UNISYS

System 80 OS/3 Distributed Data Processing (DDP)

Programming Guide

OS/3 Release 14

April 1991

Printed in U S America

Priced Item

7004 4508-000

Correspondence regarding this publication should be forwarded to Unisys Corporation either by using the User Reply form at the back of this manual or by addressing remarks directly to Unisys Corporation, SPG East Coast Systems Documentation Development, Tredyffrin Plant, 2476 Swedesford Road, P.O. Box 203, Paoli, PA, 19301-0203, U.S.A.

PAGE STATUS SUMMARY
ISSUE: 7004 4508-000

Part/Section	Page Number	Update Level
Cover		
Title Page/Disclaimer		
PSS	iii	
About This Guide	v thru x	
Contents	xi thru xix	
1	1 thru 12	
2	1 thru 7	
3	1 thru 4	
4	1 thru 40	
5	1 thru 4	
6	1 thru 36	
7	1 thru 6	
Appendix A	1 thru 3	
Appendix B	1 thru 4	
Appendix C	1 thru 3	
Appendix D	1	
Appendix E	1 thru 4	
Appendix F	1, 2	
Appendix G	1, 2	
Appendix H	1 thru 3	
Glossary	1 thru 10	
Index	1 thru 6	
User Reply Form		
Back Cover		

Unisys uses an 11-digit document numbering system. The suffix of the document number (1234 5678-xyz) indicates the document level. The first digit of the suffix (x) designates a revision level; the second digit (y) designates an update level. For example, the first release of a document has a suffix of -000. A suffix of -130 designates the third update to revision 1. The third digit (z) is used to indicate errata for a particular level and is not reflected in the page status summary.

About This Guide

Purpose

This guide is one of a series designed to instruct you in using the Unisys Operating System/3 (OS/3). It gives an overview of distributed data processing (DDP) concepts, describes DDP network requirements, and explains in detail how the Unisys DDP facilities are used.

Audience

This guide is intended for the site administrator who needs to know how Unisys DDP products fit together, the programmer who wants to examine the concepts behind the products, and the person who uses the DDP facilities.

Organization

This guide has seven sections, eight appendixes, a glossary, and an index.

Section 1. An Introduction to Distributed Data Processing

Defines DDP, describes the various DDP network configurations, and lists the advantages of a DDP system.

Section 2. DDP Products

Describes the four Unisys DDP products that are available and explains how they fit together in a DDP system.

Section 3. DDP Network Requirements

Lists the things you must have and do to make your DDP network operational.

Section 4. The DDP Transfer Facility

Describes in detail the use of this facility, which sends files and jobs from one OS/3 system to another.

Section 5. The OS/3 to UNIX O/S DDP File Transfer Facility

Describes in detail the use of this facility, which lets you transfer files back and forth between an OS/3 system and a UNIX® system.

Section 6. The DDP File Access Facility

Describes in detail the use of this facility, which lets you access and process files on remote OS/3 systems and write application programs that can communicate with other application programs on remote systems.

Section 7. BAL Program Level Interface

Explains how to access the distributed functions provided by OS/3 DDP with user-written basic assembly language programs.

Appendix A. Sample DDP File Copy/Job Submission Session

Shows a typical file copy and job submission from a local host to a remote host.

Appendix B. Command Function Summary

Lists the DDP commands by function.

Appendix C. Command Format Summary

Lists the DDP commands alphabetically.

Appendix D. Command Requirements

Lists those commands and parameters that require the use of other commands and parameters.

Appendix E. Entering DDP Commands in a Batch Stream

Explains how to submit DDP commands as a remote batch job and gives a sample enter stream.

Appendix F. Logging Chart

Provides a chart for listing your system-assigned job numbers and any other pertinent data from your log file for accounting purposes.

UNIX is a registered trademark of AT&T Information Systems.

Appendix G. Generating Your ICAM Network with DDP

Gives a sample ICAM network definition for the DDP file transfer and file access facilities.

Appendix H. DDP Program-to-Program Macroinstructions Summary

Lists by function the program-to-program macros used with the DDP file access facility.

Related Documentation

The following publications are referenced in this guide. Use the version that applies to the software level in use at your site.

Assembler Programming Guide (7004 4532)

Consolidated Data Management Programming Guide (UP-9978)

Consolidated Data Management Macroinstructions Programming Guide (7004 4607)

File Cataloging Technical Overview (7004 4615)

General Editor (EDT) Operating Guide (7004 4599)

Information Management System (IMS) Action Programming in RPG II Programming Guide (UP-9206)

Information Management System (IMS) COBOL/Assembler Action Programs Programming Guide (UP-9207)

Information Management System (IMS) System Support Functions Programming Guide (UP-11907)

Interactive Services Operating Guide (UP-9972)

Job Control Programming Guide (7004 4623)

Model 7E Integrated Communications Access Method (ICAM) Operations Guide (7002 3908)

Model 7E Installation Guide (7002 3858)

Model 7E Operations Guide (7002 3866)

Model 50 Basic Operations Guide (7004 1934)

Model 50 Detailed Operations Guide (7004 1942)

Model 50 Integrated Communications Access Method (ICAM) Operations Guide (7004 1967)

Model 50 Installation Guide (7004 1892)

Models 8-20 Integrated Communications Access Method (ICAM) Operations Guide (7004 4557)

Models 8-20 Installation Guide (7004 5505)

Models 8-20 Operations Guide (7004 5208)

Spooling and Job Accounting Operating Guide (7004 4581)

System Messages Reference Manual (7004 5190)

System Service Programs (SSP) Operating Guide (UP-8841)

Notation Conventions

This guide uses the following conventions to illustrate commands and control statements:

- Code capital letters, parentheses ( ), and punctuation marks (except braces, brackets, and ellipses) exactly as shown. An ellipsis, which is a series of three periods (...), indicates the omission of a number of obvious entries.
- Lowercase letters and terms in commands represent information that you must supply. These lowercase terms may contain hyphens for readability. Lowercase letters and terms in system response messages represent variables the system supplies to you.
- Information within braces { } represents necessary entries. You must choose one.
- Code positional subparameters in the order indicated. Code nonpositional subparameters in any order.
- Insert commas after each positional parameter except the last. When you omit a positional parameter (or a positional subparameter within a series of parameters), retain the comma to indicate the omission. When you omit a trailing positional parameter, omit its associated comma also, even when a keyword parameter follows. Code positional parameters or positional subparameters in the order shown.

- Separate keyword parameters from each other and from positional parameters by spaces. This is different from normal OS/3 format.

- Use underlines as indicated in commands. DDP uses underlines, not hyphens, to join two words in a parameter. This is different from normal OS/3 format.

- The shaded parameter is selected automatically when you omit a keyword parameter or subparameter. In this example, the default for the MODE parameter is DIRECT:

```
MODE=┌DIRECT  ┐
     {WAIT    }
     └INDIRECT┘
```

- Keyword parameters (nonpositional parameters) are normally entered in the format: keyword=value. This guide shows optional keyword parameters in alphabetical order, for easy reference. However, you may enter them in any order.

- Code zero or more spaces before a DDP command. Conclude each command with one or more spaces except for the last command on a line, which does not require the concluding space.

- Commands may continue on subsequent lines if the last character on the line to be continued is an ampersand (&). In a command, the ampersand may appear only where a space may appear, and DDP treats it as the last character on the line. The ampersand may follow zero or more spaces and must be followed by at least one space before a comment begins.

- The exclamation mark (!) is the comment character and terminates line scanning. It may appear anywhere in a command that a space may appear; it does not have to be used only at the end of a command. If you use it in the middle of a command, however, be sure to put a continuation character (&) before it so that your command may continue on the next line.

  Comments following ampersands do not need the preceding exclamation mark, since the ampersand terminates scanning of that line. After an ampersand or an exclamation mark, scanning begins again on the next line.

- Character strings begin and end with apostrophes. Apostrophes (') within strings must be doubled. Parameter values require string format if they contain a comma (,), space (△), semicolon (;), exclamation mark (!), apostrophe ('), quote ("), pound sign (#), equal sign (=), or ampersand (&). Note that the exclamation mark and ampersand within a quoted string are simply characters within that string. They do not have their usual functions as the comment and the continuation characters.

- When you use a character string as a keyword parameter value, you may continue it on the next line by using the pound sign (#) as a concatenation character. The procedure is:

  1. Break the character string into two parts. For instance, if you have this character string:

     ```
     TO=H001::'MOD1,PERSONNELFILE(READ/WRITE),,S'
     ```

     break it into the two strings:

     ```
     'MOD1,PERSONNELFILE(READ/
     WRITE),,S'
     ```

  2. Add the concatenation character (#) and the continuation character (&) to the end of the first part of the string:

     ```
     'MOD1,PERSONNELFILE(READ/'#&
     'WRITE),,S'
     ```

     The concatenation character may come immediately after the final character of the first part of the string or immediately before the first character of the second part of the string. For instance, you might have a long statement in a DDP SUBMIT REQUEST command that won't fit on one line. You can continue the command on the next line in either of two ways:

     ```
     DDP SUBMIT REQUEST='RV CGV,,O=VSN999,N=186309,'#&
     'T=30' HOST=N825
     ```

     or

     ```
     DDP SUBMIT REQUEST='RV CGV,,O=VSN999,N=186309,'&
     #'T=30 HOST=N825
     ```

     Note that:

     - the concatenation character (#) may follow zero or more spaces, and may be followed by zero or more spaces; and

     - to continue a character string on the next line, you must use both the concatenation character (#) and the ampersand (&). They may be surrounded by any number of blanks and may appear in any order. However, if the concatenation character follows the ampersand, it must be on the next line, since scanning of the first line terminates at the ampersand.

Contents

Figures

Tables

Section 1
An Introduction to Distributed Data Processing

1.1. What Is Distributed Data Processing?

Distributed data processing (DDP) is not a single hardware or software product. It's a computer management concept giving local managers control over their data and jobs while at the same time allowing central access to all data.

Just as administrative responsibilities in a large company are distributed over a number of managers, so in a DDP system jobs and data are distributed over a number of computers. To get the administrative advantages of the local computers while at the same time keeping central control, you need a DDP system.

We can look at a DDP system in three different ways. First, it has a physical aspect: DDP links independent computers. Second, it has an administrative aspect: DDP distributes both jobs and data over a number of linked computers. Third, it has a functional aspect: DDP forms a layer between you and the communications network. Let's examine each of these aspects in detail.

DDP Links Independent Computers

A DDP system contains at least two computers capable of independent processing. They are linked together through telecommunications so that each can use the other's capabilities and data.

The joined computers have independent operating systems. They can communicate and process transactions independently. But they can also be joined as equal hosts - not terminals - in a DDP system, allowing each computer to perform the job it does best. Thus, work is distributed over several computer hosts.

At any time, of course, one computer may be *in charge*, controlling, for instance, the copying of a file. But the computers may switch roles for different jobs. That's different from a master-slave relationship, in which one piece of hardware is always in charge and the other always does its bidding.

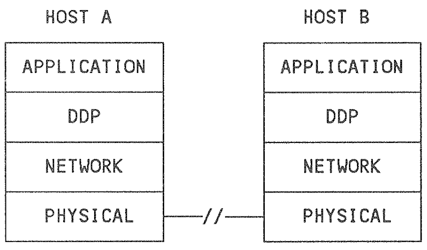
DDP Distributes Both Jobs and Data over a Number of Hosts

What is *distributed* in a DDP system? First, jobs. You can send a job to any other host to which your host is linked and get the results back just as if your own host had done the job. Second, data gets distributed. You can put files on the host where they're needed most, but others can get records from them or copies of the whole file.

DDP Forms a Layer between You and the Communications Network

The third way to view DDP is to look at the function DDP products have in your computing system. DDP is that layer of software products and their associated hardware devices that lies between the user and a communications network. Current products in the layer let you share the following tasks among linked hosts:

- Send files and jobs anywhere in your system
- Access and process files anywhere in your system
- Write programs that can communicate with other programs anywhere in your system
- Use your information management system (IMS) action programs to access records on a remote host



1.2. Requirements for a DDP System

The three requirements for a DDP system are:

1. Two or more computers must be logically separate. Each host must have an operating system so it can function independently.

2. An electronic link must join the hosts so they can work cooperatively on the same project. This link makes the capabilities of each host available at any host. It lets you distribute projects across independent hosts.

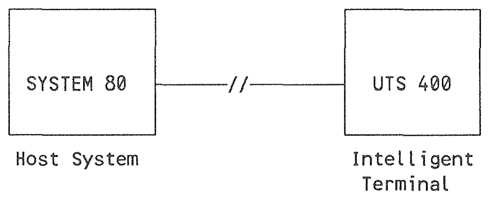
3. The communicating computers must use a common command language.

1.3. Ways to Join Computers in a DDP System

DDP configurations take the same forms you find in any communications network that joins nodes. But there is a difference. A communications network joins both hosts and terminals. Since a DDP system is a subset of a communications network, only independent hosts are joined, though of course those hosts may themselves have terminals.

So if you already have a communications network, you won't have to make any changes in it when you move to DDP.

A programmable terminal is the simplest form of independent computer. So, the simplest DDP configuration is a link between a processor and a programmable terminal. For instance, a Unisys System 80 processor and a programmable Unisys UTS 400 terminal system form a simple DDP configuration.

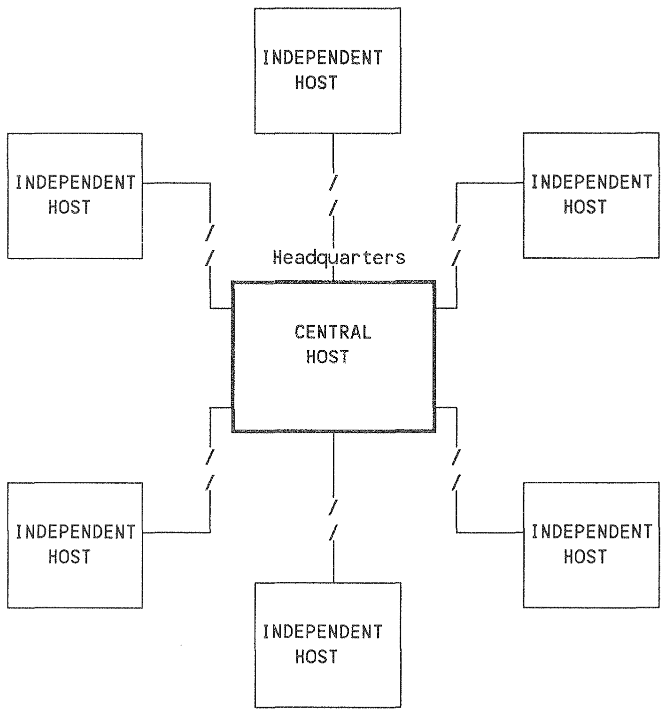


A Minimal DDP System

Although the UTS 400 is normally used as a terminal to a larger computer, it is capable of independent processing. Once its programs are loaded from the System 80 processor, it can store and call them with no further help.

In the following star and ring configurations, it is important to note that not all OS/3 DDP comands are available between hosts with no direct connection since OS/3 does not support pass-through operation. For all DDP commands to be available to all hosts in the network, all hosts must have a direct connection with all other hosts.

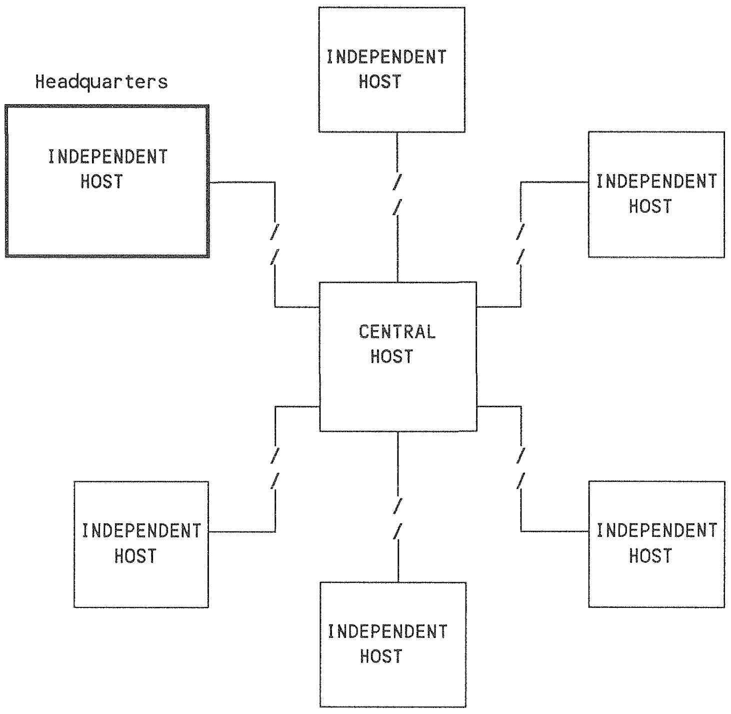
The *star* is one of the structures most used in DDP, just as it is in communications networks.



A Star System - Involving a hierarchical relationship among hosts

Star structures are hierarchical. A central host controls the system. Subordinate, though still independent, hosts have links to the central computer but not to each other. It's important to realize, though, that the DDP system itself treats all hosts as equals. Each can request data from the others. The central host, in addition to its other functions, switches messages from one host on a point of the star to another.

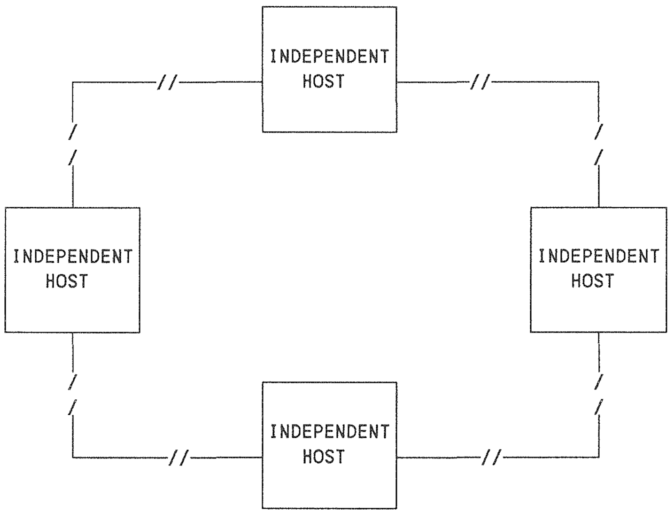
You may make the central host the largest or make it monitor the functions of the other hosts. But you may also decide differently. For instance, your headquarters computer could be on one of the points, and the central host could be a small computer mainly managing traffic. This structure frees the headquarters computer from traffic management.



A Star System - With the largest computer on one of the points

A star structure with many points transfers messages rapidly, since each message gets to its destination with no more than one intervening host. Fast communication makes the star efficient.

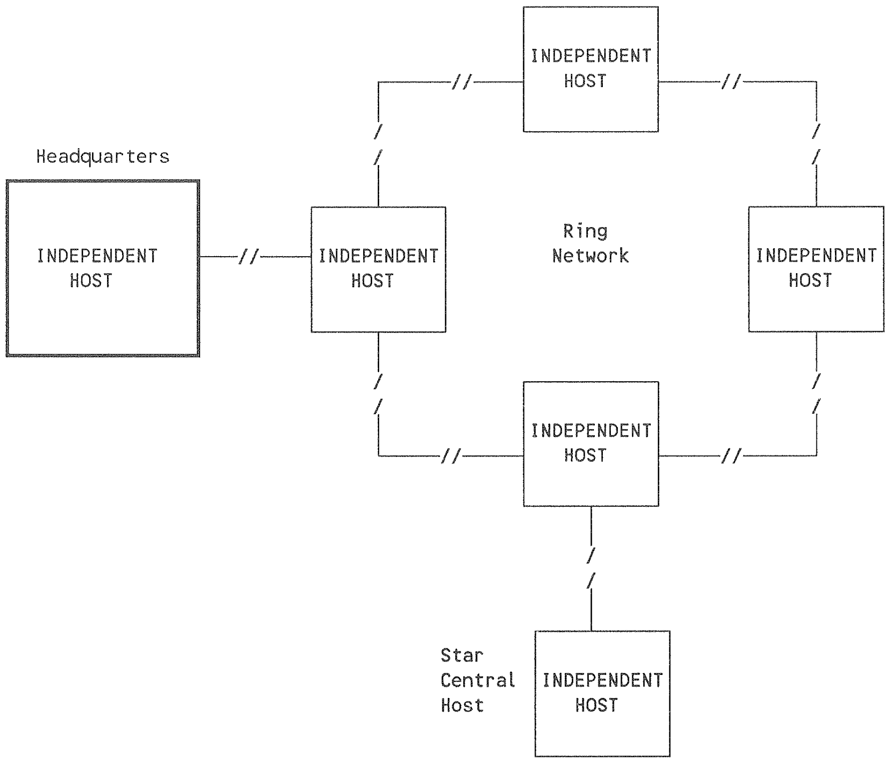
DDP configurations can also use the ring form of communication.



A Ring System

The ring is nonhierarchical. There's no central traffic manager. Each host communicates directly with its two contiguous hosts. Communications to the more remote hosts go through each intervening host. Therefore, communications are slower. For this reason, the ring is less efficient than the star. You might want to use it, though, if line costs are less for a ring than for a star structure.

Of course, you can combine these types. And most DDP systems will. A tree structure can combine star and ring structures. A single host may be in star or ring configuration with other hosts in addition to being connected to still other hosts and configurations. The important thing is that DDP treats all hosts in the network as equal.



A Tree System

So what do these various DDP structures mean to you? They mean you can have whatever configuration of hosts your company needs to process data efficiently.

1.4. How DDP Evolved

Data processing systems have evolved from centralized to decentralized to distributed. And distributed processing itself didn't emerge instantly. Instead, it evolved slowly in response to user needs. Distributed systems are well-suited to today's management needs.

1.4.1. Centralized, Decentralized, and Distributed Systems

- A *centralized computer system* is a central processor performing varied tasks for many users. It generally combines batch processing with electronically connected terminals that send and receive data but don't process it themselves.

- A *decentralized computer system* has several different computers, either large or mini. These computers are not connected as equals, where each seeks data from the other; instead, they operate independently.

  Computers in a decentralized system may be temporarily connected to work on the same task. But, unless they're connected as equals, they're not in a DDP system. For instance, the Nine-Thousand Remote (NTR) communications package of ICAM connects a Unisys Series 1100 system and a System 80 so that the 1100 system is always the master and the System 80 is always the slave. They can't change roles with NTR, so NTR isn't, strictly speaking, a DDP product.

- A *distributed computer system* combines the electronic link of the centralized system with the independent computers of the decentralized system. Functions within the computers can cooperate on the same task, and any computer in the DDP system can initiate or control a particular task.

The Evolution from Centralization to Decentralization to Distribution

Because of the cost of the earliest computers, the limited number of people trained to operate them, and the specialized site requirements, the earliest data processing systems were centralized. Very few users could justify the cost of multiple systems. Gradually, however, decentralized and then distributed systems evolved from centralized ones.

Decentralized systems evolve from centralized ones when you add an independent computer that's not connected to the old one. This new computer may be small, or it may be as elaborate as the central system.

Decentralized systems become distributed systems when you join the independent computers as equals. Each of the joined computers must be able to access data on any other.

1.4.2. Hardware and Communications Developments Leading to DDP

Centralized systems developed for a very good reason: cost. The original computer hardware cost so much that companies could generally justify the cost of only one computer and had to keep it running constantly to realize savings over manual methods. But the history of computer hardware costs is one of continually declining prices.

Many companies now have small computers in the field, where they do their own processing and send back reduced data through methods such as NTR. If communications costs are reasonable, the savings more than pay for the field computer. But as hardware costs continue to fall and as the field computers become more powerful, it makes less sense to use the field computers simply as terminals to a central computer. With DDP, computers can send jobs to each other during busy periods and can access facilities they don't have. At the same time, they can continue to send back processed data to a headquarters computer and to perform strictly local functions. Linking hosts saves money because the demand for excess capacity at each site during busy periods disappears.

You may decide to structure your DDP network using the star configuration with a main computer and subsidiary hosts.

Or you may do away altogether with a main computer and rely wholly on the combined capacity of your hosts by using a tree system.

Whichever you choose, Unisys offers you a range of hardware and software products to ensure that your system meets your demands.

1.5. Advantages of Distributed Data Processing

DDP Lets Managers Get Data Faster and More Easily

Efficiency increases when a manager has control over all the tools necessary to make decisions. Computerized data is a vital tool. Under a centralized computing system, managers may not have immediate access to their own records, which are compiled and stored elsewhere. Special reports may take longer to complete than they did under older, manual methods. And managers may not welcome new centralized computing functions, since such functions can mean loss of control.

Under a DDP system, however, local managers control the local computing site, the personnel, and the data. Increased communication between managers and data processing people can result in better services for local needs. Thus, local managers find themselves using more of the services the computer can perform.

DDP Supports the Organizational Structure of Your Company

Centralized computer systems make sense in a centralized organization. A centralized system, for example, may use remote terminals to collect raw data from branches, and use those same terminals to send back directions from the central, decision-making group. If the only managers with authority are at central headquarters, it makes sense for them to control all computer data.

Similarly, a decentralized system works well in a decentralized organization. If a manager's only responsibility to headquarters is a once-a-year profit report, and if his branch has little contact with other branches, it makes sense for him to have a strictly local computer system.

But most business organizations don't fall into either of these categories. Instead, they use what we might call a *distributed* management system, whereby decisions are made at different management levels. A distributed data processing system offers the ideal compromise: central coordination of, and access to, data processed and located at decentralized sites. Distributed organizations prevent too much central control on the one hand and *every man for himself* lack of coordination on the other. Local managers control local data, but central managers can quickly access it.

DDP Lowers Costs

If you have a central computer with many remote terminals that send raw data to it, your communications costs may be very high. But if, instead, you process that raw data on a small computer at the remote site and then send just a summary, communications costs are bound to be less.

DDP may also decrease the load on your central computer and increase throughput. The computer needs to spend less time switching among competing jobs and can devote more resources to accomplishing tasks. This may allow you to retain your present computer rather than having to purchase a larger one.

DDP also lets your organization share resources. A department that can gain access to a large computer in another branch may not need a large computer of its own. Through DDP, occasional users can access large systems promptly and efficiently as needed, avoiding demands for a larger local system.

DDP may also help you control personnel costs. Let's say you're planning to put independent computers at five different sites in your company. In the past, that might have meant a 500 percent increase in data processing personnel. But since DDP makes communications and coordination among those hosts faster and easier, you may be able to centralize skilled personnel and have smaller staffs of less skilled persons at the remote sites. Using DDP products, the central staff may be able to do a great deal of the program development and testing that otherwise would have been done at the remote sites.

DDP Helps You Safeguard Your Data

Some companies now make copies of files to send to a remote site to guard against the total destruction of records at one site. DDP can make this task easier or make the files more readily accessible. You can use DDP to make copies of your files on a remote host. Then, if files are destroyed, they can be accessed immediately on the remote host or recopied from the remote host. Thus, an accident or failure at one host doesn't destroy vital company data.

DDP Helps You Conserve Your Storage Facilities

By being able to access and process remotely located files, DDP helps you conserve your storage facilities. A file that is seldom used at a particular site need not occupy file space at that site. Instead, whenever that file is needed, it is simply accessed through the DDP file access facility.

DDP Helps You Maintain Program Control between Remote Systems

By providing the ability to have programs initiate and communicate with one another, DDP allows you to write programs that control the execution of other programs at remote sites. Thus, an application that requires several programs to be executed in a sequence determined by processing events can now be programmed to function automatically. That is, after the first program is initiated, it initiates the second, which in turn could initiate a third and fourth program, and so on. The communicating programs can be located on the same host or distributed among all the host systems in the DDP network.

DDP Helps You Communicate and Coact with Other Remote Systems

DDP lets you communicate with remote processors and initiate and control programs and also coact with remote processors working together on the same application program. Therefore, you can execute each part of your application job where it can be done most efficiently.

1.6. Computer Concepts for Distributed Data Processing

If you are an operator, programmer, or site administrator for a Unisys DDP system, you should be familiar with the following concepts. Also, see the glossary at the back of this guide for definitions of terms specific to DDP.

1.6.1. Using Spooling in DDP

Spooling is vital to a DDP system because it lets hosts store DDP data or output until devices are available at the destination host. Without spooling, you'd have to make sure that, for instance, any time you expect output back from a remote host, your printer or punch is free. With spooling, output from a remote host goes to your spool file to wait until the device is available.

1.6.2. Display Messages

Display messages tell you about your job. They may tell you that your job has been successfully submitted to the DDP network or that the remote host has completed it. The DDP software displays messages at the:

- Terminal, workstation, or system console where you entered the command
- Log of the host originating the command
- System console of the host originating the command
- Log of the receiving host
- System console of the receiving host

1.6.3. Automatic Logging and Accounting Methods with DDP

All DDP local and remote activity is automatically logged and sent to the central site printer for your reference. See 3.1.1.

Logging and job accounting for DDP is described in the *Spooling and Job Accounting Operating Guide*, 7004 4581.

1.6.4. Communications Requirements for DDP Hosts

DDP requires electronic communications, and Unisys is using its distributed communications architecture (DCA) to implement it. Each DDP host computer requires the following communications software products:

- Integrated communications access method (ICAM) software
- DCA termination system, or access to a public data network (PDN)

1.6.5. Entering Commands to a Remote Host or to Your Local Host

You can enter DDP commands at the system console, a terminal, or a workstation. Normally, you'll use a terminal or workstation that doesn't need to be dedicated to DDP.

Some DDP functions take a long time to execute. For instance, you may want to copy a very large file. Or you may want to start a program on a busy host and then have to wait to establish a connection. Fortunately, your terminal is not reserved for DDP during the entire DDP operation. Once you have entered the DDP commands to start your function, you may perform other tasks after the DDP command you entered has been accepted.

DDP commands aren't reserved just for remote tasks. You can also use them to perform local tasks. Simply use your own host's identification number for the receiving host. You may not want to do that often, since only limited kinds of local tasks can be performed through DDP commands. But if you suddenly need to perform a task, such as a job start or a file copy, while working with DDP, you can use DDP commands to do the job.

Section 2
DDP Products

2.1. Overview

Four Unisys DDP products are available:

- DDP transfer facility
- OS/3 to UNIX O/S DDP file transfer facility
- DDP file access facility
- IMS DDP transaction facility

This section presents an overview of these products, describing what they do for you. Section 3 gives their DDP network requirements, and Sections 4 through 6 explain how you use the file transfer and file access facilities.

Detailed information on using the IMS DDP transaction facility is not provided in this guide. See the appropriate IMS manuals (listed at the end of this section).

2.2. DDP Transfer Facility

The DDP transfer facility allows you to transfer files and jobs between OS/3 systems. You can also inquire about the availability of a file or the status of a job on a remote system. A simple set of English-based commands is all you need to use the file transfer facility.

2.2.1. Sending Files to a Remote Host

You can copy data files, library files, or individual modules from library files and send these files to another host.

You can transfer files in three ways:

- From a local system to a remote system
- From a remote system to a local system
- From a remote system to a remote system

You can also:

- Duplicate a data base by sending files to a remote host
- Send a library file of programs to a remote host and run the programs later
- Make a new copy of a file that has been updated at another host
- Send files to a remote host for a special project. For instance, an annual report may require one-time access to data on several hosts. If many separate points of access to those files are needed, the most efficient way to do the job may be to copy all the files onto one host and run the report program there.

2.2.2. Sending Jobs to a Remote Host

In addition to file copying, you can also:

- Send job control streams (a form of program module) from one host to another
- Start a job on a remote host
- Send, receive, and respond to messages from a job on a remote host
- Inquire about the status of your job on a remote host
- Communicate with the operator of a remote host

2.2.3. Job Control Language and Device Requirements for Remote Jobs

When you send job control streams to a remote host, you must know details about the capacities and devices of your remote host so you can write your job control stream to conform to its limits.

Job streams must be complete to be sent. And once you send them, they may be performed in any time sequence - not just in the one you originally intended. So don't send jobs to different hosts if the successful completion of one job depends on the prior execution of another. If, for instance, job A sends data to job B, execute both jobs on the same host.

2.2.4. What Happens to Output from Remote Jobs?

What happens to completed output? Normally, it returns automatically to the sending host's spool file. Then it's printed or punched, according to your instructions in the job control stream. In addition to this automatic return, you can send the output to a third host or send multiple copies throughout the DDP system.

2.2.5. Functions You Perform with Remote Jobs

Jobs are sent to remote hosts for various reasons. For example, you can:

- Move a job that's too large for one computer to a more powerful one
- Move a job needing a particular file to the host where the file resides

Perhaps you have a centralized inventory file comprised of data from 10 different sites. Each remote site needs to access or update the file only about once a month. But the site where the file resides updates it daily. It wouldn't make sense to keep copies of the file at each site and have to update them constantly. Instead, it's more efficient to keep just one file. When the remote sites need to access the file, they can submit their jobs through DDP to the site where the file resides.

Section 4 provides more detailed information on this facility.

2.3. OS/3 to UNIX O/S DDP File Transfer Facility

The OS/3 to UNIX O/S DDP file transfer facility lets you transfer files between OS/3 and UNIX systems. You can use DDP commands to create, copy, or purge files on a UNIX system from an OS/3 system or on an OS/3 system from a UNIX system.

You can copy data files or individual modules from library files to send these files to another host.

You can transfer files in three ways:

1. From a local system to a remote system
2. From a remote system to a local system
3. From a remote system to a remote system

Either the local or the remote system can be a UNIX system.

You can:

- Duplicate a data base by sending files to a remote host
- Send a library module to a remote host
- Make a new copy of a file that was updated at another host

See Section 5 for more detailed information on this facility.

2.4. DDP File Access Facility

The DDP file access facility enables you to access and process files residing on remote OS/3 systems and write application programs that can initiate, and communicate with, application programs on other OS/3 systems.

2.4.1. Remote File Processing

You can access MIRAM data files on remote OS/3 systems and process them on your local system simply by adding a host identification parameter to the device assignment set for that file. DDP will then connect your program with the remotely located file. No changes are required in your program. Remote file processing eliminates the need for you to copy a remote file on your system before you can access and process it.

2.4.2. Program-to-Program Communication

You can write basic assembly language (BAL) programs that can communicate (converse) with remote OS/3 systems in your DDP network. You can write primary/surrogate programs that you can use to transfer data from one host to another host, elaborate on the data, and then send it back to the original host.

The program that initiates the conversation is called the *primary* program, and the initiated program is called the *surrogate* program. However, the primary and surrogate can reverse roles. There is no limit to the number of status changes that can be requested between a pair of programs. In addition, up to 255 surrogate programs can be initiated by a primary program.

All program-to-program conversations are initiated and controlled by a set of consolidated data management macroinstructions embedded in both primary and surrogate programs. These macroinstructions enable a program to:

- Establish a conversation with another program (DOPEN)
- Direct a communication to a particular surrogate program (DMSEL)
- Transfer data between programs (DMOUT and DMINP)
- Transfer primary/surrogate control between programs (DMCTL)
- Terminate a conversation with a program (DCLOSE)

Section 6 provides more information on this facility.

2.5. IMS DDP Transaction Facility

The IMS DDP transaction facility lets you process IMS transactions at a remote OS/3 system. To use this facility, each OS/3 system must:

1. Define an ICAM global network that supports distributed data processing, including a LOCAP macroinstruction identifying each IMS system that will send or receive remote transactions

2. Configure a multithread IMS

3. Include a LOCAP statement in the configuration for each remote IMS

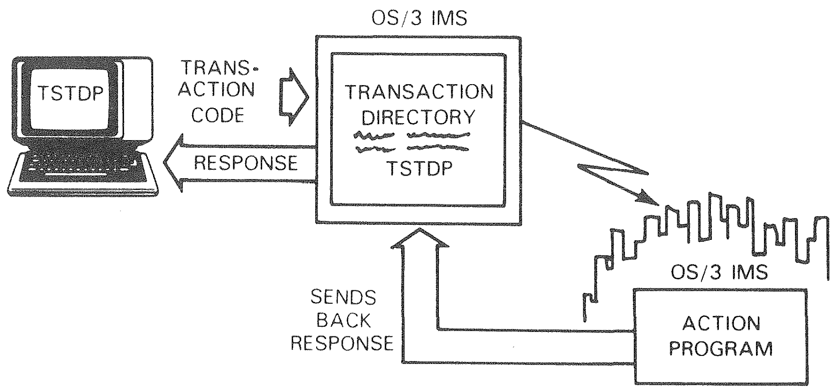
For further information, see the *IMS System Support Functions Programming Guide,* UP-11907, and the ICAM operating guide for your system.

In a DDP transaction, a terminal operator at one IMS system, called the primary IMS, initiates the transaction. The primary IMS, through the transaction facility, routes the transaction to a remote system where a secondary IMS processes the transaction and sends back a response. The remote transaction may be processed by user action programs (written in COBOL, RPG II, basic assembly language) or by UNIQUE (inquiry language). There is little difference between the way action programs process a remote transaction and the way they process a local transaction. Most IMS features are available, including the use of screen format services.

The three different ways in which IMS can route a transaction to a remote system are explained in the following subsections.

2.5.1. Directory Routing

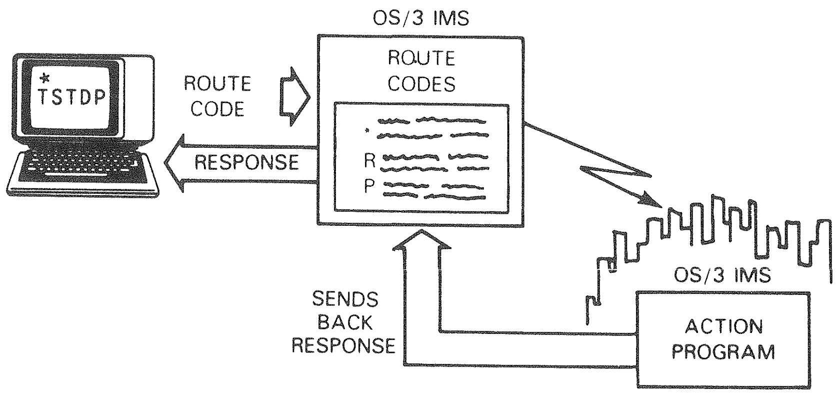
A terminal operator at the primary IMS enters a transaction code that identifies a transaction at a particular remote system. The transaction code and the remote system associated with it are defined to IMS in the configuration. The primary IMS routes the message to the secondary IMS, where action programs or UNIQUE process the transaction. Once the transaction begins, a communications link is established between the terminal operator and the remote system. This allows a dialog transaction consisting of multiple input and output messages.



Directory Routing

2.5.2. Operator Routing

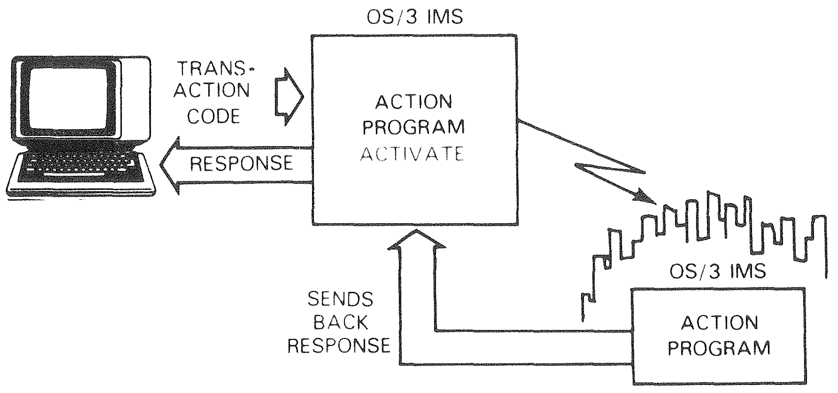
Operator routing is similar to directory routing. In this case, a special character is associated with the remote system, rather than a transaction code. The special character is defined in the IMS configuration or at IMS start-up. When the terminal operator enters a transaction code prefixed by the special character, IMS routes the message to the secondary IMS, where the transaction is processed in the same manner as in directory routing.



Operator Routing

2.5.3. Action Program Routing

The terminal operator enters a transaction code that initiates a transaction at the primary IMS system. A COBOL or basic assembly language (BAL) action program at the primary IMS issues an ACTIVATE function call to IMS, identifies the remote system in its output message header, and generates a message containing a transaction code. IMS routes this message to the remote IMS, where action programs process the transaction and return a message to the originating action program or its successor. The action program at the primary IMS can then return a message to the terminal operator or can issue another ACTIVATE call to initiate another remote transaction.



Action Program Routing

For specific network definition, configurator, and start-up requirements for IMS DDP processing, refer to the *IMS System Support Functions Programming Guide,* UP-11907.

For details on how to process IMS DDP transactions, refer to:

- *IMS Action Programming in RPG II Programming Guide,* UP-9206
- *IMS COBOL/Assembler Action Programs Programming Guide,* UP-9207

Section 3
DDP Network Requirements

3.1. The DDP System Environment

To use the DDP file transfer and file access facilities, all hosts in your system must have:

- Spooling
- Interactive services
- ICAM generated with the demand mode interface (DMI) and related ICAM software, including the DCA termination system
- Consolidated data management (CDM)
- Library utilities (for library file transfers)
- DDP transfer facility

Notes:

1. *For models 8-20 users, some of the required software is contained in Extended System Software (ESS).*
2. *On non-OS/3 systems, you must have that system's equivalent of OS/3 DDP, for example, UNIX Information Services (IS/5000 or IS/6000).*

To use the IMS DDP transaction facility, see the *IMS System Support Functions Programming Guide*, UP-11907, and the ICAM operations guide for your system.

You can improve DDP performance and reduce main storage fragmentation by having the DDP modules resident that are common to all DDP functions. When you generate the OS/3 supervisor, specify:

```
RESHARE=DDPL
```

to make all local processing modules resident, or

```
RESHARE=DDPR
```

to make all DDP modules resident, including the DDPL group. See the software installation guide for your system.

You may also modify the supervisor at initialization time. See the operations guide for your system. The same modules apply at either system generation or initialization.

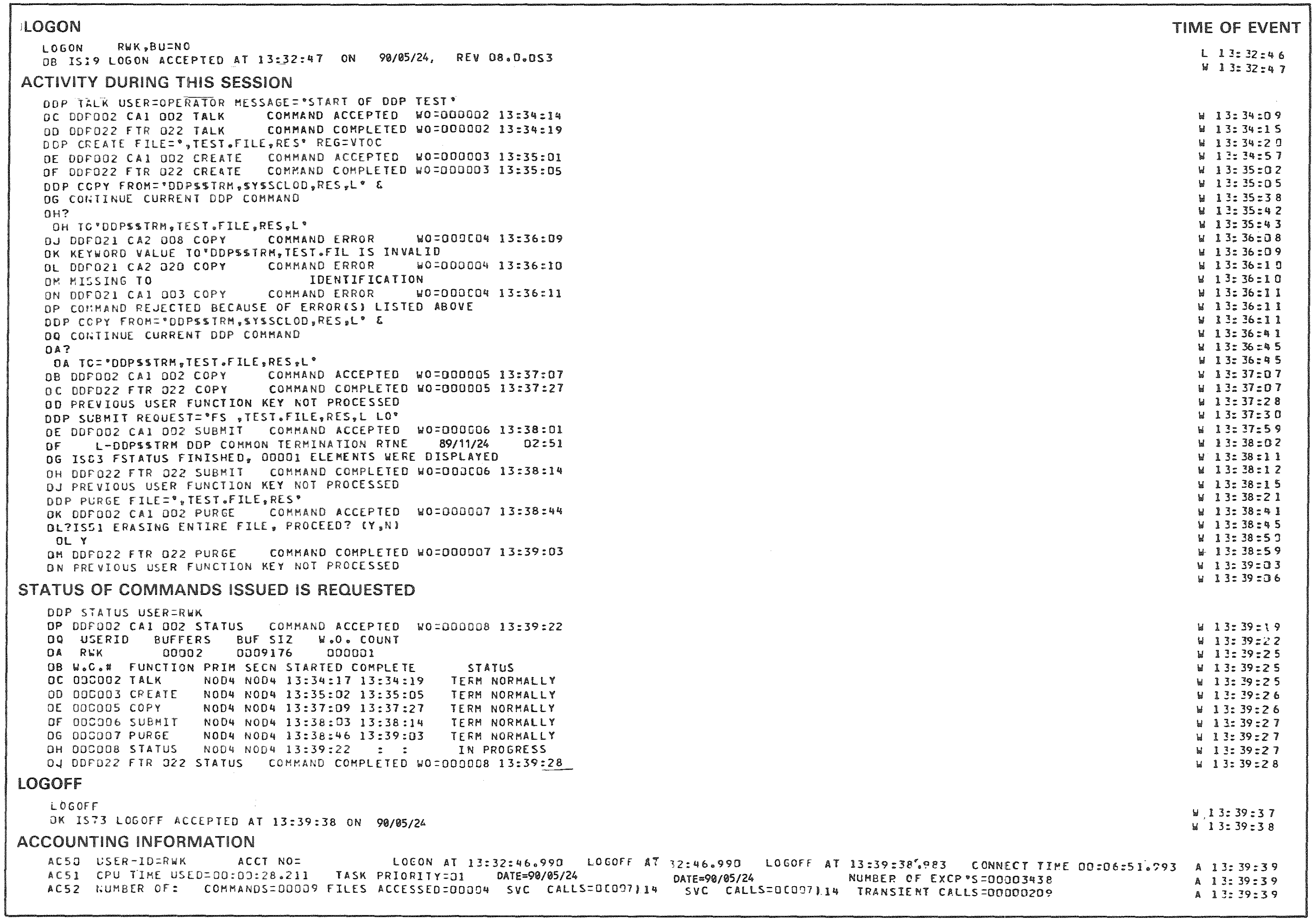
You enter DDP commands at a terminal, a workstation, or an OS/3 system console.

DDP also uses dynamic buffers, but you don't have to worry about their number or size. DDP takes care of that for you automatically. However, some DDP STATUS response messages (4.5.1) show you the number and size of buffers being used. The number varies with the DDP commands being run.

3.2. DDP Activity Logging

All DDP local and remote activity is logged in the interactive services log files on the systems involved with the DDP activity. This information is provided at the system console printer after interactive services shutdown time.

Figure 3-1 is a typical log printout of a session showing a series of commands that were entered by the operator. At the end of the activity, a DDP STATUS COMMAND was entered to provide the status summary shown. Accounting information is given at the end of the log printout.



```
LOGON                                                                                          TIME OF EVENT
   LOGON    RWK,BU=NO                                                                            L 13:32:46
   OB IS:9 LOGON ACCEPTED AT 13:32:47  ON   90/05/24,   REV 08.0.0S3                             W 13:32:47
ACTIVITY DURING THIS SESSION
   DDP TALK USER=OPERATOR MESSAGE='START OF DDP TEST'
   OC DDF002 CA1 002 TALK     COMMAND ACCEPTED  WO=000002 13:34:14                               W 13:34:09
   OD DDF022 FTR 022 TALK     COMMAND COMPLETED WO=000002 13:34:19                               W 13:34:15
   DDP CREATE FILE=',TEST.FILE,RES' REG=VTOC                                                     W 13:34:20
   OE DDF002 CA1 002 CREATE   COMMAND ACCEPTED  WO=000003 13:35:01                               W 13:34:57
   OF DDF022 FTR 022 CREATE   COMMAND COMPLETED WO=000003 13:35:05                               W 13:35:02
   DDP COPY FROM='DDP$$TRM,SYSSCLOD,RES,L' &                                                     W 13:35:05
   OG CONTINUE CURRENT DDP COMMAND                                                               W 13:35:38
   OH?                                                                                           W 13:35:42
    OH TC'DDP$$TRM,TEST.FILE,RES,L'                                                              W 13:35:43
   OJ DDF021 CA2 008 COPY     COMMAND ERROR     WO=000C04 13:36:09                               W 13:36:08
   OK KEYWORD VALUE TO'DDP$$TRM,TEST.FIL IS INVALID                                              W 13:36:09
   OL DDF021 CA2 020 COPY     COMMAND ERROR     WO=000004 13:36:10                               W 13:36:10
   OM MISSING TO                        IDENTIFICATION                                           W 13:36:10
   ON DDF021 CA1 003 COPY     COMMAND ERROR     WO=000C04 13:36:11                               W 13:36:11
   OP COMMAND REJECTED BECAUSE OF ERROR(S) LISTED ABOVE                                          W 13:36:11
   DDP COPY FROM='DDP$$TRM,SYSSCLOD,RES,L' &                                                     W 13:36:11
   OQ CONTINUE CURRENT DDP COMMAND                                                               W 13:36:41
   OA?                                                                                           W 13:36:45
    OA TC='DDP$$TRM,TEST.FILE,RES,L'                                                             W 13:36:45
   OB DDF002 CA1 002 COPY     COMMAND ACCEPTED  WO=000005 13:37:07                               W 13:37:07
   OC DDF022 FTR 022 COPY     COMMAND COMPLETED WO=000005 13:37:27                               W 13:37:07
   OD PREVIOUS USER FUNCTION KEY NOT PROCESSED                                                   W 13:37:28
   DDP SUBMIT REQUEST='FS ,TEST.FILE,RES,L LO'                                                   W 13:37:30
   OE DDF002 CA1 002 SUBMIT   COMMAND ACCEPTED  WO=000006 13:38:01                               W 13:37:59
   OF     L-DDP$$TRM DDP COMMON TERMINATION RTNE    89/11/24     02:51                           W 13:38:02
   OG IS03 FSTATUS FINISHED, 00001 ELEMENTS WERE DISPLAYED                                       W 13:38:11
   OH DDF022 FTR 022 SUBMIT   COMMAND COMPLETED WO=000C06 13:38:14                               W 13:38:12
   OJ PREVIOUS USER FUNCTION KEY NOT PROCESSED                                                   W 13:38:15
   DDP PURGE FILE=',TEST.FILE,RES'                                                               W 13:38:21
   OK DDF002 CA1 002 PURGE    COMMAND ACCEPTED  WO=000007 13:38:44                               W 13:38:41
   OL?IS51 ERASING ENTIRE FILE, PROCEED? (Y,N)                                                   W 13:38:45
    OL Y                                                                                         W 13:38:50
   OM DDF022 FTR 022 PURGE    COMMAND COMPLETED WO=000007 13:39:03                               W 13:38:59
   ON PREVIOUS USER FUNCTION KEY NOT PROCESSED                                                   W 13:39:03
                                                                                                 W 13:39:06
STATUS OF COMMANDS ISSUED IS REQUESTED
   DDP STATUS USER=RWK
   OP DDF002 CA1 002 STATUS   COMMAND ACCEPTED  WO=000008 13:39:22                               W 13:39:19
   OQ  USERID   BUFFERS   BUF SIZ   W.O. COUNT                                                   W 13:39:22
   OA  RWK        00002   0009176    000001                                                      W 13:39:25
   OB W.C.#  FUNCTION PRIM SECN STARTED COMPLETE       STATUS                                    W 13:39:25
   OC 000002 TALK     NOD4 NOD4 13:34:17 13:34:19   TERM NORMALLY                                W 13:39:25
   OD 000003 CREATE   NOD4 NOD4 13:35:02 13:35:05   TERM NORMALLY                                W 13:39:26
   OE 000005 COPY     NOD4 NOD4 13:37:09 13:37:27   TERM NORMALLY                                W 13:39:26
   OF 000006 SUBMIT   NOD4 NOD4 13:38:03 13:38:14   TERM NORMALLY                                W 13:39:27
   OG 000007 PURGE    NOD4 NOD4 13:38:46 13:39:03   TERM NORMALLY                                W 13:39:27
   OH 000008 STATUS   NOD4 NOD4 13:39:22   :  :      IN PROGRESS                                 W 13:39:27
   OJ DDF022 FTR 022 STATUS   COMMAND COMPLETED WO=000008 13:39:28                               W 13:39:28
LOGOFF
   LOGOFF                                                                                        W 13:39:37
   OK IS73 LOGOFF ACCEPTED AT 13:39:38 ON  90/05/24                                              W 13:39:38
ACCOUNTING INFORMATION
   AC50  USER-ID=RWK       ACCT NO=             LOGON AT 13:32:46.990   LOGOFF AT 32:46.990   LOGOFF AT 13:39:38.983   CONNECT TIME 00:06:51.293   A 13:39:39
   AC51  CPU TIME USED=00:00:28.211   TASK PRIORITY=01    DATE=90/05/24           DATE=90/05/24            NUMBER OF EXCP'S=00003438              A 13:39:39
   AC52  NUMBER OF:   COMMANDS=00009 FILES ACCESSED=00004  SVC  CALLS=0C007).14    SVC  CALLS=0C007).14   TRANSIENT CALLS=00000209                A 13:39:39
```

Figure 3-1. DDP Activity Log Printout Sample

DDP allocates a 1,024-byte buffer for each user accessing DDP functions. This buffer contains a log of all activity for the particular user and is the source of information displayed by the DDP STATUS USER=userid command. The buffer contains enough room for log information for the last 15 DDP commands.

3.3. Host Identification Requirements

Each host in your DDP network has a host identification (host-id). This is the same identification as the label of your DMI LOCAP macroinstruction in your ICAM network definition. You must know the host-id to send files to, or communicate with, a remote host.

The host-id is one to four alphanumeric characters long. The first character must be alphabetic.

3.4. Using the DDP Network

You must always have ICAM and interactive services running to use DDP. So, the first things to do to prepare your system for DDP are:

1. At the system console, enter Cn or Mn to load the appropriate ICAM symbiont,

   where:

   `Cn or Mn`

   Is the name specified on the MCPNAME parameter in the COMMCT phase of SYSGEN; n is a numeric digit 1 to 9 that identifies the network to be loaded.

   The message ICAM READY appears.

2. Run the GUST (ML$$GI) job stream given in the ICAM operations guide for your system.

3. You are now ready to use the DDP facilities.

In case of system or communications line failure, the DDP system will recover automatically if the automatic recovery feature has been selected by the console operator at system initialization time. When the system is available again, the recovered work orders are reinitialized automatically. See the operations guide for your system.

Section 4
The DDP Transfer Facility

4.1. What the DDP Transfer Facility Does for You

The DDP transfer facility lets you enter commands at your workstation or terminal to:

- Create a file on a host
- Copy a file from one host in your system to another
- Remove a file from a host
- Send a job control stream to a host
- Run a job control stream on a host
- Receive the output from your executed job or send the output to another host
- Find out the status of a command, host, job, file, or user in the system
- Send a request (such as an operator command) to a host
- Send a message to an operator or user at any host in your system
- Terminate a job already submitted

So, using DDP, you can perform many tasks on another computer.

But, before you start entering commands, you need some background information.

4.2. Using the DDP Transfer Facility

This subsection describes how to use the DDP transfer facility.

4.2.1. File Identification Requirements

The DDP transfer facility uses a specific form of file identification (file-id). It contains not only the name of the file, but also the name of the volume that contains the file (if the file isn't cataloged), the module name and type, and the read and write passwords. It is a maximum of 74 characters long plus required punctuation marks, such as commas, slashes, and parentheses, for a total of 81 characters. It is similar to the file-id used in other OS/3 products, such as interactive services, and is always expressed as a character string (in apostrophes).

The file-id format is:

```
'[module-name],filename[([read-password]/[write-password])],[vol],[module-type]'
```

where:

`module-name`
: Is used only when you're sending just one module from a program library file. It is the one to eight alphanumeric characters identifying the module. It must begin with an alphabetic character.

`filename`
: Is always required. It is the same as the name on the // LBL job control statement for the program. For disk files and standard format label diskettes, it is 1 to 44 alphanumeric characters long. For tape files, data set label diskettes, and logical files (spool files), it is 1 to 17 alphanumeric characters long.

`read-password`
: Is a password (one to six alphanumeric characters) enabling you to read the file. If omitted, the system assumes there is no read password on the DDP command. If the file does have a read password that you omit, you won't be able to read the file.

`write-password`
: Is a password (one to six alphanumeric characters) enabling you to write to the file. If omitted, the system assumes there is no write password on the DDP command. If the file does have a write password that you omit, you won't be able to write to the file.

`vol`
: Is the volume serial number (one to six alphanumeric characters) of the volume on which your file resides. Specify this when the file is not cataloged.

`module-type`
: Is the type of code (one to four alphanumeric characters) contained in the program module you're sending.

The module types are the same ones you're familiar with for all OS/3 library files. SAT files may have the following types:

- S (source code)
- M (macro)
- P (procedure)
- L (load code)
- O (object code)

The default is S (source).

MIRAM files may have the following types:

- F and FC (screen format modules)
- J (saved job control stream)
- MENU (menu modules)
- HELP (help screen modules)

For MIRAM files, you may create your own element type and identify it with one to four characters. (For further information on MIRAM files, see the *Consolidated Data Management Programming Guide,* UP-9978.)

Table 4-1 gives some examples of complete file-ids.

Table 4-1. Examples of File-ids

Type of File	Example
A cataloged file with no read and write passwords	`',SITE/3'`
A cataloged file with read and write passwords	`',SITE/3(XX2674/BENNIS)'`

A source module from a cataloged file with read and write passwords	`'PROG14,JOBFIL19(XX4982/RILEY)'#&`
	`',,S'`

An uncataloged file with a write password but no read password	`',SITE/A(/SMPRO),VSN149'`
An uncataloged file with a read password but no write password	`',SITE/A&B(READ/),296410'`

4.2.2. Requirements for Using DDP Commands within Your Local Host

You may use DDP commands to work with files and jobs entirely within your local host. The DDP transfer facility accepts your local host-id as a valid host-id for all commands. In addition, DDP commands use your local host as a default if you fail to provide a host-id. Thus, if you're working with DDP and must perform a local task, you can perform the task with a DDP command.

4.2.3. Preparing Your Terminal to Enter DDP Commands

If you're entering DDP commands from a terminal rather than from a workstation or system console, perform the following steps. If you're using a workstation, see 4.2.4. If you're entering DDP commands from a system console, see 4.2.5.

1. Run the GUST program (ML$$GI) once from the system operator's console.

2. Sign on the terminal.

Continue with 4.2.4.

4.2.4. Logging On (LOGON)

If you're using a workstation or terminal, enter the LOGON command with your user identification. (You don't need the LOGON statement if you're entering commands from a system console.) The format is:

```
LOGON user-id
```

where:

`user-id`
: Is the one to six alphanumeric characters identifying you as a user to the host.

Example:

```
LOGON CHAR
```

If the system accepts your LOGON command, it returns a message stating so.

4.2.5. Using the DDP Software (DDP)

If your terminal, workstation, or system console is connected to a DDP network, you enter DDP software whenever you issue a DDP command. DDP scans your commands and informs you of all detectable syntax errors. (See the *System Messages Reference Manual,* 7004 5190, for a complete list of all DDP messages.) Once the command is correct, DDP accepts it and forwards it to the DDP system.

DDP Gives You a Work Order Number to Reference Your Commands

Once DDP accepts your command, it returns a work order number to you in the format:

```
DDP002 CA1 002 cccccccccc COMMAND ACCEPTED WO=nnnnnn time
```

where:

cccccccccc
: Is the name of the command you just entered.

nnnnnn
: Is the work order number of the command.

For instance, let's say you've just entered the DDP COPY command. DDP responds:

```
DDP002 CA1 002 COPY COMMAND ACCEPTED WO=A48107 13:46:02
```

Write down the work order number, since error messages use it for identification. (You may want to use the log sheet in Appendix F to do this.) If, for example, you enter a command that cannot be completed at the remote host, DDP informs you of the problem by sending you a message, such as:

```
DDP001 ELH 023 COPY COMMAND ABORTED WO=A48107 13:48:39
```

WO=A48107 is the work order number that DDP returned to you at the time you entered your copy command. You may have entered several copy commands, but each has a different work order number so you can tell them apart.

DDP Gives You a Job Name to Reference Your Jobs

When you enter a DDP SUBMIT FILE command, DDP returns a job name to you with which you can cancel the job if you later need to.

DDP renames all jobs submitted with the DDP SUBMIT FILE= command. The job name is eight characters long with the format xxxxyyyy.

where:

xxxx
: Is the host-id of the initiated job.

yyyy
: Is a unique four-digit number.

The format for the job name message is:

```
DDP044 EJS 044 jobname JOB SUBMITTED FOR WO=work-order-number time
```

The work order number identifies the command you entered. (In this case, it identifies the DDP SUBMIT FILE command.)

DDP Sends Messages to You from the Remote Host

As your command or job is executing on the remote host, it produces messages informing you of its status and of any problems encountered. DDP sends you these messages at your terminal or workstation. If you are still logged on the system at the time the message is produced, it is displayed immediately. If you've logged off, DDP displays your messages on the system console.

For a complete list of DDP messages, see the *System Messages Reference Manual,* 7004 5190.

4.2.6. No Need to Terminate DDP

You can go on to another task as soon as your command is accepted by DDP. However, all commands must be complete for you to log off.

4.2.7. Entering DDP as a Batch Operation

DDP is designed as an interactive product. But you can use DDP commands in a job control stream. For instance, if it's 4:30 p.m. and you're leaving at 5:00 p.m., you can punch cards with DDP commands on them and submit them in the same way you'd submit any other remote batch job. Your results and messages are printed at your local host.

Appendix E gives an example of DDP commands entered as a remote batch job.

4.3. Remote File Commands

There are three remote file commands:

- The DDP CREATE command establishes a file on a host.
- The DDP COPY command copies the contents of a file or modules from a file to an established file on a host.
- The DDP PURGE command removes the file or module and all references to it from the host.

4.3.1. Creating a File (DDP CREATE)

Function and Requirements

The DDP CREATE command establishes a file on a receiving host. It allocates space for the file and either catalogs the file in your online system catalog or records it in your volume table of contents (VTOC).

The DDP CREATE command is the most complex DDP command. But, once you have the file established, copying and using it are easy.

Follow these requirements for correct file creation:

- The file name cannot already exist on the receiving host. If you try to use a file name already being used, you get an error message. For more information on file names, see 4.2.1.
- Defaults on the file specifications are those of the receiving host. If you do not want these defaults, and if your requirements cannot be met through the DDP CREATE command parameters, create your file by submitting a batch job control stream to the host. (See 4.4.1.)
- You may omit the host-id specification and create a job stream to catalog all remote files on your local system. For an example, see the *Job Control Programming Guide,* 7004 4623.
- The only required parameter is file-id.

Format

```
DDPΔCREATEΔFILE=┌                ┐
                │{host-id       }::│file-id
                │{local-host-id }  │
                └                ┘

                ┌                                                ┐
                │ΔBLOCK_SIZE={number-of-characters/1-9 digits}   │
                │            {256                            }   │
                └                                                ┘

                ┌                          ┐
                │ΔDEVICE_CLASS={DISK     } │
                │              {TAPE     } │
                │              {DISKETTE } │
                └                          ┘

                ┌                         ┐
                │ΔFILE_TYPE={SEQUENTIAL}  │
                │           {INDEXED   }  │
                │           {LIBRARY   }  │
                │           {UNDEFINED }  │
                └                         ┘

                ┌                                                   ┐
                │ΔINITIAL_SIZE={initial-number-blocks/1-9 digits}   │
                │              {3 cylinders                     }   │
                └                                                   ┘

                ┌                      ┐
                │ΔREGISTER={VTOC    }  │
                │          {CATALOG }  │
                └                      ┘
```

Parameters

`host-id`
Is one to four alphanumeric characters that name the host you create the file on. If you omit the host-id, the command creates a file on your local system. For more information, see 3.1.2.

`file-id`
Is 1 to 74 alphanumeric characters identifying your file. For more information, see 4.2.1.

```
BLOCK_SIZE={number-of-characters/1-9 digits}
           {256                            }
```
Is the number of characters in a block. If omitted, the default value is 256 characters. If DEVICE_CLASS=DISKETTE, your maximum block size is 256 characters.

```
DEVICE_CLASS={DISK    }
             {TAPE    }
             {DISKETTE}
```
Is the device class that will contain your file. The default is DISK.

- If you specify DEVICE_CLASS=DISKETTE, the diskettes can be either data set label diskettes or format label diskettes. Either type of diskette can be used for MIRAM data files. If you intend to use the diskette for library files, then the diskette must be prepped as a format label diskette with the following parameters in the prep job stream:

  ```
  FORMT=FLB  DNSTY=2  RECSZ=256
  ```

  The diskette must be double sided and double density.

  The INITIAL_SIZE parameter of the DDP CREATE command must be specified.

- If you specify DEVICE_CLASS=TAPE, tapes can be used for MIRAM data files only.

```
FILE_TYPE={SEQUENTIAL}
          {INDEXED   }
          {LIBRARY   }
          {UNDEFINED }
```
Is the type of file being created. The default is UNDEFINED.

A SEQUENTIAL file is one that you access record-by-record, according to the order of the records in the file.

An INDEXED file, such as a MIRAM file, is accessed according to one or more index keys.

A LIBRARY file is a collection of program modules.

An UNDEFINED file may be any type of file. When you create a file as UNDEFINED, and then copy a file to it, it takes on the characteristics of the originating file. However, there is one restriction: if you're copying a source library file to an UNDEFINED file that doesn't have anything in it, POSITION must equal SOF. (See 4.3.2.)

```
INITIAL_SIZE=┌initial-number-of-blocks/1-9 digits┐
             └3 cylinders                        ┘
```

Is the number of blocks (1 to 999,999,999) initially allocated to the file. Omit this parameter for tapes. The default is 3 cylinders.

If DEVICE_CLASS=DISKETTE, you must also specify INITIAL_SIZE.

```
REGISTER=┌VTOC   ┐
         └CATALOG┘
```

Specifies that the file you create is either cataloged in the catalog file (specify CATALOG) or simply registered in the volume table of contents of the volume you're using (specify VTOC). If omitted, CATALOG is assumed.

Registering the file only in the VTOC takes less time and costs less than cataloging it. However, there are two disadvantages to registering it in the VTOC only. One is that every time you refer to the file, you have to specify the volume name as part of the file-id. The other is that you cannot perform an indirect copy to an uncataloged file. (For additional information on indirect copies, see the DDP COPY command format and parameters, MODE= parameter, 4.3.2.)

Example

You want to create a cataloged file on host H001, disk volume VOL007, that has read and write passwords and a block size of 182. Records are fixed. The file is smaller than three cylinders. You're planning to copy a file into the one you're creating, so you can use the FILE_TYPE default of UNDEFINED. The command is:

```
                                   read
                 host-id         password        volume
                    |                |              |
                  ┌─┴┐             ┌─┴┐          ┌──┴─┐
DDP CREATE FILE=H001::',REM/PERS(AXS9/AXSA7),VOL007' BLOCK_SIZE=182
                        └──┬──┘       └┬─┘           └──────┬──────┘
                       file name     write              number of
                                    password            characters
                                                        in block
```

4.3.2. Copying Files or Modules (DDP COPY)

Function and Requirements

With the DDP COPY command, you can duplicate either an entire file or a module from a file on another system. Follow these requirements to ensure correct copying:

- You can copy only to an established file. Thus, you might first issue a DDP CREATE command to establish the file, and then a DDP COPY command to fill it. However, you can copy to any file on a remote host to which you have access; the file doesn't have to be established through a DDP command.
- You can copy any type of library file or element.
- You can copy only character-oriented MIRAM data files.
- You must know the locations (host-ids) of all files you access.
- You may omit the host-id specification if you create a job stream to catalog all remote files on your local system. For an example, see the *Job Control Programming Guide,* 7004 4623.
- See Table 4-2 for the ways records are stored in files of various types.
- See Tables 4-3 and 4-4 for the limitations on originating and destination files during a DDP COPY.
- The two files may be on the same or different hosts. For instance, you may copy from file X on host A to file Y on host A. Or you may copy from file X on host A to file Y on host B.
- You may copy from a remote file to another remote file. In other words, your local host doesn't have to be either the originating or the destination host.
- The record forms of the originating and destination files do not have to be the same.
- You may not use a DDP COPY command to alter the element-type of a module. Its element type remains the same in the destination file as it was in the originating file.
- You can use a DDP COPY command to alter a user-specified MIRAM library element type.
- After completion of a DDP copy command for source, macro, and procedure elements, the header record in the new element will not contain the original date and time stamp, comments, and patch indicator. The date and time stamp reflect the date and time the operation took place. The comments and patch indicator are deleted.

Format

```
DDPΔCOPYΔFROM=┌ ┌originating-host-id┐:: ┐originating-file-id
              │ {local-host-id      }    │
              └ └                   ┘    ┘

           ΔTO=┌ ┌destination-host-id┐:: ┐destination-file-id
               │ {local-host-id      }    │
               └ └                   ┘    ┘

               ┌ΔKEY┌_ ┌n┐┐ = (size,location┌, ┌DUPLICATES   ┐┐ ┐
               │    │  {1}│                 │  {NO_DUPLICATES}│ │
               │    └   ┘┘                  └                ┘ │
               │                            ┌, ┌CHANGE   ┐┐)    │
               │                            │  {NO_CHANGE}│     │
               └                            └            ┘     ┘

               ┌ΔMODE=┌DIRECT  ┐┐
               │      {WAIT     }│
               │      └INDIRECT ┘│
               └                 ┘

               ┌ΔPOSITION=┌EOF┐┐
               │          └SOF┘│
               └               ┘

               ┌ΔTRANSLATE=┌EBCDIC┐┐
               │           └NONE  ┘│
               └                   ┘
```

Parameters

`originating-host-id`
: Is one to four alphanumeric characters naming the originating host. If omitted, DDP assumes the originating file is on your local host. For more information, see 3.1.2.

`originating-file-id`
: Is 1 to 74 alphanumeric characters identifying the input file. For more information, see 4.2.1.

  If your originating file is a LIBRARY file, its element type must be the same as the element type of the destination file.

  Note: *Table 4-3 shows restrictions on the DDP COPY command for some types of originating files.*

`destination-host-id`

Is one to four alphanumeric characters naming the host to receive the data. If omitted, DDP assumes the destination file is on your local host. For more information, see 3.1.2.

`destination-file-id`

Is 1 to 74 alphanumeric characters identifying your file. For more information, see 4.2.1.

If the destination file is a LIBRARY file, its element type must be the same as the element type of the originating file.

Note: *Table 4-4 shows some restrictions on the DDP COPY command for various types of destination files.*

```
KEY ┌    ┌n┐ ┐ = (size,location ┌ , ┌DUPLICATES   ┐ ┐
    │ _  {  }│                  │   {              }│
    └    └1┘ ┘                  └   └NO_DUPLICATES┘ ┘

                                ┌ , ┌CHANGE   ┐ ┐ )
                                │   {          }│
                                └   └NO_CHANGE┘ ┘
```

Is used to change an index for the file on the receiving host. You need one of these parameters for each index key in the record (to a maximum of five), even if you change only one. You omit this parameter for any nonindexed file.

`n`

Is the number of the key being changed. If omitted, 1 is assumed. You must start with 1 and continue sequentially through 5. Specify all the keys, each in its own KEY_n parameter, even though you may be changing only one of them.

`size`

Is the number of character positions in the key being changed. The maximum size is 80 characters per key.

`location`

Is the number of bytes in the record that precedes the key.

DUPLICATES permits identical values for different keys in the file. NO_DUPLICATES prohibits keys with identical values.

CHANGE permits future file update programs to change the index. NO_CHANGE prohibits change.

To illustrate the use of the KEY parameter, let's say you're copying a nonindexed file to a remote system and you want to create a two-key index. Each record in your file has the following fields:

	employee-name	employee-number	social-security-number	pay-class
Number of characters:	50	8	9	2

You want to use the employee-number field as the first index key and the pay-class field as the second key. Therefore, in the CHANGE command, you first specify:

```
The size is 8 since the employee-                          We want to allow change, but
number field contains 8 character- ----------┐       ┌---- there won't be any duplicates
positions.                                   |     ┌-┴-┐   in employee numbers.
                                     KEY_1 =(8,50,CHANGE)
                                         |     └┬
This is the first index key.    ---------┘      └-------The location is 50, the number of
                                                        bytes in the record that precedes
                                                        the employee-number field.
```

Following this same plan, the other key you need to specify is:

```
The number of character positions --┐        ┌-------------Many employees have
in the pay-class field              |    ┌---┴------┐      the same pay class.
                              KEY_2 =(2,67,DUPLICATES,CHANGE)
                                  |    └┬              └-┬┘
     The second key    -----------┘     |                └--- The pay-class field
                                        |                     changes when employees are
                                        |                     promoted.
                                        |
                                        └-- The location is 67, the number of bytes in
                                            the record that precedes the pay-class
                                            field.
```

```
MODE=⎧DIRECT  ⎫
     ⎨WAIT    ⎬
     ⎩INDIRECT⎭
```

MODE=DIRECT means that the device or medium needed to copy the file at the destination host must be immediately available. If not, the command aborts. DIRECT is the default.

MODE=WAIT means that if either the destination file or the originating file is not immediately available, the system should hold the command until both the originating file and destination file are free and then execute the DDP COPY. In this case, of course, the COMMAND COMPLETED message won't appear until the devices are freed and the copy performed. Note, however, that each host has a time limit for which it will hold any command. If this time limit elapses and the devices are still not free, DDP aborts your command.

MODE=INDIRECT permits the facility to build a temporary file at the destination host, if necessary, to store the file copy for later automatic transfer to the proper device. If it is possible to perform a DIRECT copy, however, the system will do so. You may not specify INDIRECT unless your destination file is cataloged. MODE=INDIRECT is only available for SAT or MIRAM library files.

```
POSITION=⎧EOF⎫
         ⎩SOF⎭
```

Specifies overwriting or extending of the destination file. POSITION=EOF (end of file) means that a copy of the originating file is appended to the end of the destination file. POSITION=SOF (start of file) means a copy of the originating file overwrites the destination file, and all previous contents of the destination file are lost. The default is EOF.

If you specify EOF, then:

- The destination file can't be an UNDEFINED file that's empty.
- The block and record sizes of the originating and destination files must match.
- Record storage is as shown in Table 4-2.

Table 4-2. How Files Store Additional Records at the End of File

If the destination file is this type:	Then records are added to the destination file:
INDEXED	By record key
LIBRARY	With new modules overwriting those with the same name and additional modules added to the end of the file
SEQUENTIAL	In the order transferred, following the last record of the destination file

If you specify SOF, the destination file may have any file type, because it's completely overwritten with the originating file, whose type it adopts. RELATIVE files store records in successive relative record positions, beginning with the first.

Tables 4-3 and 4-4 give additional information on the use of the POSITION parameter.

```
TRANSLATE={EBCDIC}
          {NONE  }
```

Indicates the character code you want the file translated to as it arrives at the destination host. If omitted, EBCDIC is assumed.

If you send a file to a host using a different encoding system but you do not want the file to be translated to the host's encoding system, specify NONE. NONE means that the destination host holds the file in the original code. It can transfer the file but cannot access it to perform useful work.

The TRANSLATE= keyword parameter is not supported when the source and destination files reside on the same host.

Table 4-3. DDP COPY Restrictions for Originating Files

If the originating file type is:	And POSITION=	And you also specify:	Then:
INDEXED	EOF	An INDEXED destination file	Keys of both originating and destination files must be identical.
LIBRARY	EOF	That the complete library is to be copied, or the file-id contains an element type but no module name	Destination file must be LIBRARY and the TO=parameter must not contain a module name.
LIBRARY	EOF	Module name	Destination file may be LIBRARY (with or without module name specified), SEQUENTIAL, or INDEXED.
SEQUENTIAL or INDEXED	EOF		Destination file may be LIBRARY, SEQUENTIAL, or INDEXED.
UNDEFINED	EOF or SOF		The file is treated as a data file.
UNDEFINED with nothing in it	EOF or SOF		DDP COPY command is aborted.

Table 4-4. DDP COPY Restrictions for Destination Files

If the destination file type is:	And POSITION=	And you also specify:	Then:
INDEXED	EOF	An INDEXED originating file	Keys of both files must be identical.
LIBRARY	EOF or SOF	Module name	Originating file must be LIBRARY.
LIBRARY	EOF or SOF		Modules from the originating file with the same name as destination-file modules overwrite those modules. Other modules are added to the end of the file. If SOF, the destination file is initialized and original contents are lost.
LIBRARY	EOF	Module name and an originating file that is not a LIBRARY file	The resulting module at the destination file is source (symbolic).
UNDEFINED with nothing in it			POSITION must be SOF. If POSITION=EOF, the command is aborted.

Example

You want to copy file PERSNNEL into the blank cataloged file REM/PERS on host H001, which uses EBCDIC. If a direct connection isn't possible, you want the system to hold the command until it is. The command is:

```
                           read            destination
                         password            file name
                            |                    |
                          ┌─┴─┐             ┌────┴────┐
DDP COPY FROM=',PERSNNEL(JMS8/)' TO=H001::',REM/PERS(/ASXA7)' MODE=WAIT
                └──┬───┘          └┬┘               └─┬┘
              originating        host-id            write
               file name                           password
```

Gang Copy

You can gang copy SAT/MIRAM library modules using the DDP COPY command. The gang copy operation uses a period in the module name field. To perform the gang operation, follow the period with a comma.

Example

If you gang copy a module and use its prefix, enter the module name prefix followed by a period and a comma in the module name field of the input file name string. When you use a module prefix, specify a module type, or the default type S (source code) is assumed.

```
DDP COPY FROM='ABCD.,$Y$MIC,RES,L' &
         TO=',MICRO.FILE,RES'
```

This command copies all modules of type L with a prefix of ABCD.

Notes:

1. *If you specify a module prefix with a period followed by a non-comma character, only that module is copied. For example, ABCD.1 does not copy all modules beginning with the prefix ABCD.*

2. *When you want to use a gang operation, don't place the period in the eighth position of the module name field. DDP cannot determine if you want a gang operation and copies only the specified module.*

4.3.3. Purging Files or Modules (DDP PURGE)

Function and Requirements

The DDP PURGE command physically removes a file or module and all references to it from a host. Follow these instructions to ensure proper purging of your file:

- You may purge only existing files or modules. A DDP PURGE command for a nonexistent file results in an error.

- If the file or module has passwords, you must know them to purge it. If the file has both read and write passwords, you must specify both to purge the file.

- If the file is cataloged, the DDP PURGE command will decatalog the file.

Format

```
               ┌                   ┐
DDPΔPURGEΔFILE=│ ┌host-id       ┐::│file-id
               │ └local-host-id ┘  │
               └                   ┘
```

Parameters

`host-id`
: Is one to four alphanumeric characters naming the host you are purging the file or module from. If omitted, DDP assumes the file is on your local system. For more information, see 3.1.2.

  You may omit the host-id specification and create a job stream to catalog all remote files on your local system. For an example, see the *Job Control Programming Guide,* 7004 4623.

`file-id`
: Is 1 to 74 alphanumeric characters identifying the file to be purged. For more information, see 4.2.1.

Example

You want to purge cataloged file REM/PERS on host H001. The command is:

```
                                    read
                  host-id         password
                     |               |
                    ┌┴┐             ┌┴┐
DDP PURGE FILE=H001::',REM/PERS(AXS9/AXSA7)'
                       └──┬───┘      └┬┘
                      file name      write
                                    password
```

Gang Purge

You can gang purge SAT/MIRAM library modules using the DDP PURGE command. The gang purge operation uses a period in the module name field. To perform the gang operation, follow the period with a comma.

Example

If you gang purge a module and use its prefix, enter the module name prefix followed by a period and a comma in the module name field of the input file name string. When you use a module prefix, specify a module type, or the default type S (source code) is assumed.

```
DDP PURGE FILE='ABCD.,$Y$MIC,RES,L'
```

This command purges all modules of type L with a prefix of ABCD.

Notes:

1. *If you specify a module prefix with a period followed by a non-comma character, only that module is purged. For example, ABCD.1 does not purge all modules beginning with the prefix ABCD.*

2. *When you want to use a gang operation, don't place the period in the eighth position of the module name field. DDP cannot determine if you want a gang operation and purges only the specified module.*

4.4. Remote Job Commands

There are three DDP job commands:

- The DDP SUBMIT FILE command sends a file of job control streams to a host for execution. You also use it to initiate a file of job control streams already at the host or to bring a job control stream to your local host for execution.
- The DDP CANCEL command terminates a job that you've submitted.
- The DDP SUBMIT REQUEST command sends an operator command to a remote host.

4.4.1. Submitting Files and Modules for Execution (DDP SUBMIT FILE)

Function and Requirements

The DDP SUBMIT FILE command sends a file of job control streams to a host for execution. You also use it to initiate a file of job control streams already at the host or to bring a job control stream to your local host for execution. The receiving host returns the output to you, sends it to another host, or retains it.

Note: *For information on releasing held spool files for processing, see "Manual Release of Held Spool Files (DDP SUBMIT REQUEST,'BEGIN SPL') in subsection 4.4.3.*

Follow these instructions to ensure proper functioning of the DDP SUBMIT FILE command:

- The module or file you submit must contain complete job control streams. You can submit a maximum of 10 job streams in the element file.
- You may submit only SYMBOLIC (source code) or COMPILED_JOB (compiled jobs with expanded jprocs) files and modules.
- Output can be routed to any host in the network, but there must be a direct connection from the host on which the jobs were executed and the host to where the output is directed.
- The destination host must have all the resources required by your job control stream.
- The job stream file may be anywhere in your DDP network. It does not have to be at the destination host or at your local host. Specify its location in the FILE= parameter. The DDP SUBMIT FILE command sends the file of job control streams to the receiving host.
- You may omit the host-id specification if you create a job stream to catalog all remote files on your local system. For an example, see the *Job Control Programming Guide,* 7004 4623.
- When using DDP in a multihost environment where the host computers are using different major release levels (e.g., 8.2, 10.0), a saved job stream ($Y$AVE) cannot be sent using the DDP SUBMIT FILE command to a system operating at a different release level. Job control errors occur on the system when an attempt is made to schedule the job.

  It is possible to run such a job by saving the job on the system where it is to run and using the DDP SUBMIT FILE command to start the job.

Format

```
DDPΔSUBMITΔFILE=[ {originating-host-id}:: ]file-id
                [ {local-host-id      }   ]

      [ΔHOST={destination-host-id} ]
      [      {local-host-id      } ]

      [ΔPRINT=host-id::device-id]
```

Parameters

`originating-host-id`
: Is one to four alphanumeric characters naming the job control stream file location. If omitted, the system assumes the file is on your local system. For more information, see 3.1.2.

`file-id`
: Is 1 to 74 alphanumeric characters identifying the input file. For more information, see 4.2.1.

  The file-id in the DDP SUBMIT FILE command may be either a sequential data file, which has no module name, or a module in a library file, in which case the module name must be present. (Normally you won't be submitting your entire library for execution.) In either case, the sequential file or the module may contain more than one job name. The destination host runs all jobs submitted in this file or module.

```
HOST=⎧destination-host-id⎫
     ⎩local-host-id      ⎭
```
Is one to four alphanumeric characters naming the host you submit the file to. If omitted, the command assumes the file is to be submitted to your local system. For more information, see 3.1.2.

`PRINT=host-id::device-id`
: Specifies where job output is to be printed. You can direct printed output to the initiating host or any other host in the network.

  Output can be printed at the central printer or an auxiliary printer attached to a terminal. For example:

```
PRINT=H001::PRNTR        (Prints output at central printer)
PRINT=H001::user-id      (Prints output at auxiliary printer)
```

If you omit the PRINT keyword, output is automatically returned to the initiating host (unless overridden by job control statements).

The PRINT parameter generates an OPTION OUT statement at the end of your job control stream, overriding any other OPTION OUT or OPTION LOG statements in the job stream. A ROUTE job control stream in the device assignment set for a file overrides the PRINT parameter and any OPTION OUT statements for that file.

Example

You want to submit a job to remote host H001. You've already stored the job at your local host in module PAY06 of library file PAYJOBS. The command is:

```
                        module        read       destination
                         name       password       host-id

DDP SUBMIT FILE='PAY06,PAYJOBS(JMS17/),,S' HOST=H001

                            file name       element
                                              type
```

An alternate form of this command is:

```
DDP SUMBIT FILE='PAY06,PAY JOBS(JMS17/)' HOST=H001
```

Response Messages

As explained in 4.2.5, when DDP accepts your DDP SUBMIT FILE command, it returns a job name to you as part of the message:

```
DDP044 JNR 044 8-character-jobname JOB SUBMITTED FOR WO=work-order-number time
```

For example:

```
DDP044 JNR 044 AAAA0001 JOB SUBMITTED FOR WO=000012 13:47:56
```

Write down the job name and its work order number. (You may want to use the log sheet in Appendix F to do this.) You need your job name to cancel the job or to inquire about its status.

4.4.2. Canceling a Job or Command (DDP CANCEL)

Function and Requirements

The DDP CANCEL command terminates an executing or backlogged job or a command executing on a host. Follow these instructions to ensure proper cancellation of your job or command:

- You must specify the host on which you are running the job or command unless it is your own local host.

- You may cancel only jobs or commands you have submitted under your user-id, unless you have delegated console privileges, which allow you to:

  - Cancel any user's job or command on the local host

  - Cancel any user's job or command on a remote host where you have privileges

Format

```
              ┌                                                            ┐
DDPΔCANCELΔ   │JOB=┌ ┌host-id      ┐::┐ jobname ┌ΔOUTPUT=┌DISCARD┐  ┐  │
             {│    │{│local-host-id│} │          │        {│DELIVER│} │  │}
              │    └ └             ┘  ┘          └        └       ┘  ┘  │
              │COMMAND=work-order-number                              │
              └                                                            ┘
```

Parameters

host-id
: Is one to four alphanumeric characters naming the host that the job is executing on. If the host-id is omitted, the system assumes the job is executing on your local system. For more information, see 3.1.2.

jobname
: Is eight alphanumeric characters naming the job that the system returns on the successful completion of a DDP SUBMIT command. (See 4.2.5 for more information on the jobname parameter.)

```
OUTPUT=┌DISCARD┐
      {│DELIVER│}
```
: Specifies what you want done with the spooled output from the job you cancel. DISCARD means the system erases the spooled, completed output files. DISCARD does not apply to job output files in process. DELIVER means that the spooled output files go to the host that would have received the completed output.

COMMAND=work-order-number
: Is the one- to six-character alphanumeric work order number that was displayed when the command was accepted.

Example

You want to cancel job AAAA0001 on host H001 because there's an error in the program. If there's any output, you want it sent to you, since that might help you diagnose the problem. The command is:

```
                destination
                  host-id
                     │
                    ┌┴┐
DDP CANCEL JOB=H001::AAAA0001 OUTPUT=DELIVER
                     └──┬───┘
                        │
                job name returned
                     by DDP
```

4.4.3. Submitting a Statement for Execution (DDP SUBMIT REQUEST)

Function and Requirements

The DDP SUBMIT REQUEST command lets you send a statement to a remote host. The statement is an instruction for the remote host to perform some task, such as an operator command to run a utility program. Like anything else you submit to a remote host, it must conform to the configuration of the remote host.

When you use DDP SUBMIT REQUEST, DDP doesn't check the contents of the statement at all. You are responsible for sending a correct, executable statement to the remote host. Also, DDP can't tell you whether your statement was executed at the remote host. If, for instance, you send a request to prep a disk, DDP can tell you that your statement was delivered successfully; but it can't tell you if the disk was actually prepped. However, you do receive any messages generated by the command submitted.

Format

```
                               ┌                        ┐
DDPΔSUBMITΔREQUEST='statement' │ΔHOST=┌host-id       ┐ │
                               │      └local-host-id ┘ │
                               └                        ┘
```

Parameters

`statement`
: Is the system command or message being submitted. The statement must be enclosed in apostrophes (') if it contains any of the following: comma (,), space (Δ), semicolon (;), exclamation mark (!), apostrophe ('), quote ("), pound sign (#), equal sign (=), or ampersand (&). In addition, you must double any apostrophes within statements enclosed in apostrophes.

  Most statements you send will be character strings.

`host-id`
: Is one to four alphanumeric characters naming the destination host. If you omit this parameter, DDP submits the request to your local host. For more information, see 3.1.2.

Example

The purpose of the DDP SUBMIT REQUEST command is to save you time when you want to do routine tasks on the remote host. Let's say, for instance, that you're just starting out in your connection with a remote host, whose host-id is N852. You're going to be sending a large number of files to that host, so you need your own disk. The remote host has an 8430 disk that they've already prepped, using the volume serial number VSN999. Since this will be your disk, you want to put your own serial number on it, 186309.

One way to change the serial number would be to establish a file on your local host, put the job control stream to change the number into the file, then enter the DDP SUBMIT FILE command. But that's three steps. A much easier way would be to use the DDP SUBMIT REQUEST command with the canned job control stream that Unisys supplies to change volume numbers. All you'd have to do is enter:

```
                                                      new
                       name of the canned           serial          destination
                       job control stream           number            host-id
                                |                      |                  |
DDP SUBMIT REQUEST='RV CGV,,O=VSN999,N=186309,T=30' HOST=N852
                                  |                   |
                              old serial           type of
                                number            disk (8430)
```

The number is changed for you.

Notes:

1. *You can get more information about using the RV CGV command to change a volume serial number by consulting the* System Service Programs (SSP) Operating Guide, *UP-8841.*

2. *You cannot specify the following commands with the DDP SUBMIT REQUEST command: DISPLAY, DELETE, BREAKPOINT, FILE, IN, SU, TU, and PD. The results will be unpredictable if any of these commands are sent.*

Manual Release of Held Spool Files (DDP SUBMIT REQUEST,'BEGIN SPL')

Sometimes it's necessary to manually initiate the release of held spool files so they can be processed and displayed at a designated remote host. For example, you may have run your job with the SPOOL HOLD option or the system spool file may have been held. In order for DDP to find your files and free them for processing, you can ask the system console operator of the host where the files are held to release them for you; or you can release them yourself by entering the following command:

```
DDP SUBMIT REQUEST='BEGIN SPL' HOST=xxxx
```

Then, to initiate processing and have your files displayed at the destination host, you or the system console operator of the host where the files are held must enter the following:

```
DDRET job-name,n[,HOST=xxxx]{,PR,PU,LOG}
```

where:

`job-name`
: Is the name of the job whose output is to be processed.

`n`
: Is the job number, if known. If not, use 0.

`xxxx`
: Is the host-id of the host where the files are to be displayed. If specified, this host-id will override the current host-id specification for where the files are to be displayed.

`PR,PU,LOG`
: PR is for print files, PU for punch files, and LOG for the job log. At least one of these parameters must be present or no processing will occur. All file types known to be present should be specified since unprocessed files remain in the spool file and can be deleted only by a cold restart.

Note: *If your spool files are not held but haven't reached their destination because the connection between the source and destination hosts was not available during processing, you can enter the DDRET command without the BEGIN SPL command.*

4.5. Information Commands

There are two DDP commands that send information back and forth between hosts without involving jobs or files. They are:

- DDP STATUS, which tells you about a remote host, file, or user, or about a command or job you've sent
- DDP TALK, which allows you to send a message to a remote user or operator

4.5.1. Finding Out the Status of a DDP Command, Host, User, File, or Job (DDP STATUS)

Function and Requirements

You can find out the status of a command you entered, a host in your system, a user, a file, or a job, by using the DDP STATUS command. You might want to use this command, for instance, to see if a file your job needs is currently available or to see if a job you've submitted to a remote host via the DDP SUBMIT REQUEST command has executed successfully.

Note: *If you have delegated console privileges, you can find out the status of other users' commands and jobs on your local host and on remote hosts where you have privileges. You can also obtain information from users' current activity logs on local and remote hosts.*

This command provides statistics only for those commands that you entered for that host. Do not use this command as the object of another DDP STATUS command. For systems with main storage capacity of 512K bytes, DDP should be used in a dedicated system environment and commands should only be issued sequentially (a command must be processed before another can be issued).

Format

```
DDPΔSTATUSΔ⎡COMMAND=work-order-number                   ⎤
            ⎢HOST=host-id                                ⎥
            ⎢     ⎡⎧host-id      ⎫::⎤ jobname            ⎥
            ⎢JOB= ⎣⎩local-host-id⎭  ⎦                    ⎥
            ⎨      ⎡⎧host-id      ⎫::⎤ file-id            ⎬
            ⎢FILE= ⎣⎩local-host-id⎭  ⎦                    ⎥
            ⎢      ⎡⎧host-id      ⎫::⎤  user-id           ⎥
            ⎣USER= ⎣⎩local-host-id⎭  ⎦                    ⎦
```

Parameters

`COMMAND=work-order-number`
: Displays the status of a command you entered. If you have delegated console privileges, you can obtain the status of any user's DDP commands on your local host and remote hosts where you have privileges. For the types of information provided, see the "DDP STATUS Command Information Summary" later in this subsection.

  `work-order-number`
  : Is the one- to six-character alphanumeric work order number that is displayed to you when your command is accepted. Note and use this number when inquiring about a command you have issued. For more information, see 4.2.5.

```
HOST=host-id
```

Provides main storage usage statistics, the types and number of tasks and jobs currently active, and remote DDP session activity statistics. For the types of information provided, see the "DDP STATUS Command Information Summary" later in this subsection.

`host-id`
: Is one to four alphanumeric characters naming the host whose status you want. If you do not specify host-id, the local host is assumed. For more information, see 3.1.2.

```
JOB= ┌ ┌host-id       ┐ :: ┐ jobname
     │ └local-host-id ┘    │
     └                     ┘
```

Tells you the status of a job, such as COMPLETED, BEING PROCESSED, or IN SCHEDULER. If you have delegated console privileges, you can obtain the status of any job at your local host and remote hosts where you have privileges.

`host-id`
: Is one to four alphanumeric characters naming the host where the job was sent. DDP assumes the job is on the local host if the host-id is not specified. For more information, see 3.1.2.

`jobname`
: Is one to eight alphanumeric characters naming the job that the DDP SUBMIT FILE command returned to you. For more information, see 4.2.5.

```
FILE= ┌ ┌host-id       ┐ :: ┐ file-id
      │ └local-host-id ┘    │
      └                     ┘
```

Provides information on your data and library files. For the types of information provided, see the "DDP STATUS Command Information Summary" later in this subsection.

`host-id`
: Is one to four alphanumeric characters naming the host where the file resides. If you omit this name, DDP assumes the file is on your local host. For more information, see 3.1.2.

`file-id`
: Is 1 to 74 alphanumeric characters identifying your file. This file-id can be a library file or data file. For more information, see 4.2.1.

```
USER= [ {host-id      } :: ] user-id
        {local-host-id}
```

Lists a table of DDP functions for a particular user.

If you have delegated console privileges, you can obtain the status of any user at your local host and remote hosts where you have privileges. If you're working from a terminal or workstation, you may request only your own status. (That is, you may enter this command only with your own user-id.)

Each DDP user is allocated a buffer that contains a log of current activity. It can hold information for up to 15 DDP commands. This command displays the last 15 commands entered by the user.

`host-id`
: Is one to four alphanumeric characters naming the host where the user is located. If omitted, DDP assumes the user is on your local host. For more information, see 3.1.2.

`user-id`
: Is the one to six alphanumeric characters identifying the user. This is the same user-id a user would use in the LOGON command.

DDP STATUS Command Information Summary

The DDP STATUS command provides the following information at interactive services shutdown time for the parameters listed.

COMMAND= Parameter

This parameter displays:

- Originating (primary) host
- Destination (secondary) host
- Time the command was accepted
- Time the command was completed
- Status information, such as:

  BEING PROCESSED
  IN SCHEDULER
  COMPLETED

Example: Response to DDP STATUS COMMAND

```
W.O. # FUNCTION PRIM SECN STARTED COMPLETE STATUS
```

where:

`W.O. #`
: Is the local or remote work order number of the command whose status you're requesting. For additional information about work order numbers, see 4.2.5.

`FUNCTION`
: Is the name of the command whose status you're requesting, such as DDP COPY or DDP PURGE.

`PRIM`
: Is the host-id of the primary host involved with a command. If the command involves an originating file, the primary host is the host that has that file. Otherwise, the primary host is the host originating this command.

`SECN`
: Is the host-id of the secondary host involved with a command. If the command involves an originating file, the secondary host is the destination host of that file. Otherwise, the secondary host is the destination host for this command.

`STARTED`
: Is the time a command was sent.

`COMPLETE`
: Is the time a command was completed, if it was.

`STATUS`
: Specifies status information.

Note: *Do not use the COMMAND= parameter in an enter stream. Enter streams are discussed with batch processing in Appendix E.*

Log Printout Sample

```
DDP STATUS COMMAND=5
0B DDP002 CA1 002 STATUS   COMMAND ACCEPTED  WO=000024 12:23:35
0C W.C.#  FUNCTION PRIM SECN STARTED COMPLETE      STATUS
0D 000005 COPY     NOD4 NOD2 12:09:14 12:09:32  BEING PROCESSED
0E DDP022 FTR 022 STATUS   COMMAND COMPLETED WO=000024 12:23:42
BR CN
```

HOST= Parameter

This parameter displays:

- The current size of installed main storage, the amount of free main storage, and the size of the largest free region
- The number of current interactive tasks, enter tasks, background tasks, and the number of active batch jobs
- The device-id for every interactive user currently logged onto the system
- The names of all currently active batch jobs
- The name of each remote host for which status is being returned and the number of input and output sessions with that host

The following information is returned in response to a DDP STATUS HOST=host command:

```
SYS-SIZE=______       FREE=______     LARGEST=______
INTERACTIVE=______  ENTER=______    BACKGROUND=______BATCH-JOBS=______
CURRENT INTERACTIVE USERS:_____________________________
CURRENT ACTIVE BATCH JOBS:_____________________________
REMOTE DDP SESSION ACTIVITY_____________________________
```

where:

SYS-SIZE=
: Is the current size of installed main storage.

FREE=
: Is the amount of available main storage.

LARGEST=
: Is the largest available region of main storage.

INTERACTIVE=
: Is the current number of interactive tasks.

ENTER=
: Is the current number of enter tasks.

BACKGROUND=
: Is the number of active background tasks.

BATCH-JOBS=
: Is the number of active batch jobs.

CURRENT INTERACTIVE USERS:

Displays the device-id and user-id for every interactive user currently logged onto the system in the format:

```
dddd-uuuuuu  dddd-uuuuuu  dddd-uuuuuu
```

where dddd is the device-id, and uuuuuu is the user-id.

If no interactive users are logged on, the following is displayed:

```
NO INTERACTIVE USERS CURRENTLY LOGGED-ON
```

CURRENT ACTIVE BATCH JOBS:

Displays the names of all currently active batch jobs in the format:

```
jjjjjjjj    jjjjjjjj    jjjjjjjj
```

where jjjjjjj is the job name.

If no batch jobs are currently active, the following is displayed:

```
NO BATCH JOBS CURRENTLY ACTIVE
```

For the host specified, also displayed are:

- Name of each remote host connected
- Number of input sessions
- Number of output sessions displayed in the format:

```
REMOTE DDP SESSION ACTIVITY:
REMOTE HOST=___  INPUT SESSIONS=___  OUTPUT SESSIONS=___
```

If there is no DDP session activity, the following is displayed:

```
NO DDP SESSIONS ACTIVE AT THIS TIME
```

Log Printout Sample

```
HO=NOD1
DDP002 CA1 002 STATUS   COMMAND ACCEPTED  WO=000016 09:43:43
SYS-SIZE=  2,097,152  FREE=  1,151,488  LARGEST=  1,138,944
INTERACTIVE= 000 ENTER= 000 BACKGROUND= 002 BATCH-JOBS= 003
CURRENT INTERACTIVE USERS:
0019-CRY    001E-TOMS
CURRENT ACTIVE BATCH-JOBS:
1002-B   R-CRY1   1002-A   GUST
REMOTE DDP SESSION ACTIVITY:
REMOTE HOST= NOD2  INPUT SESSIONS= 001  OUTPUT SESSIONS= 000
DDP022 FTR 022 STATUS   COMMAND COMPLETED WO=000016 09:43:57
```

JOB= Parameter

This parameter displays:

- Command work order number
- Jobname
- Job state: whether active, backlogged, rolled out, or terminated normally or abnormally
- Seconds of CPU time used by the job
- Amount of main storage used by the job
- Number of pages, cards of output generated

Example: Response to DDP STATUS JOB

```
JOB=________PRI=__STEP=___PROG=________SIZE=________

CPU TIME=__:__:__:__  PAGES=______CARDS=______

CURRENT JOB CONDITION=__________ __________ __________
```

where:

`JOB=`
: Is the name of the job you're asking the status of.

`PRI=`
: Is the current priority of the job.

`STEP=`
: Is the current job step number.

`PROG=`
: Is the program currently being executed.

`SIZE=`
: Is the main storage being used by the job.

`CPU TIME=`
: Is the amount of CPU time used by the job.

`PAGES=`
: Is the number of pages of spooled output produced by the job.

CARDS=
Is the number of punched cards produced by the job.

CURRENT JOB CONDITION=
Is the state of the job at this moment.

Note: *Do not use the JOB= parameter in an enter stream. Enter streams are discussed with batch processing in Appendix E.*

Log Printout Sample

```
DDP STATUS JOB=NOD20003
0B DDP002 CA1 002 STATUS   COMMAND ACCEPTED  WO=000022 12:20:32
0C JOB=NOD20003  PRI=05  STEP=001  PROG=SMPLMU00  SIZE=0035840
0D CPU TIME= 00:00:12.965   PAGES= 000000   CARDS= 000000
0E CURRENT JOB CONDITION = WAITING FOR I/O
0F DDP022 FTR 022 STATUS   COMMAND COMPLETED WO=000022 12:20:44
```

FILE= Parameter

This parameter displays:

- Filename
- File type
- Number of extents
- Number of cylinders
- Number of tracks
- Block size
- Creation date
- Expiration date

The file-id parameter and security read and write keys must be expressible as part of the file-id for each system.

The volume name must also be expressible as part of the file-id for files registered in the volume table of contents (VTOC).

If the file-id contains any of the following characters embedded, then it must be expressed as a quoted string by enclosing it in apostrophes:

ampersand (&)	equals (=)	semicolon (;)
apostrophe (')	exclamation mark (!)	space ( )
comma (,)	number (#)	

Each single occurrence of an apostrophe within the file-id must be replaced by two apostrophes, for example:

JOE'S FILE would be expressed as:

```
'JOE''S FILE'
```

Example: Response to DDP STATUS FILE

```
FILE=__________________________ EXT=___

VSN=______CYL=____TRK=__TYPE=_____BKSZ=_____

RCSZ=_____CRE_DATE=__/__/__EXP_DATE=__/__/__
```

where:

FILE=
: Is the name of the file you're requesting status of.

EXT=
: Is the number of extents occupied by the file.

VSN=
: Is the VSN from the file name string, if specified.

CYL=
: Is the number of cylinders occupied by this file.

TRK=
: Is the number of tracks (beyond whole cylinders), occupied by this file.

TYPE=
: Is the type of file: indexed, sequential, relative, library, or undefined.

BKSZ=
: Is the block size of the file.

RCSZ=
: Is the record size of the file.

CRE-DATE=
: Is the date the file was created.

EXP-DATE=
: Is the expiration date of the file.

Note: *BKSZ, RCSZ, CRE-DATE, and EXP-DATE parameters are available only for files that have been opened at least once for processing. A DDP CREATE command does not open the file, and requesting the status of a newly created file will not return all of the fields described previously.*

Log Printout Sample

```
DDP STATUS FILE=',$Y$SCLOD,RES'
0G DDP002 CA1 002 STATUS   COMMAND ACCEPTED  WO=000025 12:24:32
0J FILE= $Y$SCLOD                                      EXT= 002
0K VSN= RES      CYL= 0015  TRK= 00  TYPE= SAT    BKSZ= 00256
0L RCSZ= 00256  CRE-DATE= 90/05/14  EXP-DATE= 99/12/31
DM DDP002 FTR 022 STATUS   COMMAND COMPLETED WO=000025 12:24:42
```

USER= Parameter

This parameter displays:

- The DDP functions performed for a particular user
- Uncompleted and completed commands
- Status of any user in your system (if issued from a system console)
- Your own status only (if issued from a workstation or terminal)

Example: Response to DDP STATUS USER

```
USERID   BUFFERS   BUF SIZE   W. O. COUNT
------   -------   --------   -----------

W.O. #   FUNCTION   PRIM   SECN   STARTED   COMPLETE   STATUS
------   --------   ----   ----   -------   --------   ------
```

where:

USERID
: Is the user-id of the user whose status you're requesting.

BUFFERS
: Is the number of buffers the user is using.

BUF SIZE
: Is the total buffer space the user is using.

W. O. COUNT
: Is the number of active work orders being processed.

When you issue a COPY or SUBMIT FILE command at a host not having the source file, the work order count is not incremented. While the job is being processed at the remote host, the status of the job is displayed as: IN PROGRESS. When the job is completed, the status display changes to:

```
TERM NORMALLY
```

or

```
TERM ABNORMALLY
```

W. O. #
: Is the work order number of the command whose status you're requesting. For additional information about work order numbers, see 4.2.5.

FUNCTION
: Is the name of the command whose status you're requesting, such as DDP, COPY, or PAUSE.

PRIM
: Is the host-id of the primary host involved with a command. If the command involves an originating file, the primary host is the host that has that file. Otherwise, the primary host is the host originating this command.

SECN
: Is the host-id of the secondary host involved with a command. If the command involves an originating file, the secondary host is the destination host of that file. Otherwise, the secondary host is the destination host for this command.

STARTED
: Is the time a command was sent.

COMPLETE
: Is the time a command was completed, if it was.

STATUS
: Specifies status information.

Log Printout Sample

```
DDP STATUS USER=OPERATOR
0Q DDP002 CA1 002 STATUS   COMMAND ACCEPTED  WO=000026 12:25:10
0A  USERID   BUFFERS   BUF SIZ   W.O. COUNT
0B  $Y$CON    00004    0018352    000000
0C W.C.#  FUNCTION PRIM SECN STARTED  COMPLETE     STATUS
0D 000001 PURGE    NOD2 NOD2 12:05:28 12:05:47   TERM NORMALLY
0E 000002 PURGE    NOD2 NOD2 12:05:28 12:05:44   TERM NORMALLY
0F 000003 CREATE   NOD2 NOD2 12:06:16 12:06:18   TERM NORMALLY
0G 000004 COPY     NOD4 NOD2 12:07:41 12:07:47  TERM ABNORMALLY
0J 000005 COPY     NOD4 NOD2 12:09:14 12:09:32  BEING PROCESSED
0K 000009 COPY     NOD4 NOD2 12:13:36 12:13:56  BEING PROCESSED
0L 000013 SUBMIT   NOD2 NOD4 12:15:34 12:16:05   TERM NORMALLY
0M 000015 TALKX    NOD4 NOD2 12:16:28 12:16:47   TERM NORMALLY
0P 000017 TALKX    NOD4 NOD2 12:17:04 12:17:07   TERM NORMALLY
0Q 000018 SPOOL    NOD2 NOD4 12:17:05    :  :      IN PROGRESS
0A 000021 SUBMIT   NOD2 NOD2 12:19:15 12:19:37   TERM NORMALLY
0B 000022 STATUS   NOD2 NOD2 12:20:34 12:20:44   TERM NORMALLY
0C 000023 STATUS   NOD2 NOD2 12:21:38 12:21:46   TERM NORMALLY
0D 000024 STATUS   NOD2 NOD2 12:23:37 12:23:42   TERM NORMALLY
0E 000025 STATUS   NOD2 NOD2 12:24:33 12:24:42   TERM NORMALLY
0F 000026 PURGE    NOD2 NOD2 12:25:11    :  :      IN PROGRESS
0G DDP022 FTR 022 STATUS   COMMAND COMPLETED WO=000026 12:25:22
BR CN
```

4.5.2. Communicating with a Remote Operator or User (DDP TALK)

Function and Requirements

The DDP TALK command lets you send a message to an operator or user at either the local site or a remote host. You might want to have the operator mount your disk pack, for instance. Or, you might need to ask a user if he has recently updated a file you need.

If the recipient of the DDP TALK message isn't logged on, the initiator is notified that the user is not logged on.

Format

```
                                 ┌                    ┐
DDPΔTALKΔMESSAGE='string'ΔUSER=  │ {host-id       }:: │ {OPERATOR} Δ[WAIT]
                                 │ {local-host-id }   │ {user-id }
                                 └                    ┘
```

Parameters

`'string'`
: Is the character string message you want the operator or user to get.

`host-id`
: Is one to four alphanumeric characters naming the host where the operator or user is. If you omit the host-id, DDP assumes the user is on your local host. For more information, see 3.1.2.

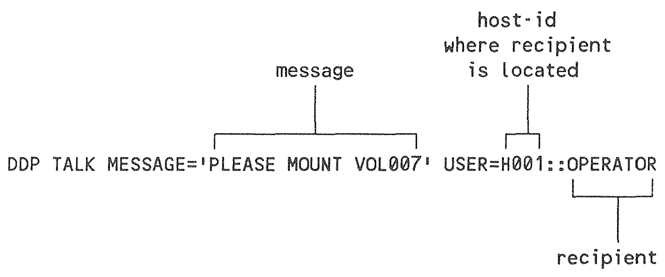
`OPERATOR`
: Is the operator of the remote host.

`user-id`
: Is one to six alphanumeric characters identifying the user to the system. This is the same id as used in the LOGON command.

`WAIT`
: Informs the operator or user that you want a reply. You do not have to wait for this reply to go on with other commands. Your keyboard isn't locked during this period.

Example

You want the operator on host H001 to mount your volume (VOL007). The command is:



```
DDP TALK MESSAGE='PLEASE MOUNT VOL007' USER=H001::OPERATOR
```

4.6. Help Screens for DDP Commands

Help screens are available for all DDP commands. To obtain information about available DDP commands and further help:

1. Type: HELP DDP

   The following information is then displayed:

DDP Commands	Help Screen Requests
DDP CANCEL	HELP DDPCAN
DDP COPY	HELP DDPCOP
DDP CREATE	HELP DDPCRE
DDP PURGE	HELP DDPPUR
DDP STATUS	HELP DDPSTA
DDP SUBMIT FILE	HELP DDPFSB
DDP SUBMIT REQUEST	HELP DDPRSB
DDP TALK	HELP DDPTAL
Parameters for CREATE	HELP DDPPAR

2. Enter the appropriate help screen request format shown in step 1.

3. Enter HELP DDPPAR if you need help with the parameters for the DDP CREATE COMMAND.

4.7. Automatic Recovery from Failure

Automatic recovery is a feature available to the console operator at system initialization time. If selected, DDP automatically recovers all work orders in progress at the time of a system crash or communications line failure. Following either type of failure, the work orders are held until the destination host is available. When it is available, the recovered work orders are reinitiated automatically.

Section 5
The OS/3 to UNIX O/S DDP File Transfer Facility

5.1. What the OS/3 to UNIX O/S DDP File Transfer Facility Does for You

The OS/3 to UNIX O/S DDP file transfer facility is a separate Unisys DDP product that lets you transfer files from a UNIX system to OS/3 or from OS/3 to a UNIX system. You can:

- Enter commands at an OS/3 workstation or terminal to:
    - Create a file on a UNIX O/S host
    - Copy a file from an OS/3 host to a UNIX O/S host
    - Copy a file from a UNIX O/S host to the same host or to another host in the DDP network
    - Remove a file from a UNIX O/S host
- Enter UNIX O/S commands at your terminal to:
    - Create a file on an OS/3 host
    - Copy a file from a UNIX O/S host to an OS/3 host
    - Copy a file from an OS/3 host to the same OS/3 host or to another host in the DDP network
    - Remove a file or element from an OS/3 host

5.2. Using the OS/3 to UNIX O/S DDP File Transfer Facility

The OS/3 to UNIX O/S DDP file transfer facility pairs DDP with Information Services (IS) on the UNIX O/S to let you send files between OS/3 and UNIX O/S systems. You can use DDP commands from OS/3 to create, copy, or purge files on a UNIX system. Or, you can use IS commands from UNIX O/S to create, copy, or purge files on an OS/3 system. You cannot use remote job commands or information commands between OS/3 and UNIX systems.

There are three OS/3 to UNIX O/S DDP file transfer commands: CREATE, COPY, and PURGE. When using these commands, you must consider the differences between DDP and IS on UNIX. Table 5-1 summarizes the differences in the CREATE, COPY, and PURGE command parameters and parameter options supported by OS/3 and UNIX O/S. Shading indicates the default.

Table 5-1. OS/3 to UNIX O/S DDP File Transfer Parameter Summary

Command	Notes	Parameter	Options - OS/3	Options - UNIX O/S
CREATE	1	FILENAME	Required	Required
		ACCESS	Ignored	Ignored
		BLOCK SIZE	Optional	
		DENSITY	Ignored	
		DEVICE CLASS	DISK DISKETTE TAPE	
		DEVICE TYPE	Ignored	
		FILE TYPE	INDEXED LIBRARY SEQUENTIAL UNDEFINED	
		INCREMENT SIZE	Ignored	
		INITIAL SIZE	Optional	
		MAXIMUM SIZE	Ignored	Ignored
		PARITY		
		RECORD FORM		
		RECORD SIZE		
	2	REGISTER	CATALOGUE VTOC	
		VOLUME	Ignored	

continued

Table 5-1. OS/3 to UNIX O/S DDP File Transfer Parameter Summary (cont.)

Command	Notes	Parameter	Options - OS/3	Options - UNIX O/S
COPY	3	FROM FILENAME	Required	Required
	4	TO FILENAME	Required	Required
		ELEMENT TYPE	ABSOLUTE ALL OMNIBUS RELOCATABLE SYMBOLIC	Ignored
		KEY_n	CHANGE NO CHANGE DUPLICATES NO DUPLICATES	
		MODE	DIRECT INDIRECT WAIT	
	5, 6	POSITION	EOF SOF	EOF SOF
	7	RECORD FORMAT	FIXED TRANSPARENT VARIABLE	FIXED TRANSPARENT (Remote host default) VARIABLE
PURGE	8, 9	FILENAME	Required	Required

Notes:

1. The only parameter in the CREATE command that is recognized by IS on UNIX is FILENAME.

2. When you specify keyword parameter REGISTER=CATALOG (the default) on the CREATE command from UNIX O/S, the job output and messages are displayed at the OS/3 system console, but are not returned to the UNIX O/S terminal.

3. Only source and proc type library modules and character-oriented data files can be copied between an OS/3 and a UNIX O/S host.

4. On an OS/3 host, a file must exist before you can copy to that file. If the file does not already exist, you must send the CREATE command to the OS/3 host before you can copy to that file.

5. The default for the POSITION parameter on OS/3 is EOF (end of file). On UNIX O/S, it is SOF (start of file). To overwrite a file, specify SOF.

6. When you specify parameter POSITION=SOF on a copy command to UNIX O/S, you do not have to use the CREATE command before you can copy to that file.

continued

Table 5-1. OS/3 to UNIX O/S DDP File Transfer Parameter Summary (cont.)

7. When OS/3 receives a zero-length record on a DDP COPY command, it writes a 1-byte space record to the file. Consequently, a file containing a zero-length record cannot be compared (source to destination) on a byte-for-byte basis. If the file containing the zero-length record is a job control stream, the job stream cannot be executed.

8. When you purge an element from a file on an OS/3 host, you must specify both the name and type of the module to be purged. If you do not specify the name and type, the entire file is purged and the file decataloged.

9. When you purge a file on an OS/3 host from UNIX O/S, the job output and associated job messages are displayed at the OS/3 system console but are not returned to the UNIX O/S terminal.

5.3. OS/3 and UNIX O/S File Names and Directories

OS/3 does not distinguish between uppercase and lowercase file names. All are read as uppercase, regardless of how you enter them. UNIX O/S does distinguish between uppercase and lowercase file names. On UNIX O/S, you can create two files with the same name, one uppercase and one lowercase.

When you send a file from OS/3 to UNIX O/S with a name that is not fully qualified by the path name, you create a file in the UNIX O/S default directory. Refer to your UNIX Information Services manual for the name of the default directory.

For example, when you enter the DDP command:

```
DDP CREATE FILE=[unix-host-id]::'OS3.1'
```

the file appears on the UNIX O/S as:

```
/default-directory-name/OS3.1
```

You can copy to or from this file, or you can purge it by specifying the file name OS3.1. You cannot reference this file by fully qualifying the path because OS/3 sends only uppercase names.

Section 6
The DDP File Access Facility

6.1. Introduction

This section describes how to:

- Access and process files residing on remote OS/3 systems
- Write application programs on OS/3 systems that can initiate and communicate with other OS/3 systems to exchange data and control information

Both of these capabilities are provided as part of the DDP file access facility.

6.2. Remote File Processing

Your programs can access and process remote MIRAM disk data (not library) files by adding a HOST=host-id parameter on the // DVC job control statement associated with a remotely located disk file. For example, if the disk file declared in the following control stream:

```
// JOB MYJOB
    .
    .
    .
// DVC 50
// VOL D00028
// LBL INVFILE
// LFD INPUTA
// EXEC PROGA
```

was located at a remote site, all that would be required to indicate that the file is at a remote site is the addition of the host-id parameter, as shown in the following:

```
// JOB
    .
    .
    .
// DVC 50,HOST=REM2
// VOL D00028
// LBL INVFILE
// LFD INPUTA
// EXEC PROGA
/&
```

No program changes or additions are needed to process a remote file.

6.2.1. Cataloging Remote Files

Remotely accessed files can be cataloged as described in the *File Cataloging Technical Overview*, 7004 4615. Further, they can be cataloged at multiple hosts. The host where the file resides would catalog it like any other file (excluding the HOST= parameter), whereas other hosts would include the HOST= parameter in their file descriptions.

6.2.2. Sharing Remote Files

Remotely accessed files can be declared as sharable read files but cannot be shared for updating purposes (read/write files).

6.2.3. Tradeoffs Involved when Processing Remote Files

Figure 6-1 shows an application where the ability to access and process data at remote hosts has distinct advantages. As shown, the inventory display program is connected with three inventory files: one local and two remote. When information is needed about a particular product, the workstation user need only enter a display request.

The inventory display program running at site A opens the applicable files and displays the requested information. The workstation user is unconcerned about where the requested information is stored.

Such an application assumes that the inventory files maintained at the remote hosts are dynamic files that are constantly being updated, processed, and maintained by the remote hosts. Consequently, it would be inefficient to make copies of these files on host A each time host A wanted to access these files.

DDP maintains an open session between hosts for each open file. Inactivity on a session for a period of 30 minutes causes the session, and then the job, to terminate. Therefore, remote files with little activity should not be opened and left open.

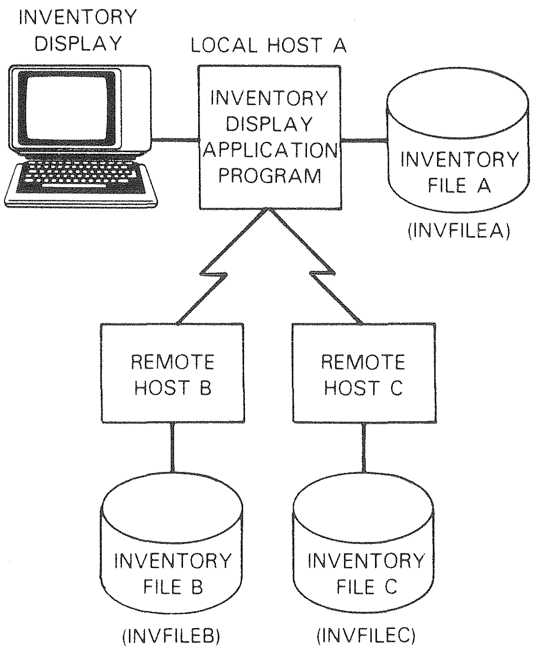


Figure 6-1. Remote File Processing Application

6.3. Program-to-Program Communication

You can write assembly language programs that can initiate and carry on a conversation with other assembly language programs at remote hosts. The communicating programs must reside on different OS/3 hosts in a DDP network.

User programs written for program-to-program communication are not scheduled if DDP shutdown processing has occurred. The job remains on queue waiting for DDP. The issuance of any DDP command loads the DDP software and automatically schedules the program.

You can translate ASCII character-oriented files to EBCDIC character-oriented files and vice versa. Or, you can transfer data in a transparent mode; that is, DDP treats the data as bit strings.

6.3.1. Types of Communication Programs

There are two basic communication programs: primary and surrogate. The program that initiates a conversation is called the primary program and the initiated program is called the surrogate program. However, primary and surrogate programs can be designed to reverse roles if the application they are being used in so dictates. A primary program can call up to 255 surrogate programs.

6.3.2. Types of Conversations

There are two types of conversations that may be conducted between communicating programs: simple and complex. By definition, a simple conversation is one that takes place between only two programs and the primary/surrogate relationship between these two programs never changes. The primary program remains the primary program throughout the life of the conversation. There is a simple exchange of information. A primary program activates a surrogate program and provides it with data to be processed. The surrogate program performs the required processing.

A complex conversation is one that allows for the communicating programs to reverse primary/surrogate status, or that allows for more than two programs to be involved in the conversation. The procedures and guidelines for preparing communicating programs follow.

6.4. Programming Considerations for Simple Conversations

6.4.1. Job Control Requirements

Programs designed for use in a simple conversation application rely on job control to identify the location and name of the surrogate program. Simple conversation applications assume that the primary program is initiated through the job control run process by an operator and that the surrogate program is initiated through the job control run process by the DDP facility. Consequently, both programs require an associated job control stream and both programs occupy job slots in the system on which they are executing.

Primary Program Device Assignment Set

The job control streams associated with communicating programs must contain a device assignment set for the program being addressed. For the primary program, this device assignment set must consist of a // DVC statement and a // LFD statement. The // DVC statement must be in the form:

```
// DVC PROG,jobname,HOST=host-id
```

where:

`PROG`
: Associates the device assignment with the program-to-program component, rather than a conventional I/O device.

`jobname`
: Identifies the name of the job you wish to communicate with.

`HOST=host-id`
: Identifies the network name of the system on which the corresponding program is located. (This name is established when the communications network for your system is defined during system installation.)

The // LFD statement must be in the format:

```
// LFD filename
```

where:

`filename`
: Is the name specified for the FILENAME= parameter on the CDIB macroinstruction issued by the primary program.

Surrogate Program Device Assignment Set

The surrogate device assignment set must also issue a // DVC statement and // LFD statement. However, the // DVC statement must have the following format:

```
// DVC PROG
```

where:

`PROG`
: Associates the device assignment with the program-to-program software, rather than a conventional I/O device.

The // LFD statement must be in the format:

```
// LFD filename
```

where:

`filename`
: Is the name specified for the FILENAME= parameter on the CDIB macroinstruction issued by the surrogate program.

Example:

Control Stream to Run Primary Program at HOST AAAA	Control Stream to Run Surrogate Program at HOST BBBB

```
// JOB PRIMJOB                      // JOB SURJOB
    .                                   .
    .                                   .
    .                                   .
// DVC PROG,SURJOB,HOST=BBBB        // DVC PROB
// LFD PRIMARY                      // LFD SURRGATE
// EXEC PROGA                       // EXEC PROGB
/&                                  /&
// FIN                              // FIN
```

6.4.2. Coding Requirements

To prepare BAL programs for use in simple conversation applications, you use a special set of consolidated data management macroinstructions to define, open and close, and transfer information between the communicating programs. These macroinstructions function the same way other data management macroinstructions do and, consequently, are designed to resemble them wherever possible. Two declarative and four imperative macroinstructions are used in this environment.

6.4.3. Declarative Macroinstructions

Declarative macroinstructions are used to supply information concerning the files required by the issuing program. These macroinstructions generate nonexecutable code, such as constants and storage areas for variables. The following two macroinstructions are used: CDIB and RIB.

Defining a Common Data Interface Block (CDIB)

The CDIB macroinstruction identifies the logical file required by the issuing program and establishes a common data interface block for the file. The CDIB is the function passing mechanism for the file; it is referenced each time you issue an imperative macroinstruction.

Format

LABEL	ΔOPERATIONΔ	OPERAND
name	CDIB	FILENAME=filename

Parameters

`name`
: Specifies the name of the CDIB. This name cannot exceed seven characters.

`filename`
: Specifies the file name that was assigned in the // LFD statement in the program execution job control stream for the issuing program.

Defining a Resource Information Block (RIB)

The RIB macroinstruction describes the file characteristics required by the issuing program. The RIB is used in combination with the CDIB macroinstruction when you issue the DOPEN imperative macroinstruction to open the file.

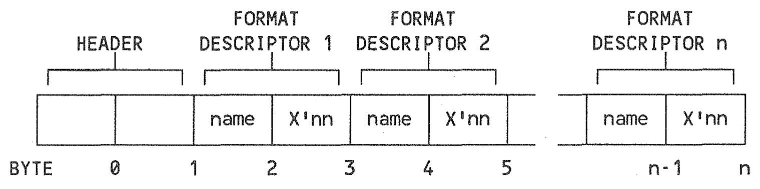
Format

LABEL	ΔOPERATIONΔ	OPERAND
name	RIB	[HOSTID=host-id] [,PROGFD=symbol] [,RCSZ=n] [,WKFM={NO | VAR | VARI}]

Parameters

`HOSTID=host-id`
: Specifies the one- to four-character alphanumeric name (first character must be alphabetic) of the host on which the destination program resides. This parameter is overridden by the HOST=host-id parameter of the // DVC PROG statement in a simple communications application environment.

`PROGFD=symbol`
: Specifies the symbolic address of a data format descriptor list that you provide to indicate the type of data that is being transferred. This list has the following format:



Notice that the data format descriptor list consists of a 2-byte header followed by one or more contiguous 2-byte data format descriptor entries. The 2-byte header contains the number of list entries in binary. The first byte of the format descriptor entry contains a name that you define for a particular type of data. The second byte of the entry contains a hexadecimal value that specifies the particular type of data. The hexadecimal values and the type of data they specify are as follows:

Hex. Value	Type of Data	Hex. Value	Type of Data
X'80'	ASCII	X'85'	Compressed Katakana
X'81'	Compressed ASCII	X'86'	Transparent Octet
X'82'	EBCDIC	X'88'	Kanji
X'83'	Compressed EBCDIC	X'89'	Compressed Kanji
X'84'	Katakana		

This list is used when you transfer data. It is used with the DMINP and DMOUT imperative macroinstructions. If the PROGFD parameter is omitted, the data type is assumed to be transparent octet.

RCSZ=

Specifies the length, in bytes, of each record to be transferred. If RCSZ is omitted and WKFM=NO or WKFM=VAR is specified, the record length is assumed to be 256 bytes.

WKFM=

Specifies the format of the work area in which input and output records are placed. If the format is variable (VAR or VARI), the work area consists of a 4-byte record descriptor word (RDW) followed by data. All record sizes are passed in the first 2 bytes of the RDW and include the 4-byte RDW.

WKFM=NO

Specifies that the work area format is fixed; it is RCSZ-bytes long. NO is the default.

WKFM=VAR

Specifies that the work area format is variable. For output operations, you must specify the amount of data in the first 2 bytes of the work area. For input operations, the work area must be large enough to hold the maximum amount of data (determined by the RCSZ keyword parameter). You do not have to specify a size in the work area; but, after receiving control back from the input operation, the first 2 bytes of the work area will contain the size of the record.

WKFM=VARI
: Specifies that the work area format is variable. For output operations, you must specify the amount of data in the first 2 bytes of the work area (same as WKFM=VAR). For input operations, you must specify, in the first 2 bytes of the work area, the maximum amount of data desired. If the specified size is greater than the actual record size, the value in the work area will be changed to reflect the actual record size.

6.4.4. Imperative Macroinstructions

The imperative macroinstructions establish and close a communications path between user application programs, begin and end conversation between programs, transfer and receive data, and control the status of the programs. The instructions you use to cause these actions are: DOPEN, DCLOSE, DMOUT, and DMINP.

Register Conventions

When you use the imperative macroinstructions, the following registers must be used as indicated:

Register 0
: The RIB address must be loaded in this register for the DOPEN macroinstruction. The work-area address must be loaded in this register for those macroinstructions that use work-area processing.

Register 1
: The CDIB must always be loaded in this register.

If symbolic notation is used, the expansion of the imperative macroinstruction loads register 0 and/or register 1 with the appropriate address. All registers are returned unchanged when control is received back from an imperative macroinstruction.

Status Checking

Status checking is required after each imperative macroinstruction. This is necessary because control is always returned inline. A standard way of checking macros is described in the *Consolidated Data Management Macroinstructions Programming Guide*, 7004 4607.

Establishing a Communications Path (DOPEN)

You use this macroinstruction to establish a communications path between user application programs. This instruction must be issued first by both the primary and surrogate application programs before any transfer of information is attempted.

Format

LABEL	ΔOPERATIONΔ	OPERAND
name	DOPEN	$\left\{\begin{array}{l}\texttt{cdibname}\\\texttt{(1)}\\\texttt{1}\end{array}\right\},\left\{\begin{array}{l}\texttt{ribname}\\\texttt{(0)}\\\texttt{0}\end{array}\right\}$

Parameters

`cdibname`
: Is the symbolic name of the CDIB required by the program issuing the DOPEN.

`(1) or 1`
: Indicates that you have preloaded register 1 with the address of the CDIB.

`ribname`
: Is the symbolic name of the RIB required by the program issuing the DOPEN.

`(0) or 0`
: Indicates that you have preloaded register 0 with the address of the RIB.

Outputting Data (DMOUT)

The DMOUT macroinstruction is used to transfer (send) data from one application program to another.

Format

LABEL	ΔOPERATIONΔ	OPERAND
name	DMOUT	$\left\{\begin{array}{l}\texttt{cdibname}\\\texttt{(1)}\\\texttt{1}\end{array}\right\},\left\{\begin{array}{l}\texttt{workarea}\\\texttt{(0)}\\\texttt{0}\end{array}\right\}$

Parameters

`cdibname`
: Is the name of the CDIB required by the program issuing the DMOUT.

`(1) or 1`
: Indicates that you have preloaded register 1 with the address of the CDIB.

`workarea`
: Is the symbolic name of the work area that contains the record to be sent.

`(0) or 0`
: Indicates that you have preloaded register 0 with the address of the work area.

Receiving Data (DMINP)

The receiving macroinstruction has the following format:

Format

LABEL	ΔOPERATIONΔ	OPERAND
name	DMINP	$\left\{\begin{array}{l}\text{cdibname}\\ (1)\\ 1\end{array}\right\},\left\{\begin{array}{l}\text{workarea}\\ (0)\\ 0\end{array}\right\}$

Parameters

cdibname
: Is the name of the CDIB required by the program issuing the DMINP.

(1) or 1
: Indicates that you have preloaded register 1 with the address of the CDIB.

workarea
: Is the symbolic name of the work area that contains the record to be sent.

(0) or 0
: Indicates that you have preloaded register 0 with the address of the work area.

If the PROGFD is specified in the RIB, you select the data format from the data format descriptor list by moving the 1-byte name you defined for the particular format into the CI$FD field of the CDIB before you issue the DMOUT macroinstruction. If you do not move a name into the CDIB, the first data format in the list will be used. On a DMINP, the data name that comes across in the input buffer will be moved into the CI$FD field.

Closing a Communications Path (DCLOSE)

You use this macroinstruction to close a communications path between user application programs.

Format

LABEL	ΔOPERATIONΔ	OPERAND
name	DCLOSE	$\left\{\begin{array}{l}\text{cdibname}\\ (1)\\ 1\end{array}\right\}$

Parameters

`cdibname`
: Is the symbolic name of the CDIB declared in the program issuing the DCLOSE.

`(1) or 1`
: Indicates that you have preloaded register 1 with the address of the CDIB.

6.4.5. Timing Considerations

We recommend that the job streams for program-to-program communications contain no other job steps than those required for the execution of these programs. The reason for this recommendation is that only 2 minutes are allotted for starting a job; otherwise, a timeout will result.

6.4.6. Designing a Simple Conversation Program

Simple conversation must contain the following three phases, which are also shown in the flow diagrams in Figure 6-2:

- *Initiation Phase* - The primary program initiates the conversation, via a DOPEN, with one and only one surrogate program, which also must issue a DOPEN.
- *Data Transfer Phase* - The primary program transfers data only to the surrogate program, after issuing DMOUTs, and the surrogate receives data after issuing DMINPs.
- *Termination Phase* - The primary program and surrogate program must each issue a DCLOSE to terminate.

The CDIBs and RIBs required by the DOPEN, DMOUT, DMINP, and DCLOSE macroinstructions are those of the program issuing the macroinstructions. Examples of simple conversation programs are given in 6.4.7.

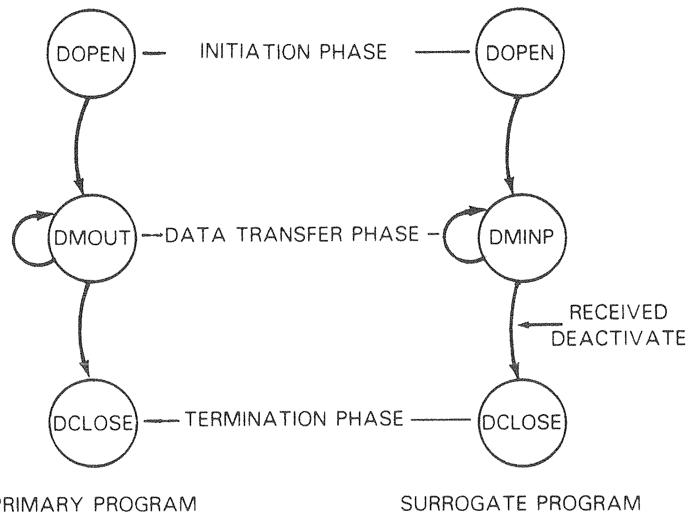


Figure 6-2. Simple Conversation Program Flow Diagrams

6.4.7. Examples of Simple Conversation Communications Programs

The following are examples of simple conversation programs.

Example 1: Primary Program and Surrogate Program Pair

Figure 6-3 shows the job control and Figure 6-4 the coding for a very simple, complete primary program and surrogate program communicating pair.

Primary Program	Surrogate Program

`// JOB PTP$PRIM`
`.`
`.`
`.`
`// DVC PROG,PTP$SURR,HOST=BBBB`
`.`
`.`
`.`
`// LFD filename field in the`
`primary CDIB`
`.`
`.`
`.`
`// EXEC load module name`
`.`
`.`
`.`
`/&`
`// FIN`
	`// JOB PT$SURR`
	
	`// DVC PROG`
	`.`
	`.`
	`.`
	`// LFD filename field in the`
	`surrogate CDIB`
	`.`
	`.`
	`.`
	`// EXEC load module name`
	`.`
	`.`
	`.`
	`/&`
	`// FIN`

Figure 6-3. Job Control for Primary Program and Surrogate Program Simple Conversation

```
Primary Program                         Surrogate Program

Line
 1  PRIMARY START  0                    SURROGATE START  0
            .                                     .
            .                                     .
            .                                     .
 2          LA     R1,PRIMCDIB                    LA     R1,SURRCDIB
            .                                     .
            .                                     .
            .                                     .
 3          LA     R0,PRIMRIB                     LA     R0,SRGATRIB
 4          DOPEN  (1),(0)                        DOPEN  (1),(0)
            .                                     .
            .                                     .
            .                                     .
 5          LA     R0,DATAMSG                     LA     R0,IOAREA
 6          DMOUT  (1),(0)                        DMINP  (1),(0)
            .                                     .
            .                                     .
            .                                     .
 7          DCLOSE (1)                            DCLOSE (1)
            .                                     .
            .                                     .
            .                                     .
 8          EOJ                                   EOJ
    PRIMCDIB DC    XL48'00'             SURRCDIB  DC     XL48'00'
    PRIMRIB  RIB   RCSZ=80              SURGATRIB RIB    RCSZ=80'
    DATAMSG  DC    CL80'00'             IOAREA    DC     XL80'00'
```

Line	Coding Description
1	Each program starts its own program.
2 and 3	The primary and surrogate programs each load their respective CDIBs and RIBs into registers R1 and R0, respectively.
4	Both the primary and surrogate programs open their own CDIBs and RIBs.
5	The primary program loads a data message into R0, and the surrogate program loads its I/O area into its R0.
6	The primary program issues a DMOUT to send the message, and the surrogate program issues a DMINP to receive the message.
7	Both programs close their respective communication paths to each other.
8	Each program ends its own program.

Figure 6-4. Coding for Primary Program and Surrogate Program Simple Conversation

Example 2: Primary Program Repeatedly Transferring Data to a Surrogate Program

Figure 6-5 shows a primary program transferring 50 messages to a surrogate program (shown in Figure 6-6) during simple conversation.

Job Control Stream

```
// JOB PRIMJOB
    .
    .
    .
// DVC PROG,SURJOB,HOST=BBBB
// LFD PRIMARY
    .
    .
    .
// EXEC PROGA
/&
// FIN
```

BAL Program for the Simple Conversation Primary Program PROGA

```
Line                                 Description

1      PROGA START  0
2            BALR   R15,0
3            USING  *,R15
4*                                   LOADING THE CDIB
5            LA     R1,PRIMARY         Load the address of the following CDIB into R1:
             .                            PRIMARY CDIB FILENAME=PRIMARY
6*           .                       OPEN A CONNECTION TO THE SURROGATE HOST BBBB
7            LA     R0,PRIMRIB         Load the address of the following RIB into R0:
                                          PRIMRIB RIB
8            DOPEN  (1),(0)            Open a session path to the surrogate program
9            .                         Check for a successful open
10*          .                       EXECUTE DATA TRANSFER LOOP
11           .                            ┌Set up counter for loop
12           LA     R0,DATAAREA           │R0= address of data area
13           DMOUT  (1),(0)          Loop {Loop is executed, sending data
14           .                            │Check for a successful send
15           .                            └Loop through 50 times
16*          .                       CLOSE THE CONNECTION TO THE SURROGATE HOST
17           DCLOSE (1)                Close the session path
18           .                         Check for a successful close
             .
             .
             EOJ
```

Figure 6-5. Job Control and Coding for Primary Program Repeatedly Transferring Data to Surrogate Program

Example 3: Surrogate Program Repeatedly Receiving Data from a Primary Program

Figure 6-6 shows a surrogate program receiving 50 messages from the primary program (shown in Figure 6-5) during simple conversation.

Job Control Stream

```
// JOB SURJOB
     .
     .
     .
// DVC PROG
     .
     .
     .
// LFD SURRGATE
     .
     .
     .
// EXEC PROGB
     .
     .
     .
/&
// FIN
```

BAL Program for the Simple Conversation Surrogate Program PROGB

```
Line                              Description

1     PROGB START  0
2           BALR   R15,0
3           USING  *,R15
4*                               LOADING THE CDIB
5           LA     R1,SURRGATE      Load the address of the following CDIB into R1:
            .                          SURRGATE CDIB FILENAME=SURRGATE
6*          .                    ATTACH TO THE PRIMARY PROGRAM
7           LA     R0,SURRIB        Load the address of the following RIB into R0:
                                       SURRIB RIB
8           DOPEN  (1),(0)          Attach to the primary session
9           .                       Check for a successful attachment
            .
10*         .                    PREPARE TO RECEIVE DATA FROM THE PRIMARY PROGRAM
11          LA     R0,INPAREA         ┌Load the address of the input area into R0
12          DMINP  (1),(0)            │Get data from primary program
13          .                    Loop ┤Receive data transfer
14          .                         └Loop to get more input
15*         .                    CHECK WHICH OF THE EXCEPTION FLAGS IS SET IN THE CDIB
16*                              TERMINATE THE ATTACHMENT TO THE PRIMARY PROGRAM
17          DCLOSE (1)              Detach from the primary program
18          .                       Check for a successful detachment
            .
            .
            EOJ
```

Figure 6-6. Job Control and Coding for Surrogate Program Repeatedly Receiving Data from Primary Program

6.5. Programming Considerations for Complex Conversations

A complex conversation enables both primary and surrogate programs to send and receive data and exchange status roles via a more controlled discipline. Additionally, a primary program can simultaneously communicate with several surrogate programs.

6.5.1. The Three Phases of Complex Conversation

A complex conversation also contains three phases:

- *Initiation Phase* - The primary program initiates a conversation with one or more surrogate programs and receives a reply from each surrogate program.

- *Data / Control Transfer Phase* - The primary and surrogate programs exchange data and/or control.

- *Termination Phase* - The primary program indicates to the surrogates that conversation is to be terminated. The surrogate programs respond to the primary and end the conversation.

6.5.2. Operational Characteristics of Complex Conversation

The following operational characteristics apply:

1. A program written as a primary program cannot be started as a surrogate program. However, once the primary program has started, it can pass to a surrogate transfer phase.

2. A program written as a surrogate program cannot be started as a primary program.

3. A job started (and running) by a primary program, whether it is running as a primary or surrogate program, cannot be attached to by another job. That is, there is no way to start or connect to a job that is already running.

6.5.3. Macroinstructions Used in Complex Conversation

The same declarative and imperative macroinstructions used in simple conversation are required in complex conversation, except that the keyword USE=PROG is also required in the primary program and surrogate program RIB for complex conversation.

Format

LABEL	ΔOPERATIONΔ	OPERAND
ribname	RIB	USE=PROG other keyword parameters

The same imperative macroinstructions used in simple conversation are also used in complex conversation, along with the following two imperative macroinstructions: DMSEL and DMCTL.

Selecting the Surrogate (DMSEL)

The select imperative macroinstruction, DMSEL, is used to begin and end conversations between paired application programs. Only the primary program can issue a DMSEL, and, since the surrogate program must reply with a DMOUT, the primary must next issue a DMINP.

Format

```
DMSEL⎧(1)     ⎫,PROG,⎧(0)           ⎫,⎧ACT  ⎫[,DATA]
     ⎩cdibname⎭      ⎩surrogate-prog⎭ ⎩DEACT⎭
```

Parameters

`cdibname`
: Is the name of the CDIB specified in the primary program. If (1) is specified, it is assumed that the address of the CDIB has been loaded into register 1.

`PROG`
: Identifies DMCTL as belonging to the program-to-program facility.

`surrogate-prog`
: Specifies the name of the surrogate with which a conversation is to be activated or deactivated. If (0) is specified, it is assumed that the address of the name of the surrogate program was loaded into register 0.

`ACT`
: Indicates a conversation with the surrogate program named is to be activated (opened).

`DEACT`
: Indicates a conversation with the surrogate program named is to be deactivated (closed).

`DATA`
: Indicates that the next macroinstruction issued will contain data to be passed with the DMSEL.

Special Control (DMCTL)

The special control imperative macroinstruction can only be used:

- by the primary program to pass control to the surrogate program via the BEQueath parameter, or
- by the surrogate program to reply to a DMSEL with DEACT, issued by the primary program.

Format

```
DMCTL┌(1)      ┐,PROG,┌BEQ┐
     └cdibname┘      └END┘
```

Parameters

`cdibname`
: Is the name of the CDIB specified in the primary or surrogate program, depending on whether the BEQ or END parameters, respectively, are specified.

`PROG`
: Identifies DMCTL as belonging to the program-to-program component.

`BEQ`
: Issued only by the primary program, indicating it is passing control to the surrogate program. Upon reception of the BEQueath, the surrogate program becomes the primary program and the primary program becomes the surrogate program. If a reply was required from the surrogate, it is sent before the BEQueath takes effect.

`END`
: Issued only by the surrogate program, and only as a reply to a DMSEL with DEACT, which is issued by the primary program, to indicate the conversation is to be terminated.

Closing a Conversation Path (DCLOSE)

The DCLOSE imperative macroinstruction is issued to close a communications path between user application programs, as for simple conversations (6.4.4), but the primary program must first issue a DMSEL with DEACT or the conversation is aborted.

6.5.4. Capabilities of Complex Conversation

The DMSEL, DMCTL, and DCLOSE macros give you flexibility in designing primary and surrogate programs for complex conversation. Your program still goes through the three basic phases - initiation, data transfer, and termination - but there are various smaller steps you can take; or you can link the procedures to pass from primary data transfer state to surrogate transfer state and then go back to the primary transfer state.

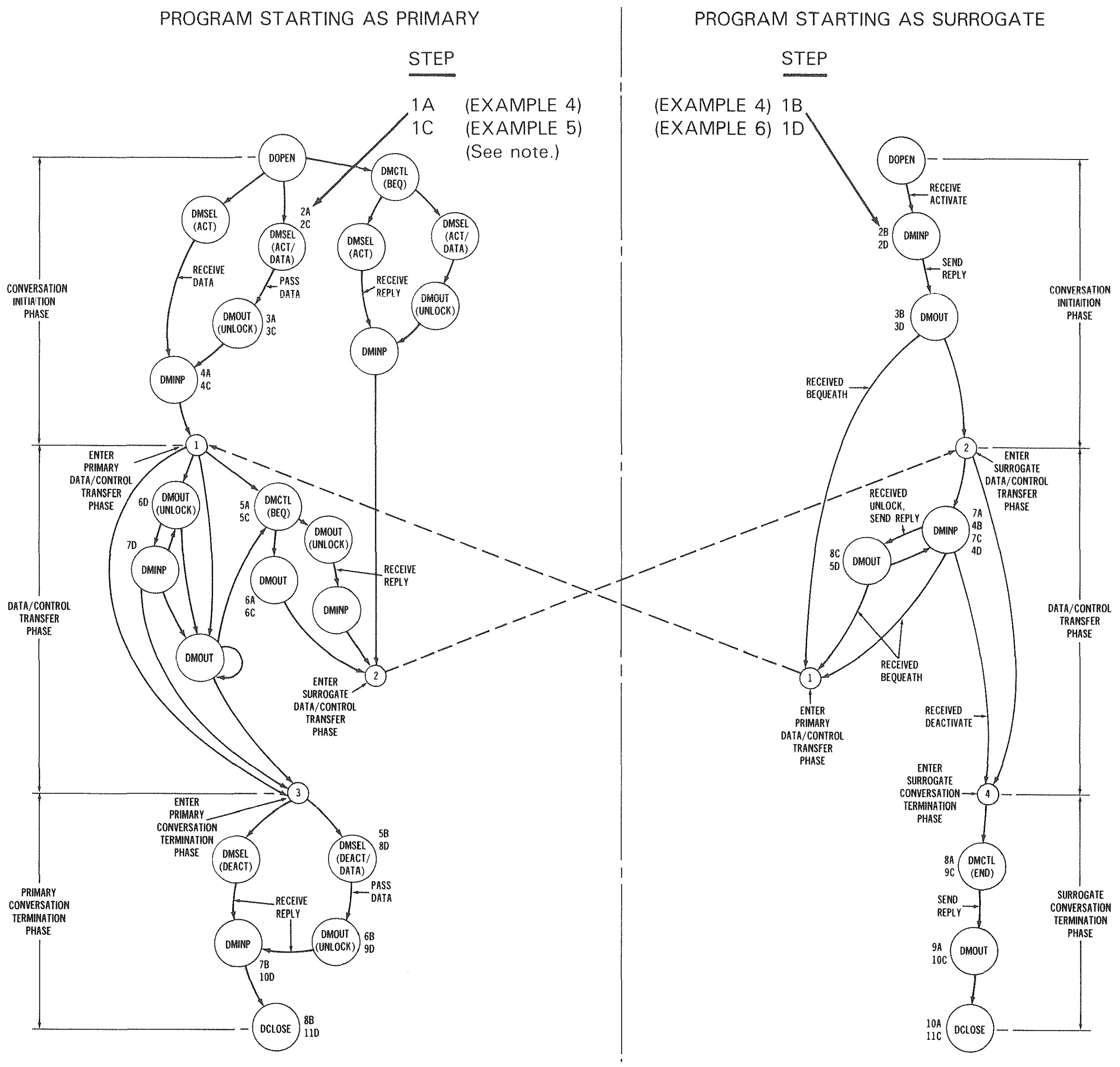
Figure 6-7 is a master flowchart showing details of the procedures in all possible combinations that you might use in writing a program set. Horizontally, the left section of the diagram shows the possible flow of a program starting as a primary program. The right section of the diagram shows a similar flow for a program starting as a surrogate program. Vertically, the diagram shows the program flow for both programs, starting with the conversation initiation phase at the top, then the data and control phase and, finally, the conversation termination phase at the bottom of the diagram.

The required macroinstructions and parameters are shown in the large circles of the procedure links and the smaller circles define the state your program arrives at. For example, the small circles containing a 1 signify the program has arrived at the primary data/control transfer phase. The small circles containing a 2 signify the program has arrived at the surrogate data/control transfer phase.

When a program starting as primary reaches the 2 state, it continues from the surrogate state 2. Similarly, when a program starting as a surrogate reaches the 1 state, it continues from the primary diagram state 1. (See steps 6A to 7A, 6C to 7C, and 5D to 6D in Figure 6-7. Examples 4, 5, and 6 in subsection 6.5.6 delineate these changes in control.)

The small circles containing a 3 or 4 indicate the program can enter the primary conversation termination phase or surrogate conversation termination phase, respectively.

The notes signifying the conditions required before being able to continue in the flow, such as RECEIVE DATA, PASS DATA, RECEIVED BEQUEATH, and so forth, indicate the protocol conditions required to issue the macroinstructions or conditions that follow the use of the macroinstruction. For further information, refer to the macroinstruction specifications in 6.5.3.



LEGEND:

Large circle ◯ indicates the imperative macroinstruction shown can be issued.

When the program starting as primary reaches the ② state, it continues from the surrogate diagram state ② .

When the program starting as surrogate reaches the ① state, it continues from the primary diagram state ① .

Note: In the examples, steps progress within the same letter. For example, after 1A go to 2A, 3A, etc.

Figure 6-7. Complex Conversation Procedures Flow Diagrams

6.5.5. Job Control Requirements

The job control for complex conversation is very simple, as shown for the following pair of communicating programs.

For the Primary Program

```
// JOB jobname
      .
      .
      .
// EXEC load module name
      .
      .
      .
/&
// FIN
```

For the Surrogate Program

```
// JOB jobname
      .
      .
      .
// EXEC load module name
      .
      .
      .
/&
// FIN
```

where:

`jobname`
: For the primary program, jobname is the job name specified for the primary program.

  For the surrogate program, jobname is the job name specified for the surrogate program.

`load module name`
: For the primary program, load module name is the name of the load module specified for the primary program.

  For the surrogate program, load module name is the name of the load module specified for the surrogate program.

6.5.6. Examples of Complex Conversation Programs

The following are examples of complex conversation programs. Examples 4, 5, and 6 are referenced on the complex conversation procedures flow diagrams (Figure 6-7), according to the coding line number listed at the left side of each program that follows. Therefore, you can relate the examples to the flow diagrams and learn how and why the examples were coded as they are and what other procedures could be used if your program required it.

Example 4: Primary Program and Surrogate Program Pair with BEQueath

Figure 6-8 shows the required job control .

The primary program (on the left side of Figure 6-9) issues a DOPEN and selects the surrogate program via DMSEL ACTivate to exchange status via DMCTL BEQueath. After receiving acceptance from the surrogate program, the program becomes the new surrogate program and receives data from the new primary program.

The surrogate program (on the right side of Figure 6-9) issues a DOPEN, replies to a message from the primary program, accepts the BEQueath, and, as the new primary program, sends data to the new surrogate program. The program then issues a DMSEL DEACTivate and, after receiving a reply, closes the conversation.

Primary Program	Surrogate Program

// JOB PTP$PRIM
.
.
.
// EXEC load module name
.
.
.
/&
// FIN
	// JOB PTP$SRGT
	.
	.
	.
	// EXEC load module name
	.
	.
	.
	/&
	// FIN

Figure 6-8. Job Control for Primary Program and Surrogate Program Pair with BEQueath

Figure 6-9 shows the coding for example 4:

```
Primary Program                              | Surrogate Program
---------------------------------------------+----------------------------------------
      PRIMARY START  0                       | SURROGATE  START  0
              .                              |            .
              .                              |            .
              .                              |            .
              LA     R1,PRIMCDIB             |            LA     R1,SURRCDIB
              .                              |            .
              .                              |            .
              .                              |            .
              LA     R0,PRIMRIB              |            LA     R0,SRGATRIB
Line                                         | Line
  1A          DOPEN   (1),(0)                |  1B        DOPEN  (1),(0)
              .                              |            .
              .                              |            .
              .                              |            .
              LA     R0,=CL8'PTP$SRGT'       |
  2A          DMSEL  (1),PROG,(0),ACT,DATA   |
              LA     R0,ACTMSG               |            LA     R0,RPLYMSG
  3A          DMOUT  (1),(0),UNLOCK          |  2B        DMINP  (1),(0)
              .                              |            .
              .                              |            .
              .                              |            .
              LA     R0,IOAREA               |            LA     R0,RPLYMSG
  4A          DMINP  (1),(0)                 |  3B        DMOUT  (1),(0)
              .                              |            .
              .                              |            .
              .                              |            .
  5A          DMCTL  (1),PROG,BEQ            |
              LA     R0,BEQMSG               |            LA     R0,IOAREA
  6A          DMOUT  (1),(0)                 |  4B        DMINP  (1),(0)
              .                              |            .
              .                              |            .
              .                              |            .
              LA     R0,IOAREA               |            LA     R0,=CL8'PTP$PRIM'
                                             |  5B        DMSEL (1),PROG,(0),DEACT
  7A          DMINP   (1),(0)                |            LA     R0,DEACTMSG
  8A          DMCTL   (1),PROG,END           |  6B        DMOUT  (1),(0),UNLOCK
              .                              |            .
              .                              |            .
              .                              |            .
              LA     R0,=RPLMSG              |            LA     R0,IOAREA
  9A          DMOUT  (1),(0)                 |  7B        DMINP  (1),(0)
              .                              |            .
              .                              |            .
              .                              |            .
 10A          DCLOSE (1)                     |  8B        DCLOSE (1)
              .                              |            .
              .                              |            .
              .                              |            .
              EOJ                            |            EOJ
     PRIMCDIB DC     XL48'00'                |  SURRCDIB  DC     XL48'00'
     PRIMRIB  RIB    USE=PROG,               |  SRGATRIB  RIB    USE=PROG,
                     HOSTID=BBBB             |                   RCSZ=80
                     RCSZ=80                 |  RPLYMSG   DC     CL80'REPLY'
     ACTMSG   DC     CL80'ACTIVATE'          |  DEACTMST  DC     CL80'DEACTIVATE'
     BEQMSG   DC     CL80'BEQUEATH'          |  IOAREA    DC     XL80'00'
     RPLYMSG  DC     CL80'REPLY'             |
     IOAREA   DC     XL80'00'                |
```

Figure 6-9. Coding for Primary Program and Surrogate Program Pair with BEQueath

Example 5: Primary Program Issuing a BEQueath and Repeatedly Receiving Data Transfers

Figure 6-10 shows the primary program issuing a DOPEN and DMSEL to select the surrogate program (shown in Figure 6-11) and passes control to it via a BEQueath. As the new surrogate program, it repeatedly receives data transfers from the new primary program, replying whenever necessary.

Job Control Stream

```
// JOB PRIMJOB
    .
    .
    .
// EXEC load module name
    .
    .
    .
/&
// FIN
```

BAL Program for the Complex Conversation Primary Program PROGC

```
Line                                      Description

     PROGC  START  0
            BALR   R15,0
            USING  *,R15
                                          LOADING THE CDIB
                                           Load address of the following CDIB into R1:
                                               PRIMARY CDIB FILENAME=PRIMARY
                                          OPEN CONNECTION TO SURROGATE
            LA     R0,PRIMRIB              Load address of the following RIB into R0:
            .                                 PRIMRIB   RIB   USE=PROG,HOSTID=ISLC
            .                                                 WKFM=VARI
            .
  1C        DOPEN  (1),(0)                 Open session path
            .                              Check for successful open
            .                            SELECT THE SURROGATE PROGRAM
            .
            LA     R0,=CL6'SURJOB'          Load surrogate program name into R0
  2C        DMSEL  (1),PROG,0,ACT,DATA      Select surrogate program
            .                               Check for successful open
            .
            .
            LA     R0,ACTMSG                Load address of ACTivate DATA
  3C        DMOUT  (1),(0),UNLOCK          Issue ACTIVATE
            .                              Check for successful ACTivate
            .                             ISSUE INPUT REQUEST TO GET REPLY FROM
            .                             SURROGATE
            LA     R0,RPLYAREA             Load address of reply area into R0
```

Figure 6-10. Job Control and Coding for Primary Program Issuing a BEQueath and Repeatedly Receiving Data Transfers (Part 1 of 2)

BAL Program for the Complex Conversation Primary Program PROGC (continued)

```
             .
             .
             .
 4C          DMINP  (1),(0)                Wait for surrogate reply
             .                             Check whether reply was received
             .                            ISSUE BEQUEATH TO SURROGATE
             .
 5C          DMCTL  (1),PROG,BEQ           Indicate BEQueath
             .
             .
             .
             LA     R0,=BEQMSG              Load address of BEQueath data
 6C          DMOUT  (1),(0)                 Send data message - No reply required
             .                             PREPARE TO RECEIVE DATA FROM THE PRIMARY
             .
             .
             LA     R0,INPTAREA             Load address of input area
             .
             .
             .
 7C          DMINP  (1),(0)                 Get data from primary
             .                              Check for successful read
             .                             IF REPLY IS REQUIRED, SEND REPLY TO PRIMARY
             .
             LA     R0,INPTAREA             Load address of reply area
 8C          DMOUT  (1),(0)                 Reply to primary
             .                              Check for successful reply
             .                              Loop to get more input (DMINP performed)
             .                             CHECK WHICH EXCEPTION FLAGS ARE SET IN CDIB
 9C          DMCTL  (1),PROG,END           Indicate REPLY END after receiving
             .                             DEACTivate
             .
             .
             LA     R0,INPTAREA            Load address of reply
             .
             .
             .
10C          DMOUT  (1),(0)                Reply to primary
             .                             Check for successful reply
             .                            CLOSE CONNECTION TO PRIMARY
             .
11C          DCLOSE (1)                    Close session path
             .                             Check for successful close
             .
             .
             EOJ
```

Figure 6-10. Job Control and Coding for Primary Program Issuing a BEQueath and Repeatedly Receiving Data Transfers (Part 2 of 2)

Example 6: Surrogate Program Receiving a BEQueath and Repeatedly Transferring Data

Figure 6-11 shows the surrogate program issuing a DOPEN and receiving an ACTivate message from the primary program shown in Figure 6-10. After replying, it receives a BEQueath from the primary program and, as the new primary, sends 500 messages to the new surrogate, checking proper receipt after each 50 messages. It issues a DEACTivate message to the new surrogate program and, after receiving a reply, END, it terminates the conversation.

Job Control Stream

```
// JOB SURJOB
       .
       .
       .
// EXEC load module name
       .
       .
       .
/&
// FIN
```

BAL Program for the Complex Conversation Surrogate Program PROGD

```
Line                                    Description

    PROGD  START  0
           BALR   R15,0
           USING  *,R15
                                        LOADING THE CDIB
           LA     R1,SURCDIB             Load address of the following CDIB into R1:
           .                                SURCDIB   CDIB  FILENAME=SURRGATE
           .
           .
           LA     RO,SURRIB              Load address of the following RIB into RO:
                                            SURRIB    RIB USE=PROG,WKFM=VARI
 1D        DOPEN  (1),(0)                Attach to primary session
           .                             Check for successful attachment
           .                            GET ACTIVATE MESSAGE FROM THE PRIMARY
           .
           LA     R0,INPTAREA            Load input area into R0
           .
           .
           .
 2D        DMINP  (1),(0)                Get input from primary
           .                             Check for successful input
           .                            REPLY TO PRIMARY
           .
           LA     R0,INPTAREA            Load input area into R0
 3D        DMOUT  (1),(0)                Reply to primary
           .                             Check for successful reply
           .                            RECEIVE BEQUEATH FROM PRIMARY
           .
           LA     R0,INPTAREA            Load address of input area
```

Figure 6-11. Job Control and Coding for Surrogate Program Receiving a BEQueath and Repeatedly Transferring Data (Part 1 of 2)

```
BAL Program for the Complex Conversation Surrogate Program PROGD (continued)

 4D        DMINP  (1),(0)                      Get message from primary
           .                                   Check whether BEQueath was specified
           .                                  ENTER DATA TRANSFER LOOP
           .                                   Loop2 is executed 49 times and a reply is
                                               expected on the 50th message.
  SENDLOOP LA     R9,10(0,0)                   Set Loop1 counter (R9)
  LOOP1    LA     R8,49(0,0)       ┌ LOOP1     Set Loop2 counter (R8)
  LOOP2    MVC    DATAAREA(84),    │┌ LOOP2    Load address of data area
           .      DATAMSG          ││
           .                       ││
           .                       ││
 5D        DMOUT  (1),(0)          ││          Send data without request for reply
           .                       ││          Check for successful send
           .                       ││
           .                       ││
           BCT    R8,LOOP2         │└───────   Loop through 49 times
           .                       │          SET UP MESSAGES FOR WHICH REPLY IS REQUIRED
           .                       │
           .                       │
           LA     R0,DATAAREA      └──┐         Load address of data area
 6D        DMOUT  (1),(0),UNLOCK      │         Send message and wait for reply
           .                          │         Check for successful send
           .                          │        GET REPLY FROM SURROGATE AND CHECK VS
           .                          │        LAST MESSAGE SENT
           LA     R0,RPLYAREA         │         Load the reply area
           .                          │
           .                          │
           .                          │
 7D        DMINP  (1),(0)             │         Get the reply
           .                          │         Check for successful reply
           .                          │
           .                          │
           BCT    R9,LOOP1            └──────   If check positive, loop through 10 times
                                               TERMINATE CONVERSATION WITH SURROGATE
           LA     R0,=CL7'PRIMJOB'              Load surrogate program name
 8D        DMSEL  (1),PROG,(0),                 Issue DEACTivate
           .      DEACT,DATA
           .                                    Check for successful DEACTivate
           .
           LA     RO,DATAMSG                    Load address of DEACTivate message
 9D        DMOUT  (1),(0),UNLOCK                Issue DEACTivate message
           .                                    Check for successful DEACTivate
           .                                   CHECK WHETHER REPLY TO DEACTIVATE MATCHES
           .                                   MESSAGE SENT TO SURROGATE
           LA     R0,RPLYAREA                   Load address of reply area
10D        DMINP  (1),(0)                       Get the reply
           .                                    Check whether reply was received
           .                                   CHECK THAT REPLY END WAS RECEIVED
           .                                   TERMINATE ATTACHMENT TO SURROGATE
11D        DCLOSE (1)                           Detach from the surrogate
           .                                    Check for successful detachment
           .
           .
           EOJ
```

Figure 6-11. Job Control and Coding for Surrogate Program Receiving a BEQueath and Repeatedly Transferring Data (Part 2 of 2)

Example 7: Primary Program Simultaneously Transferring Data to Three Surrogate Programs on Three Different Hosts

Figure 6-12 shows a primary program simultaneously transferring 50 messages to three surrogate programs on three different, remote hosts.

Job Control Stream

```
// JOB PTP$PRIM
       .
       .
       .
// EXEC load module name
       .
       .
       .
/&
// FIN
```

BAL Program for the Complex Conversation Primary Program PROGE

```
Line                                  Description

    PROGE  START  0
           .
           .
           .
           BALR   R15,0
           USING  *,R15,R12,R11,R10
           .                          BUILD THREE CDIBs FOR PROGRAM-TO-PROGRAM
           .                          OPERATION
           .
           LA     R7,50(0,0)               Load count for 50 messages
  LOOPSTRT EQU    *
           LA     R1,CDIB1                 Load address of CDIB1 into R1
           USING  CD$DCIB,R1
           BLDCDIB FILENAME=PRIME1,MSGSUP=NO
           LA     R1,CDIB2                 Load address of CDIB2 into R1
           BLDCDIB FILENAME=PRIME2,MSGSUP=NO
           LA     R1,CDIB3                 Load address of CDIB3 into R1
           BLDCDIB FILENAME=PRIME3,MSGSUP=NO
           .                          OPEN A CONNECTION TO EACH SURROGATE
           .
           .
           LA     R0,RIB1              Load address of the following RIB1 into R0:
                                          RIB1    RIB    USE=PROG,HOSTID=NOD4
                                                         WKFM=VARI
           LA     R1,CDIB1             Load address of the following CDIB1 into R1:
                                          CDIB1    CDIB
           DOPEN  (1),(0)              Open connection
           .                           Check for successful open
           .
           .
           LA     R0,RIB2              Load address of the following RIB2 into R0:
                                          RIB2    RIB
           LA     R1,CDIB2             Load address of the following CDIB2 into R1:
                                          CDIB2    CDIB
```

Figure 6-12. Job Control and Coding for Primary Program Simultaneously Transferring Data to Three Surrogate Programs on Three Different Hosts (Part 1 of 6)

```
BAL Program for the Complex Conversation Primary Program PROGE (continued)

          DOPEN  (1),(0)                Open connection
          .                             Check for successful open
          .
          .
          LA     R0,RIB3                Load address of the following RIB3 into R0:
                                           RIB3    RIB
          LA     R1,CDIB3               Load address of the following CDIB3 into R1:
                                           CDIB3    CDIB
          DOPEN  (1),(0)                Open connection
          .                             Check for successful open
          .                            SELECT EACH SURROGATE PROGRAM
          .
          LA     R0,=CL8'PTP$SRG1'       Load surrogate program name into R0
          LA     R1,CDIB1                Load address of CDIB1 into R1
          DMSEL  (1),PROG,(0),           Select surrogate program
          .      ACT,DATA
          .                              Check for successful select
          .
          LA     R0,=CL8'PTP$SRG2'       Load surrogate program name into R0
          LA     R1,CDIB2                Load address of CDIB2 into R1
          DMSEL  (1),PROG,(0),           Select surrogate program
          .      ACT,DATA
          .
          .
          LA     R0,CL8'PTP$SRG3'        Load surrogate program name into R0
          LA     R1,CDIB3                Load address of CDIB3 into R1
          DMSEL  (1),PROG,(0),           Select surrogate program
          .      ACT,DATA
          .                              Check for successful select
          .                             SET UP DATA FOR ACTIVATE COMMAND
          LA     R0,ACTVMSG1             Load activate data message into R0
          LA     R1,CDIB1                Load address of CDIB1 into R1
          DMOUT  (1),(0),UNLOCK          Issue ACTivate
          .                              Check for successful ACTivate
          .
          .
          LA     R0,ACTVMSG2             Load activate data message into R0
          LA     R1,CDIB2                Load address of CDIB2 into R1
          DMOUT  (1),(0),UNLOCK          Issue ACTivate
          .                              Check for successful ACTivate
          .
          .
          LA     R0,ACTVMSG3             Load address of CDIB3 into R0
          LA     R1,CDIB3                Load address of CDIB3 into R1
          DMOUT  (1),(0),UNLOCK          Issue ACTivate
          .                              Check for successful ACTivate
          .                             ISSUE INPUT REQUEST TO GET REPLY FROM
          .                             SURROGATE
          LA     R0,RPYAREA1             Load address of reply area into R0
          .
          .
          .
          LA     R1,CDIB1                Load address of CDIB2 into R1
```

Figure 6-12. Job Control and Coding for Primary Program Simultaneously Transferring Data to Three Surrogate Programs on Three Different Hosts (Part 2 of 6)

BAL Program for the Complex Conversation Primary Program PROGE (continued)

```
        DMINP  (1),(0)                 Wait for surrogate reply
        .                              Check for successful reply
        .
        .
        .
        LA     R0,RPYAREA2             Load address of reply area into R0
        .
        .
        .
        LA     R1,CDIB2                Load address of CDIB3 into R1
        DMINP  (1),(0)                 Wait for surrogate reply
        .                              Check for successful reply
        .
        .
        LA     R0,RPYAREA3             Load address of reply area into R0
        .
        .
        .
        LA     R1,CDIB3                Load address of CDIB3 into R1
        DMINP  (1),(0)                 Wait for surrogate reply
        .                              Check for successful reply
        .
        .
                                      ENTER DATA TRANSFER LOOP
                                       Loop2 is executed 49 times and a reply
                                       is expected on the 50th message. The
                                       reply must match the 49th message.
        LA     R9,10(0,0)     ┌ LOOP1    Set LOOP1 counter (R9)
  LOOP1 LA     R8,49(0,0)     │┌─ LOOP2  Set LOOP2 counter (R8)
  LOOP2 AP     MSGCOUNT(4),   ││         Update message counter
        .      =PL4'10'       ││
        .                     ││
        .                     ││
        LA     R0,DATAREA1    ││         Load address of data area into R0
        LA     R1,CDIB1       ││         Load address of CDIB1 into R1
        DMOUT  (1),(0)        ││         Send data - without request for reply
        .                     ││         Check for successful send
        .                     ││
        .                     ││
        LA     R0,DATAREA2    ││         Load address of data area into R0
        LA     R1,CDIB2       ││         Load address of CDIB2 into R1
        DMOUT  (1),(0)        ││         Send data - without request for reply
        .                     ││         Check for successful send
        .                     ││
        .                     ││
        LA     R0,DATAREA3    ││         Load address of data area into R0
        LA     R1,CDIB3       ││         Load address of CDIB3 into R1
        DMOUT  (1),(0)        ││         Send data - without request for reply
        .                     ││         Check for successful send
        .                     ││
        .                     ││
        BCT    R8,LOOP2       │└───────  Loop through 49 times
```

Figure 6-12. Job Control and Coding for Primary Program Simultaneously Transferring Data to Three Surrogate Programs on Three Different Hosts (Part 3 of 6)

```
BAL Program for the Complex Conversation Primary Program PROGE (continued)

                    LOOP 1 (continued)
     .
     .
     .
                                      SET UP MESSAGE FOR WHICH A REPLY IS EXPECTED
     LA     R0,DATAREA1                  Load address of data area into R0
     LA     R1,CDIB1                     Load address of CDIB1 into R1
     DMOUT  (1),(0),UNLOCK               Send data - request reply
     .                                   Check for successful send
     .
     .
     LA     R0,DATAREA2                  Load address of data area into R0
     LA     R1,CDIB2                     Load address of CDIB2 into R1
     DMOUT  (1),(0),UNLOCK               Send data - request reply
     .                                   Check for successful send
     .
     .
     LA     R0,DATAREA3                  Load address of data area into R0
     LA     R1,CDIB3                     Load address of CDIB3 into R1
     DMOUT  (1),(0),UNLOCK               Send data - request reply
     .                                   Check for successful send
     .
     .
                                        GET REPLY FROM SURROGATE CHECK IT AGAINST
                                        LAST OUTPUT MESSAGE SENT
     LA     R0,RPYAREA1                  Load address of reply area into R0
     .
     .
     .
     LA     R1,CDIB1                     Load address of CDIB1 into R1
     DMINP  (1),(0)                      Get the reply
     .                                   Check for successful reply
     .
     .
     LA     R0,RPYAREA2                  Load address of reply area into R0
     .
     .
     .
     LA     R1,CDIB2                     Load address of CDIB2 into R1
     DMINP  (1),(0)                      Get the reply
     .                                   Check for successful reply
     .
     .
     LA     R0,RPYAREA3                  Load address of reply area into R0
     .
     .
     .
     LA     R1,CDIB3                     Load address of CDIB3 into R1
     DMINP  (1),(0)                      Get the reply
     .                                   Check for successful reply
     .
     .
     BCT    R9,LOOP1           ------    Loop through 10 times
```

Figure 6-12. Job Control and Coding for Primary Program Simultaneously Transferring Data to Three Surrogate Programs on Three Different Hosts (Part 4 of 6)

BAL Program for the Complex Conversation Primary Program PROGE (continued)

```
                                        TERMINATE THE CONVERSATION WITH THE SURROGATE
                                        PROGRAM
         LA     R0,=CL8'PTP$SRG1'        Load surrogate program name with a R0
         LA     R1,CDIB1                 Load address of CDIB1 into R1
         DMSEL  (1),PROG,(0),            Select surrogate program
         .      DEACT,DATA               Check for successful select
         .
         .
         LA     R0,=CL8'PTP$SRG2'        Load surrogate program name into R0
         LA     R1,CDIB2                 Load address of CDIB2 into R1
         DMSEL  (1),PROG,(0),            Select surrogate program
         .      DEACT,DATA               Check for successful select
         .
         .
         LA     R0,=CL8'PTP$SRG3'        Load surrogate program name into R0
         LA     R1,CDIB3                 Load address of CDIB3 into R1
         DMSEL  (1),PROG,(0),            Select surrogate program
         .      DEACT,DATA               Check for successful select
         .
         .
                                        SET UP DATA FOR DEACTIVATE MESSAGE
         LA     R0,DACTMSG1              Load address of DEACTivate message into R0
         LA     R1,CDIB1                 Load address of CDIB1 into R1
         DMOUT  (1),(0),UNLOCK           Issue DEACTivate
         .                               Check for successful DEACTivate
         .
         .
         LA     R0,DACTMSG2              Load address of DEACTivate message into R0
         LA     R1,CDIB2                 Load address of CDIB2 into R1
         DMOUT  (1),(0),UNLOCK           Issue DEACTivate
         .                               Check for successful DEACTivate
         .
         .
         LA     R0,DACTMSG3              Load address of DEACTivate message into R0
         LA     R1,CDIB3                 Load address of CDIB3 into R1
         DMOUT  (1),(0),UNLOCK           Issue DEACTivate
         .                               Check for successful DEACTivate
         .
         .
                                        CHECK IF REPLY TO DEACTIVATE MATCHES
                                        MESSAGE SENT TO THE SURROGATE
         LA     R0,RPYAREA1              Load address of reply area into R0
         .
         .
         .
         LA     R1,CDIB1                 Load address of CDIB1 into R1
         DMINP  (1),(0)                  Wait for the reply
         .                               Check whether REPLY (END) was received
         .
         .
         LA     R0,RPYAREA2              Load address of reply area into R0
         .
         .
         .
```

Figure 6-12. Job Control and Coding for Primary Program Simultaneously Transferring Data to Three Surrogate Programs on Three Different Hosts (Part 5 of 6)

```
BAL Program for the Complex Conversation Primary Program PROGE (continued)

          LA     R1,CDIB2                  Load address of CDIB2 into R1
          DMINP  (1),(0)                   Wait for the reply
          .                                Check whether REPLY (END) was received
          .
          .
          LA     R0,RPYAREA3               Load address of reply area into R0
          .
          .
          .
          LA     R1,CDIB3                  Load address of CDIB3 into R1
          DMINP  (1),(0)                   Wait for the reply
          .                                Check whether REPLY (END) was received
          .
          .
                                          CLOSE CONNECTION TO SURROGATE PROGRAM
          LA     R1,CDIB1                  Load address of CDIB1 into R1
          DCLOSE (1)                       Close the connection
                                           Check for successful close
          LA     R1,CDIB2                  Load address of CDIB2 into R1
          DCLOSE (1)                       Close the connection
          .                                Check for successful close
          .
          .
          LA     R1,CDIB3                  Load address of CDIB3 into R1
          DCLOSE (1)                       Close the connection
          .
          .
          .
          BCT    R7,LOOPSTRT               Loop for execution count
          EOJ                              End of job
          CDIB1  DC     XL48'00'
          CDIB2  DC     XL48'00'
          CDIB3  DC     XL48'00'
          RIB1   RIB    USE=PROG,HOSTID=NOD4,WKFM=VARI
          RIB2   RIB    USE=PROG,HOSTID=NOD4,WKFM=VARI
          RIB3   RIB    USE=PROG,HOSTID=NOD4,WKFM=VARI
```

Figure 6-12. Job Control and Coding for Primary Program Simultaneously Transferring Data to Three Surrogate Programs on Three Different Hosts (Part 6 of 6)

Example 8: Typical Surrogate Program Used during Complex Conversation with the Primary Program of Example 7

Figure 6-13 shows a typical surrogate program used along with two other surrogate programs simultaneously conversing with the primary program shown in Figure 6-12.

Job Control Stream

```
// JOB PTP$SRGn                         (where n is 1, 2, or 3)
       .
       .
       .
// EXEC load module name
       .
       .
       .
/&
// FIN
```

BAL Program for the Complex Conversation Typical Surrogate Program PROGF

```
Line                                    Description

    PROGE  START  0
           .
           .
           .
           BALR   R15,0
           USING  *,R15
                                        LOADING THE CDIB
           LA     R1,SURCDIB             Load address of the following CDIB into R0:
           .                                SURCDIB   CDIB   FILENAME=PTP$SRGn
           .
           .
           LA     R0,SURRIB              Load address of the following RIB into R0:
                                            SURRIB   RIB   USE=PROG,WKFM=VARI
                                        ATTACH TO THE PRIMARY
           DOPEN  (1),(0)                Attach to the primary session
                                         Check for successful attachment
                                        GET ACTIVATE MESSAGE FROM PRIMARY
           LA     R0,INPTAREA            Load address of input area into R0:
           .
           .
           .
           DMINP  (1),(0)                Get input from primary
           .                             Check for successful input
           .
           .
                                        REPLY TO PRIMARY
           LA     R0,INPTAREA            Load input area into R0
           DMOUT  (1),(0)                Reply to primary
           .                             Check for successful reply
           .
           .
                                        PREPARE TO RECEIVE DATA FROM PRIMARY
```

Figure 6-13. Job Control and Coding for a Typical Surrogate Program Used during Complex Conversation with the Primary Program of Example 7 (Part 1 of 2)

```
BAL Program for the Complex Conversation Typical Surrogate Program PROGF (continued)

RECVLOOP  LA     R0,INPTAREA                  Load address of reply into R0
          .
          .
          .
          DMINP  (1),(0)                      Get data from primary
          .                        Recvloop   Check for successful read
          .
          .
                    Recvloop                 IF A REPLY IS REQUIRED, SEND REPLY TO PRIMARY
          BE     RECVLOOP                     Continue loop if no flags are set
          .
          .
          .
          LA     R0,INPTAREA                  Load address of reply into R0
          DMOUT  (1),(0)                      Reply to primary
          .                                   Check for successful reply
          .
          .
          BE     RECVLOOP                     Loop to get more input
          .                                  CHECK WHICH EXCEPTION FLAGS ARE SET IN CDIB
          .                                   Check whether DEACTivate was received
          .                                  REPLY TO THE DEACTIVATE FROM THE PRIMARY
          DMCTL  (1),PROG,END
          .
          .
          .
          LA     R0,INPTAREA                  Load address of reply into R0
          .
          .
          .
          DMOUT  (1),(0)                      Reply to primary
          .                                   Check for successful reply
          .
          .
                                             TERMINATE ATTACHMENT TO PRIMARY
          DCLOSE (1)                          Detach from the primary
          .
          .
          .
          EOJ
SURCDI    CDIB   FILENAME=PTP$SRGn (where n is 1, 2, or 3)
SURRIB    RIB    USE=PROG,WKFM=VARI
INPTAREA  DC     XL'input-area-size'
```

Figure 6-13. Job Control and Coding for a Typical Surrogate Program Used during Complex Conversation with the Primary Program of Example 7 (Part 2 of 2)

Section 7
BAL Program Level Interface

7.1. Introduction

This section describes how to access the distributed functions provided by OS/3 DDP with user-written basic assembler programs.

7.2. User Program Parameter List

The basis for all processing at the program level interface is the parameter list passed by the user program. This parameter list is generated by an assembly language procedure call.

Format

```
Label DDP positional-param, keyword-params
```

Parameters:

`positional-param`
Is one of these DDP commands:

```
CREATE
COPY
PURGE
SUBMIT FILE
SUBMIT REQUEST
TALK
```

`keyword-params`
Are command keywords used with specific DDP commands. Keyword parameters used on a procedure call may differ from those used on interactive DDP commands. These differences are shown in Table 7-1.

Keyword parameters can be:

- Direct parameters - Data is completely contained within the parameter.
- Indirect parameters - The parameter contains the address of an area within the user program where the actual parameter data is stored.

Indirect parameters are usually used for variable data, such as host-ids, file identifiers, message text, and destination user-ids. They are identified by a period (.) followed by a one- to eight-character label. For example:

```
FILE=.FLENAME
```

Indirect parameters require that data be aligned on a half-word boundary. Therefore:

```
Bytes 0/1 = Half-word specifying the length of the parameter information

Bytes 2/N = Indirect parameter data
```

Table 7-1 explains the command keywords used on a procedure call or gives the keyword parameter choices. There are no defaults.

Table 7-1. Procedure Call Command Keywords

Command	Keyword Explanation/Parameters	Interactive Keyword
CREATE	FILE= Actual file name (with optional host-id) or indirect specification giving the address of an area containing the file name string	(None)
	REGISTER= [{ VTOC / CATALOG }]	(None)
	FILETYPE= [{ SEQUENTIAL / RELATIVE / INDEXED / LIBRARY / UNDEFINED }]	FILE_TYPE=
	DEVCLASS= [{ DISK / TAPE / DISKETTE }]	DEVICE_CLASS=
	BLKSIZE= One- to nine-digit number specifying the block size	BLOCK_SIZE=
	INITSIZE= One- to nine-digit number specifying the initial size	INITIAL_SIZE=
COPY	FROM= Actual file name (with optional host-id) or indirect specification giving the address of an area containing the file name string	(None)

continued

Table 7-1. Procedure Call Command Keywords (cont.)

Command	Keyword Explanation/Parameters	Interactive Keyword
COPY (cont.)	TO= Actual file name (with optional host-id) or indirect specification giving the address of an area containing the file name string	(None)
	TRANSLAT= [ASCII / EBCDIC / NONE]	TRANSLATE=
	MODE= [DIRECT / WAIT / INDIRECT]	(None)
	POSITION= [EOF / SOF]	(None)
	KEY1= Size, location, and characteristics of first file key	KEY_1=
	KEY2= Size, location, and characteristics of second file key	KEY_2=
	KEY3= Size, location, and characteristics of third file key	KEY_3=
	KEY4= Size, location, and characteristics of fourth file key	KEY_4=
	KEY5= Size, location, and characteristics of fifth file key	KEY_5=
PURGE	FILE= Actual file name string (with optional host-id) or indirect specification giving the address of an area containing the actual file name string	(None)
SUBMIT	REQUEST= Actual text string or an indirect specification giving the address of an area containing the actual text string	(None)
	FILE= Actual file name string (with optional host-id) or indirect specification giving the address of an area containing the actual file name string	(None)

continued

Table 7-1. Procedure Call Command Keywords (cont.)

Command	Keyword Explanation/Parameters	Interactive Keyword
SUBMIT (cont.)	HOST= Actual host-id or indirect specification giving the address of an area containing the actual host-id	(None)
	PRINT= Actual print device or indirect specification giving the address of an area containing the actual print device specification	(None)
	PUNCH= Actual punch device or indirect specification giving the address of an area containing the actual punch device specification	(None)
TALK	MESSAGE= Actual text message or indirect specification giving the address of an area containing the actual text message	(None)
	USER= Actual user-id (with optional host-id) or indirect specification giving the address of an area containing the actual user-id	(None)
Applies to all commands	WAIT=$\left\{\begin{matrix}\text{YES}\\ \text{NO}\end{matrix}\right\}$ Specifies whether the user program is to wait for completion of the DDP command. If WAIT=YES (default), the program is put in the wait state until the command has completed. NOTE: If WAIT=YES and the program performs a copy from a remote host to the local host or another remote host, control is returned to the program when the COPY command is accepted by the source file host; however, there is currently no way to detect the completion of the actual copy process. If WAIT=NO, control is returned to the program when the DDP processing complex accepts the command. The user program can interrogate the X'80' bit in the first word of the parameter list contained in register 1.	(None)

7.3. Return Conditions

The DDP program interface routine (DDP$PROG) returns to the calling user program with registers 0 and 1 to indicate completion status.

7.3.1. Normal Exits

DDP$PROG returns to the user program with registers 0 and 1 set to zero to indicate normal completion status. All other user program registers are restored to the value at the time of the procedure call.

7.3.2. Error Exits

When an error is detected in DDP$PROG and the DDP processing complex is not called, DDP$PROG returns to the user program with:

- Register 0 set to the error code that caused the abnormal termination
- Register 1 set to zero when the error is detected

When the error is detected in the DDP processing complex, DDP returns to the user program with:

- Register 0 set to the value 512
- Register 1 set to the three-character EBCDIC DDP error code (see the *System Messages Reference Manual,* 7004 5190)

Table 7-2 lists errors or other abnormal conditions (sanity checks) returned to the user program by DDP$PROG. An explanation for each condition is given.

Table 7-2. DDP$PROG Abnormal Completion Status Messages

Error/Condition (Register 0 =)	Explanation
512	An error is returned from the DDP processing complex. The actual error is contained in register 1.
513	No main storage available for dynamic buffer allocation. This is a system resource problem.
514	Bad header item code (HIC) in parameters passed by the user program. This is usually the result of an error in the DDP procedure called by the user program and not the program itself, since the procedure generates the header item codes.
515	End of parameters indicator not found during parameter scan. This is a sanity check that points to a problem within the DDP procedure itself, not the processing module.
516	Bad header item code (HIC) for indirect parameter. This is a sanity check that points to a problem within the DDP procedure itself, not the processing module.
517	Indirect parameter is not half-word aligned. A problem exists in the user program data area alignment.
518	Size of the indirect parameter exceeds maximum allowed. There is an error in the size specified as the first 2 bytes of the indirect parameter.
519	Error returned from SYSCOM, DDP could not be activated. This is a sanity check.
520	DDP$PROG is cancelled. The requested function is terminated abnormally. This error results from the activation of abnormal termination island code because of a program check or an operator cancel request.

Appendix A
Sample DDP File Copy/Job Submission Session

The following example shows a typical file copy and job submission from a local to a remote host. We have divided the example into parts for easy explanation.

Shading indicates entries the user makes.

- **Part 1: Logging on the system and mounting the disk volume**

```
1.  LOGONΔJSMITH
2.  LOGON ACCEPTED date time
3.  DDPΔTALKΔMESSAGE='PLEASE MOUNT VOL007'ΔUSER=H001::OPERATORΔWAIT
4.  DDP002 CAI 002 TALK COMMAND ACCEPTED WO=000001 10:15:49
5.  H001:$Y$CON VOLUME MOUNTED. PROCEED.
6.  DDP022 TRM 022 TALK COMMAND COMPLETED WO=000001 10:15:59
```

Explanation:

1. This command attaches our job to the system. JSMITH is our user-id.
2. The LOGON command is accepted at the indicated date and time.
3. We tell the remote operator to mount our disk volume.
4. The DDP TALK command is accepted.
5. The remote operator responds to the DDP TALK command, telling us he has mounted our volume.
6. The DDP TALK command is completed.

- **Part 2: Allocating file space on the remote host and copying the file**

```
1.  DDPΔCREATEΔFILE=H001::',REM/PERS(AXS9/AXSA7),VOL007'ΔBLOCK_SIZE=182&
2.  RECORD_SIZE=91
3.  DDP002 CAI 002 CREATE COMMAND ACCEPTED WO=000002 10:20:37
4.  DDP022 mmm 022 CREATE COMMAND COMPLETED WO=000002 10:25:51
5.  DDPΔCOPYΔFROM=',PERSNNEL(JMS8/)'ΔTO=H001::',REM/PERS'#&
6.  '(/AXSA7)'''ΔMODE=WAIT
7.  DDP002 CAI 002 COPY COMMAND ACCEPTED WO=000003 10:37:21
8.  RECORD COUNT=cccccccc DATA SIZE=ssssssssss MODULES=mmm
9.  DDP022 mmm 022 COPY COMMAND COMPLETED WO=000003 10:42:39
```

Explanation:

1. & 2. We allocate space on our remote host H001 for our file, which we name REM/PERS. We give it read and write passwords (AXS9 and AXSA7). In the next command, we're going to be copying one of our local files to this file. So we have to establish the same block and record sizes as we have in our local file (182 and 91 characters, respectively). This command uses the defaults for DEVICE_CLASS, FILE_TYPE, INITIAL_SIZE, INCREMENT_SIZE, RECORD_FORM, and REGISTER.

3. The DDP CREATE command is accepted.

4. The DDP CREATE command is completed. Our file is now established and cataloged on the remote host.

5. & 6. We copy our local file, PERSNNEL, whose read password is JMS8, on our remote host H001 in our remote file REM/PERS. If a direct connection isn't possible immediately, we want the system to hold the command until it is. This command uses defaults for the POSITION and TRANSLATE parameters.

   These lines also show the method for breaking a line using character string concatenation.

7. The DDP COPY command is accepted.

8. This message indicates the number of data records, data size, and number of modules transferred.

9. The DDP COPY command is completed.

- **Part 3: Sending the job to the remote host**

```
1.  DDPΔSTATUSΔJOB=UPDATE
2.  DDP002 CA 002 STATUS COMMAND ACCEPTED WO=000004 10:49:01
3.  WAITING FOR I/O
4.  DDP022 mmm 022 STATUS COMMAND COMPLETED WO=000004 10:52:16
5.  DDPΔSUBMITΔFILE='PAY06,PAYJOBS(JMS17/),,S'ΔHOST=H001
6.  DDP002 CA 002 SUBMIT COMMAND ACCEPTED WO=000005 11:01:45
7.  DDP044 mmm mid H0010035 JOB SUBMITTED FOR WO=000005 11:01:52
```

Explanation:

1. We want to find out what happened to a job we submitted in a previous session.
2. The DDP STATUS command is accepted.
3. The job is being executed and is currently waiting for I/O.
4. The DDP STATUS command is completed.
5. We submit the job in module PAY06, located in library file PAYJOBS, to the remote host H001.
6. The DDP SUBMIT command is accepted.
7. Our job name is H0010035 (renamed by DDP on remote system). The first four characters of the jobname identify the host that initiated the SUBMIT command (not the host where the job is executing).

Appendix B
Command Function Summary

Table B-1. Command Function Summary

To do the following:	Use this command and these parameters:	Required?	See also:
Cancel a job	DDPΔCANCELΔJOB=		4.4.2
on a specific host	[{host-id local-host-id} ::]	No	
with a job name of	jobname	Yes	
where you want the job's output to be discarded or delivered to you	[ΔOUTPUT= {DISCARD DELIVER}]	No	
Cancel a command	DDPΔCANCELΔCOMMAND=		
with work-order-number of	work-order-number	Yes	
Copy a file	DDPΔCOPYΔFROM=		4.3.2
on a specific host	[{originating-host-id local-host-id} ::]	No	
with a file-id of	originating-file-id	Yes	
to a host	ΔTO=	Yes	
with a host-id of	[{destination-host-id local-host-id} ::]	No	
to a file with an id of	destination-file-id	Yes	
with an index that's being changed from the original	ΔKEY [_ {n 1}] = (size,location	No	
	[, {DUPLICATES NO_DUPLICATES}]	No	
	[, {CHANGE NO_CHANGE}])	No	

continued

Table B-1. Command Function Summary (cont.)

To do the following:	Use this command and these parameters:	Required?	See also:
Copy a file (cont.)			4.3.2
with a specification of how the copy is to be performed	[ΔMODE={DIRECT WAIT INDIRECT}]	No	
that should overwrite current file contents (SOF) or add to current file contents (EOF)	[ΔPOSITION={EOF SOF}]	No	
that needs translating to the host's code	[ΔTRANSLATE={EBCDIC NONE}]	No	
Create a file	DDPΔCREATEΔFILE=	Yes	4.3.1
on a specific host	[{host-id local-host-id}::]	No	
with a file-id of	file-id	Yes	
with a specific block size	[ΔBLOCK_SIZE={#-characters 256}]	No	
on a specific device	[ΔDEVICE_CLASS={DISK TAPE DISKETTE}]	No	
with a specific file type	[ΔFILE_TYPE={SEQUENTIAL INDEXED LIBRARY UNDEFINED}]	No	
with an initial size	[ΔINITIAL_SIZE={initial-number-blocks/ 1-9 digits 3 cyl}]	No	
that's either cataloged or not	[ΔREGISTER={VTOC CATALOG}]	No	

continued

Table B-1. Command Function Summary (cont.)

To do the following:	Use this command and these parameters:	Required?	See also:
Enter stream processing	DDP ENTER ERROR=	No	E.2
and continue or cancel if an error occurs	[{continue / cancel}]		
Purge a file	DDPΔPURGEΔFILE=		
on a specific host	[{host-id / local-host-id}::]	No	4.3.3
with a file-id of	file-id	Yes	
Find out the status	DDPΔSTATUSΔ		4.5.1
of a command you entered	COMMAND=work-order-number	Yes	
of a host	ΔHOST=host-id	No	
of a job you entered	ΔJOB=[{host-id / local-host-id}::] jobname	No	
of a file you control	ΔFILE=[{host-id / local-host-id}::] file-id	No	
of another user	ΔUSER=[{host-id / local-host-id}::] user-id	No	
Submit a job for execution	DDPΔSUBMITΔFILE=		4.4.1
from a specific host	[{originating-host-id / local-host-id}::]	No	
where the job is located in a specific file	file-id	Yes	
to a specific host	[ΔHOST={destination-host-id / local-host-id}]	No	
where you want job output printed	[ΔPRINT=host-id::device-id]	No	

continued

Table B-1. Command Function Summary (cont.)

To do the following:	Use this command and these parameters:	Required?	See also:
Send a statement	DDPΔSUBMITΔREQUEST=		4.4.3
where you make a specific request	'statement'	Yes	
to a specific host	[ΔHOST={host-id | local-host-id}]	No	
Send a message	DDPΔTALKΔMESSAGE=		4.5.2
where the message is	'string'	Yes	
to a user or operator	ΔUSER=	Yes	
located on a specific host	[{host-id | local-host-id}::]	No	
who is either the operator or a user with an id	{OPERATOR | user-id}	Yes	
where you want a reply	[ΔWAIT]	No	

Appendix C
Command Format Summary

```
DDPΔCANCELΔ ⎧ JOB= ⎡⎧host-id      ⎫::⎤ jobname ⎡ΔOUTPUT=⎧DISCARD⎫⎤ ⎫
            ⎨      ⎣⎩local host-id⎭  ⎦         ⎣        ⎩DELIVER⎭⎦ ⎬
            ⎩ COMMAND=work-order-number                            ⎭
```

```
DDPΔCOPYΔFROM= ⎡⎧originating-host-id⎫::⎤ originating-file-id
               ⎣⎩local-host-id      ⎭  ⎦

       ΔTO= ⎡⎧destination-host-id⎫::⎤ destination-file-id
            ⎣⎩local-host-id      ⎭  ⎦

       ⎡ΔKEY ⎡_⎧n⎫⎤ = (size,location ⎡,⎧DUPLICATES   ⎫⎤   ⎤
       ⎢     ⎣ ⎩1⎭⎦                  ⎣ ⎩NO_DUPLICATES⎭⎦   ⎥
       ⎢                                                  ⎥
       ⎢                             ⎡,⎧CHANGE   ⎫⎤ )     ⎥
       ⎣                             ⎣ ⎩NO_CHANGE⎭⎦       ⎦

       ⎡ΔMODE=⎧DIRECT  ⎫⎤
       ⎢      ⎨WAIT    ⎬⎥
       ⎣      ⎩INDIRECT⎭⎦

       ⎡ΔPOSITION=⎧EOF⎫⎤
       ⎣          ⎩SOF⎭⎦

       ⎡ΔTRANSLATE=⎧EBCDIC⎫⎤
       ⎣           ⎩NONE  ⎭⎦
```

```
DDPΔCREATEΔFILE=[{host-id      }::] file-id
                [{local-host-id}  ]

        [ΔBLOCK_SIZE={number-of-characters/1-9 digits}]
        [            {256                             }]

        [ΔDEVICE_CLASS={DISK    }]
        [              {TAPE    }]
        [              {DISKETTE}]

        [ΔFILE_TYPE={SEQUENTIAL}]
        [           {INDEXED   }]
        [           {LIBRARY   }]
        [           {UNDEFINED }]

        [ΔINITIAL_SIZE={initial-number-blocks/1-9 digits}]
        [              {3 cylinders                     }]

        [ΔREGISTER={VTOC   }]
        [          {CATALOG}]
```

```
DDPΔPURGEΔFILE=[{host-id      }::] file-id
               [{local-host-id}  ]
```

```
DDPΔSTATUSΔ{COMMAND=work-order-number                    }
           {HOST=host-id                                 }
           {                                             }
           {JOB=[{host-id      }::] jobname              }
           {    [{local-host-id}  ]                      }
           {                                             }
           {FILE=[{host-id      }::] file-id             }
           {     [{local-host-id}  ]                     }
           {                                             }
           {USER=[{host-id      }::] user-id             }
           {     [{local-host-id}  ]                     }
```

```
DDPΔSUBMITΔFILE=┌┌originating-host-id┐::┐file-id
                │{local-host-id      }  │
                └└                   ┘  ┘
            ┌                         ┐
            │ΔHOST=┌destination-host-id┐│
            │      {local-host-id      }│
            └      └                   ┘┘
            [ΔPRINT=host-id::device-id]
```

```
DDPΔSUBMITΔREQUEST='statement' ┌ΔHOST=┌host-id      ┐┐
                               │      {local-host-id }│
                               └      └              ┘┘
```

```
DDPΔTALKΔMESSAGE='string'ΔUSER=┌┌host-id      ┐::┐┌OPERATOR┐Δ[WAIT]
                               │{local-host-id}   │{user-id  }
                               └└             ┘   ┘└        ┘
```

Appendix D
Command Requirements

Certain DDP commands and parameters require the use of other commands or parameters. Table D-1 describes those requirements.

Table D-1. Command Requirements

If you specify:	You must also specify:/The following must be true:
DDPΔCANCELΔJOB=	a job name for a job that you submitted through the SUBMIT FILE command with the same user-id
COMMAND=	a work-order-id for a DDP command you submitted that you want to cancel. You must use the same user-id as when you issued the original command.
DDPΔCOPYΔFROM=	originating-file-id destination-file-id TO=
file-id that includes a nonsource module name	File-id must also include element type.
MODE=INDIRECT	File must be cataloged.
For additional restrictions, see Tables 4-3 and 4-4.	
DDPΔCREATEΔFILE=	file-id
BLOCK_SIZE=	either INITIAL_SIZE or INCREMENT_SIZE or both

DEVICE_CLASS=DISKETTE	BLOCK_SIZE=256 or less
	RECORD_SIZE=256 or less
	INITIAL_SIZE

REGISTER=VTOC	file-id with a volume serial number included for all subsequent commands referring to this file
DDPΔPURGEΔFILE=	file-id
DDPΔSUBMITΔFILE=	file-id
file-id that includes module name	File-id must also include element type.
DDPΔSUBMITΔREQUEST=	statement being submitted
DDPΔTALKΔMESSAGE=	string being sent USER=

Appendix E
Entering DDP Commands in a Batch Stream

E.1. Overview

You usually use DDP as an interactive product. But you can submit DDP commands as a remote batch job. The advantage of doing so is that you can perform remote tasks using these relatively simple commands during periods when you can't be at the terminal, such as overnight. The disadvantage is that you don't get your messages and output as promptly as you would if you were working interactively. Instead, all messages and output are printed at your local host for you to pick up.

E.2. DDP ENTER Command

The DDP ENTER command controls the processing of an enter stream.

Format

```
DDPΔENTERΔERROR= ┌┌continue┐┐
                 │{        }│
                 └└cancel  ┘┘
```

Parameters

continue
: To continue the processing of an enter stream when another DDP command terminates in error.

cancel
: To cancel the processing of an enter stream when another DDP command terminates in error. If any other errors occur when you specify this parameter, DDP reads the remaining unprocessed commands in the enter stream and displays them on the enter stream log.

All DDP commands in an enter stream are processed under the user-id specified in the LOGON command. You can run multiple enter streams.

When the DDP ENTER command is encountered in an enter stream, the following messages appear in the enter stream log:

```
DDP002 CAI 002 ENTER COMMAND ACCEPTED hh:mm:ss
DDP002 INT 002 ENTER COMMAND COMPLETED hh:mm:ss
```

If there are syntax errors in the command, the following message appears on the console, workstation, or DDP enter stream log following the syntax errors (see the *System Messages Reference Manual,* 7004 5190):

```
COMMAND REJECTED BECAUSE OF ERROR(S) LISTED ABOVE
```

You can only use the DDP ENTER command in an enter stream. If you enter the command from a console, workstation, or terminal, the following messages are displayed:

```
DDP002 CAI 002 ENTER COMMAND ACCEPTED hh:mm:ss
DDP103 INT 103 ENTER COMMAND REJECTED hh:mm:ss
```

E.3. Card Format

Table E-1 shows the format for the cards required to enter your DDP task as a batch job.

Table E-1. Entering DDP Commands in a Batch Stream

Card Format	Explanation
// DATA FILEID=name	name The name through which you reference this enter stream. It is one to eight alphanumeric characters long.
LOGON user-id,BU=$\begin{Bmatrix}\text{YES}\\ \text{NO}\end{Bmatrix}$	user-id The one to six alphanumeric characters identifying you as a user to the host. BU=$\begin{Bmatrix}\text{YES}\\ \text{NO}\end{Bmatrix}$ BU=YES prints the day's bulletin on your output. The default is NO. Although you can't change your DDP commands as a result of knowing the bulletin, knowing it may explain output errors.

continued

Table E-1. Entering DDP Commands in a Batch Stream (cont.)

Card Format	Explanation
DDP command cards	Type these on the cards exactly as you would type them in at your terminal. Because this is an enter stream, your DDP commands are executed in the order in which you list them. Each command is completed before the next is started. (This is different from terminal operation, where you can start your next DDP command before the previous one is complete.) Therefore, it's all right to list commands that depend on previous commands. For instance, you can first create a file and then copy to it. DDP won't attempt to copy before it creates the file. NOTE: Operations that proceed in a local-to-remote direction are done serially. However, operations in a remote-to-local, or remote-to-remote direction, are merely routed to the remote host for processing. As soon as routing is complete, the next command is executed. Care should be taken when performing these operations from an enter stream, since concurrent command execution may result.
LOGOFF	
// FIN	

E.4. Procedures

Follow these steps to enter your DDP commands as a remote batch job:

1. Place cards in the card reader.

2. At the system console, enter IN.

   A message appears on the system console indicating that the file you specified on your FILEID parameter in the // DATA statement has been created.

3. At the system console, enter:

```
ENTERΔQ=RDR,HOLD=⎡YES⎤,FIL=name
                 ⎣NO ⎦
```

   where:

```
HOLD=⎡YES⎤
     ⎣NO ⎦
```

   Indicates whether the enter stream is to be retained by the host (HOLD=YES) or deleted once the stream has been processed (HOLD=NO). The default is HOLD=NO.

name
: Is the reference name of the enter stream as indicated in your FILEID parameter in the // DATA statement.

Enter streams can also be created through the OS/3 editor and saved as source modules in a user library. They can be executed by typing the following command:

```
ENTER module-name,filename,vsn
```

For parameter specifications of the ENTER command, see the *General Editor (EDT) Operating Guide,* 7004 4599.

E.5. Sample Enter Stream

Here are the cards you'd use to enter the sample session from Appendix A in a remote batch enter stream. See Appendixes A and B for explanations of any commands you don't understand.

```
// DATA FILEID=ARCHER
LOGON JSMITH,BU=YES
DDP ENTER ERROR=CONTINUE
DDP TALK MESSAGE='PLEASE MOUNT VOL007' HOST=H001::OPERATOR
DDP CREATE FILE=H001::',REM/PERS(AXS9/AXSA7),VOL007' BLOCK_SIZE=182 RECORD_SIZE=91
DDP COPY FROM=',PERSNNEL(JMS8/)' TO=H001::',REM/PERS(/AXSA7)' MODE=WAIT
DDP SUBMIT FILE='PAY06,PAYJOBS(JMS17/),,S' HOST=H001
LOGOFF
// FIN
```

Appendix F
Logging Chart

All DDP local and remote activity is automatically logged in the interactive services log file during shutdown processing. The log printout is normally sent to the central site printer.

The following chart can be used to list your system-assigned job number and any other pertinent data from your displayed log for accounting purposes.

DDP LOG SHEET

TERMINAL: ______________

User-id	Date	Command	Time Accepted	Work Order Number	Jobname	Remote Host ID	Time Completed	Time Aborted

Appendix G
Generating Your ICAM Network with DDP

OS/3 DDP file transfer and file access facilities use ICAM's demand mode interface (DMI) and distributed communications architecture (DCA). Your host-ids appear in the label of the LOCAP macroinstructions. The REMOTE operands on the LPORT and LOCAP macroinstructions must specify OS3. To generate an ICAM network for the IMS DDP transaction facility, see the *IMS System Support Functions Programming Guide,* UP-11907.

Here is a sample ICAM network definition for the DDP file and job transfer and file access facilities. It's a very simple one, with just two hosts and one user connected to one of the hosts on a terminal. For further information on using ICAM and DMI, see the ICAM operations guide for your system.

Figure G-1 shows a definition for an ICAM network in a local system. A similar (mirror image) ICAM network would be required in the remote computer.

Note: *In a three-node network (i.e., initiator at one node, source file at a second, and destination at a third node), each node must have a communications link to every other node. For example:*

NODA must connect to NODB + NODC.
NODB must connect to NODA + NODC.
NODC must connect to NODA + NODB.

If these connections are not available, spooled output and messages for remotely submitted jobs are lost.

```
COMMCT

ACT1      CCA    TYPE=(GBL,,A),DCA=YES,GAWAKE=YES,CCAID=ACT1              X

                 FEATURES=(OPCOM)

          BUFFERS 24,256,2,ARP=36,UDUCT=(12,32,3),LINKPAK=(10,768,1)

LNE1      LINE   DEVICE=(UNISCOPE),TYPE=(9600,SYNC),ID=4

ATRM      TERM   FEATURES=(|U20 |,1920),ADDR=(21,51),     <----------- X --- This is your local
                                                                              terminal.
                 HIGH=MAIN,LOW=MAIN,MEDIUM=MAIN,INPUT=(YES)

|CUP1|    LOCAP  TYPE=(|DMI|)    <------------------------------------------- This is your local host.
                                                                              Here's where you specify
VLN1      VLINE  DEVICE=ABM,TYPE=(9600),ID=7,RSPADDR=3,                      DMI.  (Notice that the
                                                                              REMOTE= operand is not
                                                                              used.)
          LPORT  LINE=VLN1,REMOTE=OS3,PORT=1,EU1=CUP1,EU2=CUP2,          X

                 USERTP=DMI

BTRM      TERM   FEATURES=(U20,1920),ADDR=(21,51)                          X

                 HIGH=MAIN,LOW=MAIN,MEDIUM=MAIN,INPUT=(YES),REMOTE=B

                 TCTUPD=YES,REMOTE=(B)

|CUP2|    LOCAP  TYPE=(DMI),REMOTE=(|OS3|)      <---------------------------- This defines the remote
                                                                              OS/3 system.
          ENDCCA

          MCP

                 MCPNAME=C1

          CACH=(04,9600,SYNC)

          CACH=(07,9600,FULL,ILA)

END
```

Figure G-1. ICAM Network Definition in a Local System

UNISCOPE is a registered trademark of Unisys Corporation.

Appendix H
DDP Program-to-Program Macroinstructions Summary

Table H-1 is a summary of the program-to-program macroinstructions used with the OS/3 DDP file access facility.

Table H-1. DDP Program-to-Program Macroinstructions Summary

To do the following:	Issue this imperative:	Reference
Establish a communications path between primary IPC and destination IPC	DOPEN$\begin{Bmatrix}(1)\\ \text{cdibname}\end{Bmatrix},\begin{Bmatrix}(0)\\ \text{ribname}\end{Bmatrix}$ where: cdibname Address of the user's CDIB. (1) CDIB address is preloaded into register 1. ribname Address of user's RIB. (0) RIB address is preloaded into register 0.	6.4.4
Begin and end a conversation between paired programs	DMSEL$\begin{Bmatrix}(1)\\ \text{cdibname}\end{Bmatrix}$,PROG,$\begin{Bmatrix}(0)\\ \text{surrogate-prog}\end{Bmatrix},\begin{Bmatrix}\text{ACT}\\ \text{DEACT}\end{Bmatrix}$[,DATA] where: cdibname Address of user's CDIB. (1) CDIB address is preloaded into register 1. PROG Identifies the imperative belongs to the DDP file access facility.	6.5.3

continued

Table H-1. DDP Program-to-Program Macroinstructions Summary (cont.)

To do the following:	Issue this imperative:	Reference
Begin and end a conversation between paired programs (cont.)	surrogate-prog Name of the program with which conversation is to be initiated (used with ACT option only). (0) Assumes address of name of program is preloaded into register 0. ACT Indicates primary program conversation with surrogate program is to be activated. DEACT Indicates primary program conversation with surrogate program is to be deactivated (closed). DATA Indicates next imperative contains data to be passed with ACT or DEACT IPC function.	
Transfer data	$\left\{\begin{matrix}\text{DMOUT}\\ \text{DMINP}\end{matrix}\right\},\left\{\begin{matrix}\text{(1)}\\ \text{cdibname}\end{matrix}\right\},\left\{\begin{matrix}\text{(0)}\\ \text{workarea}\end{matrix}\right\}$[,UNLOCK] where: DMOUT Used to send data. DMINP Used to receive data. cdibname Address of user's CDIB. (1) CDIB address is preloaded into register 1. workarea Address of data buffer that can be fixed or variable according to the WKFM parameter description in the RIB. (0) Buffer address is preloaded into register 0. UNLOCK Indicates reply required from surrogate program (only issuable by primary program).	6.4.4

continued

Table H-1. DDP Program-to-Program Macroinstructions Summary (cont.)

To do the following:	Issue this imperative:	Reference
Pass conversation control from primary program to surrogate program	DMCTL $\left\{\begin{matrix}\text{(1)}\\ \text{cdibname}\end{matrix}\right\}$,PROG, $\left\{\begin{matrix}\text{BEQ}\\ \text{END}\end{matrix}\right\}$ where: cdibname Address of user's CDIB (1) CDIB address is preloaded into register 1. PROG Identifies the imperative belongs to the DDP file access facility. BEQ Indicates conversation control is to be passed to the surrogate program. END Indicates an end to the conversation (only a DMOUT imperative can follow; indicates a reply is expected by the primary program; usable only in TWA mode by the surrogate program).	6.5.3
Close the conversation	DCLOSE $\left\{\begin{matrix}\text{(1)}\\ \text{cdibname}\end{matrix}\right\}$ where: cdibname Address of user's CDIB. (1) CDIB address is preloaded into register 1.	6.4.4

Glossary

A

absolute code
Load code.

application program
An assembly language program, written by the user to converse or communicate with one or more communicating application programs on OS/3 remote hosts in a DDP environment. See Section 6.

B

BEQ
A parameter (of the DMCTL imperative) that indicates the primary program wishes to bequeath primary status to the surrogate program. The primary program becomes the surrogate program after the solicited response is received from the surrogate.

C

catalog
An online directory of information containing file names with their passwords that enables easy access to the files and also restricts files to certain users.

CDM
See consolidated data management.

centralized data processing (computer) system
A method of data processing in which a large processing unit accommodates the varied needs of many users. The centralized computer may have many nonintelligent terminals and peripherals.

command
An action performed on a computer.

command analyzer
The part of the DDP software that analyzes your commands and forwards them for processing.

compiled job
A series of expanded job control statements in a system library that can be executed immediately.

complex conversation
A two-way alternating communications discipline whereby primary and surrogate programs may send and receive data. The primary and surrogate programs may reverse statuses. A primary program can also communicate with more than one surrogate program.

consolidated data management
The name of the System 80 data management system.

conversation
An exchange of information between two or more communicating programs that is conducted via CDM program-to-program imperative macroinstructions. Conversations may be simple or complex.

D

data base
A collection of data fundamental to an organization.

data handling
The ability to access data on a remote host by using programs on the local host.

DDP
See distributed data processing system.

DDP command analyzer
See command analyzer.

DEACT
Parameter of the DMCTL imperative issued by the primary program to provide data to and solicit a response from the surrogate program with a request to terminate the conversation. The surrogate program may accept ((REPLY (END)) or reject (REPLY) the terminating request when sending the response message.

decentralized data processing (computer) system
A method whereby several independent computers exist within an organization but are not electronically connected.

defined record management
The facility of IMS to access a defined file.

demand mode interface
The ICAM system interface that supports both distributed data processing and all OS/3 interactive services.

destination file

In a DDP system, the file into which another file is being copied.

destination-host-id (with the DDP file access facility)

One to four alphanumeric characters naming the host to receive the data. If omitted, DDP assumes the destination file is on your local host.

destination (with the DDP file access facility)

An extension of the user-id node used in the DDP file access facility, with the format:

```
destination::=[host-id:]user-id
```

Host-id is one to four alphanumeric characters identifying the computer on which you wish your job to be executed. The host-id is optional and defaults to the local host, i.e., the host computer on which the job is executing. The generic host-id, $HOST, specifies that the host-id is the originator or master.

display messages

Information about a job being executed, displayed on a console or terminal screen.

distributed communications architecture (DCA)

The Unisys architecture that draws together all aspects of the communication products by defining a set of logical concepts and a set of rules (protocols and interfaces) and guidelines to be used in applying the concepts in the design of hardware, software, and network products.

distributed data processing system

A configuration of independent computers joined in a peer relationship.

DMI

See demand mode interface.

E

element

In OS/3, a discrete part of a module in a program library file. In DDP commands, however, an element is a module.

element type

The kind of code that elements are written in, such as symbolic (source code) or absolute (load code). *See also* element.

encoding

The converting of data through use of a code so that it can be reconverted to its original form.

F

file-id

The complete name by which a file is identified. In DDP, the file-id must include the file name and may also include the module name, read and write passwords, volume, and element type, in this format:

```
'[module-name],filename[([read-passwrd]/write-passwrd])],[vol],[element]'
```

file name

From 1 to 44 alphanumeric characters identifying your file when it resides on disk and 1 to 17 alphanumeric characters for tape files, data set label diskettes, and logical files (spool files).

H

help screens

A display of how information and additional help can be obtained for the DDP commands.

hierarchical relationship

An arrangement of hosts so that there is a different rank among them.

host

An independent computer attached to a DDP system.

host-id

The identifying name of your host. In DDP, it is the same as the node-id you specify in the REMOTE parameter of your LOCAP macroinstruction in your ICAM network definition. If host-id is omitted, the local host is assumed. See 3.1.2 and the individual commands and imperatives where host-id is used for particular specifications. Also, see the variations: destination, local, and originating host-ids.

I

ICAM

Integrated communications access method. A generalized software package for OS/3 communications that gives you multiple levels of interface to remote devices.

IMS DDP transaction facility (IMS DDP)

A Unisys IMS DDP transaction facility, which enables workstation and terminal operators to process an IMS transaction at a remote system. Multithread IMS systems are required at both the local and remote sites. IMS can route a transaction to a remote system by:

- transaction directory routing
- transaction program routing
- transaction operator routing

indexed file
A file accessed according to one or more index keys (for example, IRAM or MIRAM files).

information management system (IMS)
A software product facilitating the development and installation of online, transaction-oriented data base management applications under OS/3.

interactive services
A collection of commands and system programs that enables you to prepare, run, and control jobs and to perform system housekeeping routines easily and quickly from a workstation.

J

jobname
One to eight alphanumeric characters naming the job that the DDP SUBMIT FILE command returned to you.

jproc
A series of job control statements stored in a library that can be executed by a simple job control statement. (Same as proc or procedure.)

L

library file
A file consisting of a collection of program modules.

local host-id
One to four alphanumeric characters identifying the computer you are using at your site. See the command using the local host-id for particular specifications.

LOCAP (LOCal APplication) file
The file within ICAM that is necessary to permit program-to-program transfer, whether local or remote, and that supplies the name and type of program that other applications may access.

M

macro or macroinstruction
A source statement in an assembler program that is converted into many assembler source statements to do a particular function (for example, OPEN or CLOSE).

master-slave relationship
A communications connection in which one computer completely controls the data sending and receiving functions.

message
Consists of some arbitrary amount of information whose beginning and end are defined or implied and that is transmitted from one program to another program in a DDP network.

module
A part of a library file that is accessible by name as a unit.

N

network
The total collection of physical entities joined in a data communications system (nodes, network processors, terminals, etc).

nine-thousand remote (NTR)
A software utility program that controls data transmission to allow the Unisys System 80 to be used as a terminal to a Unisys Series 1100 system.

nonencoded characters
Standard codes (such as EBCDIC or ASCII) made up of bit patterns. There are more possible bit patterns than are actually used in these codes. Some users alter their computers to recognize patterns that are not part of the standard code. These unique bit patterns are called nonencoded characters.

NTR
See nine-thousand remote.

O

originating file
In DDP, the file from which a copy is being made.

originating host-id
Is one to four alphanumeric characters naming the originating host. Used with the DDP SUBMIT FILE command, it names the job control stream file location. If HOST-ID is omitted, the system assumes the file is on your local host.

P

password

One to six alphanumeric characters you must specify to read a file (read-passwrd) or write to a file (write-passwrd). Passwords are optional parts of the file-id. They are used only with cataloged files.

primary host

The originating host in a DDP system.

primary program

The initiating part of a pair of application programs in the program-to-program component of the DDP file access facility. *See* surrogate program for the other part of the communicating pair.

procedure/proc

See jproc.

program-to-program communication

A component of the DDP file access facility whereby you can write a BAL program that can initiate and carry on a conversation with another BAL surrogate program at a remote OS/3 host. The communicating programs must reside on different hosts in the DDP network.

protocol

A set of rules defining the structure, content, sequencing procedures, and error detection and recovery techniques for the transmission of data. Also used to establish, maintain, and control communications between two corresponding levels in a level hierarchy. Normally implies the sending and receiving of unique command and response messages or message headers.

R

read password

See password.

real time (during a DDP activity)

Pertaining to the actual time during which a DDP process transpires and during which results can be used in guiding the DDP process.

relative file

A file you may access either record by record or by the relative number of the logical record within the file.

relocatable code

Object code.

remote file processing
A function of the DDP file access facility whereby you can access and process disk files residing on remote OS/3 systems in your DDP network.

remote host
The computer geographically separated from you to which you are electronically connected.

ring structure
The joining of hosts in a circle so that each is connected to two others.

S

screen format
A form displayed on a video terminal used to input data to a program or to output data from a program.

secondary host
The destination host in a DDP system.

sequential file
A file you access record by record, according to the order of the records in the file.

simple conversation
A one-way-only communications discipline whereby the primary program is only allowed to send and the surrogate program is only allowed to receive. Only one primary program and one surrogate program are involved, and that relationship remains for the duration of the conversation.

spooling
The process by which the central processor reads and writes records to or from a high-speed storage device rather than a slower device.

stand-alone processing
The ability of a computer to process data with no connection to any other computer.

star structure
The joining of hosts so all hosts are connected to one central host.

surrogate program
A surrogate user application program that is one part of a pair of programs used with the DDP file access facility. *See* primary program for the other part of the program pair. However, the two programs can exchange statuses via the BEQueath parameter of the DMCTL imperative.

symbolic code
Source code.

system
See distributed data processing system.

T

terminal
A point in a network where data can either enter or leave. Normally operator oriented in that it provides for human interpretable input/output media.

transaction directory routing
A routing method of the IMS DDP transaction processor facility whereby the terminal operator routes a transaction via a transaction code that identifies a transaction at a particular remote system. *See* transaction operator routing and transaction program routing.

transaction operator routing
A routing method of the IMS DDP processor transaction facility whereby the terminal operator routes a transaction via a transaction code with a special character that routes the transaction to a particular remote system. *See* transaction directory routing and transaction program routing.

transaction processing
The use of an information management system to request information or to change records, and to receive a response. Each transaction involves one input request from a terminal followed by one output response from a host. DDP transaction processing involves the use of information management systems across two or more hosts.

transaction program routing
A routing method of the IMS DDP transaction processor facility whereby the terminal operator initiates a transaction at the local IMS system. The COBOL or basic assembly language action program sends a message that initiates a transaction at the remote system. *See* transaction directory routing and transaction operator routing.

tree structure
The joining of hosts so that each is connected to at least one other host but may in addition be connected to others.

two-way alternating communication
A DDP file access facility communications discipline between pairs that provides a request/response exchange to coordinate data transfer. A sense of primary and surrogate program is maintained. *See* complex conversation.

U

undefined file
Any type of file except a library file containing source (symbolic) code.

UNIQUE **(uniform inquiry update element)**
An easy-to-use inquiry language that lets you display data and update your files by entering commands from the terminal. A set of IMS-supplied action programs processes these UNIQUE commands.

user-id
The one to six alphanumeric characters identifying you as a user to the host.

V

volume table of contents (VTOC)
A directory written on your disk volume that lists the addresses and other information about the files on that volume.

W

work order number
A reference number assigned to each command entered into the DDP software.

work order request
Any command you enter into the DDP software. Some messages use this term when they're referring to the command you entered immediately preceding this message.

workstation
Any terminal, consisting of a CRT display and typewriter-like keyboard, that you use to access the interactive services.

write password
See password.

Index