information from a single file. Transfers from the AS/400 system can be sent to the personal computer display, printer, or main storage. For example, two files could be combined and transferred as one file, all records from one or more files that have a common field could be transferred as a separate file to the personal computer, or just summary information about the common records could be transferred. This allows users to pull data that they need from a number of files on the AS/400 system and put it in a personal computer file to create the reports or documents they need. When transferring data from a personal computer file to an AS/400 file, however, the entire personal computer file must be transferred.

DDM/PC (DOS)

Distributed Data Management for the IBM Personal Computer System (DDM/PC) allows application programs running on DOS systems, called source systems, to access and manage files residing on other systems, called target systems. Using the DDM licensed program for personal computers as an extension of the transfer function, data residing on remote systems can be transferred to and from personal computers.

The access to database files using DDM/PC is record-level access. With DDM/PC, an application program running on the personal computer may retrieve, add, update, and delete data records from database files that reside on an AS/400 system. This provides concurrent record input and output, as well as data sharing among multiple applications. Furthermore, the application program interface links to DDM files on the host system and sends the requests to the correct system within the network.

The router must be started when you are using the AS/400 system as the target system.

Application programs running on your personal computer can interface with DDM/PC by using a set of DDM/PC functions (the DDM/PC API). This API is a set of file-access and file-management programming functions.

Remote SQL API

This API allows you to run PC applications that issue SQL statements to an AS/400 system. With Remote SQL, a program running under the OS/2 program or DOS and written in any language that can accommodate the Pascal linkage conventions can access SQL or conventional database files on a remote AS/400 system. Using remote SQL, a program can update individual records or groups of records in a database file. Remote SQL works with the appropriate router program (DOS or the OS/2 program) to enable basic program to program communication between the DOS or OS/2 personal computer and the AS/400 system.

Copy PC Document Commands

The copy PC document commands make it easier for AS/400 database data to be accessed by PC applications and PC data to be accessed by AS/400 applications. The commands run on the AS/400 system to copy database files to and from folder documents. ASCII to EBCDIC translation is automatically performed, but no field level transformations are done.

Virtual Print

The *virtual printer* function allows personal computer users to route print jobs to AS/400 system spool queues and printers, in addition to converting data as needed. (In contrast, the *printer emulation* function allows personal computer users to print AS/400 files on the printer attached to their work station.) Personal computer users can assign up to three printers on any system within the network as their virtual printers. As seen in Figure 8-4, these printers could be AS/400 printers attached to a system on the network, personal computer printers with printer emulation attached to another personal computer using PC Support/400 on the network, or any combination of the two.

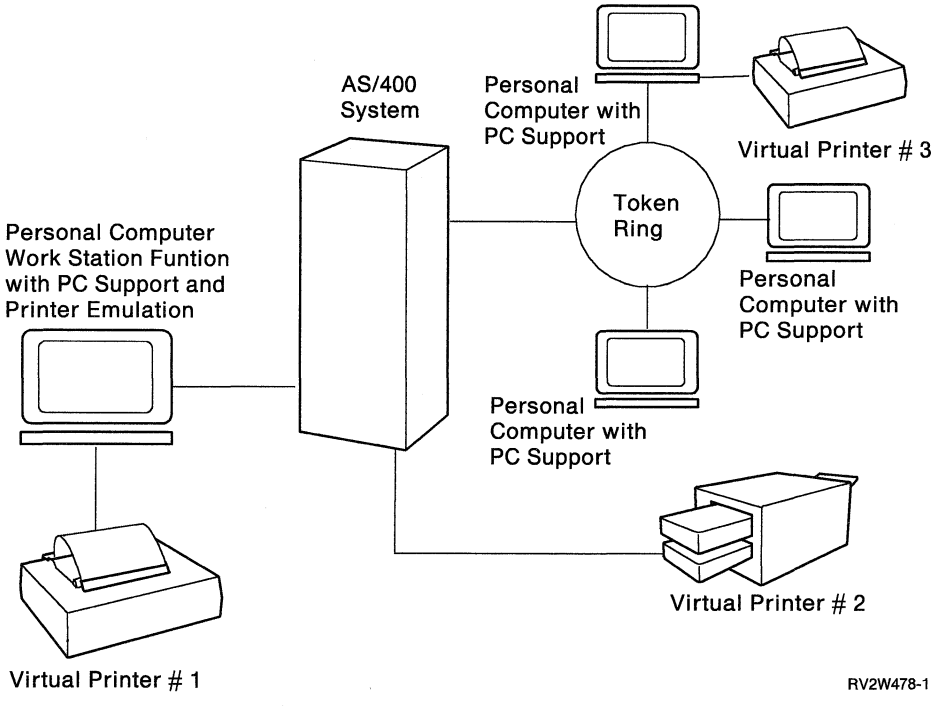


Figure 8-4. Sharing Printers in PC Support/400

With this capability, users can take advantage of the faster speed and better quality of the larger AS/400 printers, personal computer-attached printers, or not require a printer attached to their personal computer.

PC Support/400 gives users two different ways to set up a printer as a virtual printer:

- The interactive virtual printer option allows the user to set up a printer during an individual session. This method of setting up a virtual printer is most often used for printing jobs that would require the use of a printer the user does not normally use. When the personal computer is turned off, any printer defined by the interactive method is lost and must be set up again the next time the personal computer uses a virtual printer.

- The automatic virtual printer option does not need to be specified each time the personal computer is turned on. Virtual printer assignments can be saved in an alternative configuration file and accessed by an automatic virtual printer program. For example, if most of the printing for a particular user is sent to the printer attached to the user's personal computer with final versions sent to the system printer, an alternative configuration file could be created to specify this. Another user, who does not have a printer attached to a personal computer, could choose to use the print identifiers specified in the original PC Support configuration file.

When the command to configure the virtual printer is issued, the program searches the alternative configuration file specified in the alternative configuration file parameter for printer identifiers to process. The *alternative configuration file* is a file that the user sets up to define the printer and the output format to use for a virtual printer. The printer identifiers are what actually contain the information such as printer location and page length among others. If the user does not specify an alternative configuration file, the program searches the PC Support configuration file for printer identifiers specified during the original PC Support configuration.

The configure virtual printer command can be entered as a command or, if the user prefers, included in a batch file such as STARTPCS.BAT. Any virtual printer commands included in a batch file automatically set up the virtual printer specified in the alternative configuration print identifiers each time the personal computer is turned on.

PC Support Update (Command Interface)

The PC Support update function updates existing files on the target directory with the versions of those same files found on the source directory. The date and time of the files in the source directory (folder) are compared with the date and time of the same file names on the target directory. When these dates and times are not the same, the source files is copied over the target files.

One important use of PC Support update is to update the PC Support code in the PC Support subdirectory with the latest code in the PC Support folder, QIWSFLR, on the default system. The PC Support update command is placed in the startup file (STARTPCS.BAT) by the PC Support Installation program.

The user can also use the update function to update user files and programs. Update compares the dates of two versions of the same file name and so can only be used with existing files. It cannot distribute new files.

Chapter 9. Programming Compatibility

The AS/400 system provides several ways to address compatibility issues, including the following:

- Programs and data on systems for which the user is not currently programming
- Application support on systems with a release level different from the development system
- A heritage of programs and data that must be moved from an old system to a new system
- The need to design and program an application that is compatible with a wider range of systems

Compatibility can make it easier to move applications to other systems in the future.

This chapter briefly discusses the programming compatibility topics listed in the following table. If you would like more information on the topic, and there is more information available in an IBM manual, the manual listed in the right column is a good first reference.

Topics	First Reference
Release-to-release compatibility	*Backup and Recovery Guide*, SC41-8079
System/36 environment	*Programming: Concepts and Programmer's Guide for the System/36 Environment*, SC41-9663
System/38 environment	*Programming: System/38 Environment Programmer's Guide and Reference*, SC41-9755
Systems Application Architecture (SAA)	*Systems Application Architecture: An Overview*, GC26-4341

AS/400 System Compatibility, Release-to-Release

Upward Compatible: OS/400 objects are upward compatible. Objects that are created and saved on the current release level can be restored on any future release.

Downward Compatible: The OS/400 program provides one release level of downward compatibility. Objects that are created on the current release level can be saved and restored on the previous release level provided that only previous release function is used. Most OS/400 object-types provide this support including programs (*PGM) and files (*FILE).

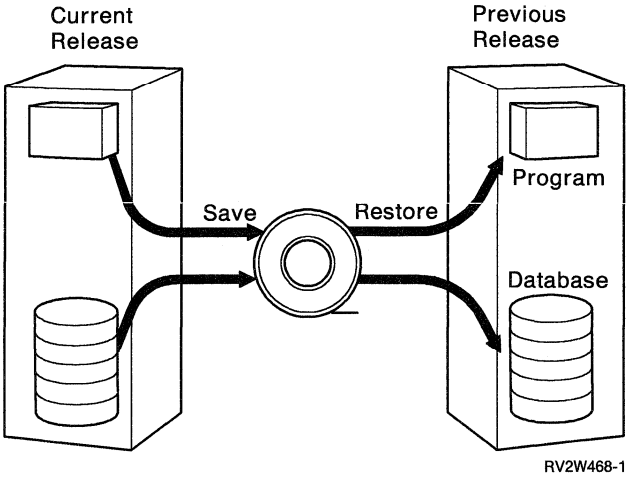


Figure 9-1. Downward Compatible Programs and Files

Downward compatibility is useful to:

- A network enterprise with a central site development system on the current release and with remote sites still on the previous release.
- An application development business with a single system on the current release that supports customers who may still be on the previous release.

System/36 and System/38 Environments

For customers who install an AS/400 system to replace or supplement either a System/36 or System/38, the AS/400 system supports the operating environment for both systems as part of the AS/400 operating system. The operating environments are available from the command line on the display and are available for both interactive and batch jobs, or the System/36 environment can be available automatically when a job starts by specifying an attribute on the user profile. The System/36 and System/38 environments provide several benefits for the move to a new system:

- In many cases, programs written for System/36 and System/38 can be moved to the AS/400 system to run on the new system with minimal change. In many cases, System/36 programs need only be recompiled. Most System/38 programs do not even need to be recompiled.
- Data files moved to the AS/400 system are identified as AS/400 files and can be accessed from either environment or from the OS/400 program.
- The AS/400 system can maintain programs and files for the System/36 and System/38.
- Program conversion, to take advantage of the OS/400 functions, can take place gradually. New parts of an application can use OS/400 functions while other parts migrated from a System/36 or System/38 can remain unchanged.
- Users can function on the AS/400 system in a familiar operating environment until they become accustomed to the new OS/400 menus, entry displays, and commands.
- Compatible, separately orderable, products are available which continue to support the System/36 and System/38 licensed programs.

To help the user migrate to the AS/400 system, a menu-driven Migration Aid leads the user through the migration process, including the migrating steps that occur on

the System/36 or System/38 and those that occur on the AS/400 system. Figure 9-2 on page 9-3 shows how the migration aid works and where the functions take place.

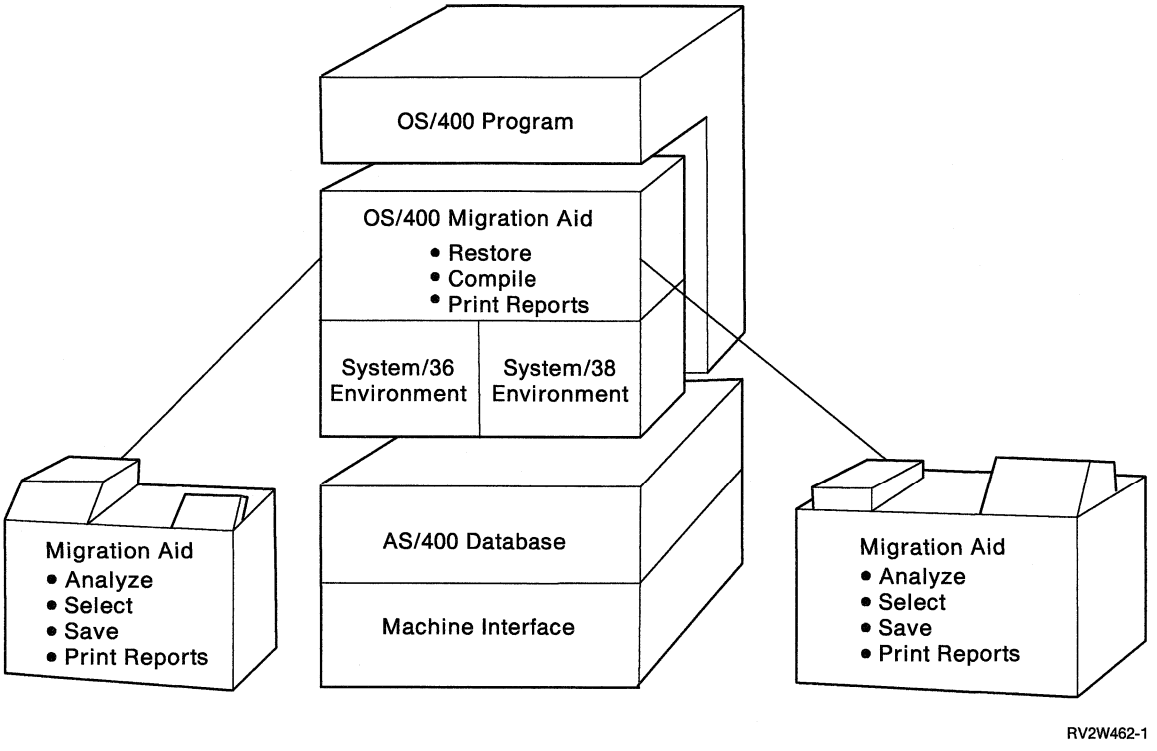


Figure 9-2. How the Migration Aid Works

After the application programs and data are moved to the AS/400 system, the following options are available, but are not mutually exclusive:

- To continue to function as if the programs and data files resided on the System/36 or System/38. The programs and files are accessed from the associated environment.
- To gradually convert programs to OS/400 commands and syntax, and to make all of the necessary changes to the source to operate completely within the OS/400 program.
- To maintain the System/36 or System/38 source to support a network that includes System/36s or System/38s. For example, System/36 or System/38 programs and files would exist within the programming environment, as part of central site programming development or as part of a network of systems including System/36, System/38, AS/400 systems, and possibly other computers.

Any combination of these options can exist within a single system or network. To discuss these subjects, the chapter is divided into the following sections:

- Application Support
  - System/36 Environment
  - System/38 Environment
  - Coexistence
  - Conversion to AS/400 programs
- Compatible products

Application Support

A programming environment includes all the programs, files, and system support required by a program to perform the task. For example, on the AS/400 system, this includes the commands and programs used by the application program, the libraries and database files, as well as, the storage used to process the information. On a completely installed AS/400 system, there are always three environments available, the OS/400 program, the System/36 environment, and the System/38 environment. These environments exist on the AS/400 system and are accessed either interactively or through a batch job. If you do not specify that a job is running under a specific environment, the job is processed in the OS/400 program, using OS/400 CL commands. Within the alternative environments, most System/36 procedures and OCL statements or System/38 commands are supported by the OS/400 program.

Figure 9-3 shows the programming environment support on the AS/400 system. On an AS/400 system, the application can directly access the data management functions of the OS/400 program to manage the data. The environment services of the System/36 environment and the System/38 environment allow users to create System/36 and System/38 applications. Printing, data management, and work management is handled by the environment services which interface to the OS/400 program.

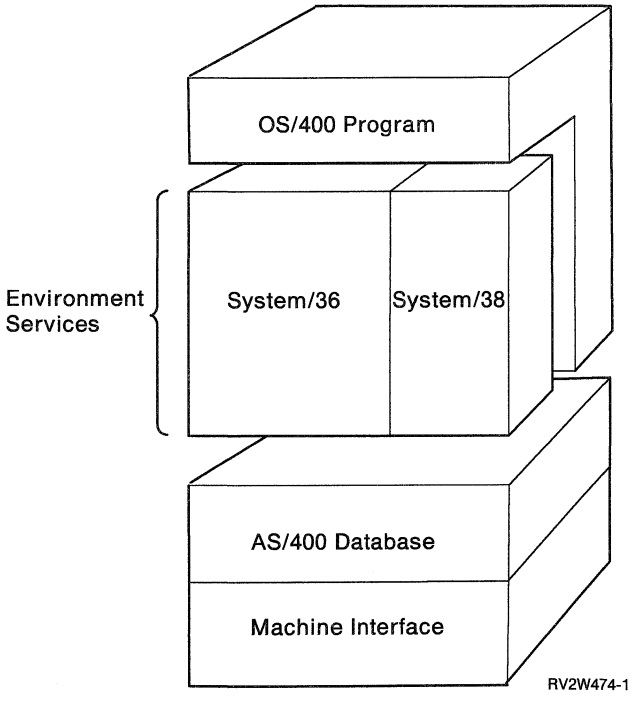


Figure 9-3. Environment Support

Because of the different architecture of the System/36 and System/38, the support provided for the System/36 environment user is more extensive than that provided to the System/38 environment user. For example, a System/38 program can simply be called. However, when accessing System/36 procedures, any of the following methods can be used:

- Use of a user profile attribute that causes a job to automatically enter the System/36 environment (for example, when an interactive user signs on or when a batch job is submitted).
- Use of the Start System/36 Environment (STRS36) command to enter the System/36 environment. A parallel end command (ENDS36) can be issued to end the session with the System/36 environment.
- Use of the Start System/36 Procedure (STRS36PRC) command to enter the System/36 environment and start a procedure. If the user is not in the environment, this command can be used to run the procedure and then return the user to the environment in which the command was issued.

OS/400 commands and functions are always available within either environment. The user profile can be set up to automatically start the System/36 environment as the user signs on to the system.

Within each environment, the AS/400 system supports:

- System/36 operation control language (OCL) procedures
- System/38 Control Program Facility (CPF) control language (CL) commands
- Compiled AS/400 control language (CL)

All environments support compiled CL. Files from System/36 environment, System/38 environment, or the OS/400 program can be accessed from any environment. Figure 9-4 shows how application programs from any environment can use information from any database file since all data is stored in OS/400 data files.

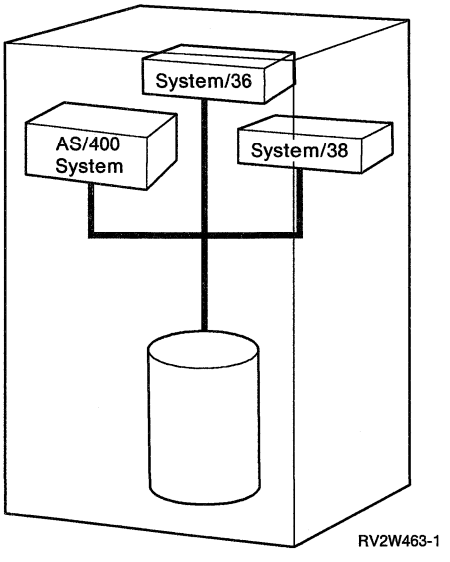


Figure 9-4. Accessing Data Files from Environments

The *mixed environment* capability of the AS/400 operating system allows users to include a mixture of programs and procedures from the various environments. For example, as illustrated in Figure 9-5 on page 9-6, an OS/400 CL program can start the System/36 environment, run a procedure, end that environment, call a System/38 program, and then run another OS/400 application program.

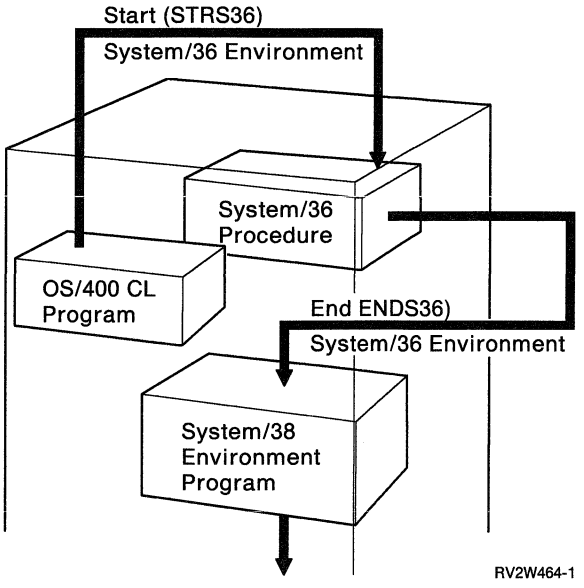


Figure 9-5. Mixed Environment Programming

Operating within the environments allows users to use commands and syntax with which they are most familiar. For example, even though the syntax differs, the OS/400 program recognizes the qualified or unqualified name in any syntax:

Environment	Command	Qualified	Unqualified
System/36 environment	// LOAD // RUN	PGMX,LIBY	PGMX
System/38 environment	CALL	PGM(PGMX.LIBY)	PGM(PGMX)
OS/400 program	CALL	PGM(LIBY/PGMX)	PGM(PGMX)

Some function keys have changed from the way they were assigned on the System/36 and System/38. Each display has an explanation of the function keys available for that task, and they are consistent throughout the OS/400 program. For example, F3 is now *exit* the function and F12 is always *Cancel* in all of the environments. Consistent function keys is part of the strategy for the Systems Application Architecture (SAA) goal for a common user access (CUA) and for greater consistency within the IBM family of computers.

Several command names were changed to provide better consistency such as the use of verbs like STR for start and END for end. In some cases, keywords on commands were dropped or parameter values were changed. Additionally, if the function of the procedure or OCL statement is no longer applicable, it is not supported. For example, in the System/38 environment, the PWRDWNSYS ADRRGN (power down system, address regeneration) parameter was dropped because address regeneration is no longer needed. In the System/36 environment, the compression of the hard disk and condense library functions are no longer required. The system does not report an error if the dropped parameters are specified, but they are ignored.

System/36 Environment

The System/36 environment supports developing and running System/36 applications that use procedures, operation control language (OCL) statements, utilities, menus, messages, and application program interfaces (APIs). The System/36 environment is operating system support that is designed to provide System/36-equivalent function, using underlying OS/400 functions wherever possible. Any OS/400 function can read, update, delete, and rename the migrated objects from a System/36. The user's compiled programs, messages, display formats, data file utility (DFU) programs, files, libraries, and so forth, are all AS/400 objects when being used by the system. Jobs running under the System/36 environment use OS/400 work management functions such as job queues, output queues, subsystems, and so on.

The environment uses the AS/400 data management functions to provide System/36 equivalent support as well as to take advantage of the additional capabilities of the OS/400 program. For example, the OS/400 program allows more files to be open. The System/36 commands call OS/400 functions. The System/36 Operation Control Language (OCL) is comparable to the program-related OS/400 CL commands. However, the System/36 OCL procedures on the System/36 are run in an interpreted manner, as opposed to the compiled CL programs of the AS/400 system. System/36 procedures running in the System/36 environment can include OS/400 program and System/38 environment commands and programs.

Within the System/36 environment, source members are stored as members within a single source file in a library instead of directly within a library. Library searches within the System/36 environment begin with the search order defined for the System/36. If the object is not found, the system searches the AS/400 library list. Members of libraries on the System/36 (procedures or programs) have the same authority as the library in which they exist. However, the added security functions provided by the OS/400 program allow authorities to be assigned individually for each item.

Operating within the System/36 environment, any OS/400 command can be used without ending the System/36 environment.

System/38 Environment

The System/38 environment is put into effect as an attribute of a program or a command. For example, if you run a program that has the System/38 environment attribute, the program uses System/38 command syntax, command definitions, and function. Objects will have a different type depending on which create command was used. For example:

Description	AS/400 System	System/38 Environment
CL Program	CLP	CLP38
Physical file	PF	PF38
RPG III	RPG	RPG38

When restoring objects to the AS/400 system that were saved from the System/38, the object attribute for files (created from source) and programs is changed to reflect the originating system.

In some cases, command and DDS parameters on the AS/400 system have different default values. For example, on SBMJOB (submit job), the default for the INLLIBL (initial library list) parameter in the OS/400 program is *CURRENT, instead of *JOBD, as it is in the System/38 environment.

The Convert to CL Source (CVTCLSRC) command can help in converting most CPF CL to AS/400 CL source. CVTCLSRC reads the commands in a source member and writes the converted commands to another source member but it does not re-create the CL program. CVTCLSRC (or a manual conversion of just command changes) is not a complete conversion because some programs are coded with variables. The converted program would fail when it was run if the system found values which are not valid for parameters on OS/400 commands.

Coexistence

Coexistence for many customers means that the AS/400 system be used in a network with other systems. Many users will continue to use the System/36 or System/38 environments for developing and testing application programs that will run on one or more System/36s or System/38s in the network. This allows them to take advantage of the AS/400 programming productivity features. Applications and data can easily be moved through a network of systems using either the communications or save-and-restore functions.

Data Interchange Between Systems: An AS/400 application program may require data stored on either a System/36 or a System/38 or vice versa. The OS/400 object differs from the System/38 object and the System/36 file, which is usable on a System/36. Consequently, program objects from an AS/400 system cannot be saved and restored onto a System/36 or System/38. A saved object or file from either system can be restored on the AS/400 system and the information is mapped into the AS/400 format.

However, users can save source and data objects from the System/36 environment and, using normal System/36 restore interfaces, restore them onto a System/36. System/38 environment source can be moved to a System/38 using normal data interchange. The System/38 objects can be created from this source.

Figure 9-6 shows how objects and source are saved from a System/38 and restored on the AS/400 system. The AS/400 system automatically converts the source member type to include the System/38 attribute.

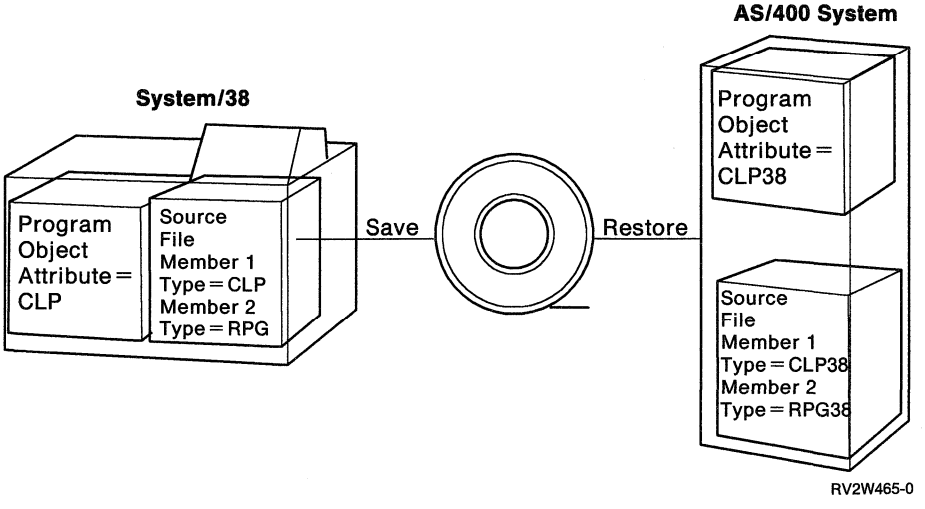


Figure 9-6. Restoring System/38 Files to an AS/400 System

The data interchange format can be used when interchanging data between a System/36 or System/38 and some other system type (for example, a System/370 computer) or when replacing a System/36 or a System/38 with an AS/400 system.

Compatible media and recording density are required on both systems. No changes are necessary in the data interchange format.

Folder Information: Personal computer data on the AS/400 system is stored in the folder object type. No support is available to extract the information from a folder on an AS/400 system and to convert it to a virtual disk for a System/38. A command Copy From a Personal Computer Document (CPYFRMPCD) does exist on the AS/400 system to copy a document from a folder to an AS/400 database file, but the internal format of this file is not the same as that used on the System/38. Folders cannot be migrated from an AS/400 system to a System/36, however, folders can be migrated from a System/36 to an AS/400 system.

PC Support/400 can convert files from System/38 and System/36 to AS/400 folders. This command is available anytime, not only during migration.

Conversion to OS/400 Programs

It is important to understand the difference between running programs in the System/36 or System/38 environment and operating the AS/400 system using only OS/400 commands and functions. Operating the AS/400 system includes such things as using OS/400 CL commands to operate the system and code the programs, using OS/400 displays, and handling messages. To convert fully to the OS/400, you must convert your programs and job streams that are operating in one or both of the environments.

Many file conversions (such as physical files) require only a simple creation from the existing source. In fact, not all files need to be converted. When files are moved to the AS/400 system that were not created directly from source, which includes all files being migrated from the System/36, they are marked as OS/400 files (for example, a physical file, PF). If a file is created using the Create Duplicate Object (CRTDUPOBJ) command, the display object description (DSPOBJD) information of the new file does not reflect the original source file or member. Only the file type, which is automatically tracked by the system, identifies the type (such as PF38) from which it was duplicated.

Many high-level language programs convert to OS/400 programs without requiring extensive changes to the source code. For most System/38 programs, the program may only need to be re-created using the OS/400 create program (for example, to create a CL program, CRTCLPGM) command, which creates the program object description information from the source. However, most System/36 environment programs would require some source code changes to be compiled as an AS/400 program.

Compatible Products

Several products from the System/36 and the System/38 are available for the environments. They can be used to change migrated programs and files or to create and update new ones within the environments. A primary objective was to ensure that the primary application interfaces on the System/36 and System/38 were supported by the environments. Information produced by these applications, and the interfaces seen by the users of these applications, are equivalent to that of the originating system.

For example, support is provided for changing and compiling Report Program Generator (RPG II) objects migrated from a System/36. Additionally, RPG II programs can be developed in the AS/400 System/36 environment to run on an existing System/36. The following is a list of some of the languages supported:

System/38-Compatible COBOL: The System/38 COBOL ANSI 74 level is supported in the System/38 environment. The AS/400 COBOL is ANSI 85. If the source is to be shared between System/38s and AS/400 systems operating within the System/38 environment, it is best to use ANSI 74. When converting to OS/400 programs, some programs may require changes.

System/36-Compatible COBOL: The System/36 environment will now check for a valid range for subscripts used as indexes for arrays. The 64KB limit does not exist.

Data File Utility (DFU): When migrating from a System/36, DFU list programs can only be run in the System/36 environment. DFU enter, update, and inquiry programs can run either in the System/36 environment or in the OS/400 program. However, maintenance of DFU programs migrated from a System/36 must be done in the System/36 environment.

All System/38 DFU functions are supported in the environment.

Query/38: System/38 Query functions are supported in the environment. The Query/400 has several enhancements including:

- Changes and query definition functions are accomplished in the same way
- Online help and index search capability
- Collating sequence support
- Character result fields are created using substring and concatenation
- Additional numeric editing

System/36-Compatible RPG II: In addition to the functions currently available to the System/36 application developer, the System/36-compatible compilers provide the following expanded capabilities:

- Greater than 64KB program size
- Maximum number of arrays increased from 75 to 200
- Ability to call any other high-level language program on the system
- Maximum number of files used by a program increased from 20 to 50

System/38-Compatible RPG III: Not only are the System/38 RPG functions supported in the environment, but the RPG/400 language has several enhancements which run in the System/38 environment including:

- Increased numeric field length from 15 to 30
- Additional I/O feedback for post operation
- New values for some keywords providing additional function

Text Management/38: Documents created using either System/38 Text Management or the System/38 Personal Support Editor can be migrated to OfficeVision/400 documents. Some changes are required in the document and the way in which you work with the document. For example, on System/38, the *type* of printer was specified (such as *5219). On the AS/400 system, the *device name* is given (such as P2). Text Management/38 in the System/38 environment allows users to do word processing tasks such as create, file, revise, and retrieve documents. From the System/38 environment, the user can access the AS/400 database interactively from the program either at editing time or printing time to include data in a text document.

SAA Support on the AS/400 System

Systems Application Architecture (SAA) is a framework in which programs are developed so that they run consistently on major IBM computing systems. The SAA framework includes:

- Consistent programming interfaces
- Consistent end-user interfaces (system and application)
- Connectivity and data interchange across major IBM systems
- Common applications, from IBM and vendors, across major IBM systems

The SAA environments are MVS, VM(CMS), the OS/400 program, and the OS/2 Extended Edition program. MVS includes the base system (TSO/E, APPC/MVS, and batch) and the CICS and IMS subenvironments.

Figure 9-7 shows the scope of SAA.

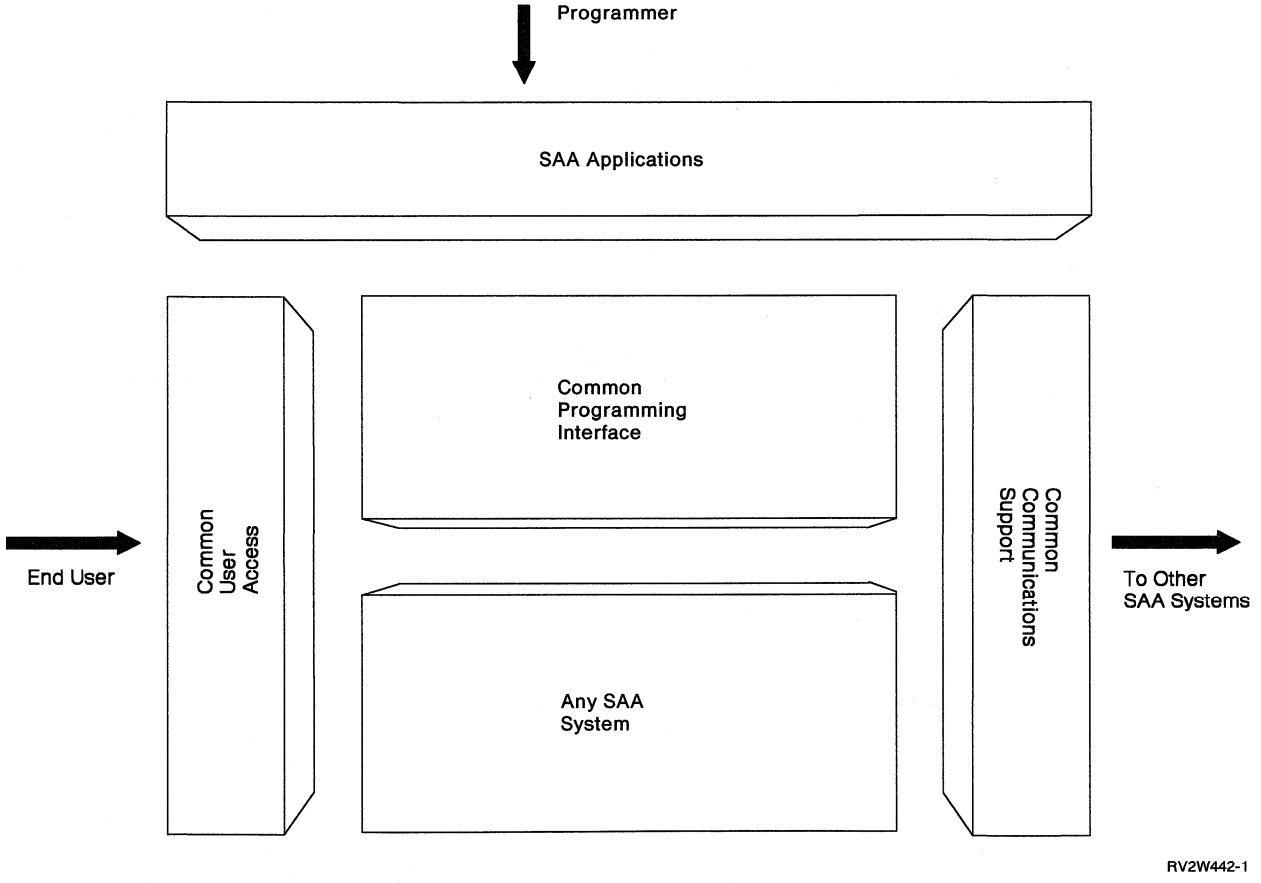


Figure 9-7. Systems Application Architecture

The focus of SAA is for enhanced ease of use and programmer productivity through consistency across applications. The key elements in SAA are:

- Common User Access (CUA)
- Common Communications Support (CCS)
- Common Programming Interface (CPI)

These elements identify the rules and conventions for how the displays look to the user, the common functions supported by the system and application programs, and the portability between operating systems.

Common User Access (CUA)

The SAA Common User Access (CUA) defines the guidelines for the dialog between the human and the computer. It establishes how information appears on a display screen and how people respond to that information. Users moving from application to application or from system to system need less time to adapt. At the most visible level, CUA provides single system image and unity.

Elements of the CUA Interface

The objective of the Common User Access is to provide a user interface that is easy to learn and use, and is also as consistent as possible across a range of devices and application types. To balance the value of consistency with the value of exploiting the advanced capabilities of the programmable work station, CUA defines two interface types: graphical and entry. Where practical, similar base components are used across the models to enhance user transfer of learning.

Graphical Model: This is best suited for the programmable work station running with OS/2 program, and makes extensive use of that system's window capabilities. The graphical model includes action bars and their associated pull-down options, windows for messages and parallel dialogs, such as help. Additionally, it defines the use of standard graphical cues, such as check boxes. It is intended for all programmable work station applications.

- Text is a Graphical model version to support the nonprogrammable display station. It maintains a close similarity to the Graphical model, including action bars and their associated pull-down options and windows. It is intended for decision intensive applications.
- Workplace is an extension of the Graphical model that supports integration of applications into an electronic version of a working environment. For example, Workplace for the office contains such things as mail baskets, file cabinets, telephones and printers, which all applications can share.

  The separate applications can be integrated as objects, which appear as icons. Workplace can use a mouse for moving objects on the screen and providing direct manipulation. Workplace applies only to the programmable work station.

Entry Model: This is best suited for the nonprogrammable display station environment and for applications that have panels with menus and prompts. It is intended for data entry intensive applications on the nonprogrammable display station, and may also be appropriate for such applications on the programmable work station.

Common Communications Support (CCS)

The SAA Common Communications Support (CCS) gives users the ability to connect multiple systems, including work stations, with hosts. It provides architectures and protocols that allow standardized communication among different devices, application programs, systems, and networks. By using CCS, the user can unify information processing across an entire enterprise and manage it as a single entity.

Elements of CCS are grouped into six broad categories. Many of these forms of CCS are based on IBM SNA or OSI international standards. The CCS protocols allow the user to link components, IBM systems, and other systems into one integrated enterprise.

- Object Content Architectures allow data objects (text, graphics, and images) to be created in a standard format and easily transferred among SAA system.
- Data Streams allow programs to generate output destined for a printer, a work station, or another application program elsewhere in the network.
- Application Services enhance the activity of the network by providing architectures that allow data distribution, document interchange, and network management.
- Session Services allow two application programs in the network to establish a dialog, communicate, and transfer data.
- Network Services provide assistance in linking across the network.
- Data Link Controls enable the user to establish communications and transmit data across any combination of telephone lines, microwave beams, fiber optics, satellite links, or coaxial cables, using local area networks, telecommunication links, or packet-switched networks.

In short, CCS provides interconnect and data interchange to SAA application users and applications through SAA interfaces, while also specifying the means by which non-SAA systems can participate in SAA environments.

Common Programming Interface (CPI)

The SAA Common Programming Interface (CPI) provides programmers with functions they can use to more easily develop applications. CPI functions are divided into two categories: languages and services.

Languages: The most widely used programming languages are offered: COBOL, FORTRAN, PL/I, RPG, and C. These languages have proven their value over the years; their strengths are well known and need no explanation. They provide an easy and familiar way to code almost any application.

Also available is an Application Generator (sometimes referred to as a fourth-generation language). Based on the IBM CSP generator, it provides increased productivity by allowing coding specifications that are less detailed, and application creation by personnel with more limited data processing skills.

In addition, a Procedures Language (based on REXX) is provided, which programmers find helpful for a variety of tasks—most frequently in building command strings for system-related procedures.

Services: Services can be called from an application to perform several types of functions: user interaction, database and repository access, and interprogram communications.

Dialog and Presentation interfaces are available to allow an application to easily converse with the user at a programmable work station. The Dialog Interface is especially convenient for writing menu-driven applications. The Presentation Interface provides a full range of screen design and graphics possibilities. Both interfaces provide CUA support.

Access to the IBM relational database is provided through:

- The Database Interface (based on the SQL language) which lets an application read and write information from the database.
- The Query Interface which supports query functions and gives the ability to quickly produce results in a formatted, easy-to-read reports.
- The Resource Recovery Interface which provides applications with a commitment coordination technique for change integrity even in multiple systems.
- The PrintManager Interface which provides a means to request print services throughout the entire enterprise.
- The Communications Interface which permits easier construction of cross-system applications. Through a simple program-to-program call mechanism, various parts of an application can reside in several systems in the enterprise and yet communicate with each other and function as if they are one complete unit.

Some CPI functions are available in both local and distributed forms. For example, I/O statements can access files and relational databases that are on either the local system or systems elsewhere in the network. Programmers can write applications that use all the resources of the enterprise without having to learn a new language or coding techniques.

AS/400 System Participation

The AS/400 system carries out full SAA support. This support will be delivered in stages and is not all available at this time. Support of SAA by the AS/400 licensed programs permits the system to be used as a consistent part of a network of SAA systems. SAA support also allows the AS/400 customer to take advantage of the common SAA applications, training, and skills that exist.

CUA support is provided in the AS/400 licensed program displays. These displays are organized to meet the CUA guidelines so that they provide good usability characteristics and assure that there is consistency in the operator interface across the system and across IBM SAA platforms. The displays that are created for customer applications may conform to the level of CUA supported by the AS/400 licensed programs or to the more advanced Graphic level CUA guidelines.

The CPI elements supported by AS/400 licensed programs include:

- COBOL
- RPG
- Application Generator
- Procedures Language
- FORTRAN
- C

- Communications
- Database
- Query
- Print Manager

Other SAA CPI elements will be added in the future. The AD/Cycle repository interface, for instance, is a SAA CPI and the application development information model will be accessed through this CPI. This means that over time, the AS/400 development environment will support these components. Support of these architectures permits the AS/400 user to create applications that can be easily adapted to other IBM SAA systems, to use the same interfaces and designs across different IBM SAA systems, to obtain and adapt applications created for another IBM SAA system, and to take advantage of the training and skills that may already exist in the use of these application development interfaces.

For example, the application generator interface is supported in the IBM program named Cross System Product (CSP). The definition and generation of an application is performed on an IBM System/370 system using MVS CSP Application Development (CSP/AD) Version 3.2 or later. The new application is then loaded to the AS/400 system and run using the OS/400 CSP Application Execution (CSP/AE) support. In addition to run-time support, the operating system also provides commands and interactive menus that aid in the creation and management of CSP/AE objects on the AS/400 system.

The AD/Cycle application development framework and life cycle tools are intended to assist with the development and maintenance of applications that operate in an IBM SAA environment. Furthermore, IBM and selected vendors will provide initial sets of SAA-compliant application development tools, plus the model of information needed by those tools to support application development.

The SAA CCS elements are supported primarily by the OS/400 licensed program. Some elements are supported by the communications utility programs, OfficeVision/400, PC Support/400, and other licensed programs. Support of the CCS elements assures the ability to connect and exchange data and documents with the other IBM SAA systems (and other systems that support the SAA architectures).

Chapter 10. Work Management

Work management supports the commands and operating system functions necessary to control system operation and the daily work load on the system. It controls resources for applications, such as work stations and storage, so that the system can support multiple applications and system tasks.

All the work done on the system is submitted through the work management functions. When the OS/400 program is installed, it includes a work management environment that supports interactive, batch, and communications jobs. The operating system can be tailored to create an individual, user-defined work management environment.

This chapter briefly discusses the work management topics listed in the following table. If you would like more information on the topic, the manual listed in the right column is a good first reference.

Topics	First Reference
Work management structure	*Programming: Work Management Guide*, SC41-8078
Subsystems	
Jobs	
Performance	

Work Management Structure

Figure 10-1 shows the objects included in and managed by work management. The reference numbers refer to the descriptions following the figure.

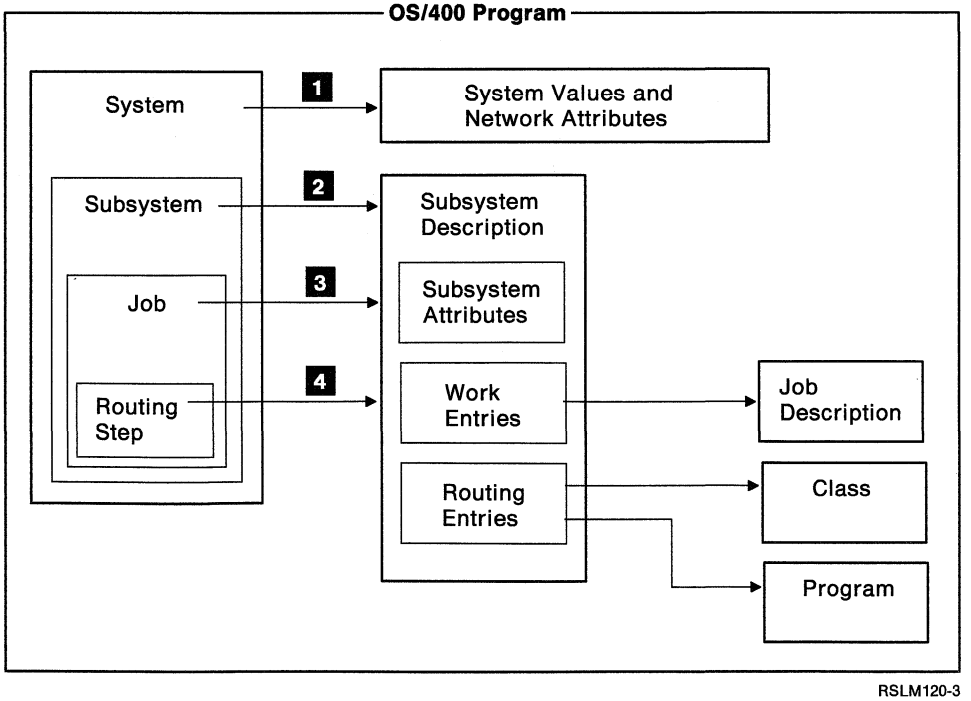


Figure 10-1. Work Management Structure

1. *System values and network attributes* are control values that affect how the system operates. System values are set to the default values when the system is shipped. Examples of system values are date, time, name of the controlling subsystem description, and maximum activity level of the system. Network attributes also apply to the local system but determine how the system works in a network of systems. Network attributes include alert status, system name, and default message queue used for object distribution. These values can be changed by the user to change system operation.

2. A subsystem is a predefined operating environment through which the system coordinates the work flow and use of resources. The system can contain several subsystems, all operating independently of each other. Subsystems share resources. The run-time characteristics of a subsystem are defined in an object called a *subsystem description.* It defines what jobs can run within a subsystem and whether they can be run in batch or interactively. The subsystem is known to the system by the subsystem description name. Depending on their processing needs, users can create their own subsystem descriptions or use the subsystem descriptions supplied by IBM.

3. Each piece of work run in a subsystem is called a *job.* Each job is an identifiable sequence of processing actions that represent use of the system. In a subsystem description, *work entries* identify the source from which jobs can be started for running in that subsystem. The source is where the job came from, for example, the work station or job queue. The subsystem reads the *job description* associated with each job to determine the attributes of the job. This includes the library list, message logging level, job queue, routing data, assigned or default printer, output priority, and user profile.

4. Jobs are processed in a subsystem as one or more consecutive *routing steps,* identified by routing entries in the subsystem description.

Subsystems

Subsystems provide the ability to have as many, or as few, unique operating environments as necessary to meet the processing needs of an installation. The number of subsystems that can be active at one time is limited only by the resources available on the system. Each subsystem can be controlled independently. Although many subsystems can be active at the same time, at least one subsystem is required at all times. Figure 10-2 on page 10-3 shows the flow from the system input device (work station, job queue, communications device) to the work entries in the subsystem through the routing entries in the subsystem to the actual start of the job.

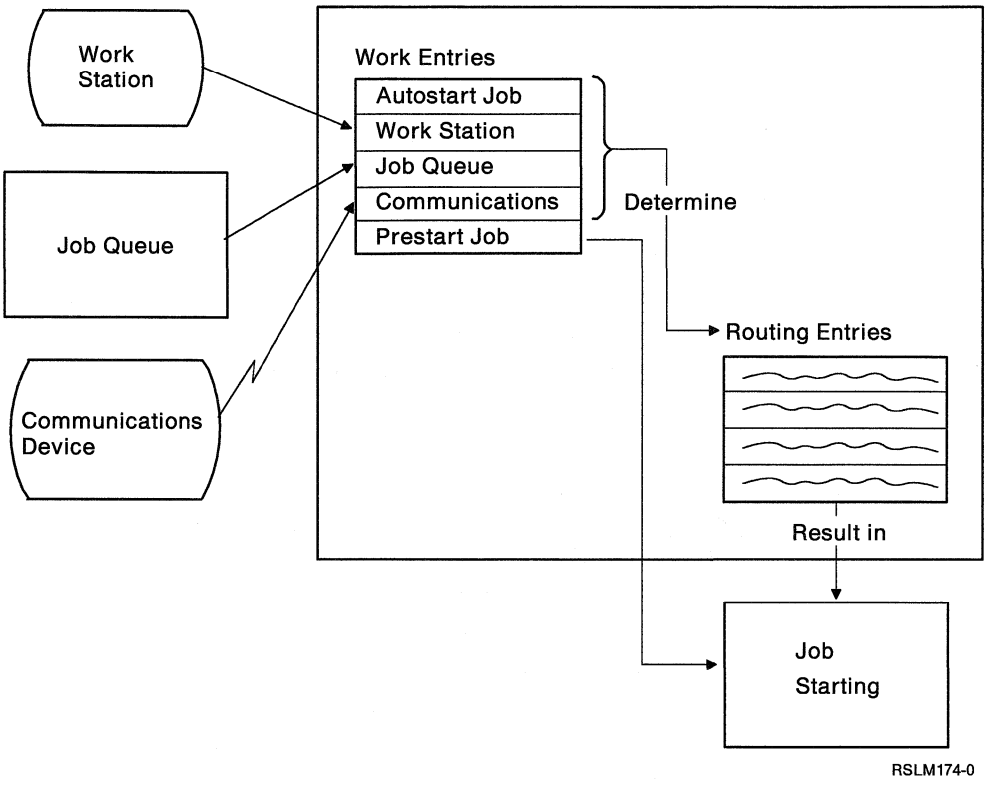


Figure 10-2. Starting a Job

IBM-Supplied Subsystems

In the library QSYS, IBM supplies two complete controlling subsystem configurations, QBASE (the default controlling subsystem) and QCTL. Each can be used as shipped or changed to meet the needs of the individual users. The controlling subsystem starts automatically when the system starts. The controlling subsystem description value specified in the system value (QCTLSBSD) determines which configuration (QBASE or QCTL) the system uses.

The QBASE default configuration allows the user to run all the same functions that the QCTL configuration does and is easier to manage because it consists of fewer subsystems. The following QBASE subsystem descriptions are found in the QSYS library:

- QBASE, supports interactive, batch, and communications jobs. It has an autostart job that automatically starts the QSPL subsystem.
- QSPL, the spooling subsystem, supports reader and writer jobs. Users can change the subsystem to meet the needs of the application.

The QCTL configuration allows more individual control over system operations by dividing the system activity into different subsystems based on the type of activity. If there is a need to extensively change the subsystem configuration or to create a new one, it is easier to use the QCTL configuration as a starting point. For example, if it is necessary to run batch jobs over the weekend or overnight without allowing users to sign on to work stations, that is easily done with the QCTL configuration by ending the QINTER subsystem.

The following QCTL subsystem descriptions are found in the QSYS library:

- QCTL is an autostart job that automatically starts the QSPL, QINTER, QBATCH, and QCMN subsystems
- QINTER has work station entries for all work station types and supports interactive jobs
- QBATCH has job queue entries for the QBATCH job queue and supports batch jobs
- QCMN has communication entries for all communications protocols and supports communications jobs
- QSPL has job queue entries for spooled jobs and supports reader and writer jobs

User-Defined Subsystems

In addition to the subsystems supplied by IBM, the OS/400 program also allows the customer to copy and change existing subsystems or create new ones to support special data processing requirements. For example, unique subsystems might be needed to do the following:

- Control an application that must be continuously available and that must provide a rapid response to its users.
- Provide an operating environment that must be separately controlled. For example, if some work stations are to be used only during a specific period of the day, these work stations could be specified as work entries in a user-defined subsystem that the system operator starts and ends at the same time each day.
- Process different kinds of batch jobs, such as:
  - Long running batch jobs
  - Batch jobs set to run at specific times
  - High priority batch jobs
- Provide specific control over a critical or unique application. By having a separate subsystem for this type of application, the performance and consistency of the application can be controlled more easily.

Subsystem Descriptions

A subsystem description is required to define each subsystem to the system. The OS/400 program uses information contained in the subsystem description to define the environment provided by the subsystem. Subsystem descriptions can be changed or new subsystem descriptions can be created for specific processing needs.

Subsystem descriptions contain the following categories of information:

- Subsystem attributes
- Work entries
- Routing entries
- Communications entries
- Autostart job entries
- Prestart job entries
- Job queue entries

After a subsystem description is created, the subsystem entries can be added to the subsystem description. The subsystem can be started and ended by control language commands.

Subsystem Attributes

The subsystem attributes of the subsystem description contain:

- The storage pool definition determines what amount of main storage is allowed for jobs running in that subsystem. A subsystem can have its own storage pool, or it can share a common storage pool (called the *BASE storage pool) with other subsystems. A storage pool definition within a subsystem also contains an activity level that determines how many jobs can compete for system resources at the same time.
- The name of the sign-on display file and the library where the sign-on display file is stored that the subsystem will use when displaying the sign-on screen.
- The maximum number of jobs that can be active in the subsystem at one time.
- The descriptive text for the subsystem description can also be included, as a reminder for the user.

In addition to the subsystem controlling the number of jobs that can be running at one time, each storage pool also has an activity level associated with it. The storage pool activity level limits the number of jobs in the storage pool that can be competing for the processing unit.

Work Entries

The work entries in a subsystem description specify the sources from which jobs can be selected to run in that subsystem. Each work entry in a subsystem description, except the job queue entry, refers to a job description. The job description for batch jobs is specified by the user or as part of a program when the job is submitted. Figure 10-3 on page 10-6 shows the relationship of work entries in a subsystem description to the other elements within a subsystem.

Job queue entries: Job queue entries specify the name of a job queue from which the subsystem can start batch jobs. Jobs are placed on the job queue when they are read by a spooling reader or submitted to the queue from another job. The job queue does not need a job description as it merely tells the system where to get the work. The job description for the job tells the system what to do.

Autostart job entries: Autostart job entries specify jobs that are to be automatically started when the subsystem is started. The jobs specified as autostart jobs are not started from a work station, so they are processed as batch jobs. Because they are automatically started by the subsystem, however, they are not placed on a job queue.

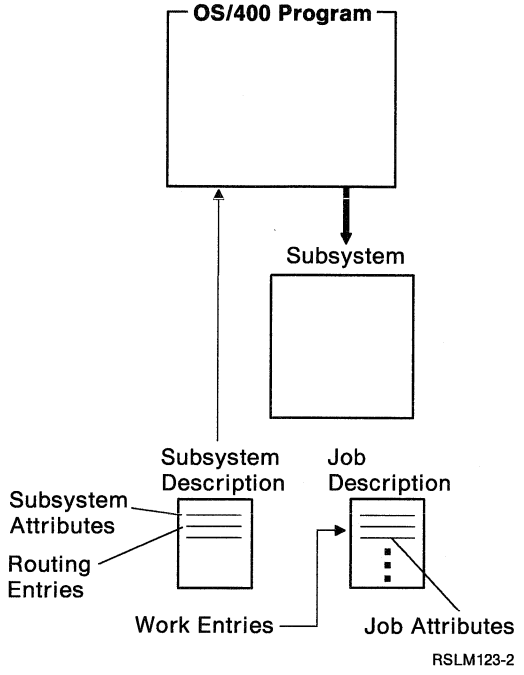


Figure 10-3. Work Entries in a Subsystem Description

Work station entries: Work station entries specify the work stations from which interactive jobs can be started in the subsystem.

Communications entries: A communications entry specifies one or a group of communication device descriptions by name or type, or a remote location name from which communicating batch jobs can be started. The program on the system which is to be started from another system is specified in a routing entry in the subsystem description or in the routing data in the job description.

Prestart job entries: Each prestart job entry identifies the program name that is to be started, the number of jobs that need to be started with the program, and the user profile under which the jobs will be run. The prestart job entries can be used to start a job on the local system before a remote system sends a program start request. By adding the prestart job entry to the subsystem description, the jobs start when the subsystem is started.

Routing Entries

The routing entries in a subsystem description specify the programs to be called, and specify the classes that define the environment of a specific job. The processing performed as a result of calling the program specified in a routing entry is called a routing step. Typically, each job will have only one routing step. Work management establishes the environment for a routing step when the routing step is started. The environment of a job includes the run priority, storage pool identification, the default wait time, as well as other information. The environment can be changed during a job. The program that is called when the job is started controls the processing within a job. That program might call other programs to perform the functions that are required in the job.

The routing entries in a subsystem description form a routing table. When a job is started, the appropriate routing entry is selected by means of routing data. The routing data is extracted from the job description associated with the job, or is specified when a batch job is placed on a job queue. The OS/400 program compares the

routing data with the values in the routing entries to determine which routing entry to select from the routing table.

For example, the subsystem descriptions provided by IBM specify the OS/400 control language processor (program QCMD) in the routing entries. Thus, the control language processor is called when routing steps are started in these subsystems, and work can be submitted through control language commands. To make the system easier to use, the control language processor can then call programs specific to the user who signed on.

Routing steps for batch jobs are started the same as for interactive jobs. That is, the routing data is compared with the routing entries in the subsystem description. When a match is made, the program specified in the routing entry is called and the routing step is started. For the subsystem QBASE, the program QSYS/QCMD is called to process the commands in the job. QSYS/QCMD supports both interactive and batch jobs.

The routing data is taken from the job description specified on the commands that placed the job on the job queue, or from the routing data (RTGDTA) parameter on the submit job (SBMJOB) command. If routing data is specified in the command, it is used instead of the routing data contained in the job description specified.

Jobs

The system operator can manage the work load on the system by starting and ending the subsystems needed for the various kinds of work to be performed. However, because jobs can be individually controlled, the work load can be further managed at the individual job level. This is done by defining attributes to meet the special requirements of an individual job.

Within a job, any number of related or unrelated functions can be performed. The functions might be requested in:

- A series of control language commands
- A single program
- One or more programs that are each made up of a series of programs

On many systems, the running of programs within a job is controlled only through the use of job steps, which are identified in the control language statements that make up the job. However, on the AS/400 system programs can call other programs directly. Thus, the job is simply made up of whatever sequence of processing actions a user wants performed.

Autostart Jobs

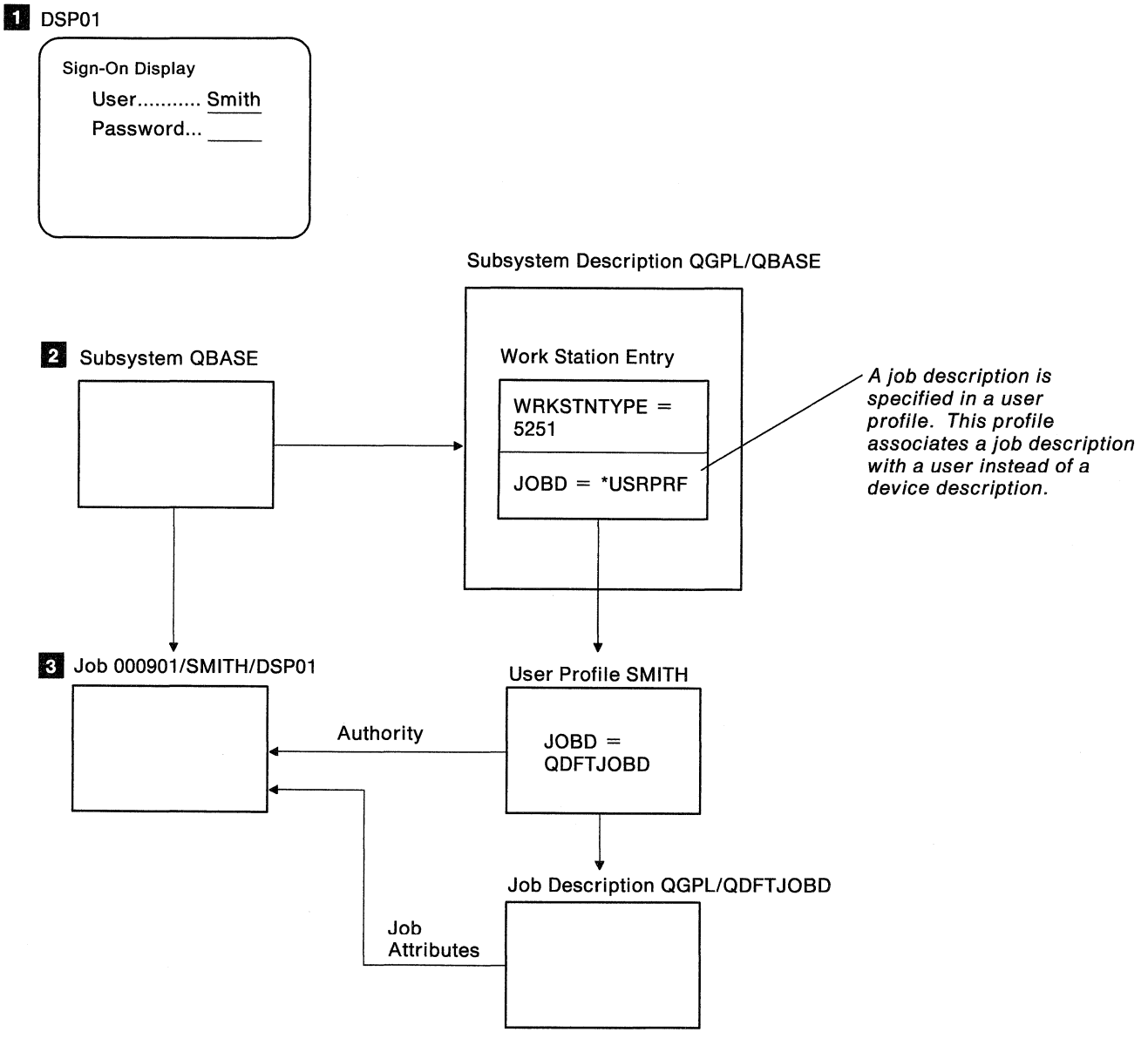
Autostart jobs are started automatically when a subsystem is started. The routing data used to start the routing step in the job is taken from the job description.

Interactive Jobs

Interactive jobs are started when a work station user signs on. An interactive job consists of all the work performed as a result of input received from the time the work station user signs on until the user signs off. Processing actions performed after the user signs off, such as writing spooled output to a printer are still considered part of that interactive job because the input for that processing was received while the user was signed on.

The first display the user sees when the work station is turned on is the Sign-On display. When the work station user enters a user ID, a valid password (if security is active), and presses the Enter key, the AS/400 Main Menu is displayed. The system does not require a password unless the security level is set in the system value at a value of 20 or higher.

The work station user can then select options from this menu. When the user selects the sign-off option the interactive job ends and the Sign-On display appears again. Figure 10-4 shows what happens when user SMITH types the user ID and presses the Enter key.



1 User SMITH types his user name and password and presses the Enter key.

2 Subsystem detects a sign-on attempt.

3 Job is started.

RSLM121-2

Figure 10-4. Subsystem Activity Leading Up to the Creation of a Job

Batch Jobs

Batch jobs are placed on a job queue to be started by a subsystem. Jobs can be placed on a job queue even if the subsystem that owns the job queue is not started. When the subsystem QBASE (or QBATCH, if QCTL is the controlling subsystem) is started, it processes the jobs on the queue in order according to their priority. The user can specify the maximum number of batch jobs that can be processed at the same time through a job queue by specifying the number of jobs in the job queue entry.

Not all jobs on a job queue are necessarily started when the subsystem is started; jobs can be held on a queue until the system operator releases them, the MAXJOBS (maximum jobs) value for the subsystem, MAXACT (maximum active) job queue is reached, or the maximum active for the priority level is reached. If the subsystem is ended before all the jobs are started, the jobs remain on the queue until the subsystem is started again or until another subsystem allocates the same job queue.

Spooled Jobs

Spooled jobs are taken from an input device, arranged by order of importance, and placed on a job queue. They are then started from the job queue. Spooling functions are available for both input and output. Spooling functions help system users to manage input and output operations more efficiently. The system supports both output spooling and input spooling.

Output Spooling: Output spooling sends job output to a spooled file, rather than directly to a printer or diskette output device. Output spooling allows the job producing the output to continue processing independently of the speed or availability of output devices.

For output spooling, the system places output records produced by a program in a spooled output file and places the spooled file on an output queue. These files are later written to external devices (tapes, printers, or diskette) by system programs, called *writers*. A separate job description is used for each type of reader or writer, therefore, the system can uniquely handle different types of spooled processing.

Spooling is especially important in a multiple-user environment where the number of jobs running often exceeds the number of available output devices. Using output spooling, the output can be easily redirected from one device to another.

The main elements of output spooling are:

Device description	A description of the printer or diskette unit
Spooled output file	A file containing spooled output records that are to be produced on an output device
Output queue	An ordered list of spooled output files
Writer	An IBM-supplied program that takes spooled output files from an output queue and produces them on an output device
Application program	A high-level language program that creates a spooled output file using a device file with the spooling attribute specified
Device file	A description of the format of the output, and a list of attributes that describe how the system should process the spooled output file

Figure 10-5 shows the relationship of the output spooling elements.

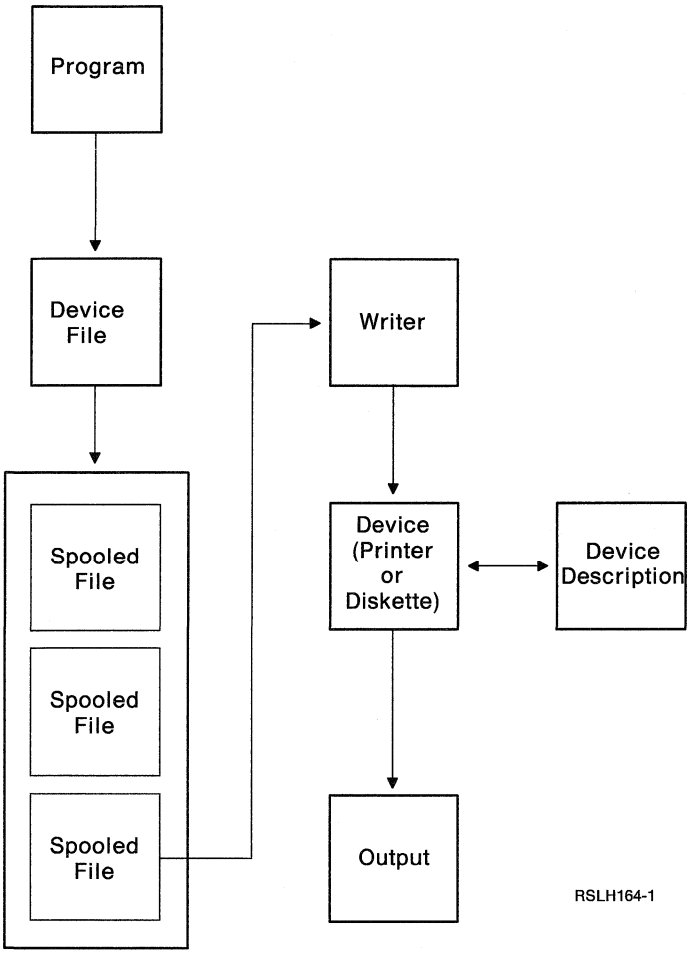


Figure 10-5. Relationship of Output Spooling Elements

Output spooling functions are performed by the OS/400 program without requiring any special instructions from the program that produces the output. When a device file is opened by a program, the operating system determines whether the output is to be spooled. When a printer or diskette file specifying spooling is opened, the spooled file containing the output of the program is placed on the appropriate output queue in the system.

A spooled file can be made available for printing when the printer file is opened, when the printer file is closed, or at the end of the job. A writer is started in the spooling subsystem to write the records to the printer. The spooled output is selected from an output queue. The same general description applies for spooled diskette files.

Input Spooling Support: Input spooling applies to diskette and database file input. Input spooling takes the information from the input device, prepares the job for scheduling, and places an entry in a job queue. Using input spooling, you can usually shorten job run time and increase the number of jobs that can be run sequentially while making the most efficient use of the device. For input spooling, a system program, called a *reader*, transfers jobs from an input device (work station, diskette, or database file) to a job queue.

The main elements of input spooling are:

Job queue — An ordered list of batch jobs submitted to the system for running and from which batch jobs are selected to run.

Reader — An IBM-supplied program that takes jobs from an input device or a database file and places them on a job queue.

When a batch job is read from an input source by a reader, the commands in the input stream are stored in the system as requests for the job, the inline data is spooled as inline data files, and an entry for the job is placed on a job queue. The job information remains stored in the system where it was placed by the reader until the job entry is selected from the job queue for processing by a subsystem.

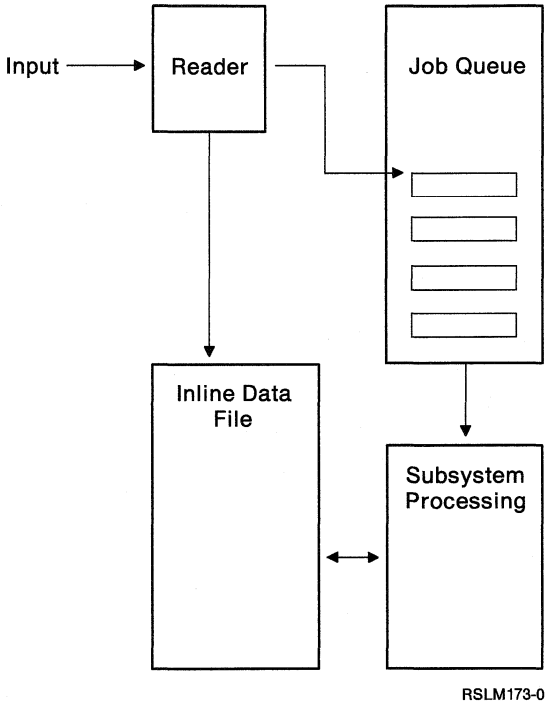


Figure 10-6. Relationship of Input Spooling Elements

Spooled jobs are supported in a manner similar to batch jobs. Figure 10-7 on page 10-12 shows the job queue entry that was added to the subsystem description and how the spooling subsystem QSPL operates. QSPL supports the processing of spooling readers and writers that transfer data from and to devices independently of the application program.

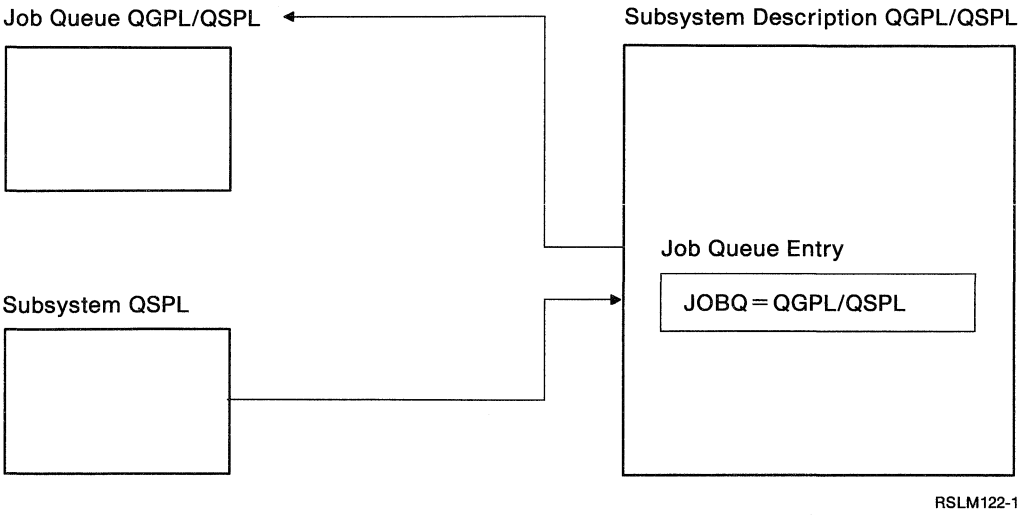


Figure 10-7. How the Job Queue Is Identified to the Spooling Subsystem

In the illustration, the subsystem description of QSPL includes a job entry for the QSPL job queue in library QGPL. When the QSPL subsystem is started, it starts the jobs on the QGPL/QSPL job queue until it is empty or the subsystem is ended.

Figure 10-8 on page 10-13 shows an example of spooling. The user's program specifies that a line should be printed on printer PRTF3. Using the printer file description, the system checks whether spooling has been specified. Using the job description information, the system uses the output queue (OUTQ) specified in the user profile (*USRPRF). For this user, the output queue is OUTQ1. The file is placed in the spool database file in the library QSYS and a pointer to the file is added to the output queue OUTQ1. (The placement in the queue reflects the priority of the job. If no priority is specified then the file is printed in the order in which the request was received.) When the spool writer starts, it selects the first file in the queue and begins sending it to the printer. This continues until the queue is empty, until only files to be *held* remain, or until the writer is stopped. (Files can be held in the queue until the user decides to print them.) The system uses the device description of the printer to determine what codes to send for special characters, such as underlining, bold type, and so on.

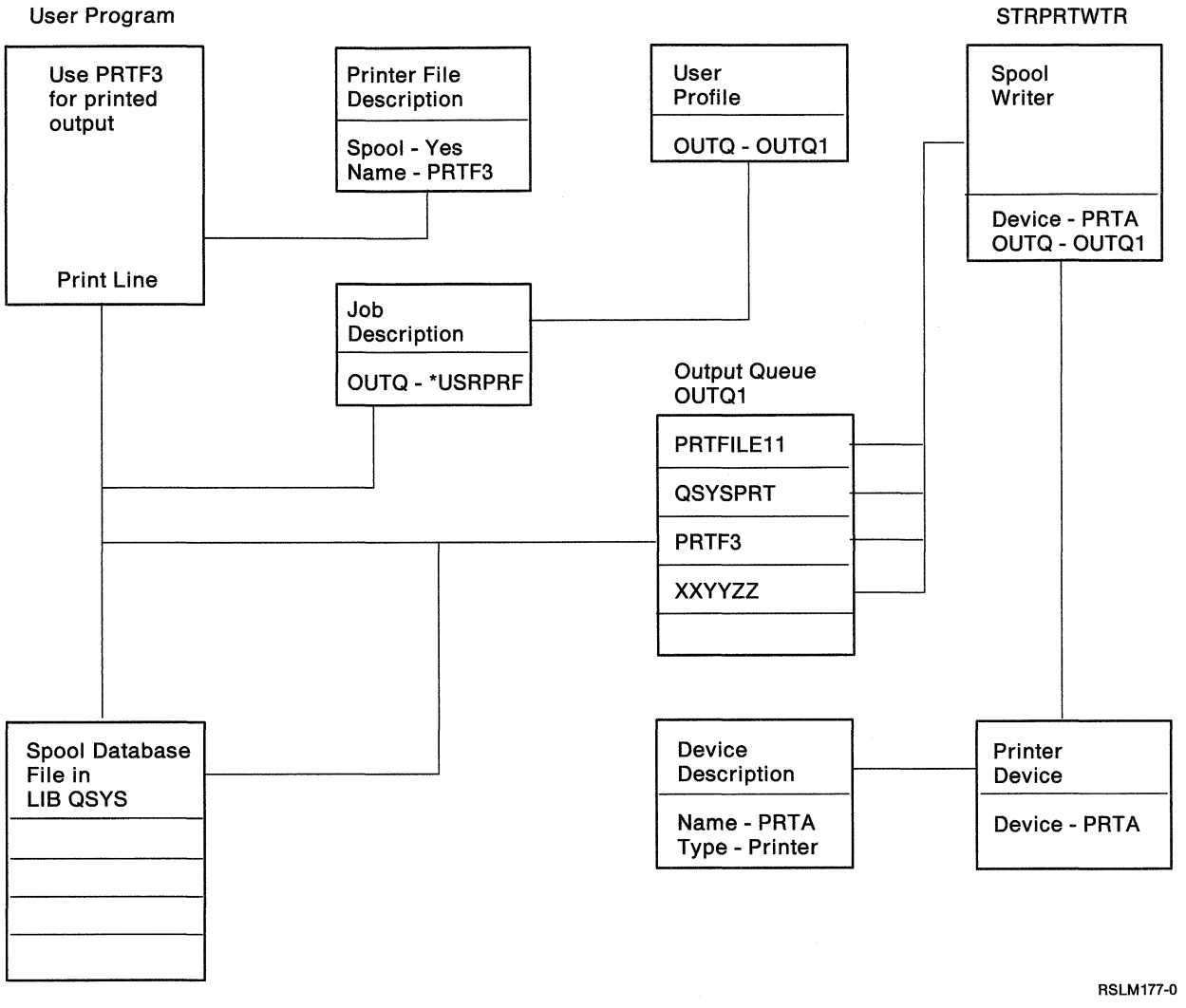


Figure 10-8. Example of Spooling

Communications Jobs

Communications jobs are started when a subsystem receives a program start request from a remote system. The requests are sent to the logical unit services (QLUS) system job which allocates devices for all communications jobs.

Before a communications device can successfully process a program start request, the device must be allocated to a subsystem. Logical unit services (QLUS) is notified when a communications device is available for starting jobs. This notification occurs when the link between the local and remote system is established for that device. When QLUS is notified, it attempts to allocate the communications device to a subsystem based on communications entry definitions. If there is no subsystem active that wants to use the device, QLUS maintains allocation of the device until the device is varied off, or a subsystem starts that wants to use the device.

Prestart Jobs

When a prestart job entry is added to the subsystem description, prestart jobs are started based on information contained in the prestart job entries. A prestart job runs under the user profile specified on the program-start-request when it is servicing that request. If a user profile is not specified on a program start request, the default user profile specified on the communications entry is used as the program-start-request user profile. The prestart job function is available to all communications protocols that support program-start-request (PSR) processing.

When a request to start a program is received, the subsystem determines if the program name sent on the program start request matches the program name on one of the prestart job entries. If so, the request is attached to one of the prestart jobs.

For example, a prestart job can initialize the environment where the application program is to be run. This includes running any programs or opening any files that are required in preparation for a request for the job from a remote user. Using a prestart job, a subsystem on the AS/400 host system in a finance enterprise can quickly respond to users' requests from the remote sites. The jobs are ready to be connected at the start of the subsystem as opposed to the more traditional approach where the job and application program initialization process occurs as the request is made.

Job Descriptions

Each job has a set of attributes. Different sets of attributes are needed for different jobs to meet the special requirements of each job. Because specifying all the attributes each time a job is submitted would be a tedious and lengthy task, the OS/400 program supports an object called a *job description* in which the attributes of a job can be predefined. The system is shipped with default job descriptions for batch, spooled, and interactive jobs. These attributes can be changed as processing needs change. The attributes specified in a job description include:

- The job queue on which the job should be placed when it is submitted (for batch and spooled jobs only)
- The scheduling priority to be used to select the job to be run from the job queue
- The user portion of the library list that is in effect when the job is started
- The routing data used by the subsystem to determine the correct routing entry for the job
- The output queue onto which spooled output should be placed.
- The output scheduling priority to be used for producing spooled output
- The user profile for the job

For a job placed on a job queue, the job description is specified when the job is submitted to the queue. The job attributes specified in the job description can be overridden when a job is submitted. For example, a different user library list might be specified. The user library list specified in the job description would then be ignored.

Job Classes

The conditions in which a job is run is specified through an object called a *class*, which contains the parameters for a routing step and further defines the environment of the job. The class is specified in the routing entry. The same class can be specified for any number of routing entries. The parameters that can be specified in a class include:

- The processing priority given to the routing step
- The time slice, or quantity of processor time, allowed for the routing step before other waiting routing steps are given the opportunity to run
- The maximum processor time allowed for the routing step
- The maximum time to wait when an instruction in the routing step must wait for some resource

Most jobs consist of only one routing step, which is the one begun at the start of the job, and only one routing step is ever active at a time in any job. If a routing step environment is to be changed, a new routing step is started. This may change the program that controls the routing step, the storage pool in which the program runs, the class that describes the environment, or the subsystem in which the routing step runs.

Performance

The system provides functions for collecting and reporting the current operating performance of the system. These reports can be collected regularly to produce a history of system performance. By monitoring system performance, changes can be made in work management tasks to accommodate the system work. For example, when a payroll program is processing payroll information to print checks, a typically resource intensive task, the system should not be used to save the complete system which requires numerous read and write operations between the main storage and auxiliary storage, tape, or diskette.

Capacity planning: Capacity planning is available to predict your hardware configuration for future data processing needs. It involves a review of current performance levels and an assessment of the future direction of the business to determine what additional hardware or programming changes should be made to improve performance or throughput with minimal cost, both in time and money. Issues to be considered include:

- Current performance does not meet objectives
- An increase in the number of transactions to be processed is expected
- A functional change in application programs is anticipated
- Another AS/400 system is being installed or added to the network of the existing system

Performance analysis: System functions, called performance tools, are available to help the user analyze the system performance for the following parts of system and program performance:

Tool	Tasks Analyzed
Job trace	Input/output operations, file use, transaction time, job flow
Program statistics	Time spent in each high-level language instruction; modifications could lead to improvement in the program's performance perhaps through changes in data types, algorithms, or arrays
File and access	Sharing of display and database files, arranging files by frequency of use, closing files when access is complete, freeing storage. This looks at all jobs, or a group of jobs, in the system at a given time.
Disk activity	Balance the use and improve disk performance by distributing heavily used files among disk storage units
Lock conflicts	Requests for system-locking functions, used to ensure integrity, are made for the same record by different jobs

Chapter 11. System Management and System Recovery

The AS/400 system is designed to help you manage one or more systems efficiently. Built into the system and its attached devices are functions that test, diagnose, and recover from errors on many of the devices and operations on the local system. If the system is part of a communications network, it can also provide information to analyze and recover from network errors. The operating system provides the communications and programming support required to connect electronically to a service provider. This can be IBM, a service vendor, an employee, or department assigned to help with problems detected on the system.

The SystemView* System Manager/400 licensed program is available to help control system management cost and complexity. It is most helpful in a network of several AS/400 systems. This set of tools is an implementation of the IBM SystemView architecture. The system management tools available with the OS/400 program also fit within the SystemView framework.

In addition to the system providing comprehensive problem analysis and error recovery methods, the AS/400 system also provides several methods for protecting and recovering data in case of a system or disk failure.

This chapter briefly discusses the topics listed in the following table. If you would like more information on the topic, and there is more information available in an IBM manual, the manual listed in the right column is a good first reference.

Topics	First Reference
AS/400 system management	*System Operator's Guide*, SC41-8082
SystemView concepts	*SystemView Concepts on the AS/400 System*, GA21-9607
OS/400 system management to SystemView disciplines	*Systems Application Architecture* SystemView* System Manager/400 User's Guide*, SC41-8201
Business management	*System Operator's Guide*, SC41-8082
Change management	*Systems Application Architecture* SystemView* System Manager/400 User's Guide*, SC41-8201
Configuration management Operations management	*System Operator's Guide*, SC41-8082
Performance management	*Programming: Work Management Guide*, SC41-8078
Problem management	*System Operator's Guide*, SC41-8082
System recovery support	*Backup and Recovery Guide*, SC41-8079

The SystemView Program and the AS/400 System

The SystemView program defines six systems management tasks necessary to support a business establishment information processing environment. The systems management functions to support these tasks can be integrated into the system's operating system or be part of a licensed or software vendor program. Each task is called a discipline, and the SystemView application dimension defines the disciplines as follows:

Business Management

This discipline addresses the activities involved in the effective and efficient management of the business aspects of an information system environment, such as inventory control, usage of and access to information system resources, financial management of the system, business planning, and process management services.

Change Management

This discipline controls the introduction of change into an information system environment. The goals of change management are to minimize the effect of the change, reduce the skill level needed to manage the change, and reduce the change process to a series of small repeatable steps that can be automatic. The change management discipline addresses initiation, planning, assessment and approval, scheduling, distribution, synchronization, installation, activation, monitoring, and subsequent analysis of changes made to an information system.

Configuration Management

This discipline controls how you plan, develop, and maintain the way the resources of an information system relate to each other. The configuration management discipline addresses configuration design, environmental planning, and updating and accessing configuration information.

Operations Management

This discipline controls how you plan for, evaluate, and support the work load placed on an information system. The operations management discipline addresses work load planning, operations structure definition, work control, operational control, and work and operational processing.

Performance Management

This discipline defines how you plan, evaluate, and control the delivery of an information system's services to its users. The performance management discipline addresses performance measurement and data collection, capacity planning, performance policy definition, and performance processing and control.

Problem Management

This discipline controls how you detect, analyze, recover from, resolve, and track problems that occur in an information system. The problem management discipline addresses policy planning and definition, process planning and tracking, incident detection and recognition, incident notification logging and filtering, problem correlation and determination, problem analysis and diagnosis, problem bypass and recovery, problem assignment, problem correction determination, problem escalation, and problem resolution and verification.

Three integrated licensed programs for the AS/400 system support the SystemView strategy and provide functions necessary for managing information systems. These licensed programs are:

- Operating System/400 program
- AS/400 Performance Tools program
- SystemView System Manager/400 program

Figure 11-1 on page 11-3 shows how the OS/400 program, the Performance Tools, and the SystemView System Manager/400 licensed programs are packaged across the disciplines in the application dimension of the SystemView structure.

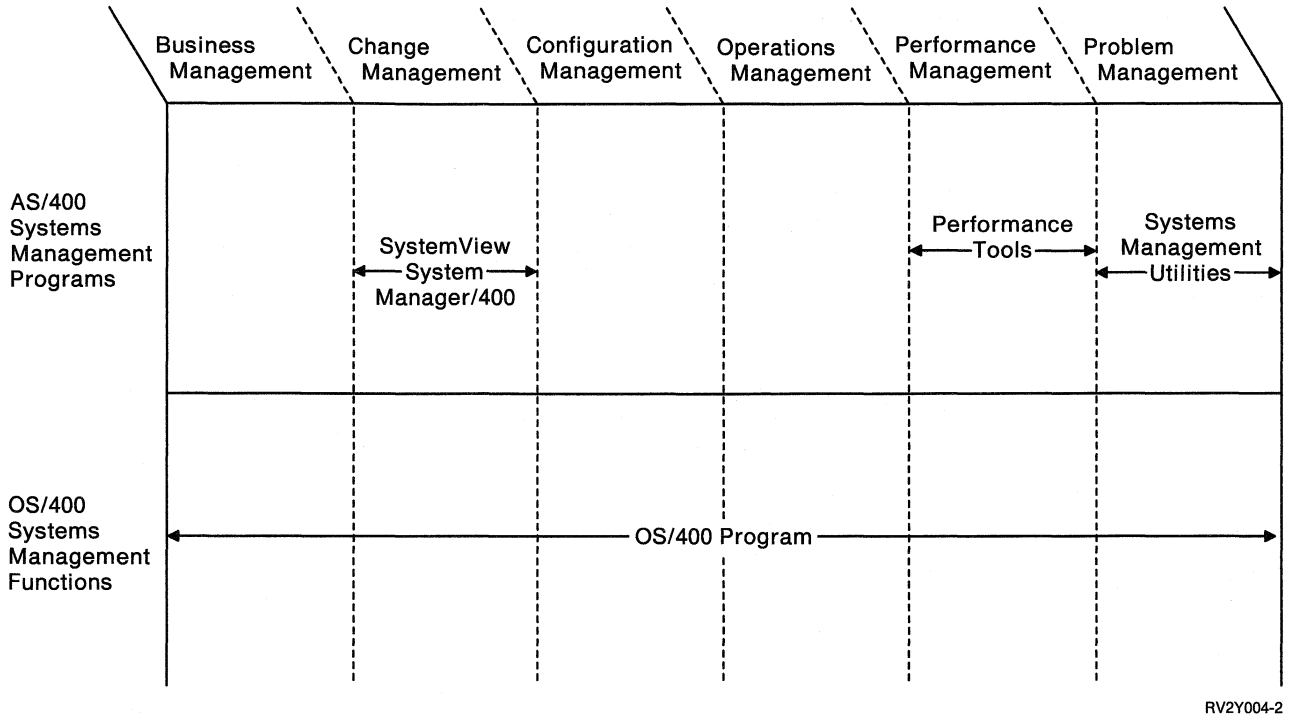


Figure 11-1. AS/400 Packaging across SystemView Disciplines

The OS/400 licensed program provides systems management functions across all the disciplines in the application dimension of the SystemView structure. The SystemView System Manager/400 and the Performance Tools licensed programs provide additional support for the problem management, change management, and performance management functions available with an AS/400 system.

The systems management functions integrated into the OS/400 licensed program allow you to manage a single AS/400 system, as well as allow another system to manage the AS/400 system.

The SystemView System Manager/400 licensed program allows customers and business partners with multiple AS/400 systems to manage those systems from an AS/400 system at a central site, or from an intermediate managing system when included in a System/370 host or System/390 host environment.

The AS/400 Performance Tools licensed program provides tools to help you with performance analysis and capacity planning that enhance the performance functions provided by the OS/400 licensed program.

SystemView Concepts

The AS/400 systems management functions that comply with the SystemView strategy are designed to adhere to the basic SystemView objectives:

- Integration of systems management applications
- Automation of systems management tasks
- Creation of an open structure

Mapping AS/400 System Management to SystemView Disciplines

The following section describes the system management functions available on the AS/400 system, and shows how and where they fit into the SystemView application dimension structure.

Figure 11-2 shows the six SystemView disciplines with a list of the corresponding AS/400 functions for each discipline. The OS/400 functions are listed separately from the other licensed program functions so that you can get a clear picture of the integration across AS/400 systems management functions.

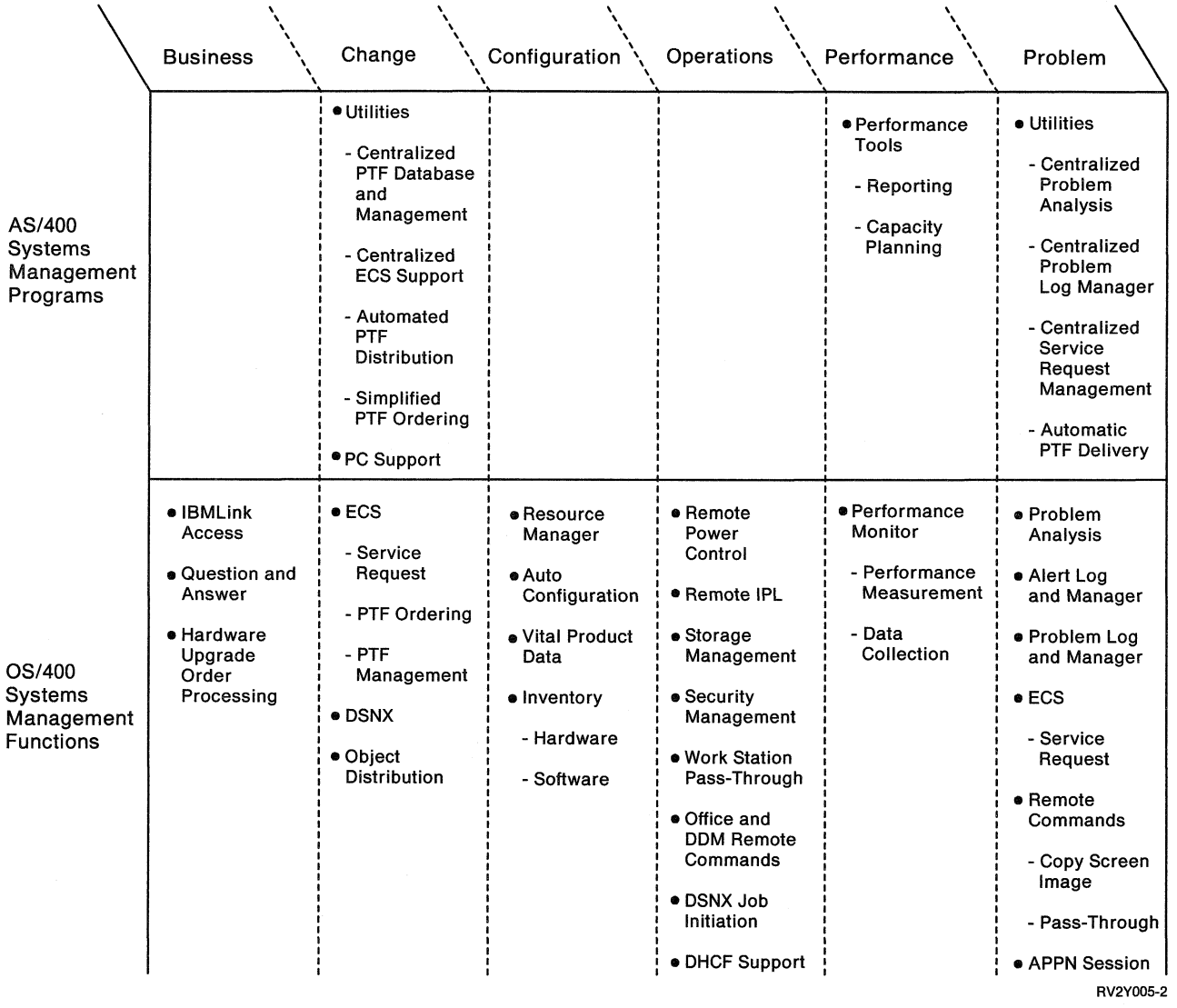


Figure 11-2. AS/400 SystemView Functions by Discipline

Business Management

The OS/400 program provides several integrated business management functions as shown in Figure 11-3 on page 11-5.

- The IBMLink interface transmits resource and configuration information to the IBM order processing systems and is used to automate the upgrade order process. The IBMLink interface also provides two-way file and message transfer capability using the IBM internal network functions.

- A question and answer (Q & A) database is maintained by IBM technical support and marketing personnel. Through the use of the AS/400 local Q & A function, electronic customer support, and IBMLink, a wide range of AS/400 information is available to assist you with your systems management tasks.
- IBM product information provides access to marketing information, such as system library subscription service (SLSS) lists and announcement letters. IBM product information and technical information exchange (TIE) work together to provide a set of technical support functions and information for you. IBM product information provides access to marketing information available on the marketing support network.
- The TIE function is an asynchronous file routing function used to exchange files between your system and the IBM marketing support system.

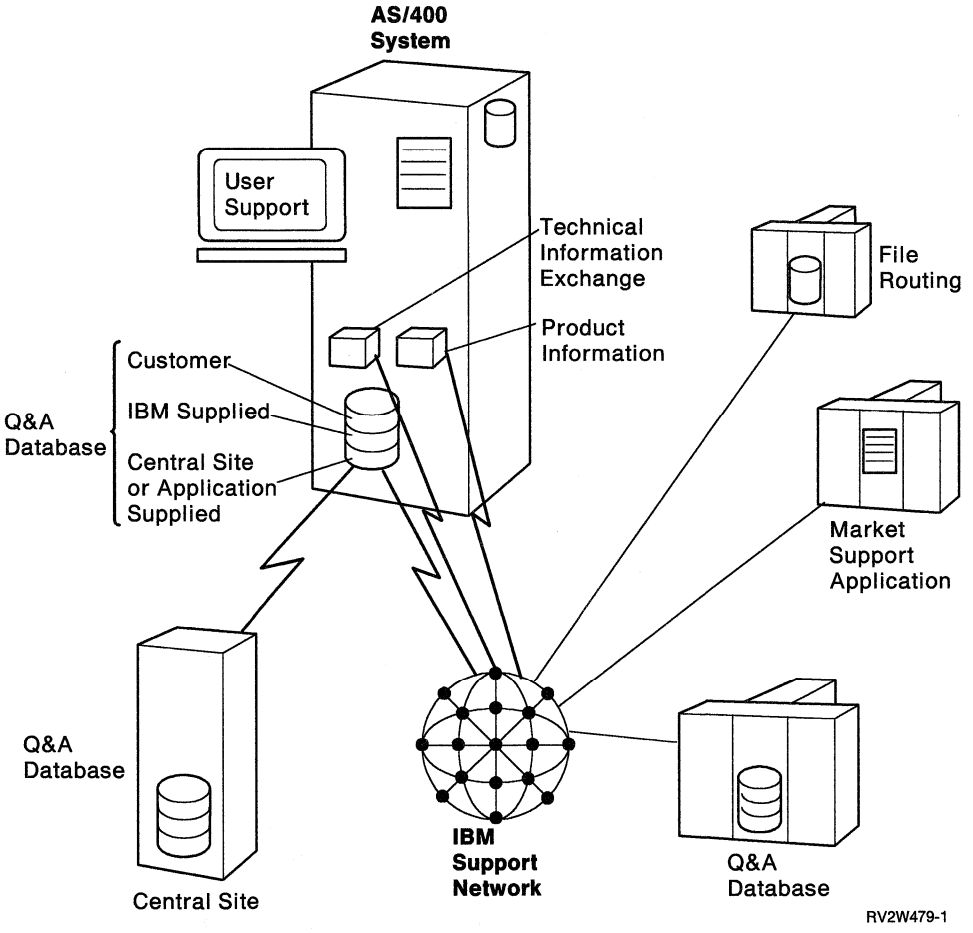


Figure 11-3. Technical Support and Information Access Functions

Change Management

System changes can be as small as program temporary fixes (PTFs) necessary to resolve a system code problem, or as large as upgrading the entire operating system.

Through the use of the electronic customer support available with the OS/400 licensed program, the AS/400 system can manage changes in a single system environment or have changes managed from another AS/400 system in a network or a System/370 or System/390 host system.

- In the AS/400 single system environment, electronic customer support is used to:
  - Send requests for service or PTF orders directly to IBM service support
  - Receive PTFs electronically or by mail (tape)
- When an AS/400 system manages other AS/400 systems, use of the SystemView System Manager/400 licensed program allows one AS/400 system to be designated as the service provider. The service provider:
  - Creates and maintains a central database of PTF information based on licensed programs that are installed or supported
  - Uses the electronic customer support service request and PTF ordering capability for both the local and remote AS/400 systems in a network
  - Distributes PTFs automatically or with manual control to remote systems in the network
  - Can process cumulative PTF orders and preventive maintenance information for the remote systems
- When the AS/400 system is in an environment managed by a System/370 or System/390 host, the OS/400 distributed systems node executive (DSNX) works with the System/370 or System/390 NetView Distribution Manager (NetView DM) licensed program to:
  - Send, retrieve, and delete files, programs, formats, and procedures in a network
  - Use DSNX support to distribute files and job streams in the System/370 or System/390 network
  - Use DSNX support for central site programming, maintenance, and distribution of AS/400 objects
  - Use OS/400 DSNX support to allow the AS/400 system to act as a direct node to the System/370 or System/390 host system or as an intermediate node (or nested focal point) between the host system and other AS/400 systems, personal computers, and the System/36
  - Use DSNX support with the AS/400 PC Support licensed program to transfer files to personal computers in the network

  Distributions to AS/400 systems and to personal computers use SNA distribution services (SNADS) and object distribution

Configuration Management

OS/400 resource and configuration management provides a system resource manager component that is the interface to a system resource database. The resource database is a record of the AS/400 system's hardware and software vital product data and configuration information. Vital product data is collected from attached hardware during each initial program load (IPL) or when resources are added after an IPL (when the device is powered on). A software installation manager collects and records software vital product data and configuration information when program products or components are added or changed. The information in this database is made available for use by applications, problem analysis procedures, and other operating system functions through the use of application program interfaces.

The configuration manager component provides an interface to create, change, or delete configuration objects automatically or under user control. Configuration objects can be automatically created for all locally attached devices when they are installed and powered on. The user can create the configuration objects using

menus or commands. Default values are provided with the OS/400 licensed program to minimize the amount of data that must be typed. All local and local area network (LAN) attached device configurations can be created automatically. To configure remote devices, use the system-supplied menu or command interface.

The software vital product data and configuration information is used by the software installation manager component to correctly install and maintain all licensed programs and licensed internal code groups.

Operations Management

The OS/400 licensed program provides functions for both local and remote operations management tasks.

Local OS/400 operations management is provided through the use of system-supplied menus and CL commands that allow the user to:

- Control spooling, storage utilization, system security, and batch and interactive jobs
- Customize the above operations for different business environments

The OS/400 Operational Assistant provides a more simplified set of operations management menus. Functions like printing, file reorganization, storage management, and security management are further automated to make these functions easier to use and more transparent to the user.

If the AS/400 system is in a network environment, remote operations functions are available through the use of remote commands.

- Using NetView DM and OS/400 DSNX, the System/370 or System/390 host system can submit remote commands to the AS/400 system
- Using the PC Support licensed program, OS/400 DSNX allows the user to perform AS/400 system operations management from a personal computer in an AS/400 environment
- Using HCF/DHCF at the System/370 or System/390 host system, the user can sign on to a remote AS/400 system with a 3270 display station that emulates the AS/400 5250 display station

Performance Management

Performance management support for the AS/400 system is provided through a combination of commands within the OS/400 licensed program and the AS/400 Performance Tools licensed program.

The AS/400 Performance Tools licensed program, along with the function provided by the OS/400 licensed program, provides:

- Functions to collect, measure, and report system performance data at various levels of detail
- Functions to assist with system tuning, work load scheduling, application tuning, capacity planning, and system sizing on the AS/400 system
- Capability to display or print performance data in table or graphic formats
- Capability to use the data in other applications or reports through the Query/400 licensed program functions

Problem Management

The AS/400 system provides integrated problem management functions in both the OS/400 licensed program and the AS/400 SystemView System Manager/400 licensed program.

The OS/400 licensed program provides problem management support for both system-detected and user-detected system problems occurring on an AS/400 system. Figure 11-4 provides an overview of problem management functions. OS/400 problem management functions include:

- Alert support for system-detected problems
- Alert management focal point capability
- Problem analysis procedures
- Problem logging and tracking
- Electronic customer support service and PTF requisition

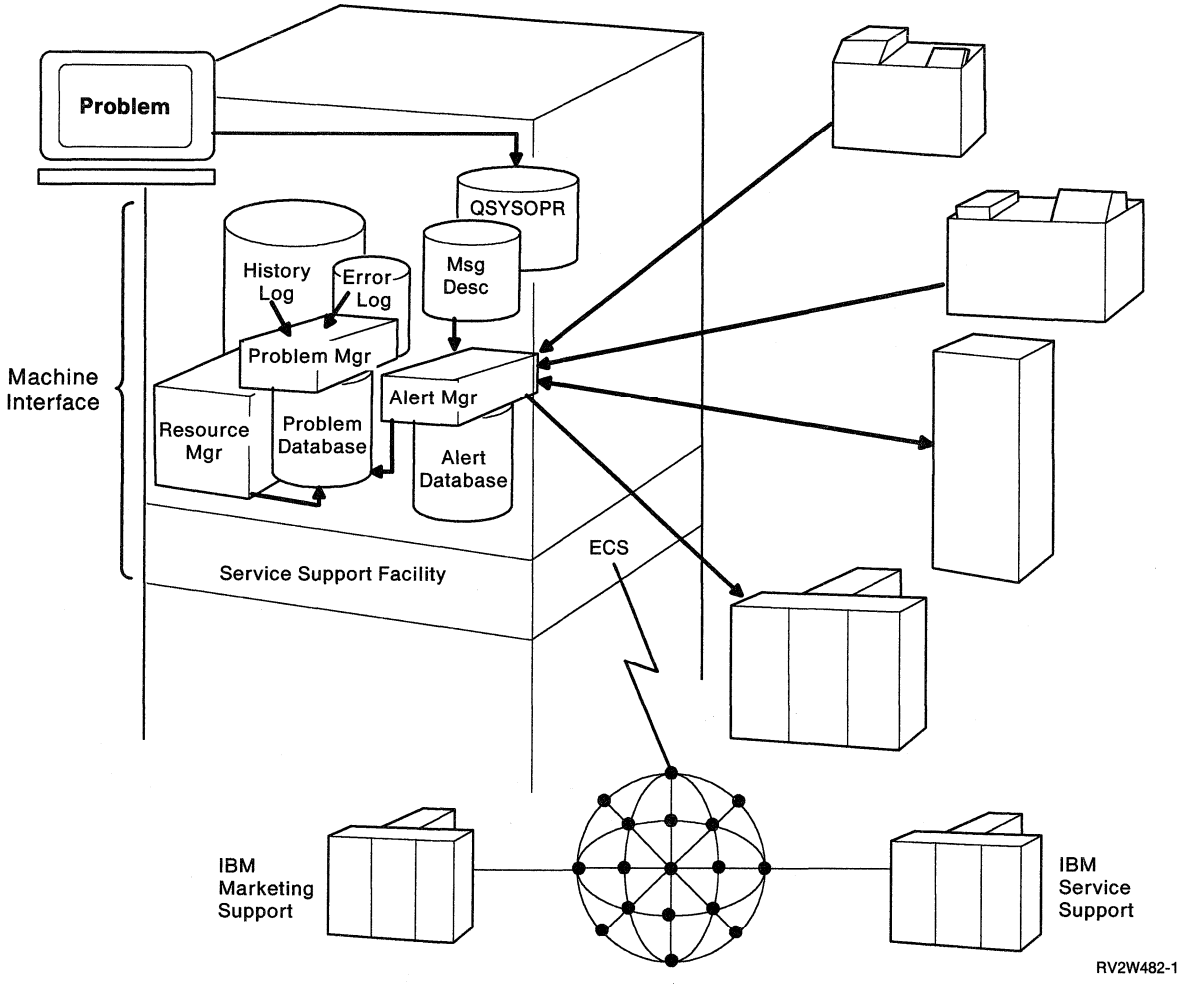


Figure 11-4. Problem Management Functions

When a problem occurs on an AS/400 system, an alert is created and logged. The alert includes vital product data and configuration information about the resource, the reference code, and other available information about the failing resource. When the alert is created, an error log entry is also created and a system message is sent to you.

From this message, problem analysis can be started and the results stored in the system's problem log. The problem record can then be sent as an electronic service request to IBM service support.

When the AS/400 SystemView System Manager/400 licensed program is installed on a managing AS/400 system in a network, the AS/400 system becomes a service provider for other AS/400 systems in the network. The service provider becomes a central tracking point for AS/400 problems in the network. The central site AS/400 system can:

- Receive, process, and track alerts
- Receive, process, and track service requests and PTF orders
- Run remote analysis for other AS/400 systems in the network
- Distribute PTFs to the other AS/400 systems in the network

Copy Screen Image: The copy screen image operation, which is part of the system service support, allows a user at one work station to see the same displays being viewed by a second user at another work station. Through commands or menus, the first user specifies that screen images on a specific work station are to be copied to some other work station. Screen images can also be copied to a database file at the same time they are copied to another work station. This allows users to process this data later and prepares an audit trail for the operations that occur during the transaction.

Service Tools: Service tools are called from the system menus, through the system service tools (SST) option, the dedicated service tools (DST) option, or by control language commands. SST and DST allow you and service personnel to analyze problems in a nonstructural (freelance) manner. SST runs one or more licensed internal code or hardware functions under the control of the operating system. SST lets you perform service functions at the same time the application programs are running. DST is used to solve problems occurring in the licensed internal code and to work with disk configurations. The system cannot be doing other work when the DST is running.

Recovery Support

The AS/400 system provides several methods for protecting the system programs, application programs, and data from being permanently destroyed. Depending on the type of failure and the level of protection chosen, most of the programs and data can be protected, and the recovery time can be significantly reduced.

The loss of the site can be caused by fire, flood, explosion, or sabotage. Though the probability of such disasters is remote, preparation for the recovery from these events must be considered.

The loss of the system programs or data can be caused by a disk unit failure or a system program error. When these failures occur, you must be in a position to install the entire system programs again, including the operating system, all the application programs, and data.

The most common type of loss is the loss of an object or group of objects. An object can be lost or damaged due to several factors, including power failure, hardware failures, system program errors, application program errors, or operator errors. For example, if an operator selects the wrong tape, incorrect data can be loaded and cause out-of-date databases.

To protect the system from loss because of power failure, power interruptions, or drops in voltage, an uninterruptible power supply that can be ordered separately can supply power to the system devices until power can be restored. Normally, an uninterruptible power supply does not provide power to all work stations. With the AS/400 system, the uninterruptible power supply:

- Continues operations during brief power interruptions or momentary drops in voltage
- Allows the system to end operations normally by closing files and maintaining object integrity.

Save and Restore Processing

Saving and restoring data and programs allows recovery from a program or system failure, exchange of information between systems, or storage of objects or data offline. Saving the system on external media is also the only method available to protect the system programs and data from disasters, such as fire or flood. The saving and restoring also allow objects from a current release to be restored to a previous release via tape or diskette. This is particularly beneficial to sites with many systems. The central site can install and test the new release while the other systems in the network are still functioning with the previous release level. The other systems can install the new release at their convenience.

When information is saved, a copy of the information is written onto one or more diskettes, reels of magnetic tape, tape cartridges, or to a disk file called a *save file*. A save file is a disk-resident file used to store data until it is used in input and output operations or for transmission to another AS/400 system over communication lines. Using a save file allows unattended save operations because an operator does not need to load diskettes or tapes.

When information is restored, the information is written from diskette, tape, or a save file into auxiliary storage where it can be accessed by system users.

Journal Management

Journal management can be used as a part of the backup and recovery strategy for database files and access paths. Journaling must be started and stopped by you.

When changes are made to the database, the changes are recorded in an object, called a *journal receiver*, before the changes are written to the journal or made to the database. The journal receiver always has the latest database information. All activity is recorded for a physical file regardless of how the change was made.

Journal receiver entries record activity for a specific record (added, changed, or deleted), and for a file (opened, file or member saved, and so on). Each entry includes additional control information identifying the source of the activity, the user, job, program, time, and date.

The system records some file-level changes, including moving a file and renaming a file. The system also records member-level changes, such as initializing a physical file member, and system-level changes, such as initial program load (IPL). You can add entries to a journal receiver to identify significant events (such as the check-point at which information about the status of the job and the system can be recorded so that the job step can be restarted later) or to help in the recovery of application programs.

An access path describes the order in which records are read from a database file. When access paths are recorded in the journal, the system can recover the access path and avoid spending a significant amount of time rebuilding access paths during the IPL following an abnormal system end.

Commitment Control

Commitment control is an extension of the journal management function on the AS/400 system, and must also be started by you. The system can identify and process several changes to database files as a single transaction. When the system or a job ends abnormally, journaling alone ensures that no more than the changes to the last record are lost by synchronizing database files with the journal. Because programs can make multiple changes to files in one transaction, the journal alone may reflect only a partially completed transaction.

With commitment control started, the system ensures that:

- All changes within a transaction are completed for all files affected
- All changes within a transaction are rolled back if processing is interrupted
- Changes are not saved if you cancel the transaction
- An application can be started again if a job or system fails

The commit and rollback operations are available in several AS/400 programming languages including the C/400, COBOL/400, Pascal, PL/I, Control Language (CL), RPG/400 and Structured Query Language (SQL) languages.

Data Recovery after Disk Failures

Disk recovery is the process of recovering from the loss of data because of failures or damage of the storage media contained within the disk units. In addition to the regular backup procedures, there are several OS/400 functions that protect the current data that has not been copied to external media by a save procedure.

If all objects are not saved on external media immediately before a failure, recovery is not possible for recently entered data. Therefore, once the previously saved objects are restored, the system is operational, but the database is not current.

Even if journaling of database files is active, the journal receiver that contains the recently entered transactions is not available if it was stored on the disk that failed. Alternative methods are available to recover recently entered data by using the following disk recovery tools:

- Auxiliary storage pools (ASP)
- Checksum protection
- Mirrored protection

Auxiliary Storage Pools (ASPs)

An auxiliary storage pool (ASP) is one or more physical disk units assigned to the same storage area as shown in Figure 11-5 on page 11-12. ASPs allow you to control where certain types of objects are stored on auxiliary storage devices, thus allowing certain objects to be isolated on specified physical disk units.

The system ASP isolates the system programs and the temporary objects that are created as a result of the processing by system programs.

User ASPs can be used to isolate some objects such as libraries, files, journals, journal receivers, save files, applications, and data. The AS/400 system supports up

to 15 user ASPs. Isolating libraries or objects in a user ASP protects them from disk failures in other ASPs and reduces recovery time.

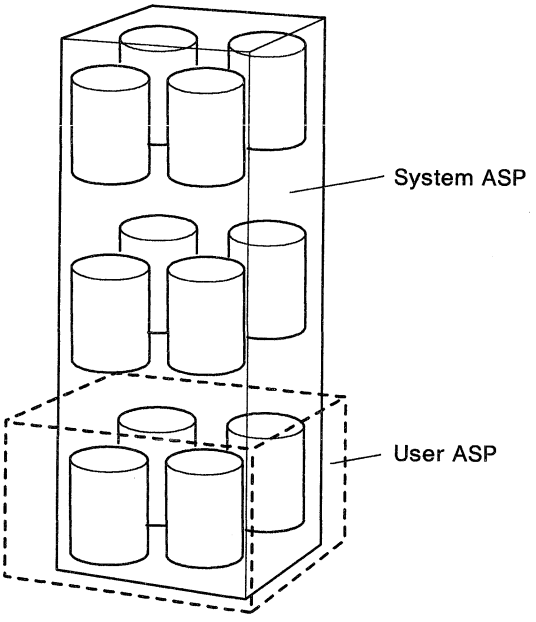


Figure 11-5. Auxiliary Storage Pools

In addition to this recovery advantage, placing objects in an ASP can improve performance. If a journal receiver is isolated in a user ASP, the disks associated with that ASP are dedicated to that receiver. In a programming environment that requires many read and write operations to the database files, this can reduce arm contention on the disks in that ASP, and can improve journaling performance.

Checksum Protection

Checksum protection is available for any ASP on the system. The checksum software maintains a coded copy of ASP data in special checksum data areas within that ASP's auxiliary storage. If the system fails or if a disk is damaged, the system recovers the data when the failure is corrected. Starting checksum protection is a dedicated function that requires exclusive use of system resources. Before starting checksum protection, system performance should be monitored to determine storage requirements.

Figure 11-6 on page 11-13 shows that when checksum protection is configured, the system automatically groups the disk units of the ASP into checksum sets of up to eight disk units each. Space equivalent to one disk unit in each set is used to store the checksum data protects the data stored on the other units of the set. During checksum configuration all of the disk space in an ASP is erased, and the data in the ASP must be reinstalled when the system is started again.

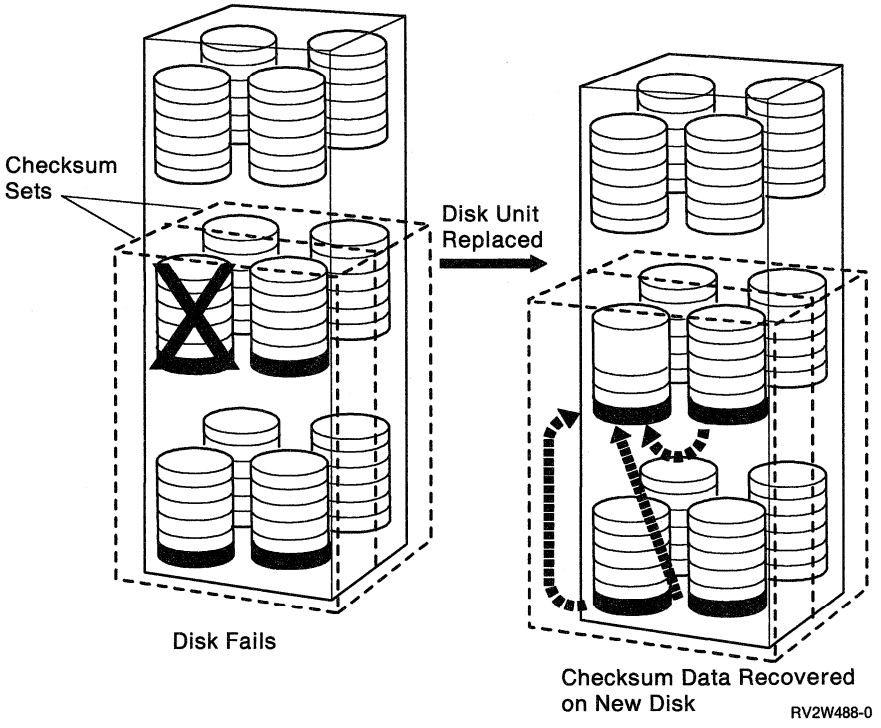


Figure 11-6. Checksum Protection

Checksum protection does not prevent the system from ending abnormally. When a disk unit fails, the system becomes unusable regardless of whether checksum protection is active or not. The advantage of checksum protection is in avoiding a total ASP (or system) reinstallation if the failing unit must be replaced and data is lost.

Any changes made to permanent objects in a checksum protected ASP are automatically maintained in the checksum data of the checksum set. If any single disk unit in a checksum set is lost, the system reconstructs the contents of the lost device using the checksum and the data on the remaining functional units of the set. In this way, if any one of the units fails, its contents may be recovered. This reconstructed data reflects the most up-to-date information that was on the disk at the time of the failure.

Over time some objects can become damaged because of slight imperfections on a disk surface. If this occurs to objects in the ASP when checksum protection is in effect, the data is automatically re-created. This avoids having the system mark the object as damaged.

Mirrored Protection

Mirrored protection is a function that increases the availability of the AS/400 system in the event of a failure of a disk-related hardware component. It can be used on any model of the AS/400 system and is a part of the licensed internal code. Different levels of mirrored protection are possible, depending on what hardware is duplicated. The system remains available during a failure of a disk-related hardware component such as a disk unit, a disk controller, a disk I/O processor, or a bus, if the failing hardware component and hardware components attached to it are duplicated. For the 9406 system unit, some failed hardware components can be serviced while the system remains available.

Figure 11-7 shows an example of a system that has three different levels of mirrored protection.

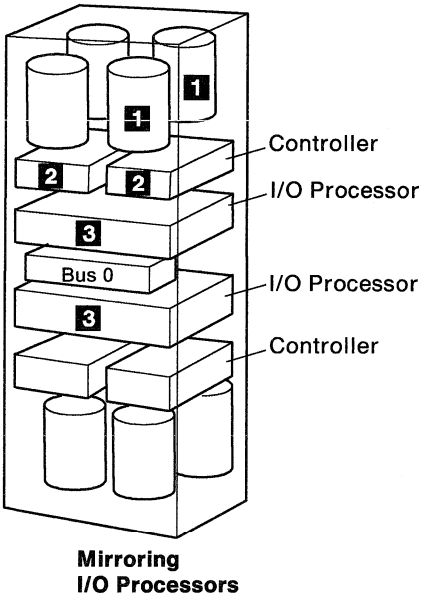


Figure 11-7. Mirrored Protection

1 Disk unit

2 Controller

3 Input/output processor

All mirrored units on the system must have identical disk unit-level protection and reside in the same ASP. The units in an ASP are automatically paired by the system when mirrored protection is started. The system pairs the units to provide the maximum level of protection for the current hardware configuration.

Different levels of protection provide different levels of system availability. For example, if only the disk units on a system are mirrored, all disk units have disk unit-level protection so the system is protected against the failure of a single disk unit. In this situation, if a controller, I/O processor, or bus failure occurs, the system cannot run until the failing part is repaired or replaced.

Checksum and mirrored protection should not be used as a replacement for system backup procedures. Although both mirrored and checksum protection can fully recover from most single disk drive failures, data may be lost as a result of multiple failures or other disasters, such as fire and flood. Even though double failures and disasters are rare, the risk of unrecoverable data loss is too great to consider either checksum protection or mirrored protection as a substitute for saving the entire system on a regular basis.

Chapter 12. Application Design

An application program is a program or group of programs that assist the user in performing business tasks. Primary considerations in developing an application are to:

- Understand the task to be accomplished
- Analyze the problem to develop the simplest logical process that leads to a solution
- Create the programs and define the objects (for example, database files and output queues) that will direct the system to follow the process

This chapter briefly discusses the topics listed in the following table. If you would like more information on the topic, and there is more information available in an IBM manual, the manual listed in the right column is a good first reference.

Topics	First Reference
Definition of the application	No additional manuals
Requirements	No additional manuals
Design	*National Language Support Planning Guide*, GC41-9877
Implementation Development Tools	• *Application Development Tools: Programming Development Manager User's Guide and Reference*, SC09-1339 • *Application Development Tools: Source Entry Utility User's Guide and Reference*, SC09-1338 • *Application Development Tools: Data File Utility User's Guide and Reference*, SC09-1381 • *Application Development Tools: Screen Design Aid User's Guide and Reference*, SC09-1340 • *Application Development Tools: Report Layout Utility User's Guide and Reference*, SC09-1416 • *Utilities: Interactive Data Definition Utility User's Guide*, SC41-9657 • *Programming: Control Language Programmer's Guide*, SC41-8077 • *Programming: Procedures Language 400/REXX Programmer's Guide*, SC24-5553 • *Application Development Tools: Advanced Printer Function Guide*, SC09-1361 • *Languages: Pascal User's Guide*, SC09-1209 • *Languages: PL/I User's Guide and Reference*, SC09-1156 • *Languages: BASIC User's Guide and Reference*, SC09-1157 • *Languages: Systems Application Architecture* AD/Cycle* COBOL/400* User's Guide*, SC09-1383 • *Languages: RM/COBOL-85** for the AS/400* User's Guide*, SC41-9865 • *Languages: Systems Application Architecture* FORTRAN/400* User's Guide*, SC41-9845 • *Languages: Systems Application Architecture* C/400* User's Guide*, SC09-1347 • *Languages: Systems Application Architecture* AD/Cycle* RPG/400* User's Guide*, SC09-1348

Topics	First Reference
Providing online help for users	*Guide to Programming Application and Help Displays*, SC41-0011
Creation	No additional manuals
Testing	No additional manuals
Implementation	No additional manuals
AD/Cycle	*AD/Cycle Concepts*, GC26-4531

Definition of the Application

There are many things that must be considered when creating an application program. For example:

- What is the problem to be solved?
- Who will use the programs and who will use the reports?
- What should the reports look like and what information is required?
- What is the current state of the data? Does the data already exist in the computer or must new data files be created?
- What conventions exist within the current file structures? (For example, file naming, and program variables)
- What backup and recovery procedures should be in place?
- What are the system resources? (For example, printers, displays, and so on)
- How will the program handle input and output, program messages and files, passing of variables, data, system messages, and so on?

Using the design of a payroll program as an example in this chapter, each topic will be discussed along with the AS/400 system support for handling the creation of the application program.

The payroll program will need employee, tax, and salary information. Checks and payroll reports will need to be created including end-of-the-year benefit forms. The same files could be used to produce data to be included in the quarterly corporate income and expense report. Management may also need summaries of overtime from the employee time card files to project employee requirements for the coming year. Designing the files for current, as well as future needs, involves careful thought and planning. The first step is to determine what exists today.

Naming Conventions: If programs and data files exist on the system, is there a naming convention that has been used? If so, following that convention makes managing of the applications easier. A naming convention helps organize the data and makes the program documentation and other types of operations, such as displaying or copying files, easier. If there are no conventions, then consideration of such conventions is recommended. For example, all employee data files could begin with EMP and the payroll files could begin with PAY. Additionally, the OS/400 programs allow operations to be performed on multiple files using the generic names. For example, to copy all EMP or PAY files, the system permits copying groups of files by using generic names such as EMP* or PAY* instead of listing each file name separately.

The file name, record format name, and field name can be as long as 10 characters and must follow all system naming conventions. Keep in mind that some high-level languages have more restrictive naming conventions than the AS/400 system does. For example, RPG/400 allows only 6-character names. In some cases, the system name can be changed to temporarily match the high-level language restrictions. In addition, names must be unique as follows:

- Field names must be unique in a record format
- Record format names and member names must be unique in a file
- File names must be unique in a library

Source Files: If data currently exists on the system or network, determine the type of data, where it is stored, who owns it, and who can change it. Some existing program may update the files the new application will be using. If so, the interaction between programs must be considered to ensure that there are no conflicts. Updates to each program should be made so as to have as little negative affect on the other application programs as possible. There may also be source data that could be incorporated into the overall application. For example, a report format using the employee file could be changed to produce a summary payroll report.

User Interface Guidelines

Another consideration for the programmer is to match the users who will be using the application program and how they will use it. For example, menus that will be used by the data entry operator should take into account:

- The existing company standards for application programs
- The task that will be done using the display (entering data, selecting jobs, and so on)
- The type of work station that will be used and its characteristics (the device description files, for both printers and displays, provided by the operating system help the programmer control work station differences)
- The flow of information (which menu is presented first, second, and so on)
- The number of fields that should be included on each display

IBM has established SAA common user access (CUA) guidelines for designing a user interface. Following these guidelines allows interactive application programs to have similar appearance across several SAA operating system environments with a minimum amount of retraining for the application user.

Consideration should also be given to what type of messages and help information will be provided by the application program. For example, if the wrong data type is entered, will the program detect an error and what kind of assistance will the program provide to help the user correct the entry? In the payroll example, suppose letters are entered for the rate of taxation (*l* instead of *1*), how will the system prompt the user to enter only numeric values? Using DDS to describe files, validity checks can be incorporated into the design of the database.

Application Environment

The environment of the application program includes:

User profiles	Who can run the program and who will be accessing which files with what authority
Libraries	Where the program and file objects will reside
Library lists	What search sequence will be used for locating related information

Output queues Where printed output will be sent by the application program

Programming environment
System/36 environment, System/38 environment, or the OS/400 program

Each of these will need to be defined in terms of the specific application program. For example, the AS/400 system supports a group profile that could be created for data entry personnel to allow several people the authority to run the program that updates the employee files. Individual user profiles could apply to the person(s) responsible for maintaining the application programs.

Libraries should be created to contain all of the related data files and programs (for example, payroll). If employee files exist in more than one library or if separate libraries contain data the new application will be using, a library list will be needed to help the application program locate the required files. (While testing the programs, the library list should be changed to direct the programs to a test library containing test data so that any existing *real* data is not affected.)

Output for the payroll application will include pay checks and tax forms, as well as various summary reports. Output queues need to be created to manage when and where the output will be produced. For example, because the output is printed on company checks, it needs to be sent to the printer that will handle printed forms.

System Resources

A final task in the survey of the current system involves identifying system resources. This includes:

Hardware: Determine what hardware is available. This includes the amount of storage available for processing and storing the information as well as any devices attached to the system that can be used by the application program. When planning the printed output, what printers are available and what is the current work load? (An option may be to hold the output until the work load on the system is lightest.)

Network Considerations: Determine how the program will share data and resources with work stations and local, remote, and distributed systems in the network. Will users at remote sites have access to the information? Will the processing be done locally or at another host system? If it is a large network, will the information be entered at remote system work stations and then sent to a central site for processing or will all processing take place locally?

Specific OS/400 work management and data management functions provide support to manage these kinds of tasks including DDM, the communications device (ICF) files, and prestart jobs. The communications utilities and problem determination functions provide networking support such as APPN and monitoring alerts.

Programming Tools: Determine what programming tools are available. Several high-level programming languages are available for the AS/400 system including:

- AS/400 BASIC
- AS/400 Pascal
- AS/400 PL/I
- C/400 language
- COBOL/400 language
- RM/COBOL-85 language
- FORTRAN/400 language
- Procedures Language 400/REXX

- RPG/400 language

The languages can be supplemented by many high-level functions available with the OS/400 control language. For example, CL allows the programmer to use the AS/400 database and work management functions such as spooling and managing job processing.

The AS/400 application development tools provide utilities to help the programmer create the application program. These programming tools include:

- BGU for creating graphics
- CGU for creating DBCS characters
- DFU for working with data files
- PDM for interfacing with other programming tools
- SEU for entering and editing source
- SDA for designing menus and user interface displays

After the application guidelines, environment, and system resources are defined, the next step is to identify specific requirements for the application program.

Requirements

Identifying requirements for the application program will determine what programs, reports, data files, and other objects will be needed. This includes defining the purpose of the application, identifying users, determining data security required, and establishing performance requirements.

Purpose of the Application: Talking to the prospective users is a good way to identify how the task is currently being done, what information is required to do the task, and what reports would be helpful. In terms of the program, this can be converted into the display formats, the fields in the database file, and the printed output. For example, if the users indicate that the checks are mailed, then the employee database must have the name and address fields; the input display must allow users to enter and update the address information; the name and address must appear on the check to be displayed for mailing.

Identify Users: Once the purpose is clearly defined, it is easy to match the users to each task performed by the application. It will help to think in terms of your user and how to make their job easier. For example, too many options on a menu or fields on an input display can be overwhelming and confusing. The flow from menu options to data entry displays to printed output should be easily understood. The prompts should be clearly stated so that the user has no trouble understanding what is expected in the data entry fields. The testing phase of the application program should include a review of the displays and the flow by the prospective users.

Security: Matching the users to the tasks they will perform will determine the security level required. This matching should be detailed for each database file and program. OS/400 security functions allow user and group profiles to be easily created to specify each level of security for each database file or program. In the payroll example, the user responsible for updating salary information would have read and update authority to a different set of database files than the user whose task is limited to updating employee demographic information. System support gives the user of the program the same authority associated with the program unless otherwise specified in the individual user profile. If users have authority to run a program which updates the salary database, they can update the database unless their user profiles specifically prohibit it.

Performance: When considering the performance requirements of the application program, decide which tasks will be performed interactively and which will be processed as a batch job. For example, updating employee record information is best done interactively. Depending on the size of the company, this may need to be done on a daily basis. Processing and printing checks, however, is a batch operation, because the only direct interaction between the system and the operator required is to load the correct forms. The timing of this task would depend on when checks were issued and when reports were due. When the application program is put into effect, the OS/400 performance tools can be used to improve the system performance. Factors which affect system performance include:

- Interactive throughput
- Internal and external response time
- Active display stations
- Keyed input + think time
- Resource utilization
- Programming environment

Concentrating on one area, could adversely affect another. For example, if users want fast response time, the system needs to be designed and operated so that the users receive stable, consistent response time over a range of system loads. This choice, however, could cause batch jobs to run slower.

Design

After defining the user's needs and specifying the requirements for the application program, you are now ready to begin the design process. A very complete design specification will make the application program much easier to create. Figure 12-1 on page 12-7 provides an overview of the environment for the application program that will be created. This includes the files, languages, and other objects such as journals for recovery purposes. Including these in the design process makes it easier to accomplish than adding it after the application programs are completed.

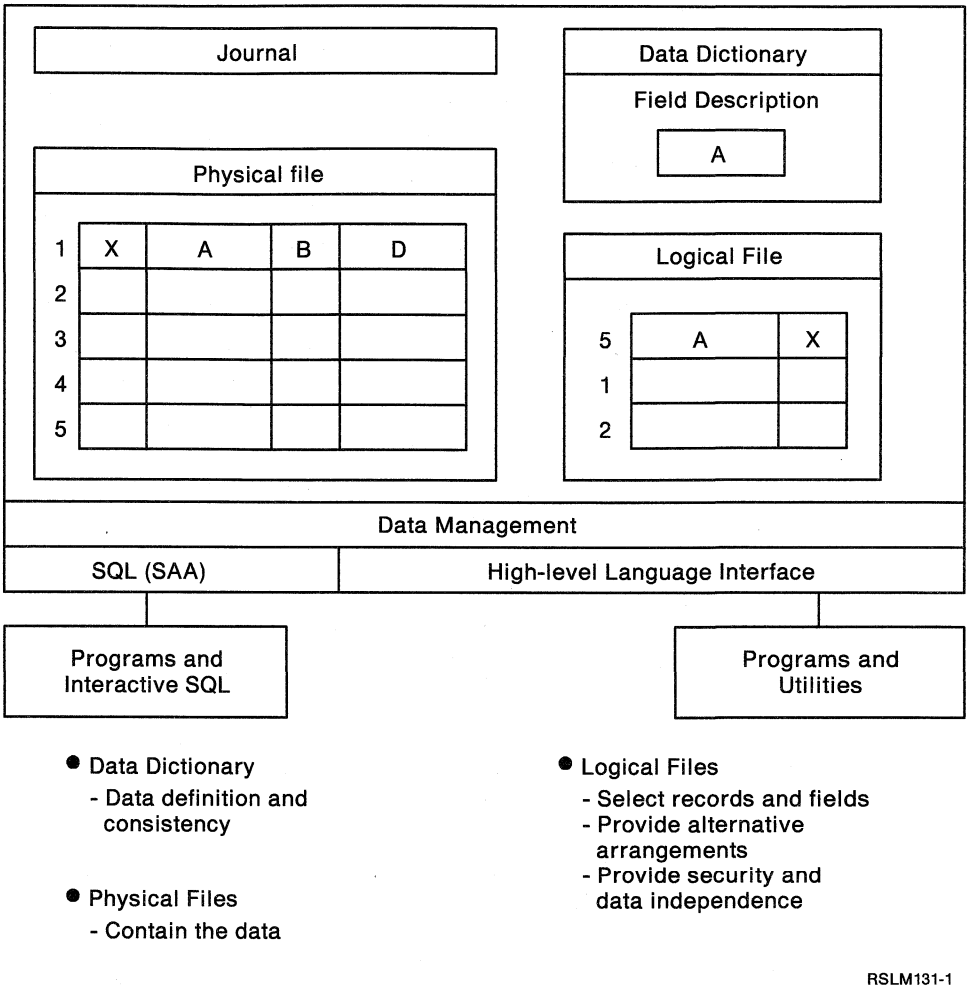


Figure 12-1. Environment of the Application Program

Designing the application program includes determining:

- Input and output methods and devices
- Processing options
- Performance specifications
- Security levels
- Program structure
- Message requirements
- Implementation tools
- Testing strategy

Input and Output

Designing how input and output will be managed by the application program includes:

- Determining which physical and logical database files are required
- Determining how the data will be described to the application programs
- Determining the characteristics of the printer and the format for each printed report or document
- Determining the characteristics of the displays and the format for the menu, entry, and list displays

Describing the Data

There are two ways of describing records in a database file:

- Field-level description. Each field in the record is described to the database system in terms of: name, length, data type, validity checks needed, and text description.
- Record-level description. Only the length of the record is described to the system. Each program must include the information about the fields.

The OS/400 program provides several methods to describe data for the database:

OS/400 IDDU
: The interactive data definition utility (IDDU) file definitions are stored in the OS/400 data dictionary. IDDU is a menu-driven, interactive function of the operating system that allows you to describe multiple-format physical files for use with QUERY, PC Support/400, OfficeVision/400, and the data file utility (DFU). Users familiar with describing data on the System/36 might choose IDDU.

SQL/400 language
: The Structured Query Language/400 (SQL/400 language) is the SAA database language for defining and processing data.

OS/400 DDS
: Data description specifications (DDS) provide the most options, including describing fields in logical files and key fields used to define the order in which data is presented to the program. DDS keywords also provide some options to process data before it is presented to the program.

The AS/400 system data management function allows field information to be defined as part of each physical database file, as was discussed earlier, or in a separate *field reference physical file*. The field reference file contains no data and is used only as a reference for the field descriptions for other physical and logical files. This is done by telling the system to copy descriptions from the field reference file into the new file descriptions. It is used to simplify record format descriptions and to ensure field descriptions are consistent. All fields required for an application program or group of files can be defined in the field reference file.

After the field reference file is created, physical file record formats can be specified without describing the characteristics of each field in each file. As the physical file is specified, the programmer can make changes to the field reference file instead of to the individual file descriptions. Changes to the field descriptions and keywords specified in the physical or logical file descriptions override the descriptions in the field reference file.

In the example, the field reference physical file could contain a description of all of the possible fields such as salary, tax rates, time card data, and employee information. The data would reside in separate database files whose descriptions would be copied from the field reference file. For example, the field descriptions for the employee name could be referred to by the time card and employee database files. The following must be determined for each field:

Data type
Length
Decimal positions
Text
Column headings
Alias
Attributes
Validity checking

Editing
Keyboard shift

Any high-level programming language restrictions, national language considerations, or programming conventions that may affect the description format must be identified. For example, not all HLLs support all data types. Some national languages use periods, commas, or slashes as date and monetary delimiters.

National Language Support

In those systems or network configurations where more than one language is supported, the users and applications must be aware of the following items:

- What national language version for the primary language is installed
- What national language version for the secondary language is installed, if any
- What release level of the national language version for the primary language is specified
- What release level of the national language version for the secondary language is specified
- What local work station controllers support the language
- What display stations and keyboards support the language
- What printers support the language
- What keyboard ID is used for the local devices
- What remote work station controllers support the language
- What display stations and keyboards support the language from a remote location
- What printers support the language from a remote location
- What keyboard ID is used for the remote devices
- What applications support the language on the local system
- What applications support the language on the remote system

Format of Printed Output

In determining the printed output for the application, there are two basic topics:

1. The device: where and how will the output be printed?
2. The format: what will the output look like?

Device: The capabilities of the printer will determine how the system will communicate with the printer and what the printed output can look like. You will need to determine which printer will be used for which report. This will determine the output queue that will receive the spooled files. For example, the printing of the payroll checks may require a printer with a form feed attachment to feed in the check forms. The summary report for the end of the corporate year may require a printer capable of graphics or different type styles.

The device description identifies the hardware capabilities to the OS/400 program, including whether spooling is active. The application program refers to the device file rather than to the actual device. Using these objects, OS/400 data management support controls the processing of data between the program and the printer.

The current work load of the printer needs to be assessed so you can plan the timing of the actual printing. For example, printing the checks will require loading

the forms and interrupting the flow of other work to that printer. Other spooled files to that printer will need to be held until the checks are finished and the regular paper is reloaded. The OS/400 work management functions easily manage these spooling tasks.

You will also need to determine what will happen to the output when printing is complete. For example, the overtime summary report could be saved in a database file to later be merged into a document, which management will present to justify increasing the number of employees.

Printing conditions that could cause errors will also need to be considered in the design of the application program. For example, will there be a message to the program or operator if an error occurs? How should the program handle unprintable characters? Should they be ignored or should the program stop? Will the program restart the process if it is interrupted? In the case of printing checks, the program will need to *mark* the last check that was successfully printed so that it can continue from that point when restarted.

Format: When the printer characteristics are identified, the format for the output can be determined. For example, if the printer can accommodate the various national languages (for example, is all points addressable), then the AS/400 character generation utility can be used to produce the output in the DBCS format. (There are already over 7000 DBCS characters available.) Depending on the capabilities of the printer, the AS/400 advanced function printing (AFP) allows the user to print the Advanced Function Printer Data Stream (AFPDS) data sent from a System/370 host to place data not just at row and column positions, but at any addressable point on the page. This includes graphic images, various fonts, logos, signatures, electronically designed forms, and so on. In the payroll example, the blank check forms could even be printed on site using AFP.

The content of the printed output must be selected and considered when planning the application program. This can be done by meeting with those who will be using the final product and creating prototypes of the printed output. The prototype of each report can be used to plan the layout of the fields.

Focus on the purpose and audience of each report and printed output. For example, if it is a summary report, then pages of numbers that lead to the *bottom line* are not appropriate. However, if the purpose is to determine how the *bottom line* was achieved, then the numbers are important and, perhaps, those which deviate substantially from the average should be highlighted.

With the prototypes and the purpose clearly defined, the next step is to determine the fields. Using the field reference file as a starting point, plan any changes that need to be made to the fields such as: data type, field length, decimal positions, and other attributes. Any data that is not included in the field reference file needs to be added either to the existing field reference file or to a new database file. System variables, such as the date and time, also need to be described.

A final step is the actual form of the output. This can be specified in the OS/400 printer files that can be referred to by each program within the application. Printer attributes can be specified for each field, such as underlining, type size and style, and so on. DDS can also be used to describe the printed format or the format can be specified in the program. DDS provides more functions.

In addition to using the high-level languages to create the output, the AS/400 system provides the RPG/400 language available in three forms. System/36-Compatible

RPG/400 language is used in the System/36 environment. System/38-Compatible RPG/400 language is an enhanced version of the System/38 program and runs in both the System/38 environment and in the OS/400 licensed program. The RPG/400 language can be run in the OS/400 environment and can be incorporated into other RPG/400 and HLL application programs using a simple call statement.

Format of Displayed Output

The displays serve as the user's interface to the application program. These displays must be designed to accommodate the user, the type of work station where they will be displayed, the purpose of the display, and the content of the display.

Define the Device: Like the printer, the AS/400 system maintains a device description for each display station so the programmer need not define the device for each program. The following items must be considered when designing the display:

Display size
Color
Type of keyboard
Is it a programmable work station?
Is it a local or remote work station?

Flow of Information: A good way to determine the flow of display screens in the application program is to map the user's route through the programs. For example, in the example application program, the first display is a general menu that gives the user choices among the payroll tasks. Each subsequent display depends on the task the user chooses from the menu. The following example shows a menu that could be used for the payroll application.

```
                        Company XYZ Payroll Menu

Select one of the following:

     1.  Data entry and update
     2.  Run reports
     3.  Process payroll
     4.  Application maintenance menu
     5.  Batch job menu

    90.  Sign off









Selection or command
===> _____________________________________________________________
_____________________________________________________________

PF1=HELP              PF3=Exit                    PF12=Cancel
```

Depending on the user's authorization to each of the database files, the menu options would display a different set of options. The application program can be designed so that the display reflects the user's experience or security level. For example, the personnel manager might see only options 1 to 3. The user responsible for systems operations might only see 2 and 4.

Design Displays: As discussed earlier, there are four types of displays that can be used in a program:

Menu	From the first menu, the subsequent menu choices might allow the user to specify the database to be updated, the report to be printed, the payroll information to be processed, and so on.
List	If several reports need to be printed, the list display allows users to enter multiple selections. The program then processes each report.
Entry	The data entry displays would include the fields used to type employee data as well as any other information that is to be included in the database.
Information	This might include information, such as how to load the payroll check forms into the printer or even an explanation for each option on the menu. A function key can be programmed into the application as a Help key. The user moves the cursor to the option and presses the Help key and the information panel is displayed.

Displays can also be a combination of all four types.

For each display, determine the following:

- *Fields*: Which fields are required and what are their attributes. For example, when entering the number of hours an employee worked, a decimal point is required to indicate a partial hour.
- *Function keys*: Which function keys will be active for which displays and how will they be indicated on the display. For example, F1 is a CUA standard for help.
- *Help*: What help is available and how does the user access it. For example, is it task specific or is does it contain overview information about a menu?
- *Message handling*: What messages are to be sent and how will they be displayed to the user. For example, if an incorrect employee number is entered, how is that communicated to the user?
- *Overwrites*: Which information or fields are cleared, overwritten by a subsequent display panel, and which are saved to be displayed again. For example, when entering employee data on multiple screens, it is helpful to keep the employee name for each data entry display for that employee.

The application development tool screen design aid (SDA) is an interactive utility to design menus, to design displays, and to create online help information. Some of the advantages of using SDA are:

- Generates data description specifications (DDS) from the information that appears on the screen. An extensive knowledge of DDS coding forms, keywords, or syntax is not necessary.
- Presents displays in groups to make DDS keyword selection easier at the file, record, or field level.
- Allows field selection from existing database files to design a display.
- Allows the programmer to view the display as it is designed or changed.
- Provides assistance for testing of the displays with the data and status of the conditioning indicators.

Validity Checking

Determine what type of validity checking the system should perform when the user is entering data. Using the DDS keywords in the display file that was created, the AS/400 system performs the validity checking specified in the source statements before sending the data to a program when the program is run. Therefore, any errors can be corrected before the data is sent to the program. For example, when using display files, you can have the system check the validity of data entered by the work station user. If errors are found, a message can be issued so that the work station user can correct the input before the record is passed to the application program. In the payroll application program, validity checking could be used to check the validity of department and employee numbers, comparing those entered by the operator with those in the master database file.

Providing Online Help

Determine what help information should be available for each area of the display and how the help should function. Online help information can be defined in an office document, DDS record formats, or user interface manager (UIM) panel groups. Using the DDS keywords in the display file, the AS/400 system displays the text when a user presses the Help key

The user can define the display file to supply online help information for the following:

- Smaller rectangular parts within a record, such as individual fields
- Parts of the display that are not part of any smaller rectangular parts that already have online help information defined for them

Each help area is defined with H specifications in DDS with the Help Area (HLPARA) keyword. A help area is defined by specifying the upper-left row and column and lower-right row and column of the rectangular area. These coordinates must be located in the screen area, but are not required to be in the record area.

Only one file-level help keyword (either HLPDOC, HLPRCD, or HLPPNLGRP) can be defined, but the user can specify one or more H specifications for a record.

The UIM help facility uses panel groups to hold online help information. To use panel groups for online help information, the user must specify them in the DDS coding for the applications display. Panel groups can contain three types of online help information:

- Display help
- Command help
- Search indexes

Panel groups can be linked to each other using hypertext.

DDS uses help records to define online information.

Online documents can be used for online help information. These documents are created using the word processing functions in the OfficeVision/400 licensed program. The document and folder do not have to exist when you create the display file. Users do not need the OfficeVision/400 program on the system to run online help information, but they do need the licensed program to create an online help document.

Identify Database Requirements

During application design, the following decisions must be made about the use of database files:

- What files, record formats, and keywords are needed? As the physical and logical files are described to the system, DDS, IDDU, and SQL allow keywords to be specified to create access paths to the data. These are used by the programs to retrieve information in the order required by the program.
- Are new physical files needed, or does the data already exist on the system?
- Are new fields or changes to existing fields needed?
- Are new logical files necessary to provide the record formats and/or access paths needed by the program?
- Are the record formats and fields used by the program already defined in the system?

The AS/400 programming development manager (PDM) and other system commands provide the functions necessary to display the file descriptions, record formats, and file use information for files that already exist on the system.

Determine Requirements for the Database: Knowing what the output of the application should be helps to identify what input is required. Whether the information already exists on the system or whether new files are required, the following is a list of some of the requirements that must be determined:

- Security and file integrity requirements
- Fields and their attributes
- Program-described or externally-described data
- Types of logical files that will be used
- Arrangement of the data
- Transaction management requirements
- Strategies for avoiding duplication
- Data sharing requirements

Design Physical Files: The physical file contains the actual data that is used in the application program. Where possible, the design should include the use of existing record formats and a strategy for avoiding duplication of information. Additionally, the design includes determining:

- Which fields are required in each physical file (For example, an employee database should contain name, address, telephone number, and so on)
- Attributes of the fields (For example, length, data type, name, decimal positions, and so on)
- Access method (For example, keyed or arrival sequence)
- Join fields (For example, joining the first name and last name fields to create a *name* field instead of duplicating the data)
- The creation method
  - Size of the files
  - Storage requirements

Design Logical Files: The logical files contain no data, but define the record formats for the data used in the application program. To create the paycheck in the payroll example, logical files may be used to present a view of:

- The employee information
- The amount of time worked during the specific time frame
- The rate of pay
- The taxes and other amounts to be deducted
- The net pay

Processing all of this information from the various physical files is made easier because the data management functions provide the application program with all of the data as though it were a single file. The combination of fields used to create the various logical files must be determined. Like the physical file, the fields and their attributes must be chosen and identified. However, with logical files there are additional things to be determined:

- The type of logical file
  - Simple
  - Join
  - Multiple-format
- The source of the data
  - Existing physical file
  - Calculation

Because data can have several logical views, the data can be read and processed using different record formats and access paths than the ones used to store the data. Figure 12-2 shows how data of three different kinds can be linked together in one relational database but handled in three different views.

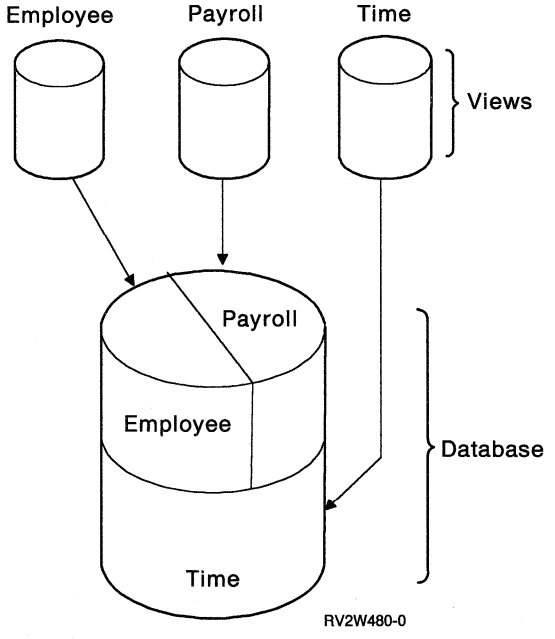
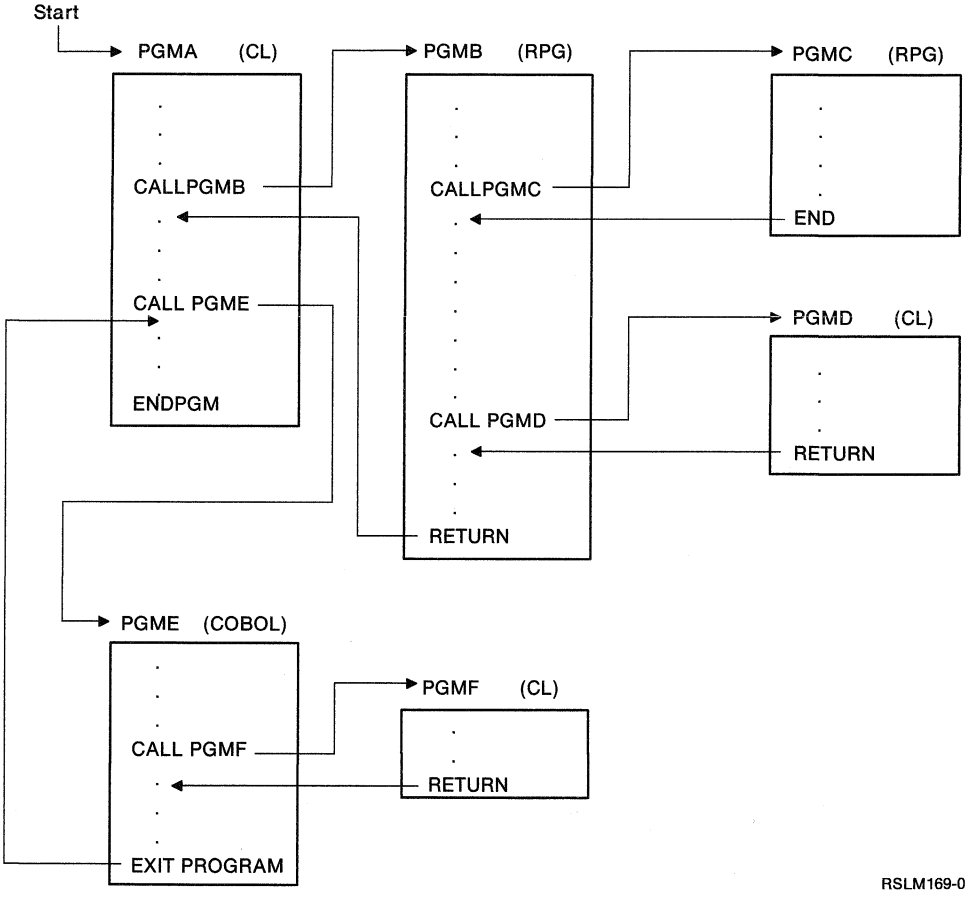


Figure 12-2. Example of a Relational Database

Structure

Within an application, each program can be designed to perform a specific function. Whenever the function is needed, that program can be called. When applications are organized in this manner, one program in the application (normally a CL program but it could be a REXX program) controls the flow of activity within the application. The following illustration shows how control could be passed between a control language program, RPG/400 programs, and COBOL/400 programs in an application. To use the application, a work station user would request program A (PGMA), which controls the entire application. The illustration shows:

- A CL program (PGMA) calling an RPG/400 program (PGMB)
- An RPG/400 program (PGMB) calling another RPG/400 program (PGMC)
- An RPG/400 program (PGMB) calling a CL program (PGMD)
- A CL program (PGMA) calling a COBOL program (PGME)
- A COBOL program (PGME) calling a CL program (PGMF)



When the application is developed, each program should be written in the high-level language that best provides the functions needed. When deciding which language to use performance should be considered. For example, an interpretive language does not normally run as fast as a compiled language. If performance is critical, a CL program would be preferred over a REXX program if both languages provide the desired level of support. Any program can call any other program regardless of the high-level languages in which the programs are written. For example, control language statements can be used by a work station user to request application functions. Control language statements also can be used to call other programs based

on conditions that exist when the programs run. The high-level languages provide the functions needed to perform operations on the data processed by the application. These programs can request database processing through the data management functions provided by the system.

Using the operating system functions, applications that are made up of more than one program can pass information from one program to another using any of the following:

- Parameters of the command that calls the program
- Data areas
- Files
- Data queue
- Messages supported by the OS/400 message handling functions

Data Areas: A data area is an object that contains common data that can be shared by different programs in a job or by programs in different jobs. It exists independently of the programs that use it. Values can be placed in the data area to control the functions performed by programs that access those values. The values can be changed by the programs or by commands entered by a work station user. The system synchronizes the use of values in the data area so that one program does not change a value while another program is using it.

A local data area is created automatically for each job in the system, including autostart jobs, jobs started on the system by a reader, and subsystem monitor jobs.

A local data area does the following:

- Pass information to a submitted job by loading information into the local data area, then the data can be accessed from within the submitted job.
- Improve performance over other types of data area accesses from a CL program.
- Store information without the overhead of creating and deleting a data area for each submitted job.

Data Queues: Data queues are a type of AS/400 system object that you can create, to which one HLL program can send data, and from which another HLL program can receive data. The receiving program can be already waiting for the data, or can receive the data later. Normally, the programs are in different jobs.

Some of the advantages of using data queues are:

- Data queues are the fastest means of asynchronous communication between two jobs. Using a data queue to send and receive data requires less overhead than using database files, message queues, or data areas to send and receive data.
- More than one job can receive data from the same data queue.
- If the job is an interactive job, this can provide better response time and decrease the size of the interactive program. This, in turn, can help overall system performance.

Figure 12-3 shows the relationship of programs, database files, message files and queues, device files, data area objects, and data queues. Notice that all of the programs, no matter which HLL language was used, can access the same files and queues.

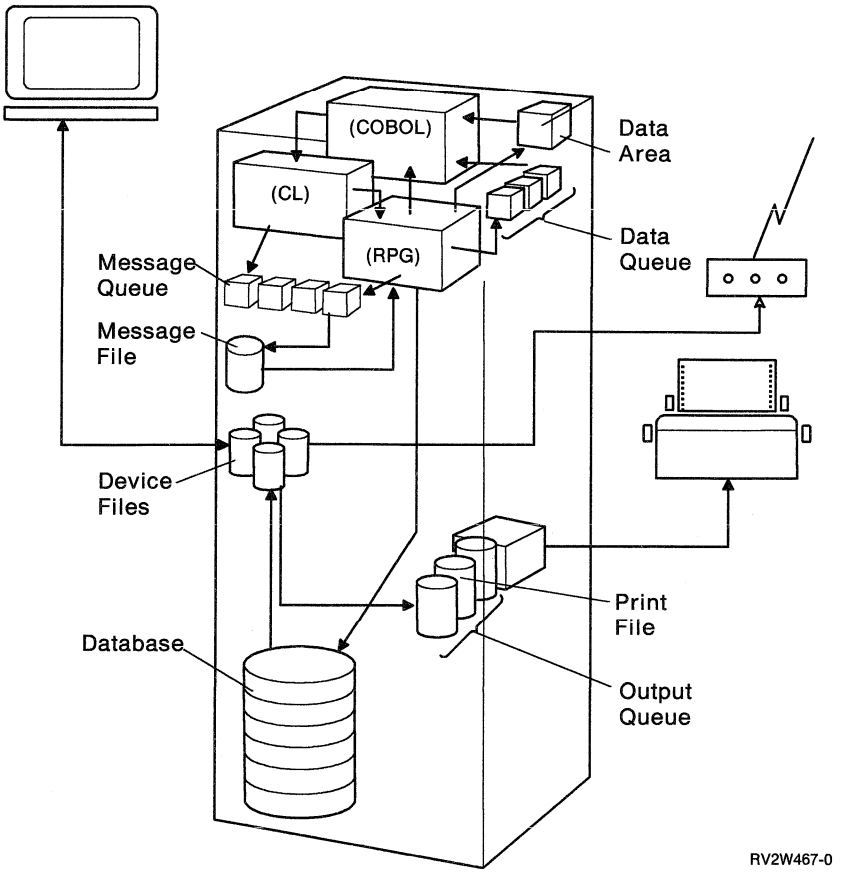


Figure 12-3. Accessing Files

Batch or Interactive

Another major design consideration is whether to do an interactive application or a batch application. In many cases, a complete application is a combination of interactive and batch programs. For example, if an application requires documents to be printed (such as picking slips or purchase orders), that part of the application is best performed by a batch job. However, the entry of data that updates master files on a timely basis is best performed interactively. Generally, any task that ties up a work station for more than a short time (while the task runs and the work station user cannot interact with the system) should be run as a batch job.

Three primary elements should be considered when an interactive or batch application is designed:

- The user interface needed to call and to communicate with the application
- The files needed for the application:
  - Database files for storing data
  - Device files for output or communicating interactively with users
  - Messages files
- The structure of the application; that is, the programs to be included and the method used to control flow among the programs

Using the work management functions on the AS/400 system, batch jobs can be submitted by work station users, started by the system operator, or called from an interactive program. Work station users can submit jobs whenever they are needed.

Batch jobs can also run at a specified time or when a subsystem is started. The scheduling priority for the jobs can be set in the job description.

When an interactive program is running, communication is needed between the program and the work station user. This communication is normally performed through displays defined in a display device file.

User and Interprogram Messages

A *message* is a communication sent from one user or program to another. Most data processing systems provide communications between the system and the system operator to handle errors and other conditions that occur during processing. In addition to this type of support, the OS/400 program provides message handling functions that support two-way communications between programs and system users, between programs, and between system users. Two types of messages are supported:

- Impromptu messages, which are created by the program or system user when they are sent and are not permanently stored in the system.
- Predefined messages, which are created before they are used. These messages are placed in a message file when they are created, and retrieved from that file when they are used. The messages are numbered and can be defined by either the program or the system. For example, if an application program attempts to write information to the tape drive and the drive is not ready to receive the data, the system message file contains a predefined message for that condition.

Because messages can be used to provide communications between programs and between programs and users, using the OS/400 message handling functions should be considered when developing applications. The following concepts of message handling are important to application development:

- Messages can be defined in messages files, which are outside the programs that use them, and variable information can be provided in the message text when a message is sent. Because messages are defined outside the programs, the programs do not have to be changed when the messages are changed. This approach also allows the same program to be used with message files containing translations of the messages into different languages.
- Messages are sent to and received from message queues, which are separate objects on the system. A message sent to a queue can remain on the queue until it is explicitly received by a program or work station user.
- A program can send messages to a user who requested the program regardless of what work station that user has signed on to. Messages do not have to be sent to a specific device. A program can be used from different work stations without changing the program.

Because replies can be returned by a user or a program that receives a message, the message handling function allows two-way communications.

Messages in a CL Program

You can monitor for escape, notify, and status messages that are sent to your CL program's program message queue from other programs using commands in your program. The Monitor Message (MONMSG) command monitors the messages sent to the program message queue for the conditions specified in the command. If the condition exists, the CL command specified on the MONMSG command is run. The MONMSG command does not detect Information, Diagnostic, or Completion messages.

Escape Messages: Escape messages are sent to tell your program of an error condition that forced the sender to end. By monitoring for escape messages, you can take corrective actions or clean up and end your program.

Notify Messages: Besides monitoring for escape messages, you can monitor for notify messages that are sent to your CL program's program message queue by the commands in your program or by the programs it calls. Notify messages are sent by these programs to tell your program of a condition that is not typically an error. By monitoring for notify messages, you can specify an action different from what you would specify if the condition had not been detected.

Status Messages: Another type of message you can monitor for is the status message. You can monitor for status messages that are sent by the commands in your CL program or by the programs your program calls. Status messages tell your program the status of the work performed by the sending program. By monitoring for status messages, you can prevent the sending program from proceeding with any more processing.

Information to Support the Application

After the reports and database files are designed, you may want to design the help information that will support the application programs. This can include help provided by the system through the user interface functions, education in the form of either online tutorials or user guides, and manuals and run books. The level of difficulty and detail should reflect the user's experience. For example, the individual responsible for updating payroll information requires a run book that provides the basic keystrokes necessary to make menu selections and to enter data. The system operator responsible for application maintenance tasks requires a reference guide with problem determination and resolution information as well as backup and recovery strategy. In a small business, the backup and recovery procedures are often part of the application. In larger enterprises, they are included in the overall management of the system.

Implementation Tools

To a great extent, the choice of tools depends on the tools and programming languages available on the system, the language with which the programmer is most familiar, and existing application programs on the system. The AS/400 system provides a wide range of tools and programming languages to help you develop applications.

Programmer Menu

The Programmer Menu provides a convenient method for accomplishing general programming tasks. The programmer menu options include:

- Work with DFU and Query applications
- Create objects from a source file
- Call a program
- Run a command
- Submit a job
- Go to another menu to change a source file member
- Design a display format using SDA

Application Development Tools

Application Development Tools is a set of programs specifically designed to help programmers develop application programs. The tools provide help for such tasks as managing objects, entering source, and designing displays.

Programming Development Manager (PDM): The programming development manager (PDM) utility provides easy access to objects on an AS/400 system. PDM allows the programmer to:

- Work with lists of libraries, programs, files, and other application objects
- Perform a number of different operations such as copy, delete, and rename quickly and easily

PDM performs these operations by running commands with known parameter values passed from the list. You can perform many operations without having to know particular commands. For example:

- Using the list displays, the user can perform different operations or the same operations on more than one item in the list at the same time. This includes libraries, files, members, and user-defined options. For example, in the payroll example, you may need to move or delete the application test files. Using this option, select the objects and the operations. PDM handles the multiple moves or deletes.
- The work option allows the user to work with all objects in a library or all members in a file. In the development cycle of the application program, the programmer can choose to work with all of the programs that will be used to create the payroll checks.
- Users can also create their own options, known as user-defined options, and use them to perform operations on libraries, objects, and members. Any frequently done task can be added or deleted as the application development progresses (for example, save and restore options or compile sequences).

Because operations can be performed on more than one item at a time, productivity is increased.

Source Entry Utility (SEU): SEU provides specification displays for DDS, RPG, CL and other HLLs for entering and updating source members. SEU functions include:

- Creating new members
- Updating existing members in a source physical file
- Searching by character strings and making global changes on that string
- Displaying prompts for source languages, CL commands, and BASIC
- Checking syntax for all source statements

Data File Utility (DFU): DFU creates data entry programs from the information in the descriptions of existing database files (such as length, alphanumeric, and so on) and then uses these descriptions to build the application program. DFU programs can perform several jobs, such as entering new records, updating fields, and performing file inquiry tasks. Therefore, database maintenance programs run faster.

A DFU list utility is available on in the System/36 environment and helps the programmer work with list programs.

Screen Design Aid (SDA): As discussed under the display design, SDA helps the programmer create display files that can be referred to by the HLL or called from a CL command.

Character Generator Utility (CGU): If double-byte characters (DBCS) are required for the application program, CGU helps the programmer create, delete, and change DBCS characters as well as maintain the font and sort tables for the character set.

Advanced Printer Function Utility (APF): When working with IBM 5224 and 5225 printers, programmers can use APF to:

- Define alternative character sets and symbols
- Define large characters
- Print large characters
- Build and print logos and emblems
- Generate bar graphs
- Produce a customized form that can be merged with spooled application report data

Report Layout Utility (RLU): When programmers are designing reports RLU allows them to focus on the layout and contents of the report rather than the DDS which describes it. RLU allows programmers to create and edit report images described in DDS on the AS/400 system. A report image is a prototype that looks like an actual listing produced by a high level language application. The RLU report development environment allows programmers to design reports by:

- Laying out an image on the display while RLU interprets the design and generates DDS
- Defining the record formats and fields while RLU builds the report image

Interactive Data Definition Utility (IDDU)

The interactive data definition utility (IDDU) defines data to be used by DFU, AS/400 Query, PC Support/400, OfficeVision/400, HLL, and to be embedded in documents. (DDS can also be used.) IDDU allows the programmer to enter database file and format descriptions to create database files. IDDU is the only way to describe multiple-format physical files, which are used by Query, PC Support, and DFU. These can be referred to by the programs in the application to provide external description capabilities.

Programming Languages

The choice of the programming language for the application is usually determined by the programmer's skill in that language and by the functions provided by the language. With the AS/400 system, programmers can choose among the following languages, and use the CL commands to call system functions and tie the various programs together:

AS/400 BASIC
AS/400 Pascal
AS/400 PL/I
C/400
COBOL/400
RM/COBOL-85
FORTRAN/400
Procedures Language 400/REXX
RPG/400

CL Programs: CL programs are made up of CL commands. The commands are compiled into a program that can be called whenever the functions provided by the program are needed. Advantages of using CL programs include:

- The user running the program does not have to enter individual CL commands.

- The CL programs are consistent.
- Using CL programs is faster than entering and running the commands individually, because the commands are compiled and stored in a form that can be run immediately.
- CL programs can perform various functions depending on the parameters used.
- Some commands cannot be entered individually and must be part of a CL program.
- CL programs can be tested and debugged like other high-level language (HLL) programs.
- Syntax checking is performed on the commands as they are entered.
- CL commands in CL programs can perform many functions not available in other high-level languages.

CL programs can be used for many kinds of applications. For example, they can be used to:

- Provide an interface to the user of an interactive application. This enables the user to request application functions without understanding the commands used in the program. This makes the work station user's job easier and reduces the chances of syntax errors occurring when commands are entered.
- Control the operation of an application by establishing variables used in the application (such as date, time, and external indicators) and specifying the library list used by the application. This ensures that these operations are performed in the proper order whenever the application program is run.
- Provide predefined procedures for commonly performed functions, such as procedures to start a subsystem, to provide backup copies of files, or to perform any other procedures. The use of CL programs to perform these procedures reduces the number of frequently used commands and ensures that the functions are performed consistently.
- Assist in the flow of programs and access to system functions. Using CL programs, applications can be designed with a separate program for each function, and with a CL program controlling which programs are run within the application. The application can consist of both CL and other HLL programs. In this type of application, CL programs are used to:
  - Determine which programs in the application are to be run.
  - Provide system functions that are not available through other HLL languages.
  - Provide interaction with the application user.

Most of the CL commands provided by the system can be used in CL programs. In addition, some functions designed for use in CL programs are not available when commands are entered individually. These functions include:

- Logic control functions that can be used to control which operations are performed by the program according to conditions that exist when the program is run. For example, *if* a certain condition exists, *then do* certain processing, *else* do some other operation. These logic operations provide both conditional and unconditional branching within the CL program.
- Data operations that provide a way for the program to communicate with a work station user. Data operations let the program send formatted data to and receive data from the work station, and allow limited access to the database.
- Functions that allow the program to send messages to the display station user.

- Functions that receive messages sent by other programs. These messages can provide normal communication between programs, or indicate that errors or other exception conditions exist.
- The use of variables and parameters for passing information between commands in the program and between programs.

CL programs provide the flexibility needed to let the application user select what operations to perform and run the necessary programs.

Procedures Language 400/REXX: REXX, the AS/400 system implementation of the SAA procedures language, is a versatile, easy-to-use, structured, programming language. It offers powerful string handling functions, extensive mathematical capabilities, and the ability to issue commands to a command environment that is outside of REXX. This allows CL commands to run from within a REXX program. REXX is thus an alternative to a CL program. You can combine the powerful instruction set of REXX with CL commands to control the overall flow of an application.

REXX is an interpretive language rather than a compiled one. As a result, REXX programs can be written, tested, and put into production faster than is normally the case with a compiled language. However, because it is interpreted, REXX programs may not run as fast as an equivalent compiled program.

AS/400 Pascal Programming Language: Pascal is a high-level programming language that can be used to process both numeric and textual data and provides automatic syntax error detection. Pascal can call procedures written in other programming languages and can be called by other programming languages. This means that existing programs can be incorporated into new applications.

AS/400 PL/I Programming Language: PL/I is a general-purpose programming language suited for commercial and scientific programming applications. PL/I can call procedures written in other programming languages to provide services not directly available in PL/I. Additionally, PL/I can be called by other programs. This allows the programmer to take advantage of existing application programs.

AS/400 BASIC Programming Language: BASIC programs are usually run interactively. However, CL commands can be used to run BASIC programs as a batch job. BASIC programs can be run either as source or compiled programs. BASIC supports the following kinds of files:

- Arrival-sequence and keyed-sequence database files
- Record-oriented device files
- Work station files (for devices)
- Printer files
- Streaming files
- Source file members
- AS/400 data areas
- Internal data files or program-described files

COBOL/400 Programming Language: COmmon Business Oriented Language (COBOL) is especially efficient for processing business problems. It emphasizes the describing and handling of data items and input and output records. This makes it well adapted for managing large files of data. The COBOL/400 language includes several enhancements to the ANSI COBOL'85 standard:

- Allows records to be sent and received from work stations
- Allows externally described files to be used
- Supports extended and Boolean data types

- Provides verbs to support the AS/400 database
- Allows the use of an apostrophe instead of a quotation mark
- Supports the SKIP, EJECT, and TITLE statements
- Supports field-level work station I/O
- Provides ways to define data name characteristics by copying them from previously defined data names

RM/COBOL-85 Programming Language: The RM/COBOL-85 programming language for the AS/400 system is an application language that conforms to the ANSI COBOL 85 standard, plus extensions that aid in writing interactive work station applications that are portable between several IBM and non-IBM platforms. It is a full level-2 implementation with the exception of variable length record support. It also accepts the syntax from RM/COBOL 2.0, which conforms to the ANSI COBOL 74 standard, to support easy migration of RM/COBOL-85 applications from System/36.

FORTRAN/400 Programming Language: FORTRAN is a high-level programming language used in applications for maintaining data and generating reports. It is useful in mixed environment programs that involve some numeric computations. The FORTRAN/400 language includes the following characteristics:

- Conforms to SAA FORTRAN Common Programming Interface (CPI)
- Adds useful extensions from FORTRAN/2* and VS/FORTRAN
- Provides an interactive syntax checker and debugger
- Can call programs written in other languages and be called by other languages
- Provides AS/400 system-unique extensions

C/400 Programming Language: C is a relatively low-level language and is well-suited for system programming. Other uses for C include data management, communication, and text processing. Programs written in C cover engineering, scientific, and commercial applications.

The C/400 language has the following characteristics:

- Conforms to SAA Common Programming Interface (CPI)
- Complies with the ANSI C standard
- Provides an interactive syntax checker and debugger like the FORTRAN/400 language and AS/400 Pascal
- Can call programs written in other languages and can be called by other programming languages
- Makes use of AS/400 system capabilities to provide features such as flexible file manipulation, data security and data integrity

RPG/400 Programming Language: The RPG/400 language uses the SAA definition of the RPG programming language and provides many AS/400 features that make the RPG/400 language well-suited for the AS/400 system. For example, the RPG/400 language uses externally described files and record formats. The features of the RPG/400 programming language allows for faster development. The RPG/400 language provides structured programming options such as sequential operations, conditional branching, and a full set of input and output support. The AS/400 system also supports compatible versions for the System/36 environment and the System/38 environment to run migrated RPG/II and RPG/III programs on the AS/400 system.

System Programmer Interfaces

OS/400 APIs are provided to give highly experienced system programmers the ability to create system-type applications. APIs are callable interfaces to low-level system functions. They provide access to information and function that is not available through CL commands. They also provide a more efficient access to select system information and functions such as objects, jobs, and spool files. APIs can be called from high-level language programs.

Creation

After an application is designed, the files used in the application must be described and created and the various programs must be coded and compiled. Data description specifications (DDS) are source statements for externally described data files. Control language (CL) commands and other high-level language source statements must be provided for the application program.

For programs or externally described files to be created, source files containing the source statements are needed. The user can place source statements in a source file by copying them from a device file or by entering the source statements through SEU. SEU provides special display formats to help the user enter specifications for other programs and for data description. The utility also can perform syntax checking on the source statements.

The AS/400 system provides the commands that are needed to create files and programs from the source statements contained in source files. These commands can be used in either interactive or batch jobs. You can also create files and programs through system menus or the Programmer Menu.

Testing

Testing the application programs involves setting up a test environment that is similar to the implementation environment. Authorities, files, utilities, hardware, and a communications test network should be available. Finally, verify that the output, database changes, and performance are as desired. Testing is typically a matter of test, correct, test, and so on.

The system includes functions that let a programmer observe operations performed as a program runs. These functions can be used to locate operations in the program that are not performing as intended. The OS/400 testing and debugging functions let the programmer:

- Stop a running program at any named point in the program's source statements
- Display and change information about program variables at any point
- Issue any command while the program is stopped
- Trace the use of variables in the program by recording the steps in the program that change the variables, and what those changes are

The testing functions narrow the search for errors that are difficult to find in the program's source statements. Often, an error is apparent only because the output produced is not what is expected. The AS/400 system supports a CL command that allows the programmer to add a breakpoint anywhere in the programming code. The breakpoint allows the programmer to examine variable information in the program to see if it is correct. The programmer might want to make changes to those variables before letting the program continue running. Through OS/400 group

job support, the editor can be running in one job group while the testing session is working in another. The programmer can specify a function key to move between the jobs as the need arises.

It is extremely important that the application program be thoroughly tested. As in the example application, it is easy to see that undetected errors could cause data integrity problems and unhappy users.

The test cases should resemble reality as closely as possible. If data exists on the system that will be used in the final implementation, select a subset of the data and copy it to a test library. (Remember to change the library list so the testing of the program involves only the test data.) It is also helpful to have *real* users contribute ideas to the design of the test cases. They can usually think of a few irregularities that the programmers, removed from the actual task, may not anticipate.

Implementation

The implementation phase of the application program includes educating the users with the appropriate tutorials or demonstrations, planning for maintaining the application with performance and application enhancements, and identifying strategies for dealing with any corrections. Since the application has been designed using the database, work, and system management functions of the AS/400 system, updates and enhancements should be easy to implement.

AD/Cycle Development Framework

The AD/Cycle development framework is the IBM SAA application development environment. Its primary goal is to help programmers improve the quality and productivity of application development and maintenance. This set of offerings for application development is based on the following:

- Comprehensive set of application development tools

  These are grouped conceptually into sets that focus on specific development life cycle activities. The tools share product information throughout the life cycle, from requirements definition through maintenance.

- Broad application development platform

  This platform provides tools that support:

  - Consistent user interfaces from tool to tool
  - PS/2 work station tools
  - Shared repository of information for development and maintenance of applications
  - Application development information model
  - Common tool services

- Open and extendable application development framework

  This framework provides both well-defined interfaces and common support functions for the tool set. To assist you with the integration of tools operating within this framework, IBM will publish specific rules and guidelines.

Figure 12-4 on page 12-28 shows that the AD/Cycle framework supports the traditional application development tasks that extend across the development life cycle. Moreover, the AD/Cycle framework integrates these tasks strategically with enterprise requirements and goals. Actual development approaches may vary from

these traditional phases, but the activities are typical of those in any application development process.

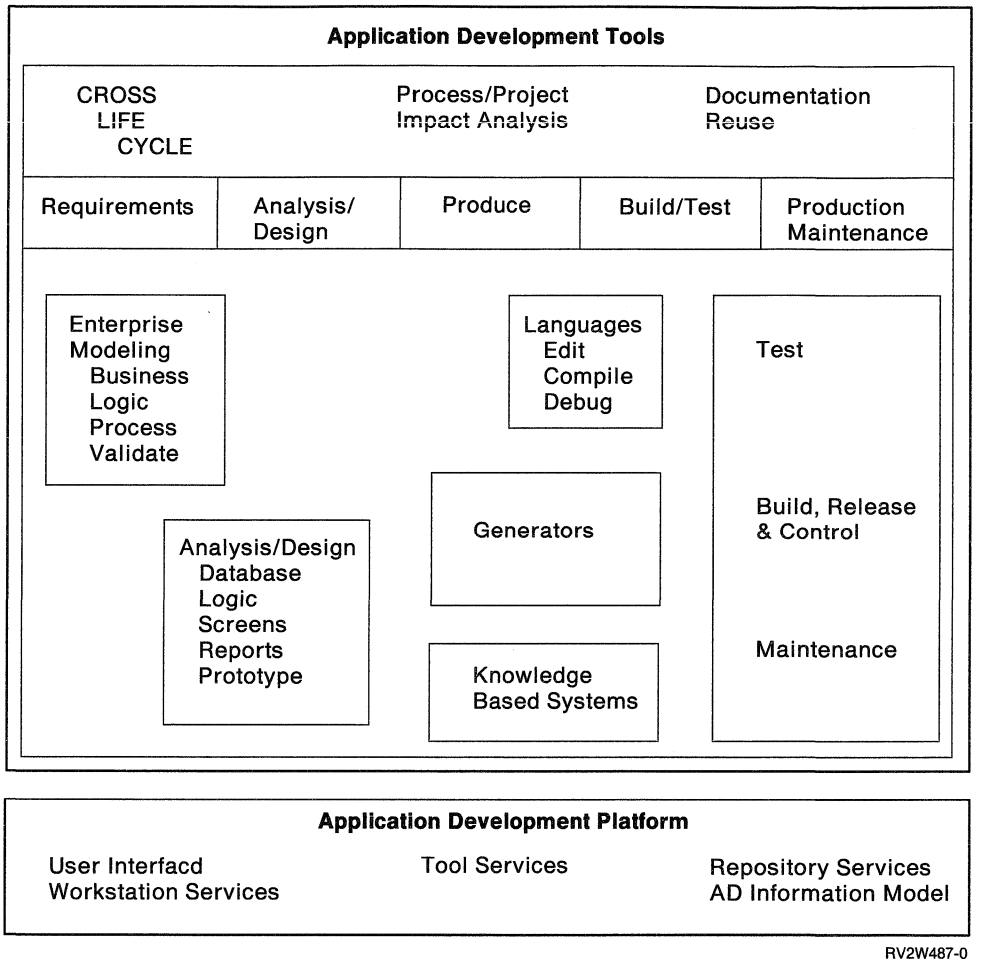


Figure 12-4. AD/Cycle Program: An Open Framework for Application Development

Integrated Application Development Cycle

The basic approach to application development that IBM and selected vendors support is based on the important and strategic relationship between an enterprise's business requirements and the applications that support them. By integrating and synchronizing business requirements with application development and function, The AD/Cycle framework promotes a more efficient, effective, and timely creation of business solutions.

This integrated approach supports the following activities:

- Modeling of the enterprise to reflect both the enterprise strategy and its data processing requirements.
- Validating the enterprise model to ensure that requirements are correct.
- Analyzing requirements and designing the applications necessary to meet those requirements, based on information in the enterprise model.
- Creating a prototype from the design information to verify the requirements and to ensure the quality and useability of the resulting applications.

- Producing the applications from the design information with the technology that best meets the application and business needs.
- Testing and maintaining the applications based on business requirements or on iterative design changes. These process and design changes are further reflected in the enterprise and design information maintained in the AD/Cycle information model.

IBM and vendor tools are available to perform many of the above activities today and more will be available over time for the AS/400 AD/Cycle environment.

Enterprise Modeling

Enterprises are dynamic, and their processes and information requirements are ever-changing. To assist in defining and maintaining accurate, useful, and up-to-date application requirements that reflect this changing environment, the AD/Cycle framework supports an enterprise (or business) model. Moreover, IBM and other vendors supply tools for keeping this model current. These tools include facilities for validating the model itself.

Analyzing and Designing the Application

Using the requirements defined during the enterprise modeling activities, AD/Cycle analysis and design tools assist in developing the application design specification that support these requirements. Because there are many accepted methods for designing applications, AD/Cycle supports the integration of tools that are consistent with today's software engineering and information engineering principles. Analysis and design techniques, such as decomposition diagrams, data-flow diagrams, ER diagrams, and data structure diagrams, are all supported by AD/Cycle tools.

Producing the Application

To facilitate application production, the AD/Cycle program supports traditional third-generation languages such as RPG and COBOL, application generators such as the IBM Cross System Product, and emerging technologies for knowledge-based applications. The IBM Cross System Product—the application generator element of the SAA CPI—generates applications from information created by the design tools. Knowledge-based application-enabling products are also part of the IBM SAA strategy, and over time these products integrate into the AD/Cycle framework by using AD/Cycle information and services.

Testing and Maintaining Applications

Application testing is often separated into two activities. If the application system is large, programmers are usually responsible for testing the major logic paths of their programs. They may write *scaffolding* code to simulate inputs and capture outputs. Depending on the completeness of their modules, they may run in either a test run environment or a simulated environment.

As application units are combined for component and final system test, AD/Cycle tools will assist in the development of test cases, capture test case information, and report on the results. Test tools for interactive applications capture the activity of test sessions, allowing the same tests to be rerun for retesting of new applications and regression testing following maintenance activities. Tools will also provide test coverage measurement and analysis.

Over time, AD/Cycle tools will support restructuring and reverse engineering of some existing application components, allowing design-level information to be cap-

tured in the AD/Cycle process. This will greatly enhance the maintenance of older applications.

Using Application Components Again: A long term goal of the AD/Cycle framework is to support an inventory of application components so that they can be used again and again. Information on such components will be maintained in a repository. This capability will eventually help users create applications from stored components as well as from components created specifically for the application.

Documentation: The development of applications often includes design, specification, and procedural documentation. IBM plans to provide facilities that enable the production of documentation using existing application development information where appropriate. This support will include the ability to create documents that can be printed or displayed at the work station.

Glossary

This glossary includes terms and definitions from the *ISO Vocabulary—Information Processing* and the *ISO Vocabulary—Office Machines*, developed by the International Organization for Standardization, Technical Committee 97, Subcommittee 1. Definitions of published segments of the vocabularies are identified by the symbol (I) after the definition; definitions from draft international standards, draft proposals, and working papers in development by the ISO/TC97/SC1 vocabulary subcommittee are identified by the symbol (T) after the definition, indicating final agreement has not yet been reached among participating members.

access path. The order in which records in a database file are organized for processing by a program. See *arrival sequence access path* and *keyed sequence access path*.

adopted authority. Authority given to the user by the program for the duration of the job that uses that program. The program must be created with owner authority.

advanced peer-to-peer networking (APPN). Data communications support that routes data in a network between two or more APPC systems that do not need to be adjacent.

advanced printer function (APF). A function of the Application Development Tools/400 licensed program that allows a user to design symbols, logos, special characters, large characters, and forms tailored to a business or data processing application. The function supports printing of any design on the 5224 or 5225 dot matrix printer.

advanced program-to-program communications (APPC). Data communications support that allows programs on an AS/400 system to communicate with programs on other systems having compatible communications support. APPC on the AS/400 system provides an application programming interface to the SNA LU type 6.2 and node type 2.1 architectures.

alert. In SNA, a record sent to a focal point to identify a problem or an impending problem.

American National Standard Code for Information Interchange (ASCII). The code developed by the American National Standards Institute for information exchange among data processing systems, data communications systems, and associated equipment. The ASCII character set consists of 7-bit control characters and symbolic characters, plus one parity bit.

APF. See *advanced printer function (APF)*.

API. See *application program interface (API)*.

APPC. See *advanced program-to-program communications (APPC)*.

application program interface (API). A functional interface supplied by the operating system or a separately orderable licensed program that allows an application program written in a high-level language to use specific data or functions of the operating system or the licensed program.

application requester. The source of a request to a remote relational database management system (DBMS).

application server. The target of a request from an application requester. The database management system (DBMS) at the application server site provides the data.

APPN. See *advanced peer-to-peer networking (APPN)*.

arrival sequence access path. An access path to a database file that is arranged according to the order in which records are stored in the physical file. See also *keyed sequence access path* and *access path*.

ASCII. See *American National Standard Code for Information Interchange (ASCII)*.

ASP. See *auxiliary storage pool (ASP)*.

authority holder. An object that specifies and reserves an authority for a program-described database file before the file is created. When the file is created, the authority specified in the holder is linked to the file.

authorization list. A list of two or more user IDs and their authorities for system resources. The system-recognized identifier for the object type is *AUTL.

autostart job. A job doing repetitive work or one-time initialization work that is associated with a particular subsystem. The autostart jobs associated with a subsystem are automatically started each time the subsystem is started.

auxiliary storage. All addressable disk storage other than main storage.

auxiliary storage pool (ASP). A group of units defined from the disk units that make up auxiliary storage. ASPs provide a means of isolating certain objects on specific disk units to prevent the loss of data due to disk media failures on other disk units. See also *system ASP*, and *user ASP*.

basic rate interface (BRI). In ISDN, an interface that provides two 64 000 bps data channels (B-channels)

and one 16 000 bps signaling channel (D-channel). Also known as *2B + D*.

binary synchronous communications (BSC). A data communications line protocol that uses a standard set of transmission control characters and control character sequences to send binary-coded data over a communications line. See also *synchronous data link control (SDLC)*.

binary synchronous communications equivalence link (BSCEL) support. The intersystem communications function (ICF) support on the AS/400 system that provides binary synchronous communications with another AS/400 system, System/36, System/38, and many other BSC computers and devices.

BRI. See *basic rate interface (BRI)*.

BSC. See *binary synchronous communications (BSC)*.

BSC 3270 device emulation. A function of the operating system that allows an AS/400 system to appear to a BSC host system as a 3274 Control Unit.

BSCEL support. See *binary synchronous communications equivalence link (BSCEL) support*.

bus. One or more conductors used for transmitting signals or power.

CCITT. The International Telegraph and Telephone Consultative Committee.

CCS. See *Common Communications Support (CCS)*.

CGU. See *character generator utility (CGU)*.

character generator utility (CGU). A function of the Application Development Tools/400 licensed program that is used to define and maintain user-defined double-byte characters and related sort information.

checksum protection. A function that protects data stored in the system auxiliary storage pool from being lost because of the failure of a single disk. When checksum protection is in effect and a disk failure occurs, the system automatically reconstructs the data when the system program is loaded after the device is repaired.

commit. To make all changes permanent that were made to one or more database files since the last commit or rollback operation, and make the changed records available to other users.

commitment control. A means of grouping database file operations that allows the processing of a group of database changes as a single unit through the Commit command or the removal of a group of database changes as a single unit through the Rollback command.

Common Communications Support (CCS). The Systems Application Architecture (SAA) component that defines architectures and protocols that interconnect systems and devices in an SAA environment and allow data to be interchanged among them.

Common Programming Interface (CPI). In the Systems Application Architecture (SAA) solution, a set of software interfaces, conventions, and protocols that provide a framework for writing applications with cross-system consistency.

Common User Access (CUA). A Systems Application Architecture (SAA) specification that gives a series of guidelines describing the way information should be displayed on a screen, and the interaction techniques between users and computers.

connection list. An AS/400 communications object for ISDN that provides a list of information used to determine when to accept incoming calls and what information to send with outgoing calls.

cooperative processing. Processing that accomplishes the work of an application by running various parts of the application on different processors. Cooperative processing allows application programmers to match tasks to the processor that can best perform the task.

CPI. See *Common Programming Interface (CPI)*.

Cross System Product/Application Execution (CSP/AE). A licensed program used to run an application using the information produced by generation. The application definitions are compatible across the environments in which CSP/AE operates (CICS/VS OS and DOS, MVS/TSO, VM/SP CMS, Distributed Processing Programming Executive/System Product, PC/DOS, Operating System/2 Extended Edition, and the Operating System/400 licensed program).

cross-reference listing. The part of the compiler listing that tells where files, fields, and indicators are defined, referred to, and changed in a program.

CSP/AE. See *Cross System Product/Application Execution (CSP/AE)*.

CUA. See *Common User Access (CUA)*.

current library. The library that is specified to be the first user library searched for objects requested by a user. The name for the current library can be specified on the Sign-On display or in a user profile. When you specify an object name (such as the name of a file or program) on a command, but do not specify a library name, the system searches the libraries in the system part of the library list, then searches the current library before searching the user part of the library list. The current library is also the library that the system uses when you create a new object, if you do not specify a library name.

data area. A system object used to communicate data, such as CL variable values between the programs within a job and between jobs. The system-recognized identifier for the data area is *DTAARA.

data description specifications (DDS). A description of the user's database or device files that is entered into the system in a fixed form. The description is then used to create files.

data file. (1) A group of related data records organized in a specific order. A data file can be created by the specification of FILETYPE(*DATA) on the create commands. Contrast with *source file*. (2) In BASIC, the table containing the values from the DATA statements of a program.

data management. The part of the operating system that controls the storing and accessing of data to or from an application program. The data can be on internal storage (for example, database), on external media (diskette, tape, or printer), or on another system.

data queue. An object that is used to communicate and store data used by several programs in a job or between jobs. The system-recognized identifier is *DTAQ.

data/text merge. In OfficeVision/400 and Query/400, the process of combining data from a file or another document (such as names and addresses) with the text of a document (such as a form letter).

database file. One of several types of the system object type *FILE kept in the system that contains descriptions of how input data is to be presented to a program from internal storage and how output data is to be presented to internal storage from a program. See also *physical file* and *logical file*.

DBCS. See *double-byte character set (DBCS)*.

DDM file. A system object with type *FILE, created by a user on the local (source) system, that identifies a data file that is kept on a remote (target) system. The DDM file provides the information needed for a local system to locate a remote system and to access the data in the remote data file.

DDS. See *data description specifications (DDS)*.

dedicated service tools (DST). The part of the service function used to service the system when the operating system is not working.

device configuration. The physical placement of display stations, printers, and so forth; and the configuration descriptions that describe the physical configuration to the system and describe how the configuration will be used by the system.

device file. One of several types of the system object type *FILE. A device file contains a description of how data is to be presented to a program from a device or how data is to be presented to the device from the program. Devices can be display stations, printers, diskette units, tape units, or remote systems.

DHCF. See *distributed host command facility (DHCF)*.

display station pass-through. A communications function that allows a user to sign on to one system (either an AS/400 system, System/38, or System/36) from another system (either an AS/400 system, System/38, or System/36) and use that system's programs and data. Sometimes called pass-through.

distributed host command facility (DHCF). A function of the operating system that supports the data link between a System/370 terminal using an AS/400 application in an HCF (Host Command Facility) environment.

Distributed Relational Database Architecture (DRDA). A connection protocol for distributed relational database processing that IBM's relational database products use. DRDA comprises protocols for communication between an application and a remote database, and communications between databases. DRDA provides the connections for remote and distributed processing. DRDA is built on the Distributed Data Management Architecture.

distributed systems node executive (DSNX). A function of the operating system that receives and analyzes requests from the NetView Distribution Manager licensed program on a host system. If the request is directed to the system that receives it, the request is processed on that system or on a personal computer directly attached to that system. If the request is intended for a different system, it is routed toward its destination.

document. Any collection of data stored in a document object. All documents and folders on a single AS/400 system make up the document library. A document can contain any type of data stored in it by an application. For example, the OfficeVision/400 application can store notes, memos, reports, and other items; the PC Support/400 shared folders application can store any data that could otherwise be stored in a PC file; an AS/400 application can store any data into a document by using CL commands, such as FILDOC and RPLDOC. The system-recognized identifier for the object type is *DOC. Synonymous with *document library object*.

document library services. The system support that lets office users manage the contents of the document library.

double-byte character set (DBCS). A set of characters in which each character is represented by 2 bytes. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires 2 bytes, the typing,

displaying, and printing of DBCS characters requires hardware and programs that support DBCS. Four double-byte character sets are supported by the system: Japanese, Korean, Simplified Chinese, and Traditional Chinese. Contrast with *single-byte character set*.

DRDA. See *Distributed Relational Database Architecture (DRDA)*.

DSNX. See *distributed systems node executive (DSNX)*.

DST. See *dedicated service tools (DST)*.

end node. In SNA, a node in an APPN network that can be a source or target node, but does not provide any routing or session services to any other node.

Ethernet. A type of local area network that is supported by the Operating System/400 licensed program.

extended help. Help that explains the purpose of the display. Extended help appears if the user presses the Help key when the cursor is outside the areas for which contextual help is available.

externally described data. Data contained in a file for which the fields and the records are described outside of the program (such as with files created by DDS, IDDU, or the SAA SQL/400 licensed program) that processes the file. Contrast with *program-described data*.

field level specifications. In DDS, specifications coded on the same line as a field name or on lines immediately following a field name. See also *file level specifications*, *record level specifications*, and *help level specifications*.

file. A generic term for the object type that refers to a database file, a device file, or a save file. The system-recognized identifier for the object type is *FILE.

file level specifications. In DDS, specifications coded on the lines before the first record format name. See also *field level specifications*, *record level specifications*, and *help level specifications*.

File Transfer Protocol (FTP). In TCP/IP, an application protocol used for transferring files to and from host computers. FTP requires a user ID and possibly a password to allow access to files on a remote host system. FTP assumes that the Transmission Control Protocol is the underlying protocol.

file transfer support. A function of the operating system that moves file members from one system to another by using asynchronous, APPC, or BSCEL communications support.

file transfer, access, and management (FTAM). The OSI standard for transferring files between nodes.

fill-in document. In OfficeVision/400, a document that allows a user to merge data into documents without using Query/400.

finance communications. The data communications support that allows programs on an AS/400 system to communicate with programs on finance controllers, using the SNA LU session type 0 protocol.

folder. A directory for documents. A folder is used to group related documents and to find documents by name. The system-recognized identifier for the object type is *FLR. Compare with *library*.

folder path. A folder name, followed by one or more additional folder names, where each preceding folder is found.

FTAM. See *file transfer, access, and management (FTAM)*.

FTP. See *File Transfer Protocol (FTP)*.

general-purpose library. The library shipped with the system that contains IBM-provided objects required for many system functions and user-created objects that are not explicitly placed in a different library when they are created. Named QGPL.

graphic character set. A set of graphic characters in a code page.

group job. One of up to sixteen interactive jobs that are associated in a group with the same work station device and user.

help level specifications. In a display file, data description specifications coded between the record and field level that define areas on the screen and associate help information with those areas. See also *file level specifications*, *field level specifications*, and *record level specifications*.

high-level language (HLL). A programming language, such as RPG, BASIC, PL/I, Pascal, COBOL, and C used to write computer programs.

HLL. See *high-level language (HLL)*.

hypertext. A weblike structure of nonlinear information nodes linked together by author-defined associations that allow users to freely select nodes of interest.

I/O. See *input/output*.

I/O controller. See *input/output controller (IOC)*.

I/O processor. See *input/output processor (IOP)*.

IBM Application Development Tools/400. The IBM licensed program that provides an integrated set of application development tools, or utilities, to be used

by programmers, analysts, and support personnel. This package includes the following utilities: programming development manager (PDM), source entry utility (SEU), screen design aid (SDA), data file utility (DFU), report layout utility (RLU), and advanced printer function (APF). In addition, the character generator utility (CGU) is added to the package if the user's system supports the double-byte character set (DBCS).

IBM Operating System/2 (OS/2). Pertaining to the IBM licensed program that can be used as the operating system for personal computers. The OS/2 licensed program can perform multiple tasks at the same time.

IBM PC Support/400. The IBM licensed program that provides system functions to an attached personal computer.

IBM Performance Tools/400. The IBM licensed program that allows a user to collect performance data, display performance data, print performance reports, and perform capacity planning.

IBM SAA Operating System/400 (OS/400). Pertaining to the IBM licensed program that can be used as the operating system for the AS/400 system.

IBM SAA SQL/400. Pertaining to the IBM licensed program that is the Systems Application Architecture (SAA) platform of SQL.

ICF. See *intersystem communications function (ICF)*.

IDDU. See *interactive data definition utility (IDDU)*.

index search. The portion of the online help information that allows a user to select help topics that provide additional conceptual help. The system-recognized identifier for the object type is *SCHIDX.

input/output controller (IOC). A functional unit that combines the I/O processor and one or more I/O adapters, and directly connects and controls one or more input or output devices.

input/output processor (IOP). A functional unit or the part of an I/O controller that processes programmed instructions and controls one or more input/output devices or adapters.

integrated services digital network (ISDN). A specification of the CCITT that defines a non-SNA network that can provide voice, data, and image over the same communications line. See also *basic rate interface (BRI)*.

interactive data definition utility (IDDU). A function of the operating system that can be used to externally define the characteristics of data and the contents of files.

interactive terminal facility (ITF). An asynchronous communications function that allows an AS/400 system to communicate with applications that can send and receive data, such as electronic mail, memos, library members, and data files.

intersystem communications function (ICF). A function of the operating system that allows a program to communicate interactively with another program or system.

IOC. See *input/output controller (IOC)*.

IOP. See *input/output processor (IOP)*.

IPL. See *initial program load (IPL)*.

ISDN. See *integrated services digital network (ISDN)*.

ITF. See *interactive terminal facility (ITF)*.

job description. A system object that defines how a job is to be processed. The object name is *JOBD.

journal. A system object used to record entries in a journal receiver when a change is made to the database files associated with the journal. The object type is *JRN. See also *journal receiver*.

journal receiver. A system object that contains journal entries recorded when changes are made to the data in database files or the access paths associated with the database files. The object type is *JRNRCV. See also *journal*.

keyed sequence. An order in which records are retrieved that is based on the contents of key fields in records.

keyed sequence access path. An access path to a database file that is arranged according to the contents of key fields contained in the individual records. See also *arrival sequence access path* and *access path*.

LAN. See *local area network (LAN)*.

library. (1) A system object that serves as a directory to other objects. A library groups related objects, and allows the user to find objects by name. The system-recognized identifier for the object type is *LIB. Compare with *folder* and *document library*. (2) The set of publications for a system.

library list. A list that indicates which libraries are to be searched and the order in which they are to be searched. The system-recognized identifier is *LIBL.

licensed program. A separately orderable program, supplied by IBM, that performs functions related to processing user data. Examples of licensed programs are PC Support/400, SAA COBOL/400, Application Development Tools/400, OfficeVision/400, and so on.

link level. (1) In SNA, the combination of the transmission connection, protocol, devices, and programming joining network nodes. (2) A part of Recommendation X.25 that defines the link protocol

used to get data into and out of the network across the duplex line connecting the subscriber's equipment to the network.

local area network (LAN). The physical connection that allows the transfer of information among devices located on the same premises.

local work station. A work station that is connected directly to the system without a need for data transmission functions. Contrast with *remote work station*.

logical file. A description of how data is to be presented to or received from a program. This type of database file contains no data, but it defines record formats for one or more physical files. See also *database file*. Contrast with *physical file*.

logical unit (LU). In SNA, one of three types of network addressable units that serve as a port through which a user accesses the communications network. See also *physical unit*.

LU. See *logical unit (LU)*.

machine interface (MI). The interface, or boundary, between the operating system and the licensed internal code.

main storage pool. A division of main storage, which allows the user to reserve main storage for processing a job or group of jobs, or to use the pools defined by the system. Contrast with *auxiliary storage pool*.

MI. See *machine interface (MI)*.

mirrored protection. A function that protects data by duplicating all disk data in an auxiliary storage pool (ASP) to another disk unit (mirrored unit) in the same ASP. If a disk failure occurs, the system keeps running, using the mirrored unit of the mirrored pair until the disk unit is repaired or replaced.

mode. The session limits and common characteristics of the sessions associated with advanced-program-to-program communications (APPC) devices managed as a unit with a remote location.

Multiple Virtual Storage (MVS). An alternative name for OS/VS2. See also *operating system* and *virtual storage (VS)*.

MVS. See *Multiple Virtual Storage (MVS)*.

national language support (NLS). The modification or conversion of a United States English product to conform to the requirements of another language or country. This can include the enabling or retrofitting of a product and the translation of nomenclature, online information, or documentation of a product.

NetView Distribution Manager. A licensed program available for IBM host systems (System/370, 43xx, and 30xx computers) that allows the host system to use, send, and delete files and programs in a network of computers.

NetView DM. See *NetView Distribution Manager*.

network interface description. An AS/400 communications object that represents the physical interface to the ISDN. The network interface description must be configured in addition to the line, controller, and device.

NLS. See *national language support (NLS)*.

node. (1) One of the systems or devices in a network. (2) A location in a communications network that provides host-processing services. (3) An information unit containing information about a single topic and linked to one or more other nodes. (4) For APPN, see *end node*.

object. (1) A named storage space that consists of a set of characteristics that describe itself and, in some cases, data. An object is anything that exists in and occupies space in storage and on which operations can be performed. Some examples of objects are programs, files, libraries, and folders. (2) In SQL, anything that can be created or manipulated with SQL statements, such as databases, tables, views, or indexes.

object authority. A specific authority that controls what a system user can do with an entire object. For example, object authority includes deleting, moving, or renaming an object. There are three types of object authorities: object operational, object management, and object existence.

object distribution. A function that allows a user to send source and data files, save files, job streams, spooled files, and messages to another user, either locally or on an SNADS network.

object management authority. An object authority that allows the user to specify the authority for the object, move or rename the object, and add members to database files.

object name. The name of an object. Contrast with *qualified name*.

open systems interconnection (OSI). (1) The interconnection of open systems in accordance with specific ISO standards. (T) (2) The set of standards defined by ISO that define how systems from different vendors communicate. (3) The use of standardized procedures to enable the interconnection of data processing systems.

Note: OSI architecture establishes a framework for coordinating the development of current and future standards for the interconnection of computer systems. Network functions are divided into seven layers. Each layer represents a

group of related data processing and communication functions that can be carried out in a standard way to support different applications.

OS/2. See *IBM Operating System/2 (OS/2)*.

OS/400. See *IBM SAA Operating System/400 (OS/400)*.

OSI. See *open systems interconnection (OSI)*.

output queue. An object that contains a list of spooled files to be written to an output device, such as a printer or a diskette. The system-recognized identifier for the object type is *OUTQ.

override. (1) To specify attributes at run time that change the attributes specified in the file description or in the program. (2) The attributes specified at run time that change the attributes specified in the file description or in the program.

paragraphs document. In OfficeVision/400, a document that contains a paragraph or paragraphs that can be combined to create other documents.

pass-through. See *display station pass-through*.

PC Support. See *IBM PC Support/400*.

performance monitor. A function of the operating system that observes system and device activity, and records these observations in a database file.

physical file. A description of how data is to be presented to or received from a program and how data is actually stored in the database. A physical file contains one record format and one or more members. See also *database file*. Contrast with *logical file*.

physical unit (PU). In SNA, one of three types of network addressable units. A physical unit exists in each node of an SNA network to manage and monitor the resources (such as attached links and adjacent link stations) of a node, as requested by a system services control point logical unit (SSCP-LU) session.

prestart job. A job that starts running before the remote program sends a program start request.

private authority. The authority specifically given to a user for an object that overrides any other authorities, such as the authority of a user's group profile or an authorization list. Contrast with *public authority*.

program temporary fix (PTF). A temporary solution to, or bypass of, a defect in a current release of a licensed program.

program-described data. Data contained in a file for which the fields in the records are described in the program that processes the file. Contrast with *externally described data*.

PTF. See *program temporary fix*.

PU. See *physical unit (PU)*.

public authority. The authority given to users who do not have any specific (private) authority to an object, who are not on the authorization list (if one is specified for the object), and whose group profile has no specific authority to the object. Contrast with *private authority*.

QGPL. See *general-purpose library*.

qualified name. The name of the library containing the object and the name of the object. Contrast with *object name*.

query. A request to select and copy from a file or files one or more records based on defined conditions. For example, a request for a list of all customers in a customer master file, whose balance is greater than $1000.

queue. A list of messages, jobs, files, or requests waiting to be read, processed, printed, or distributed in a predetermined order.

record level specifications. Data description specifications coded on the same line as a record format name or on lines immediately following a record format name (until the first field is specified). See also *field level specifications*, *file level specifications*, and *help level specifications*.

remote job entry (RJE). A function of the Communication Utilities/400 licensed program that allows a user to submit a job from a display station on the AS/400 system to a System/370-type host system.

remote system. Any other system in the network with which your system can communicate.

remote work station. A work station that is connected to the system by data communications. Contrast with *local work station*.

requester. In PC Support/400, a program that requests services from another program (a server). Each PC Support/400 function has a server and a requester.

retail communications. The data communications support that allows programs on an AS/400 system to communicate with programs on point-of-sale systems, using SNA LU session type 0 protocol.

retail pass-through. An OS/400 system program that supports routing of user data between a System/370-type host processor and a retail controller using a single AS/400 system. Both the SNA upline facility and the retail communications support use separate intersystem communications function sessions.

RJE. See *remote job entry (RJE)*.

routing entry. An entry in a subsystem description that specifies the program to be called to control a routing step that runs in the subsystem.

RPG/400. See *IBM SAA RPG/400*.

save file. A file allocated in auxiliary storage that can be used to store saved data on disk (without requiring diskettes or tapes), to do I/O operations from a high-level language program, or to receive objects sent through the network. The system-recognized identifier for the object type is *FILE.

SBCS. See *single-byte character set (SBCS)*.

screen design aid (SDA). A function of the Application Development Tools/400 licensed program that helps the user design, create, and maintain displays and menus.

SDA. See *screen design aid (SDA)*.

SDLC. See *synchronous data link control (SDLC)*.

SEU. See *source entry utility (SEU)*.

shell document. In OfficeVision/400, a prearranged document (report, letter, memo, or note) where the user inserts variable information. An example of a shell document is a form letter, to which the user adds the receiver's name, address, and personal salutation.

Simple Mail Transfer Protocol (SMTP). In TCP/IP, an application protocol for transferring mail among users in the internet environment. SMTP specifies the mail exchange sequences and message format. SMTP assumes that the Transmission Control Protocol is the underlying protocol.

single-byte character set (SBCS). A character set in which each character is represented by a one-byte code. Contrast with *double-byte character set*.

SMTP. See *Simple Mail Transfer Protocol (SMTP)*.

SNA. See *Systems Network Architecture (SNA)*.

SNA distribution services (SNADS). An IBM asynchronous distribution service that defines a set of rules to receive, route, and send electronic mail in a network of systems.

SNA upline facility (SNUF). The communications support that allows the AS/400 system to communicate with CICS/VS and IMS/VS application programs on a host system. For example, DHCF communicates with HCF and DSNX communicates with NetView Distribution Manager.

SNADS. See *SNA distribution services (SNADS)*.

SNUF. See *SNA upline facility (SNUF)*.

source entry utility (SEU). A function of the Application Development Tools/400 licensed program that is used to create and change source members.

source file. A file of programming code that is not compiled into machine language. Contrast with *data file*.

source system. In communications, the system that issues a request to establish communications with another system.

special authority. The types of authority a user can have to perform system functions, including all object authority, save system authority, job control authority, security administrator authority, spool control authority, and service authority. Contrast with *specific authority*.

specific authority. The types of authority a user can be given to use the system resources, including object authorities and data authorities. See also *object authority*. Contrast with *special authority*.

spool. The system function of putting files or jobs into disk storage for later processing or printing.

SQL. See *Structured Query Language (SQL)*.

SQL/400. See *IBM SAA SQL/400*.

SST. See *system service tools (SST)*.

stop code. In OfficeVision/400, a position marked in a document where you can insert variable information.

storage pool. A logical division of storage reserved for processing a job or group of jobs.

structure. In PL/I, a collection of data items that need not have identical attributes.

Structured Query Language (SQL). A language that can be used within host programming languages or interactively to put information into a database and to get and organize selected information from a database. SQL can also control access to database resources. See also *IBM SAA SQL/400*.

subsystem. (1) An operating environment, defined by a subsystem description, where the system coordinates processing and resources. (2) An informal name for the IBM OSI/Communications Subsystem/400 licensed program.

subsystem description. A system object that contains information defining the characteristics of an operating environment controlled by the system. The system-recognized identifier for the object type is *SBSD.

synchronous data link control (SDLC). (1) A form of communications line control that uses commands to control the transfer of data over a communications line.

(2) A communications discipline conforming to subsets of the Advanced Data Communication Control Procedures (ADCCP) of the American National Standards Institute (ANSI) and High-Level Data Link Control (HDLC) of the International Standards Organization (ISO), for transferring synchronous, code-transparent, serial-by-bit information over a communications line. Transmission exchanges may be duplex or half-duplex over switched or nonswitched lines. The configuration of the connection may be point-to-point, multipoint, or loop. Compare with *binary synchronous communications (BSC)*.

system ASP. The auxiliary storage pool where system programs and data reside. It is the storage pool used if a storage pool is not defined by the user. See also *auxiliary storage pool* and *user ASP*.

system distribution directory. A list of user IDs and identifying information, such as network addresses, used to send distributions.

system library. The library shipped with the system that contains objects, such as authorization lists and device descriptions created by a user; and the licensed programs, system commands, and any other system objects shipped with the system. The system identifier is QSYS.

system security. A system function that restricts the use of files, libraries, folders, and devices to certain users.

system service tools (SST). The part of the service function used to service the system while the operating system is running.

system value. Control information for the operation of certain parts of the system. A user can change the system value to define his working environment. System date and library list are examples of system values.

System/36 environment. A function of the operating system that processes most of the System/36 operator control language (OCL) statements and procedure statements to run System/36 application programs and allows the user to process the control language (CL) commands. Contrast with *System/38 environment*.

System/38 environment. A function of the operating system that processes most of the System/38 control language (CL) statements and programs to run System/38 application programs. Contrast with *System/36 environment*.

Systems Network Architecture (SNA). In IBM networks, the description of the layered logical structure, formats, protocols, and operational sequences that are used for transmitting information units through networks, as well as controlling the configuration and operation of networks.

Systems Network Architecture distribution services. See *SNA distribution services (SNADS)*.

target system. The system that receives a request from another system to establish communications.

TCP/IP. See *Transmission Control Protocol/Internet Protocol (TCP/IP)*.

TDLC. See *twinaxial data link control (TDLC)*.

technical information exchange (TIE). A part of the electronic customer support function that allows a user to send files to and receive files from an IBM support system, and to search for information on an IBM support system. The files are sent and received through an IBM Information Network.

TELNET. In TCP/IP, an application protocol that allows a user at one site to access a remote system as if the user's display station were locally attached. TELNET uses the Transmission Control Protocol as the underlying protocol.

temporary library. A library that is automatically created for each job to contain temporary objects that are created by the system for that job. The objects in the temporary library are deleted when the job ends. The system name for temporary library is QTEMP.

TIE. See *technical information exchange (TIE)*.

token-ring network. A local area network that sends data in one direction throughout a specified number of locations by using the symbol of authority for control of the transmission line, called a token, to allow any sending station in the network (ring) to send data when the token arrives at that location.

Transmission Control Protocol/Internet Protocol (TCP/IP). A set of vendor-independent communications protocols that support peer-to-peer connectivity functions for both local and wide area networks.

twinaxial data link control (TDLC). A communications function that allows personal computers, which are attached to the work station controller by twinaxial cable, to use advanced program-to-program communications (APPC) or advanced peer-to-peer networking (APPN).

UIM. See *user interface manager (UIM)*.

uninterruptible power supply. A source of power from a battery installed between the commercial power and the system that keeps the system running, if a commercial power failure occurs, until it can complete an orderly end to system processing.

user ASP. One or more auxiliary storage pools used to isolate some system objects from the other system objects stored in the system ASP. See also *auxiliary storage pool (ASP)* and *system ASP*.

user ID. See *user identification (user ID)*.

user interface manager (UIM). A function of the operating system that provides a consistent user interface by providing comprehensive support for defining and running panels and dialogs.

user profile. An object with a unique name that contains the user's password, the list of special authorities assigned to a user, and the objects the user owns. The system-recognized identifier for the object type is *USRPRF.

validity checking. To verify the contents of a field.

virtual printer. In PC Support/400, a printer attached to a host system that can receive output from a personal computer for printing. A virtual printer allows you to use a printer attached to the host system as though the printer were directly attached to a personal computer.

virtual storage (VS). An addressing scheme that allows external disk storage to appear as main storage.

virtual terminal manager (VTM). A VLIC component that provides an interface to handle input/output to virtual devices on the AS/400 system.

VM/MVS bridge. A function of the Communication Utilities/400 licensed program that provides distribution services between an AS/400 SNADS network and both a VM/370 Remote Spooling Communications Subsystem (RSCS) network and a Multiple Virtual Storage/Job Entry Subsystem (MVS/JES) network. Formerly known as RSCS/PROFS bridge.

VS. See *virtual storage (VS)*.

VTM. See *virtual terminal manager (VTM)*.

work entry. An entry in a subsystem description that specifies the source from which jobs can be accepted for processing in the subsystem.

work station entry. An entry in a subsystem description that specifies the work stations from which users can sign on to the subsystem or from which interactive jobs can transfer to the subsystem.

work station user profile. The system-supplied user profile that has the authority required by work station operators. Named QUSER.

X.21. In data communications, a specification of the CCITT that defines the connection of data terminal equipment to an X.21 (public data) network.

X.25. A CCITT Recommendation that defines the physical level (physical layer), link level (data link layer), and packet level (network layer) of the OSI reference model. An X.25 network is an interface between data terminal equipment (DTE) and data circuit-terminating equipment (DCE) operating in the packet mode, and connected to public data networks by dedicated circuits. X.25 networks use the connection-mode network service.

3270 device emulation. The operating system support that allows an AS/400 system to appear as a 3274 Control Unit in a BSC multipoint network or an SNA network.

3270 display emulation. The function of the operating system 3270 device emulation support that converts 3270 data streams intended for a 3278 display station into data streams that can be recognized by a display station attached to the AS/400 system.

3270 display station. Any display station, attached by coaxial cable, that uses 3270 data streams.

5250 display station. Any display station, attached by twinaxial cable, that uses 5250 data streams.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Numerics

Reader's Comments

Application System/400
System Concepts
Version 2

Publication No. GC41-9802-00

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name

Address

Company or Organization

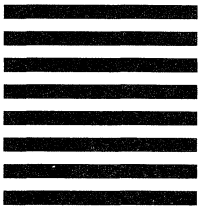
Phone No.

Reader's Comments
GC41-9802-00

Cut or Fo
Along Li

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN DEPT 245
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-7899

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fol
Along Lin

Program Number: 5738-SS1

Printed in Denmark by
J. H. Schultz Print a|s
Copenhagen

GC41-9802-00