SPERRY UNIVAC
COMPUTER SYSTEMS

PUBLICATIONS RELEASE

Operating System/3 (OS/3)

Information Management System (IMS)

Concepts and Facilities

UP-9205

This Library Memo announces the release and availability of "SPERRY UNIVAC® Operating System/3 (OS/3) Information Management System (IMS) Concepts and Facilities", UP-9205.

The Information Management System (IMS) Concepts and Facilities manual is one of five books replacing the IMS 90 Applications User Guide/Programmer Reference, UP-8614, Rev. 1. Other manuals replacing UP-8614 are:

- IMS Action programming in RPG II User Guide, UP-9206
- IMS Action Programming in COBOL and Basic Assembly Language (BAL) User Guide, UP-9207
- IMS Terminal Users Guide, UP-9208
- IMS Data Definition and UNIQUE User Guide, UP-9209

This manual explains the fundamental concepts of IMS and how it works. It is presented in thirteen sections as follows:

Section 1.	IMS Overview
Section 2.	IMS Environment
Section 3.	Action Programming — The Heart of IMS Programming
Section 4.	Generating an Online IMS
Section 5.	Beginning and Ending IMS Sessions
Section 6.	Terminal Operations and Input/Output Message Processing
Section 7.	File Activity With IMS
Section 8.	Additional Features Available to Action Programs
Section 9.	IMS and Distributed Data Processing
Section 10.	IMS and DMS Together

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists BZ, CZ, and MZ	Mailing Lists A00, A07, A08, B00, B07, 18, 18U, 19, 19U, 20, 20U, 21, 21U, 28U, 29U, 75, 75U, 76, and 76U (Cover and 225 pages)	Library Memo for UP-9205
		RELEASE DATE:
		September, 1982

Information Management System (IMS)

Concepts and Facilities

OS/3

H

SPERRY UNIVAC

UP-9205

This document contains the latest information available at the time of preparation. Therefore, it may contain descriptions of functions not implemented at manual distribution time. To ensure that you have the latest information regarding levels of implementation and functional availability, please consult the appropriate release documentation or contact your local Sperry Univac representative.

Sperry Univac reserves the right to modify or revise the content of this document. No contractual obligation by Sperry Univac regarding level, scope, or timing of functional implementation is either expressed or implied in this document. It is further understood that in consideration of the receipt or purchase of this document, the recipient or purchaser agrees not to reproduce or copy it by any means whatsoever, nor to permit such action by others, for any purpose without prior written permission from Sperry Univac.

Sperry Univac is a division of the Sperry Corporation.

This document was prepared by Systems Publications using the SPERRY UNIVAC UTS 400 Text Editor. It was printed and distributed by the Customer Information Distribution Center (CIDC), 555 Henderson Rd., King of Prussia, Pa., 19406.

PRINTED IN U.S.A.

PAGE STATUS SUMMARY

ISSUE: UP-9205

Part/Section	Page Number	Update Level
Cover/Disclaimer		
PSS	1	
Acknowledgment	1	
Preface	1 thru 4	
Contents	1 thru 7	
1	1 thru 19	
2	1 thru 7	
3	1 thru 34	
4	1 thru 18	
5	1 thru 3	
6	1 thru 22	
7	1 thru 14	
8	1 thru 30	
9	1 thru 8	
10	1 thru 5	
11	1 thru 7	
12	1 thru 15	
13	1 thru 5	
Glossary	1 thru 10	
Index	1 thru 13	
User Comment Sheet		

All the technical changes are denoted by an arrow (→) in the margin. A downward pointing arrow (↓) next to a line indicates that technical changes begin at this line and continue until an upward pointing arrow (↑) is found. A horizontal arrow (→) pointing to a line indicates a technical change in only that line. A horizontal arrow located between two consecutive lines indicates technical changes in both lines or deletions.

Acknowledgment

We are indebted to the many systems analysts and staff members of Sperry Univac branch offices and customer organizations who helped us develop the OS/3 IMS library. They gave us suggestions, answered numerous questions, reviewed the manuals, and provided us with ''real-life'' programming examples. The customer organizations assisting us include:

- Gay and Taylor Insurance Adjustors, Winston-Salem, NC
- Penn Ventilator Company, Philadelphia, PA
- Victor Valley Community College District, Victorville, CA

The Sperry Univac organizations assisting us include:

- Los Angeles Access Center, Customer Support Services, Los Angeles, CA
- Charlotte Commercial Branch, Raleigh Office, Raleigh, NC
- Minneapolis Marketing Branch, Minneapolis, MN
- Wellesley General Branch, Wellesley, MA
- Philadelphia Manufacturing Branch, Wayne, PA
- Des Moines Marketing Branch, West Des Moines, IA
- System 80 Benchmark and Demonstration Services, Blue Bell, PA

Preface

This manual is one of a series designed to instruct and guide you in using the SPERRY UNIVAC Information Management System (IMS) for Operating System/3 (OS/3). It gives an overview of the facilities provided by IMS and how to use them. IMS Concepts and Facilities should be read before the other manuals in the series.

This manual is divided into 13 sections, a glossary, and an index. The topics discussed are:

- Section 1. IMS Overview

  Introduces basic IMS concepts such as transaction processing; action programming; action programs; single-thread and multithread environments; and pre-online, online, and offline processing. Briefly discusses requirements for generating IMS, uses of IMS, and IMS documentation.

- Section 2. IMS Environment

  Explains how messages are processed and describes the interface between action programs, IMS, ICAM, and data management.

- Section 3. Action Programming: The Heart of IMS Processing

  Discusses the purpose of user-written action programs, their design, the action program interface with IMS, and how action programs process transactions.

- Section 4. Generating an Online IMS

  Discusses all phases of preparation for online processing, including system generation, creating a communications network to support IMS, pre-online processing, and IMS configuration.

- Section 5. Beginning and Ending IMS Sessions

  Describes IMS start-up and termination procedures.

- Section 6. Terminal Operations and I/O Message Processing

  Describes the types of terminals supported by IMS, input and output messages, master and standard terminal commands, user and IMS-supplied transaction codes, and transaction processing from the system console.

- Section 7. File Activity with IMS

  Describes action program file processing, types of data files supported, record locking, IMS internal files, and file recovery.

- Section 8. Additional Features Available to Action Programs

  Lists and describes special features provided by IMS including continuous output, output-for-input queueing, line disconnect, snapshot dumps, UTS downline loading, screen format services, edit tables, and initiating an OS/3 job.

- Section 9. IMS and Distributed Data Processing

  Discusses types of remote transaction routing, configuration requirements, programming for distributed data processing, and how to process remote transactions at the terminal.

- Section 10. IMS Access to DMS Data Bases

  Describes the IMS/DMS interface including COBOL/DML action programs, data definition using a data base as a source, and configuration considerations.

- Section 11. Batch Processing of Transactions

  Discusses online and offline batch transaction processing, preparation for batch processing, input to the batch processor, starting up IMS for batch processing, and control of batch processing.

- Section 12. Defined Files and Data Definitions

  Compares conventional and defined files and describes how to create and access defined files.

- Section 13. The IMS File Processing Package – UNIQUE

  Briefly describes how to use UNIQUE, the UNIQUE commands, and a sample UNIQUE application.

- Glossary

  Defines IMS terms used in this manual and throughout the IMS library.

As one of a series, this manual is designed to guide you in programming and using the OS/3 information management system. Depending on your need, you should also refer to the current version of other manuals in the series. Complete manual names, their ordering numbers, and a general description of their contents and use are as follows:

- UP-8364

  Information management system (IMS) system support functions user guide

  Describes the procedures to generate, initiate, and recover an online IMS system.

- UP-9206

  Information management system (IMS) action programming in RPG II user guide

  Describes how to write action programs in RPG II, with extensive examples.

- UP-9207

  Information management system (IMS) action programming in COBOL and basic assembly language (BAL) user guide

  Describes how to write action programs in COBOL and BAL, with extensive examples.

- UP-9208

  Information management system (IMS) terminal users guide

  Describes terminal operating procedures, standard and master terminal commands, and special purpose IMS transaction codes. Also includes UNIQUE command formats with brief descriptions. The manual is in easel format for ease of use at the terminal.

- UP-9209

  Information management system (IMS) data definition and UNIQUE user guide

  Describes how to create defined files for use with UNIQUE or your action programs and explains how to use UNIQUE. Includes extensive examples of data definitions and UNIQUE dialogs.

- UP-8748

  IMS/DMS interface user guide

  Describes how to access a data base management system (DMS) data base from IMS.

Contents

2. IMS ENVIRONMENT

3. ACTION PROGRAMMING: THE HEART OF IMS PROCESSING

4. GENERATING AN ONLINE IMS

11. BATCH PROCESSING OF TRANSACTIONS

12. DEFINED FILES AND DATA DEFINITIONS

13. THE IMS FILE PROCESSING PACKAGE – UNIQUE

GLOSSARY

INDEX

FIGURES

TABLES

1. IMS Overview

1.1. INTRODUCTION TO INFORMATION MANAGEMENT

What is information management and what does it do? One way to understand this is to break it down into two words: **information** and **management.**

Definition of information

Information can mean many things. To you, it may mean information concerning your business – personnel, payroll, inventory, or customers – that you have on data files. Once these facts are on data files, you must be able to access and manipulate them. That's where management comes in.

Definition of management

Management allows you to easily access, add, change, and delete data (facts) from your files so that the most up-to-date information about your business is right at your fingertips. And, because individual needs are never exactly alike, you must be able to tailor this management to suit your needs.

That's where the SPERRY UNIVAC Information Management System (IMS) comes in.

Characteristics of IMS

What does IMS do for you?

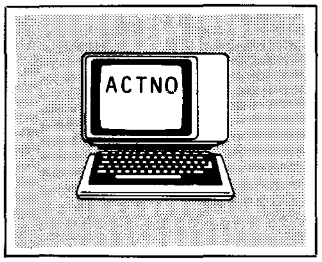
▷ It's interactive. You carry on a conversation with IMS through a terminal.

▷ It's versatile. You tailor IMS to suit your applications.

▷ It's a transaction processing system. Every time you enter an input message, you receive a response (output message). By processing transactions, IMS performs the task you want it to do.

▷ It works with existing data files. They can be in two forms: conventional files or defined files.

File function



To perform a file function with IMS, just key in a short message at your console or workstation. The system sends an appropriate response based on the nature of your message. Thus, you have quick and easy access to your files without complex programming or the long wait of batch processing.

1.2. HOW IMS REALLY WORKS

In a very broad sense, this is what happens...,

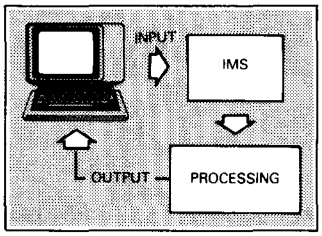
Transaction code

Action programs

When you key in a message beginning with a **transaction code,** you're actually accessing one of many different programs, called **action programs,** that are known to IMS. How does IMS know which one you want? By the transaction code.

Processing and output

All transaction codes are associated with an action program. Based on the transaction code you key in, IMS knows what action program to access. Then, the action program is processed and you get some kind of response (output). The output, of course, depends on the kind of action program requested.



Definition of action

This entire process, from keying in a message to the completion of the programming function, is called an **action.**

An action is the basic unit of work in IMS.

Definition of transaction

Figure 1-1 illustrates the action process. Since the action is the basic unit of work, everything in IMS revolves around it. Because IMS is a transaction processing system, one or more actions comprise a **transaction.**

Simple Transaction

Simple transaction

In some cases, one action can accomplish all the work you want done. When this is true, that action is called a **simple transaction.** The action shown in Figure 1-1 is an example of a simple transaction.

The terminal operator keys in the transaction code BNKACT, a space, and an account number. The appropriate action program is accessed and, when it processes, it produces an output message that gives the account number and balance for the designated account.

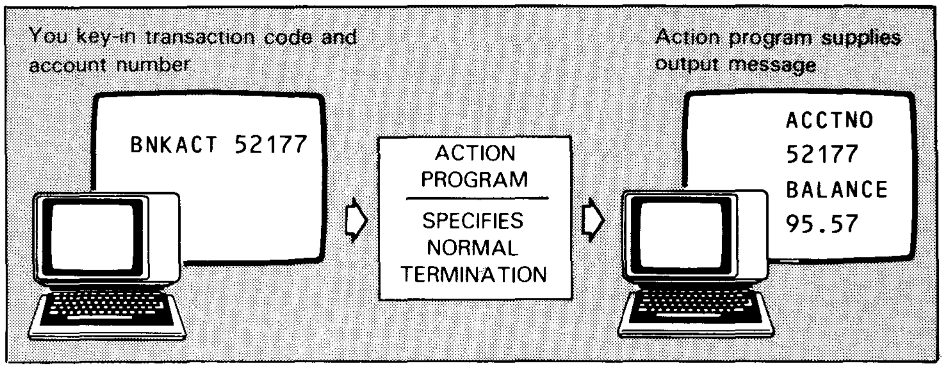

You key-in transaction code and account number
BNKACT 52177
ACTION PROGRAM
SPECIFIES NORMAL TERMINATION
Action program supplies output message
ACCTNO
52177
BALANCE
95.57

Figure 1-1. What is an Action?

Dialog Transaction

Dialog transaction

In most cases, you need a sequence of two or more actions to complete a desired function. When this is true, the sequence of actions is called a **dialog transaction** (Figure 1-2).

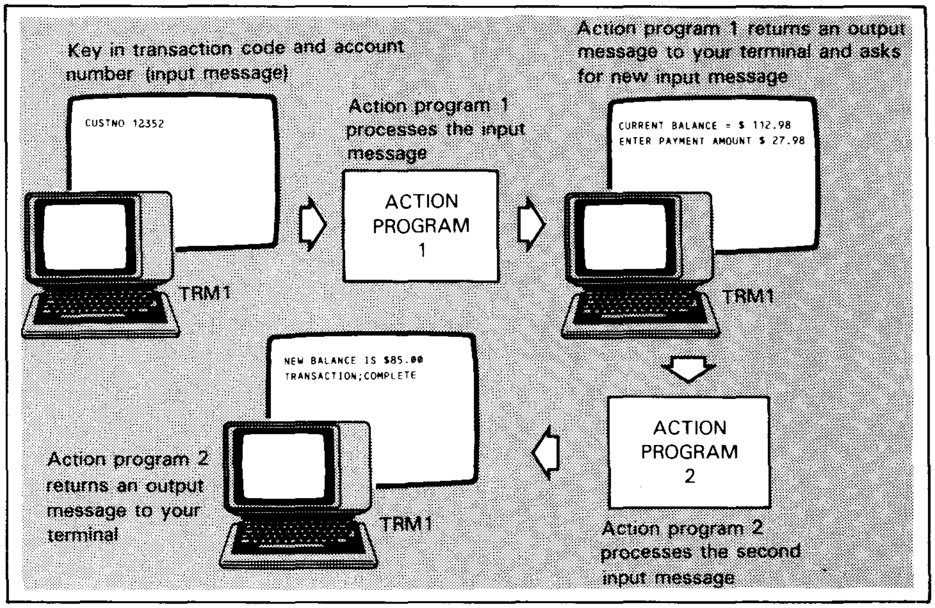

Key in transaction code and account number (input message)
CUSTNO 12352
TRM1
Action program 1 processes the input message
ACTION PROGRAM 1
Action program 1 returns an output message to your terminal and asks for new input message
CURRENT BALANCE = $ 112.98
ENTER PAYMENT AMOUNT $ 27.98
TRM1
ACTION PROGRAM 2
Action program 2 processes the second input message
NEW BALANCE IS $85.00
TRANSACTION;COMPLETE
TRM1
Action program 2 returns an output message to your terminal

Figure 1-2. Example of a Dialog Transaction

The terminal operator keys in the transaction code CUSTNO and a customer account number. The appropriate action program is accessed and, when it processes, it supplies the output message giving the operator the current balance for that customer's account.

Action program 1 then asks for a new input message in the form of a payment amount for that customer. A second action program is accessed, and when it processes, it gives the operator a new balance for that customer's account and indicates that the transaction is complete.

Note that in a dialog transaction, the terminal operator does not key in another transaction code to access the second action program. The first action program tells IMS the name of the action program that will process the second message.

Action Programs

Now that you know about the two types of transactions (simple and dialog) and how to use them, let's look at action programs.

Function of action programs

Action programs are appropriately named because each program usually performs an action. Sometimes two or more programs perform an action. Action programs work with IMS to enable you to **access, retrieve, add, change,** and **delete** data from your files. There are two types of action programs:

Types of action program

1. Those you write yourself
2. Those supplied by IMS

You write action programs in three languages

You can write your own action programs in one of three computer languages:

1. COBOL
2. Basic assembly language (BAL)
3. Report program generator II (RPG II)

Action programs are similar to standard programs in these languages, but you must follow some special rules because action programs operate under the control of IMS.

IMS action programs

You can use a set of action programs provided by IMS, called the UNiform InQuiry Update Element (UNIQUE). These programs perform a variety of file retrieval and updating functions and you access them by special UNIQUE commands. UNIQUE is somewhat limited in scope; so, for most applications, you'll probably want to write your own action programs.

We tell you more about writing your own action programs and using UNIQUE in Sections 3 and 13.

Now that you know a little about IMS, let's look at some of the characteristics and common uses of IMS.

1.3. TYPES OF IMS ENVIRONMENT

There are two types of IMS environment:

1. Single-thread
2. Multithread

Which type to choose

You decide which type you need based on your processing requirements and the amount of main storage you have.

Single-thread IMS

Single-thread system

With single-thread IMS, only one action is processed at a time, but actions from different transactions may be interspersed.

Example: single thread

In Figure 1-3, if terminal A is using action program 1 to process an account, and terminal B wants to use action program 1, terminal B must wait until terminal A is finished before it can access action program 1 or any other action program. If terminal A comes back with another input message while terminal B is still using action program 1, terminal A must wait.

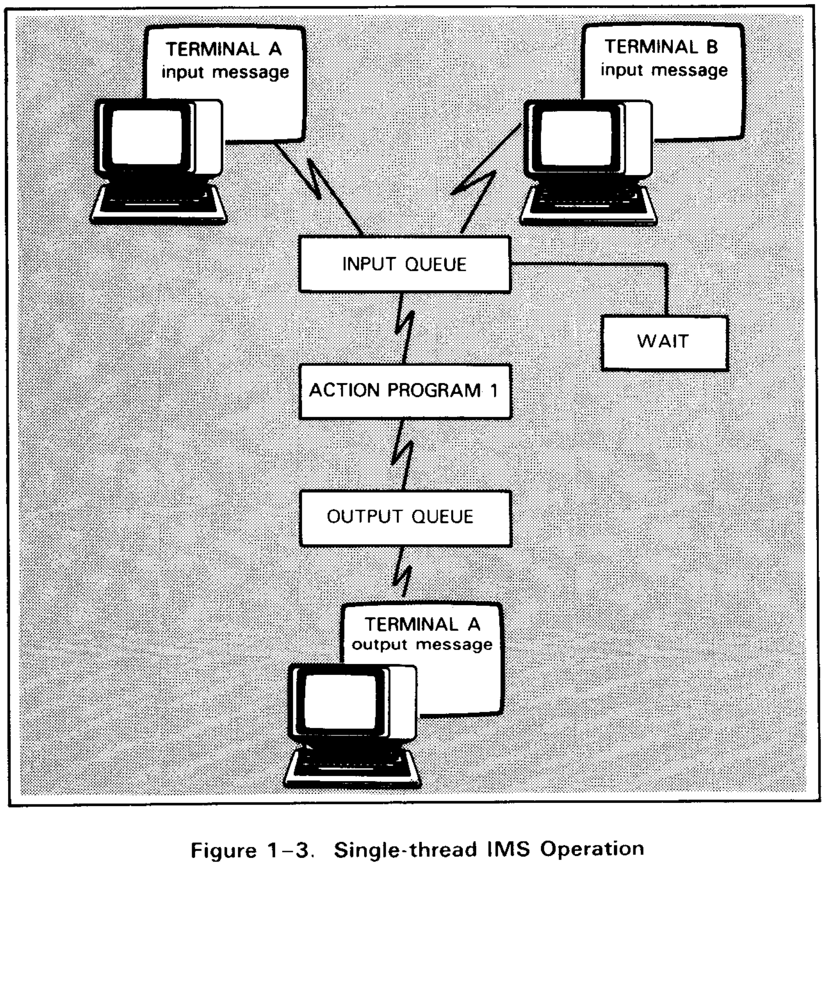


Figure 1-3. Single-thread IMS Operation

Multithread IMS

Because actions are usually short, IMS can process transactions from several terminals with little delay in response time.

Multithread system

When using multithread IMS, actions are processed concurrently for different transactions.

Example: multithread

In Figure 1-4, terminals A and B each send an input message to action program 1. Terminal C sends an input message to action program 2. The input messages go into a queue which operates on a FIFO (first in, first out) basis. Whichever message gets there first is put into the queue and processed first.

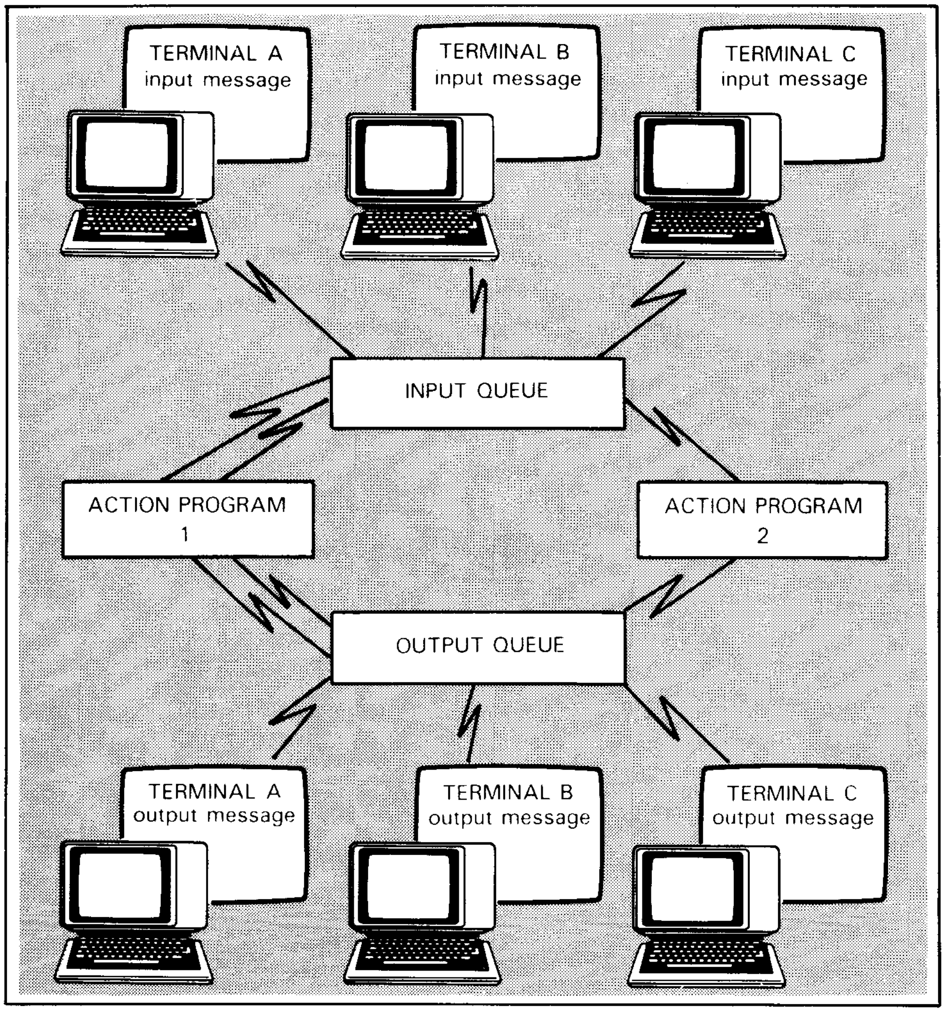


Figure 1-4. Multithread IMS Operation

Assume input message A in Figure 1–4 reaches the input queue first. IMS begins processing action program 1. When input message B, which also wants to use action program 1, reaches the queue, IMS either allows A and B to share program 1 (if program 1 is sharable or reentrant) or it waits until program 1 terminates and then schedules B to use it.

The input message from terminal C wants action program 2, and if enough main storage is available, terminal C's input message is processed immediately.

Now that you know the types of IMS environment, let's look at what you need to generate IMS.

1.4. IMS SOFTWARE PACKAGE

Software breakdown

The IMS software package you receive contains many components. Not all of them are required, but because you can tailor IMS to your needs, you may want to include some of them. To understand how some of these components fit together, let's break IMS down into three processing time frames:

Processing time frames

1. Pre-online
2. Online
3. Offline

Prepare IMS

Pre-online processing prepares and configures IMS to define your online IMS environment.

Use IMS

Online processing consists of operations that process transactions interactively. Most of these operations are transparent to the applications programmer because they are performed within IMS.

Recover damaged files

Offline processing consists of operations you perform after an IMS session. These operations include file recovery and printing of statistics gathered during the online session. Certain components of online processing make offline recovery possible.

Form of IMS components

The pre-online and offline components are utility programs (load modules) that are ready to use. The online components are stored as source and object modules that you configure into an IMS tailored to your needs.

1.5. PRE-ONLINE PROCESSING

There are four components that function under pre-online processing:

Pre-online components

1. Configurator
2. Named record (NAMEREC) utility
3. Data definition processor
4. Edit table generator

The most important part of pre-online processing is the configuration process.

Configuration process creates cutom tailored online IMS system

The configurator (Figure 1-5), through the job control procedure IMSCONF, enables you to create an online IMS that performs the functions you want. IMSCONF generates and executes an IMS configurator program, produces and links an IMS load module that becomes your online IMS system, and stores it in the designated load library.

Creation and contents of named record file

The configurator also writes tables into the named record (NAMEREC) file. These tables identify transaction codes, action programs, and the files related to those transaction codes. This configuration process tailors IMS. You supply the characteristics of your system in the different configurator input sections. We tell you more about configuring IMS in Section 4.

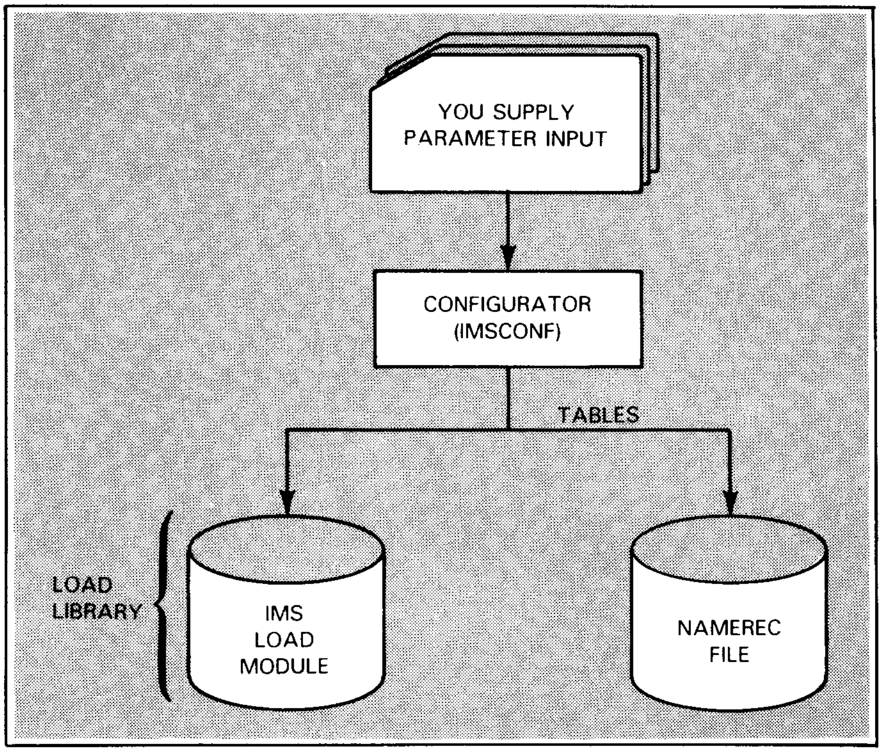


Figure 1-5. Configurator Output

NAMEREC utility initializes named record file

While the NAMEREC utility has several functions, its main purpose is to establish the named record (NAMEREC) file (Figure 1-6). The named record file is an internal IMS file that contains all tables and records needed by online IMS. These tables include configuration tables, data definition records, edit tables, and password definition tables.

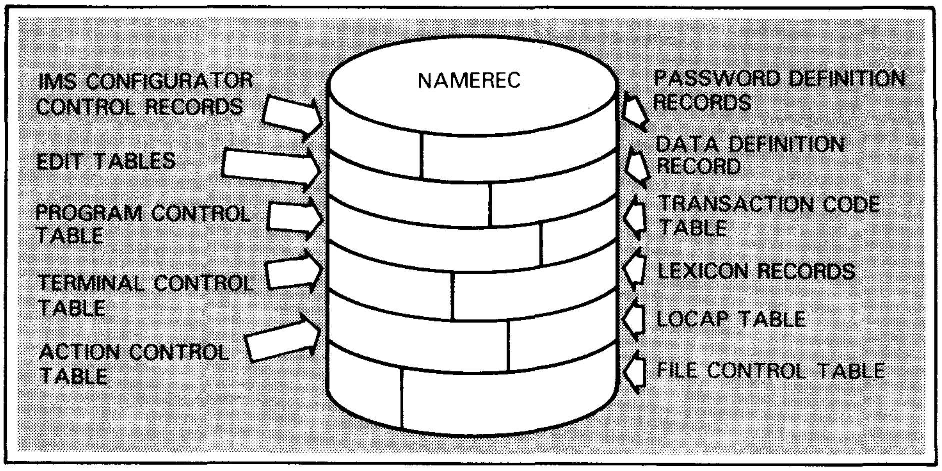


Figure 1-6. NAMEREC File

Data definition processor creates defined files

The data definition processor enables you to create a defined files. Defined files are built from conventional files. They provide a means of creating a file for a particular application from existing data without having to physically create the file. Figure 1-7 shows how the data definition processor works.

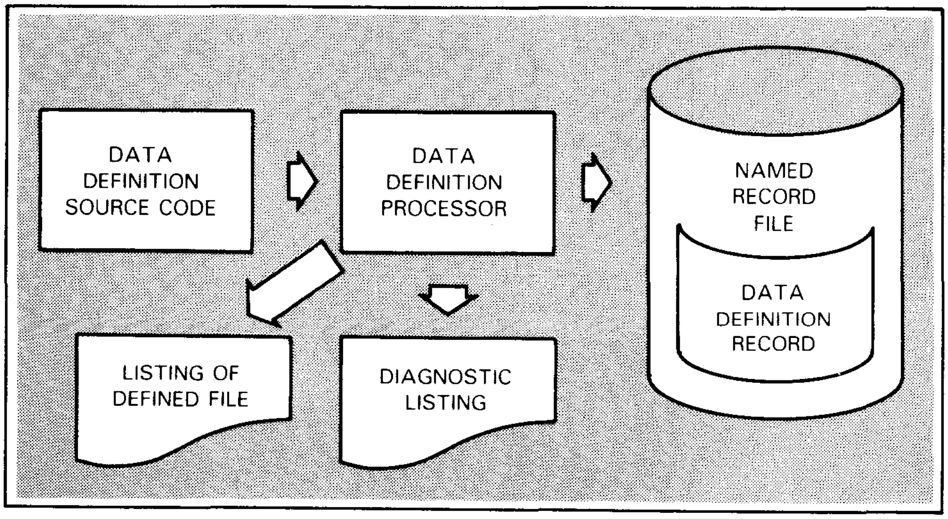


Figure 1-7. Data Definition Processing

The data definition processor writes your data definitions into the NAMEREC file and produces a printed listing of the defined file as well as a diagnostic listing.

Edit table generator formats input

The edit table generator (Figure 1-8) allows you to convert freeform input received from terminals into the fixed formats required by action programs. By using assorted parameters, you define edit tables (stored on NAMEREC as shown in Figure 1-8). These tables contain the characteristics of each field in your expected input. Based on these tables, IMS converts the input into usable form and checks it for types of data, value ranges, and the presence of required fields.

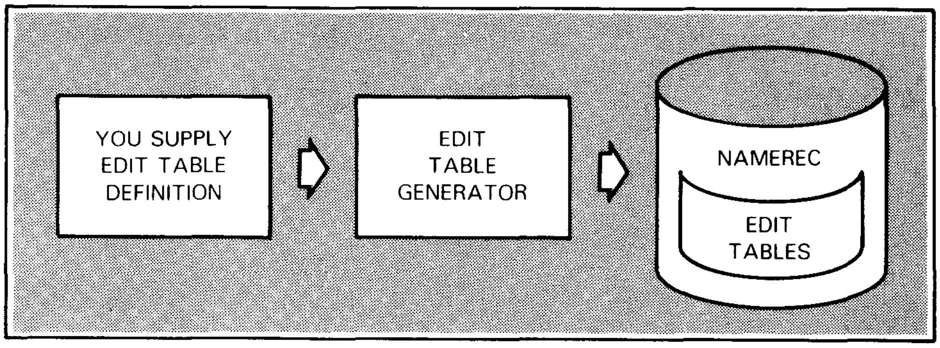


Figure 1-8. Edit Table Generator

1.6. ONLINE PROCESSING

Online components

Online IMS has five main components:

1. Start-up modules
2. Applications management
3. Internal message control
4. Shutdown modules
5. Action program component

Function of components

These five components are transparent to the applications programmer and function internally to process transactions interactively (Figure 1-9). We mention them here because, based on parameters you designate at configuration time, you have some control over how they process transactions. As a result of configuration, these components become part of the IMS load module.

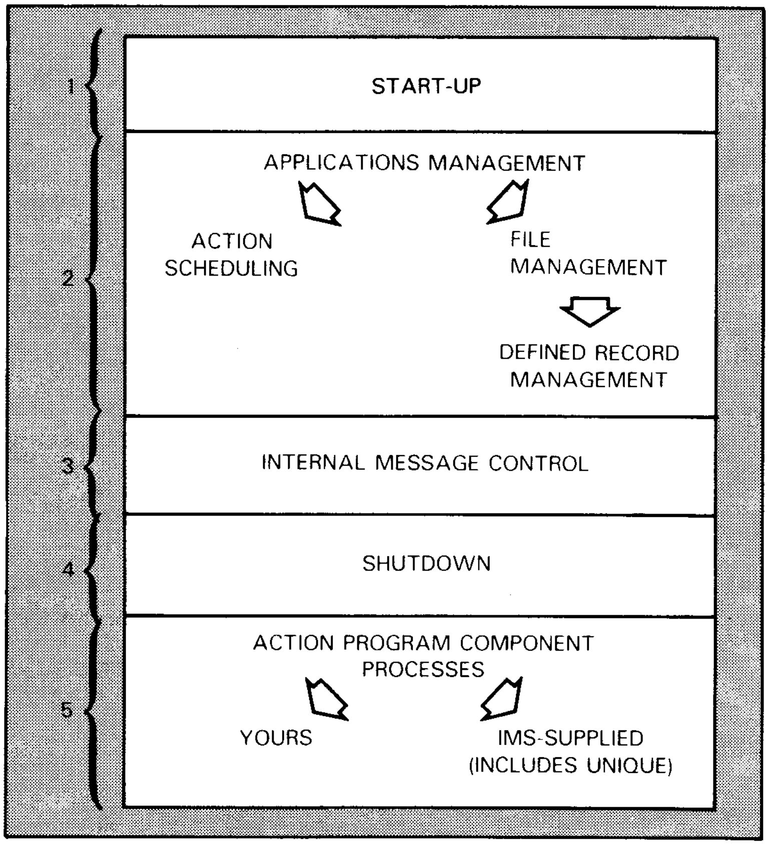


Figure 1-9. IMS Load Module

Start-up modules initialize online functions

Start-up modules initialize functions needed for online operations. This process is transparent to you but you have some control over it by using optional start-up parameters when you execute the IMS load module. We tell you more about this in Section 4.

Applications management has two parts

The IMS applications management component is also transparent, but it plays an integral part in handling transactions. It has two parts:

1 Action scheduling

2 File management

Action scheduling

Action scheduling schedules an action for each input message by allocating main storage and other resources needed by that action. It also handles termination of action programs.

File management

File management (Figure 1-10) provides all action program I/O functions and provides access to your data files through OS/3 data management.

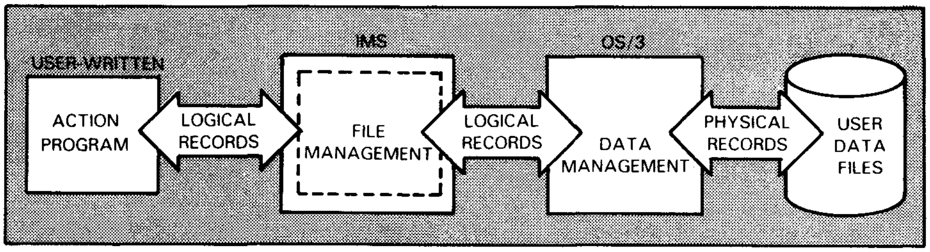


Figure 1-10. How IMS Accesses Your Data Files

Files types used with file management

File management interfaces with two types of files:

1 IMS internal files

2 Your data files

The IMS internal files are:

IMS internal files

- Named record file (NAMEREC)
- Continuity data file (CONDATA - multithread only)
- Audit and continuity data file (AUDCONF - single-thread only)
- Audit file (AUDFILE - multithread only)
- Trace file (TRCFILE)
- Terminal output message file (TOMFILE)
- Statistical data file (STATFIL)
- Fast loader file (LDPFIL)

We tell you more about these files in Section 7.

Your data files

File management controls access to your data files and performs file and record locking functions. The locking capabilities are an IMS feature that guards against concurrent updating of the same data by multiple actions.

Defined record management

Defined record management, a subcomponent of file management, accesses defined files.

Internal message control

The internal message control component:

- Keeps track of your messages
- Processes input and output messages
- Executes terminal commands
- Controls message editing
- Provides an output message when your transaction ends without one

Action program component

The action program component of online IMS handles the processing of two kinds of action programs:

1. IMS-supplied action programs
2. User-written action programs

IMS-supplied programs

The IMS action programs consist of UNIQUE and other IMS-supplied action programs. We talk more about UNIQUE in Section 13 and other IMS-supplied programs in Section 6.

User written action programs

The handling of your action program involves the activation record (3.6) and an action program load module, which consists of your source program (written in COBOL, RPG II, or BAL) linked to an IMS interface module.

1.7. OFFLINE PROCESSING

Offline components

There are three utility programs available with offline processing:

1. The offline recovery utility
2. The tape copy program
3. The statistical file print program

Offline recovery utility used when online recovery fails

The offline recovery utility reconstructs your files when online recovery fails or cannot correct the problem. You determine the type of offline recovery you want at configuration time.

Tape copy program recovers open tape trace file

A tape copy program recovers a tape trace file that is left open as the result of a system failure. This program then produces a new trace file (containing the open file's contents) that you can use for offline recovery.

Statistical file print program

The statistical print program prints statistics recorded during the online IMS session.

Now that you're familiar with IMS components, let's look at what you need to generate an online IMS.

1.8. WHAT YOU NEED TO GENERATE ONLINE IMS

When you operate any system, there are assorted system operations you perform to get various system functions. These operations are performed at various times in the processing environment. Some are external to IMS (which we call system requirements) and others are part of IMS software (which we call IMS requirements).

System Requirements

To prepare OS/3 to support online IMS

You Must:

System generation

- First, generate an OS/3 operating system to support online IMS. This is done in a process called system generation (SYSGEN). There are three phases of SYSGEN you are concerned with when generating IMS. They are the SUPGEN, the RESGEN (Series 90 only), and the COMMCT phases. System generation is discussed in Section 4.

ICAM generation

- Second, because online IMS functions as part of a communications system, generate an integrated communications access method (ICAM) symbiont that supports your IMS. You do this in the COMMCT phase of system generation. Generating an ICAM symbiont involves defining a communications network for the transaction control interface (TCI), which is the only communications interface that supports IMS. See the ICAM network definition manual for information on ICAM generation.

User written action program compilation

- Finally, compile or assemble any user written action programs.

Now that you've taken care of the external system operations, we will look at the IMS operations you must perform to ready your system for online processing.

IMS Requirements

In 1.4, we cover the components of IMS and the processing time frames in which each occur. The IMS functions needed to prepare online IMS are performed in the pre-online processing time frame. They are:

- Initialize the NAMEREC file

- Configure IMS

Initialize the NAMEREC file

You must initialize the NAMEREC file before any other pre-online processing can take place because all of the IMS processors, including the configurator, contribute to this file.

There are two ways to initialize the NAMEREC file:

1. Using the NAMEREC utility (ZP#NRU)
2. Using the INIT parameter of the IMSCONF job control procedure during configuration

We tell you more about initializing NAMEREC in Section 4.

Configure IMS

Configuring IMS is also a required step. The configuration process initializes IMS internal files and generates your online IMS system. Configuration allows you to tailor IMS to fit your needs. Configuring your system is described in Section 3.

Now that you know the system and IMS requirements to create an online IMS, let's look at some of the common uses of IMS and other documentation you need for smooth operation of your system.

1.9. USES OF IMS

Here are some of the IMS applications.

Common applications

Most Common Uses of IMS

- Order entry
- Accounts payable
- Accounts receivable
- Payroll

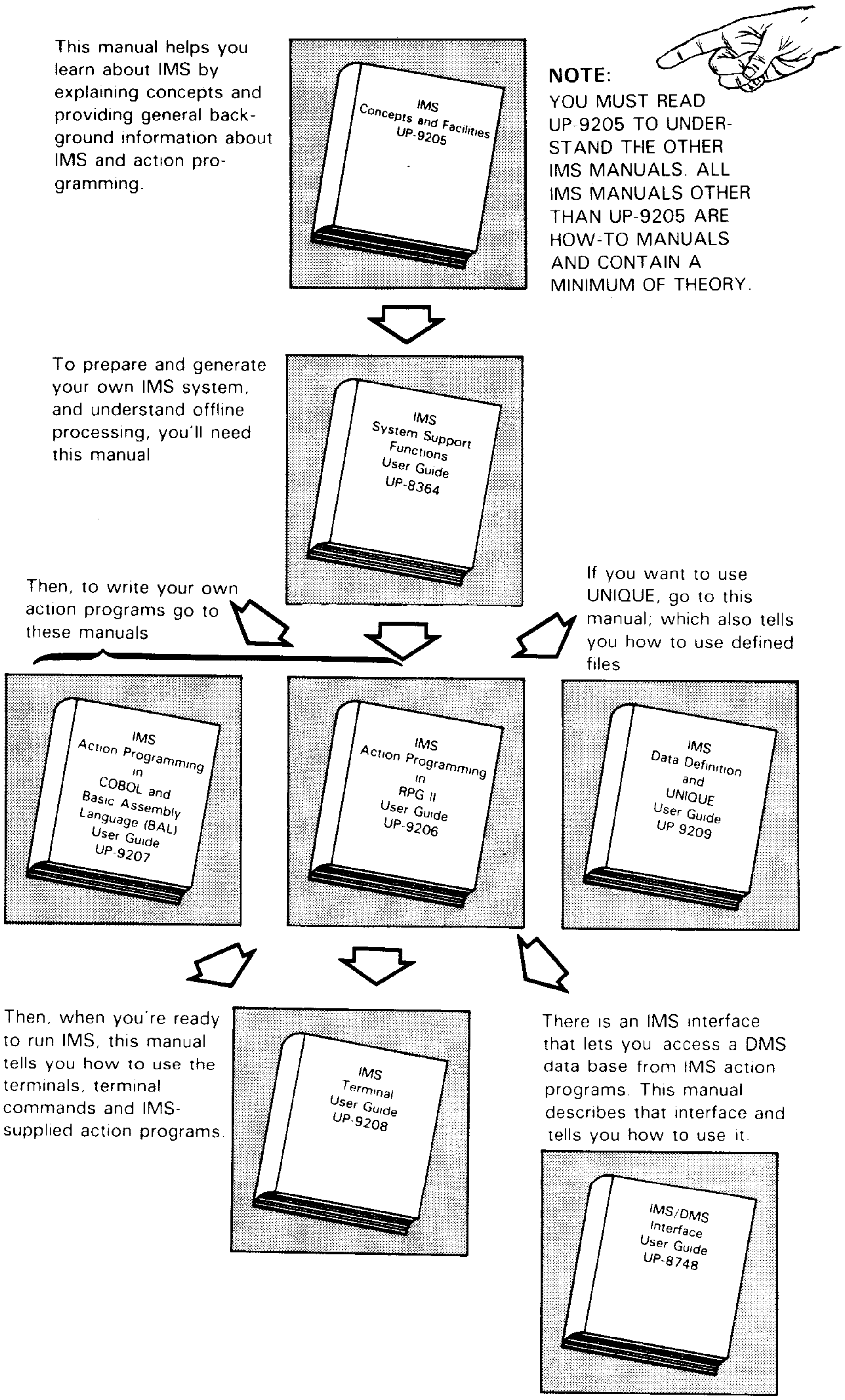
Additional applications

Other uses of IMS are:

- Inventory control
- Publications subscriptions
- Medical records
- Tax records
- General ledger
- Manufacturing
- Cost accounting
- Sales inventory

1.10. IMS DOCUMENTATION

In addition to this manual, you'll need to reference these other IMS manuals:



This manual helps you learn about IMS by explaining concepts and providing general background information about IMS and action programming.

NOTE:
YOU MUST READ UP-9205 TO UNDERSTAND THE OTHER IMS MANUALS. ALL IMS MANUALS OTHER THAN UP-9205 ARE HOW-TO MANUALS AND CONTAIN A MINIMUM OF THEORY.

To prepare and generate your own IMS system, and understand offline processing, you'll need this manual

Then, to write your own action programs go to these manuals

If you want to use UNIQUE, go to this manual; which also tells you how to use defined files

Then, when you're ready to run IMS, this manual tells you how to use the terminals, terminal commands and IMS-supplied action programs.

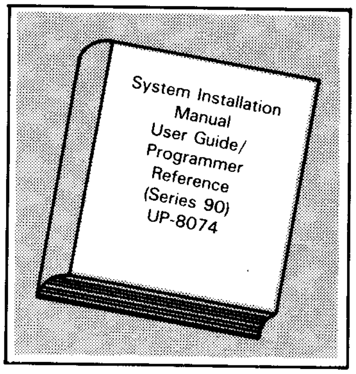
There is an IMS interface that lets you access a DMS data base from IMS action programs. This manual describes that interface and tells you how to use it.

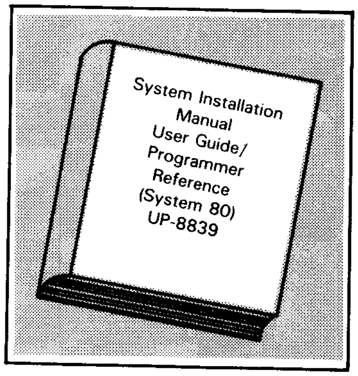
IMS Concepts and Facilities UP-9205
IMS System Support Functions User Guide UP-8364
IMS Action Programming in COBOL and Basic Assembly Language (BAL) User Guide UP-9207
IMS Action Programming in RPG II User Guide UP-9206
IMS Data Definition and UNIQUE User Guide UP-9209
IMS Terminal User Guide UP-9208
IMS/DMS Interface User Guide UP-8748

1.11. OTHER HELPFUL DOCUMENTATION

To aid you in essential non-IMS functions, you may find the following manuals helpful.

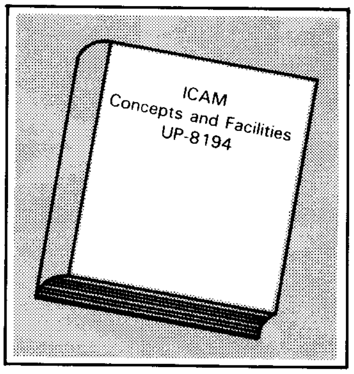
▶ To generate your OS/3 operating system to support IMS, you'll need one of these manuals:





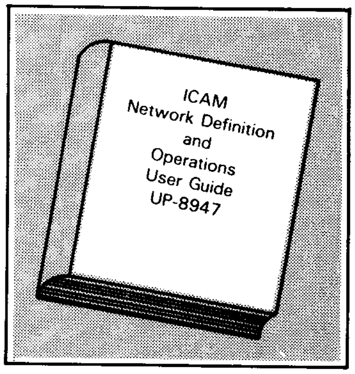
▶ Then, to generate your communciations network to support IMS, you'll need the following ICAM manuals:

First...



This manual tells you about ICAM and its requirements

Then...



This manual has the macroinstructions and commands used to define the communications network and create the ICAM symbiont

2. IMS ENVIRONMENT

2.1. BUILDING AN IMPRESSION OF IMS

Scope of section

Up to this point, we have skimmed over some of the IMS terminology. In this section, we get more specific and tell you about the IMS environment. As you read this section, you will come across many new terms, which up to this point we may not have mentioned. Don't try to understand each new term fully at first; simply concern yourself with the concept being presented. We provide the details in later sections. For now, concentrate on how IMS functions within its environment.

2.2. IMS – THE FIRST IMPRESSION

Computer environmments

All computer systems operate in environments. The word **environment** simply means surroundings. A computer environment generally includes all hardware and software that a system needs to operate. Within an environment, there also exists hardware or software that is not needed for the basic operation of the system, but is needed for specific applications. The same is true with IMS.

IMS has many parts; some required, some optional. To help you build an impression of IMS, we start with a simple diagram, and then build on it to encompass a possible IMS environment (Figure 2-1).

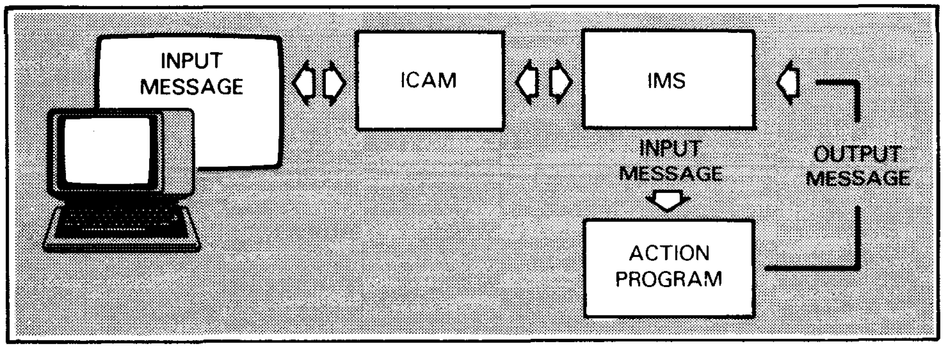


Figure 2–1. Simple Representation of IMS Flow

Your input message, like most IMS messages, is sent from the terminal. It is then passed by ICAM to IMS.

IMS interprets your input message, processes the requested action program, and passes an output message back through ICAM, to your terminal.

What is ICAM?

Based on what you already know about IMS, the terms in Figure 2-1 (except ICAM) should be familiar to you. The acronym ICAM stands for the Integrated Communications Access Method.

ICAM is the communications software that handles all input and output between your terminals and IMS. To do this, it must know how many and what kind of terminals you have and what your needs are. You tell this to ICAM by defining an ICAM network. We tell you more about ICAM network definition in Section 4.

Now that you've seen the basic IMS flow, let's expand on each of the components of Figure 2-1 so you can see what's really involved.

2.3. AN EXPANDED VIEW

Message passage

In Figure 2-1, ICAM is shown as an empty box, mysteriously passing messages back and forth. The lines on which your messages travel seem to go directly from your terminal to ICAM. This is really not so. Figure 2-2 adds another box, the communications adapter (Series 90) or the single line communications adapter (System 80). To eliminate confusion, regardless of your system, we refer to it simply as the **communications adapter**. The communications adapter is a piece of hardware; ICAM is software.

Communications adapter connects terminals to ICAM

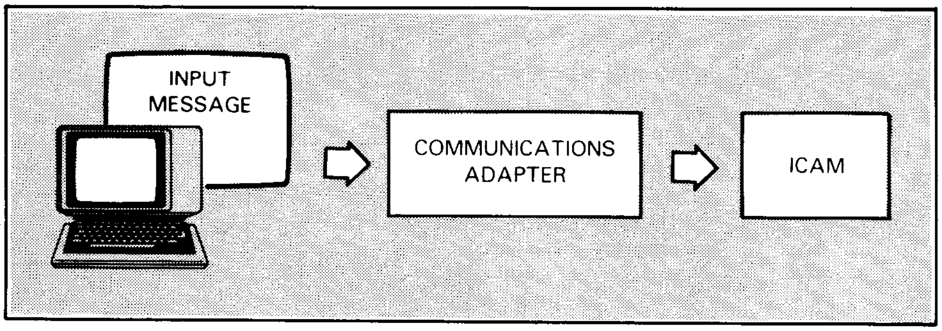


Figure 2–2. How Your Terminals are Connected to ICAM

Communications adapter function

The communications adapter controls and coordinates data transfers, status, and commands between ICAM and your terminals. It, in effect, polices the message flow so that your system can operate efficiently.

Inside ICAM

Let's look at what happens in ICAM when your terminal sends an input message (Figure 2-3).

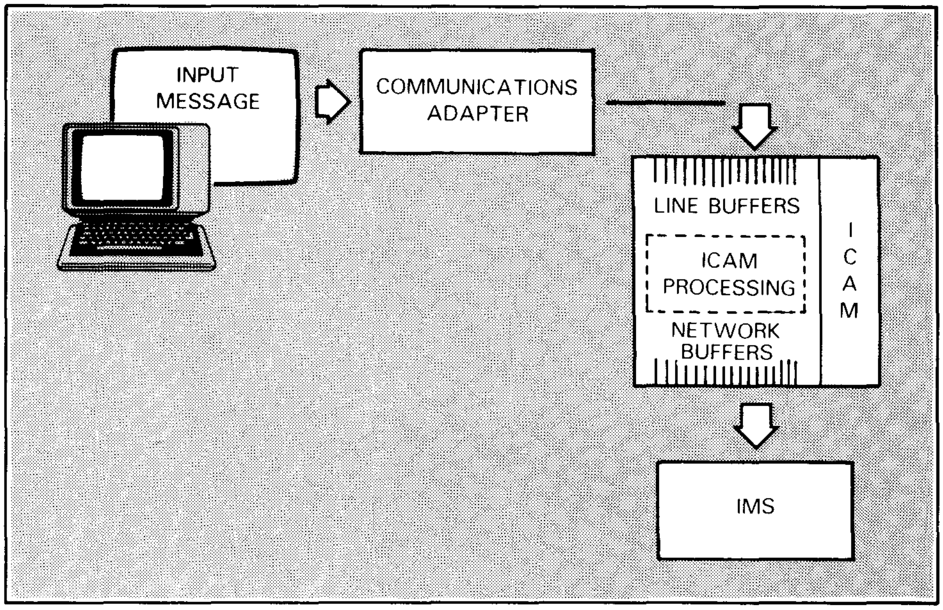


Figure 2-3. What Happens Inside ICAM?

First, your message passes through the communications adapter. When ICAM receives the message from the communications adapter, it goes into an ICAM line buffer. Line buffers temporarily hold messages until ICAM can process them.

ICAM does the internal message handling, places the message in an ICAM network buffer for passage to IMS. ICAM stores the input message in main storage until IMS is ready for it.

Inside IMS

Inside IMS

Let's expand our diagram even more. Up to this point, IMS is shown as an empty box that processes your input messages. Let's take a look inside and see what really happens.

IMS is a program

Action programs are subprograms to IMS

First of all, although IMS appears to be a separate entity, it really is a program (called a communications user program) that functions as part of the communications network. The action programs you write are actually subprograms within the IMS program. Look at Figure 2-4 to see inside IMS.

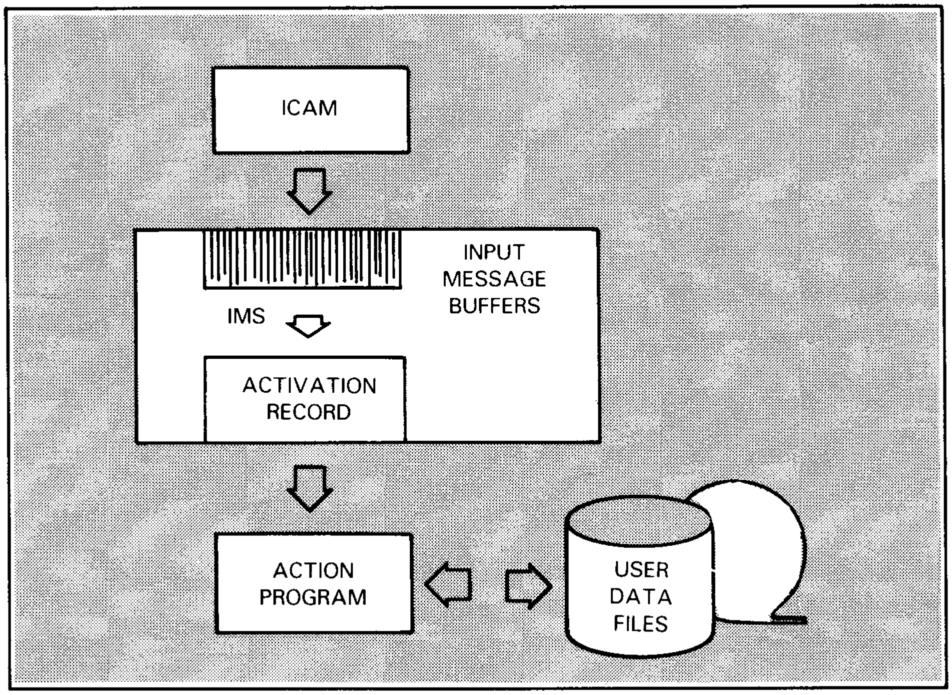


Figure 2-4. Inside IMS

Message passage to IMS

When ICAM has a message for IMS, it passes the message into an input message buffer within IMS. You designate the size of this input buffer when you configure your IMS. Among the things included in this message is the transaction code for the action program you want to access.

IMS processing

IMS, using its online components (described in 1.4), processes the message. IMS:

- reads the transaction code;
- checks the NAMEREC tables (in main storage) to find the action program and files that are assigned to the action; and
- assigns an activation record for the action.

Function of NAMEREC file

Until now, all we've mentioned about NAMEREC is that it's an IMS internal file that contains records from all other parts of IMS. Included in this NAMEREC file, we said, are tables from the configuration process. These tables, among other things, tell IMS the names of all your transaction codes, the names of the action programs related to that transaction code, and the names of the files related to each action.

The contents of the NAMEREC file are read into main storage during IMS start-up. When IMS needs information from these tables, it goes to the main storage area where these tables now reside.

Also during processing, an activation record is assigned for the action program.

Activation record definition

The **activation record** is a main storage interface area that contains and passes control information and data between your program and IMS. An activation record can contain as many as six parts:

Parts of the activation record

ACTIVATION RECORD

- Program information block (PIB)
- Input message area (IMA)
- Output message area (OMA)
- Work area (WA)
- Continuity data area (CDA)
- Defined record area (DRA)

Required parts

Two parts are always required – the program information block and the input message area. The output message area is not required but is almost always used. The other parts depend on which programming language you're using and the needs of the individual action program.

We talk more about the activation record and its function in Section 3.

But, back to Figure 2–4 . . . .

Input message area

The input message is written into the input message area (IMA) of the activation record to await transfer to the action program.

Program processing

Output

Then, the action program accesses your data files and processes the data. If you don't have an output message area, no message can be sent from your completed action program to your terminal, so IMS sends a message saying TRANSACTION COMPLETE. (This message is sent from the internal message control component of online processing). Normally you would provide an output message. It's only when you do not, that IMS sends this message.

Program to file interface

To expand a little on Figure 2-4, your action programs do not actually interface directly with your data files, they use the file management component of IMS and OS/3 data management. Figure 2-5 shows what happens when an action program requests a record. The process is similar for writing a record. Data management receives the logical record from IMS file management and writes it into your data file..

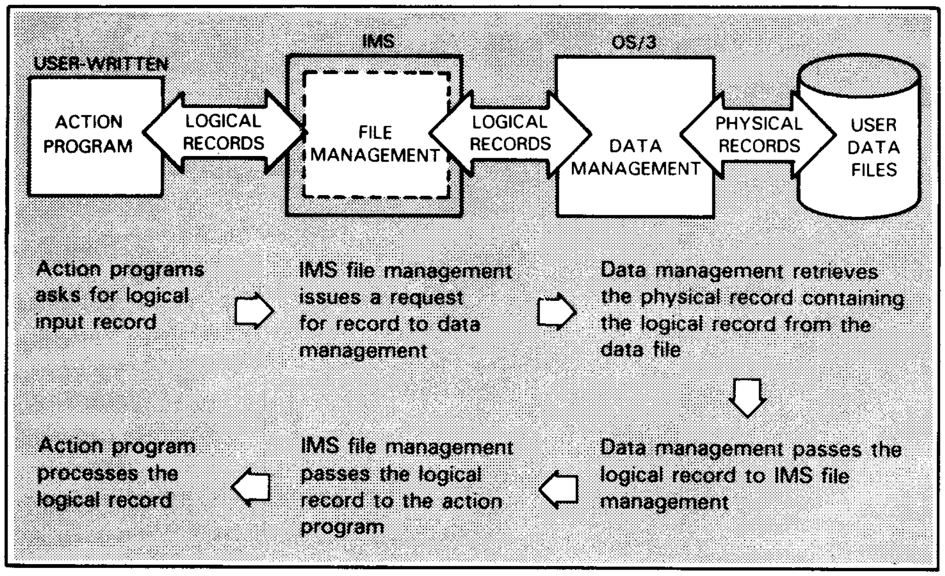

USER-WRITTEN
ACTION PROGRAM
LOGICAL RECORDS
IMS
FILE MANAGEMENT
LOGICAL RECORDS
OS/3
DATA MANAGEMENT
PHYSICAL RECORDS
USER DATA FILES
Action programs asks for logical input record
IMS file management issues a request for record to data management
Data management retrieves the physical record containing the logical record from the data file
Data management passes the logical record to IMS file management
IMS file management passes the logical record to the action program
Action program processes the logical record

Figure 2-5. How Your Action Programs Interface with Your Data Files

2.4. THE OVERALL PICTURE

Now that you've seen IMS broken down into its parts, let's look at the whole picture (Figure 2-6) so you can see how it all fits together.

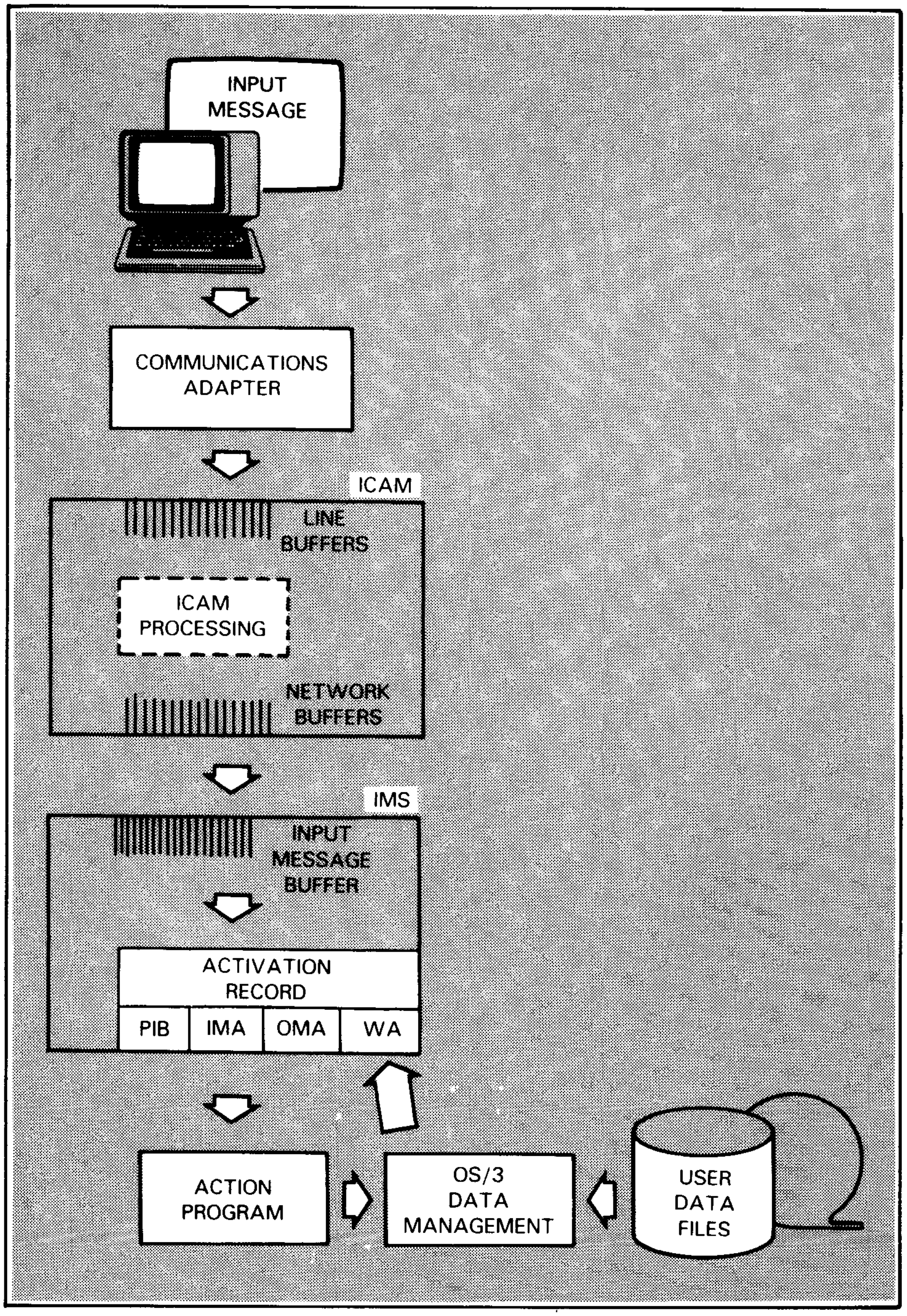


Figure 2-6. IMS – The Big Picture

3. Action Programming: The Heart of IMS Processing

3.1. PURPOSE AND TYPES OF ACTION PROGRAMS

Function of Action programs

Action programs are the heart of IMS. Without them you can't process messages. The main purpose of an action program is to process input messages and generate output messages. But they also perform file retrieval and updating functions.

As you know, there are two kinds of action programs:

Type of action programs

1 Those that IMS supplies

2 Those you write yourself

IMS supplied action programs

One set of action programs that IMS supplies is the Uniform Inquiry Update Element (UNIQUE). UNIQUE is a set of programs that performs file retrieval and updating functions. UNIQUE functions are initiated by UNIQUE commands. When you use UNIQUE, you do not need to write your own action programs. (UNIQUE is discussed in Section 13. Other IMS-supplied action programs are discussed in Section 6.)

User written Action programs

This section concentrates on the action programs you write yourself; how they're set up, and how they are processed. You can write action programs in COBOL, basic assembly language (BAL), and RPG II. We do not go into much detail about the specifics of these languages. Rather, we present general concepts of action programming. If you want information on how to write action programs in these languages, see IMS action programming in COBOL and BAL user guide, UP-9207, or IMS action programming in RPG II user guide, UP-9206.

3.2. WHY WRITE YOUR OWN ACTION PROGRAMS

Limitations

UNIQUE is an easy-to-use file retrieval and updating system. So why, if UNIQUE is so easy to use, should you write your own action programs? Because UNIQUE, like many packaged programs, is limited in how much it can do. And, you may want to use IMS in ways not possible with UNIQUE.

UNIQUE CAN'T:

- Update files that require extensive data validation
- Use special output formats
- Perform calculations on your data

Custom-tailoring action programs

When you write your own action programs, you design them to fit your needs. If your application requires validation of all terminal input, you code that into the program. If you want to perform calculations on terminal input and update your data files on the basis of these calculations, you code that into your program also.

If you want output messages to appear in a particular way on the terminal screen, you provide the format in your action program. If there are special messages or prompts you want to give the terminal operator, you can also code them into the program. You can also use screen format services to format output messages.

3.3. DESIGNING AN ACTION PROGRAM

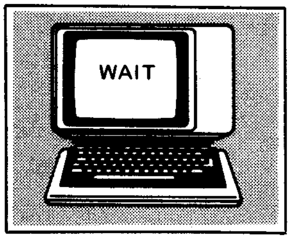
What makes action programs different

Actually, action programs are not that different from any other program. They handle all the file processing activities your regular programs currently handle. What is different is that they handle them in an interactive environment by carrying on a conversation with the terminal operator.

Keep action programs simple

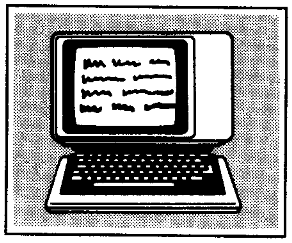
Since action programs carry on a conversation with a terminal operator, the most important thing to remember when designing an action program is to keep it simple. Present the data that the program requests, and the answers or output it will provide, in a clear and concise way so that it can be easily understood by the terminal operator.

Maintaining efficiency



Also, action programs operate most efficiently when they perform short, precise tasks. When you converse with someone, you do not listen to their question and then leave them "hanging" for an answer.

Provide constant feedback to the operator



Terminal operators should get almost constant feedback to their inquiries. Design the action program to take the operator's inquiry, process it, and provide a quick and appropriate response. Then, the program requests more data from the operator. This process continues for as long as necessary, but the key is that the operator is never stranded. The action program keeps the conversation going.

3.4. HOW IMS SUPPORTS ACTION PROGRAMS

You already know that the major activity of action programs is processing input messages and producing output messages. To understand what this involves, there is something much more fundamental that you must understand first.

IMS/action program relationship

You must understand that IMS is itself a program. The action programs you write operate as subprograms of IMS. What does this mean= It means that IMS and action programs share a special relationship. They work as a team.

IMS's supporting role

IMS handles message I/O

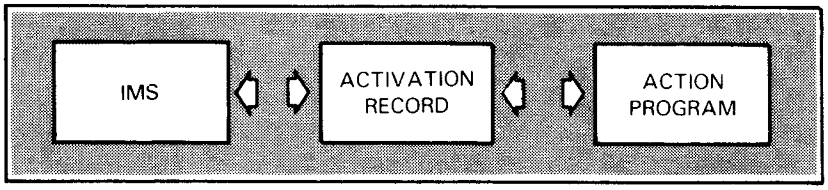
Because of this relationship, there are many things that action programs depend on IMS to do for them. For instance, action programs don't handle I/O. The internal message control component of IMS handles all input messages coming into the program and all output messages generated by the program.

IMS handles file I/O

File I/O is performed by the file management component of IMS. Action programs direct all file I/O functions to IMS which, in turn, passes them on to OS/3 data management.

Activation record handles communication between IMS and action programs

This relationship between IMS and action programs demands a lot of communication. That's why action programs must set up an area where this communication takes place. We've mentioned this area before. We talk about it more in this section. It's called the **activation record**. All communication between IMS and the action program passes through the activation record.



The Activation Record Interfaces IMS to Action Programs

3.5. CONTENTS OF THE ACTIVATION RECORD

The activation record can contain up to six areas. These areas are the program information block, the input and output message areas, the work area, the continuity data area, and the defined record area.

PROGRAM INFORMATION BLOCK

The program information block passes control information between IMS and your action program during program processing. This information is very valuable to an action programmer. For instance, after each I/O file request made to IMS, certain fields in the program information block contain the results of that request.

The action program can examine these fields to determine if the request was successful or not. The program information block also contains the accurate calendar date and clock time. Programmers often want to make this data part of an output message.

INPUT MESSAGE AREA

The input message area receives input from the terminal. The input message remains here until the action program is ready to process it.

OUTPUT MESSAGE AREA

The output message area holds the output message that the action program generates.

WORK AREA

COBOL and BAL use the work area to hold logical records during file processing and as a working storage area.

RPG II action programs never actually use the work area; however, the compiler does for certain applications.

CONTINUITY DATA AREA

The continuity data area holds data that an action program wants to save and pass to the action program that follows it (successor program). When the first program finishes processing, IMS stores the saved data until the successor program is scheduled. Then, IMS makes the saved data available to the successor program.

DEFINED RECORD AREA

The defined record area is used by defined record management when action programs use defined records. Action programs cannot access this area.

3.6. SETTING UP THE ACTIVATION RECORD

Determining activation record needs

Every action program must define an activation record. It is coded as part of the action program. How many areas the activation record contains depend upon the needs of the application. Define only those areas you need. For instance, most action programs begin processing by receiving an input message from the terminal. There must be an area to receive that message. So, the action program must define an input message area as part of its activation record.

On the other hand, if the program do not pass data to a successor program, then don't define a continuity data area. Define only those areas you need.

Loading activation record

Each time a terminal requests an action program, IMS schedules it and loads it in main storage. IMS also loads the activation record set up for that program. Once the program begins executing, the data that IMS and the program exchange is stored in the activation record.

3.7. CODING ACTION PROGRAMS

When writing your own action programs, there are three types of coding you can use:

Types of coding

1. Serially reusable
2. Reentrant
3. Shared

Efficient code

The type of coding you use depends on the programming language you're using. The most efficient way to code an action program is to make the code reentrant or shared, but this is not always possible. You designate the type of coding you're using at configuration time. You do this with the TYPE parameter in the PROGRAM section of the configuration.

Configuration requirements

Using serially reusable code

With serially reusable code, only one user can access the program at a time. Another terminal requesting the same program must wait until the first user is finished.

Example

Figure 3-1 shows terminals A and B both requesting action program PROG1. The first terminal to request the program (in this case, terminal A) gets access to the program. Terminal B must wait until terminal A finishes processing.

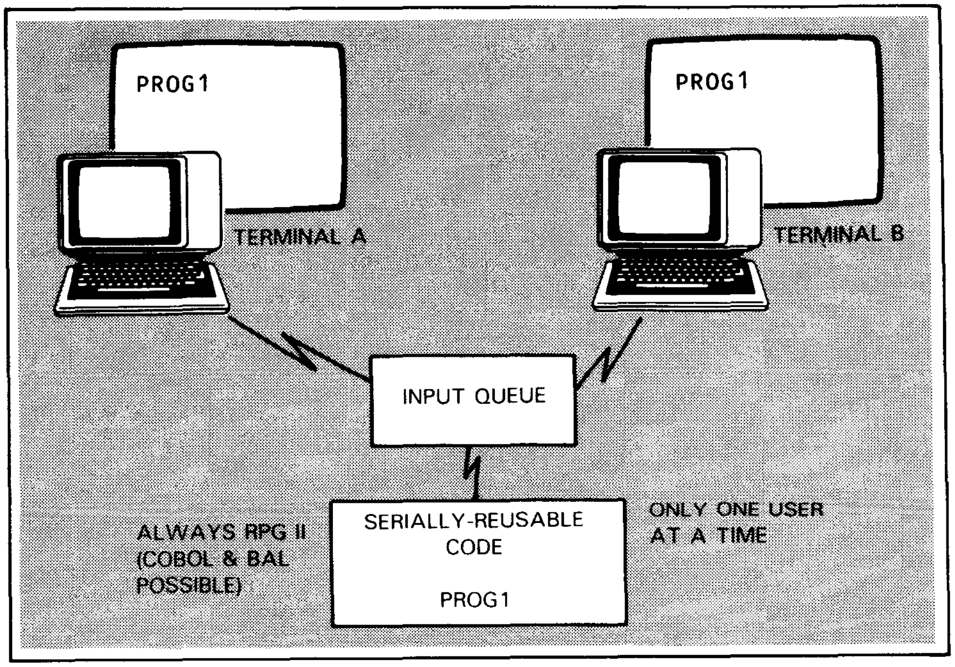


Figure 3-1. Action Program Written in Serially Reusable Code

Efficiency considerations

In a multithread IMS, if there is sufficient space, serially reusable programs reside in main storage. This way they do not have to be reloaded each time they're requested. A single-thread IMS will keep the program loaded only if used by the next input.

Serially reusable code is used most efficiently in a single-thread environment where you process only one action at a time; but this is also permitted in a multithread environment. All action programs can be written in serially reusable code; however, RPG II programs must be since they cannot function as reentrant or shared.

When serially reusable code must be used

Coding requirements

With serially reusable code, you must reset all fields to their original values at the end of the program so that the program is ready for the next user.

Using reentrant code

With reentrant code (Figure 3-2), the action program is loaded into main storage and can be used concurrently by two or more users.

Reentrant code shareable

Restrictions on using reentrant code

Reentrant programs are completely shareable. Thus, no parts of the program can change during action processing because more than one action is using the program. BAL action programs are the only type of program where you use reentrant code.

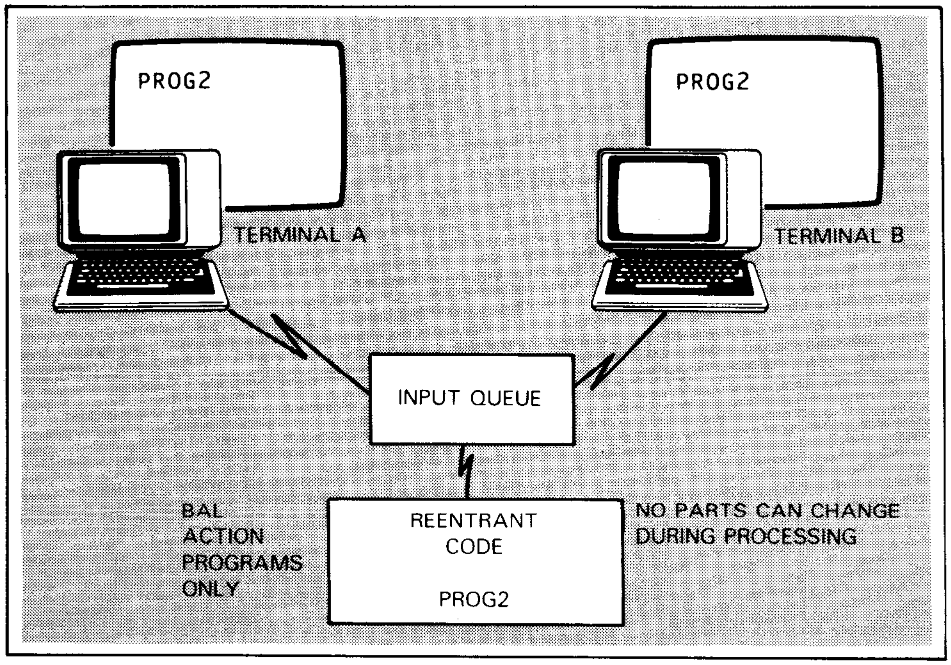


Figure 3-2. Action Program Written in Reentrant Code

Using shared code

Restrictions on using shared code

Shared code programs (Figure 3-3) are the same as reentrant code programs except that, instead of being completely shareable, they're only partially shareable. With shared code, each user has its own activation record. COBOL action programs are the only type of action programs that use shared code.

A shared code parameter in the COBOL compiler lets you designate the program as shareable. When you use shared code in a COBOL program, only the procedure division and the working storage sections are shareable (Figure 3-3).

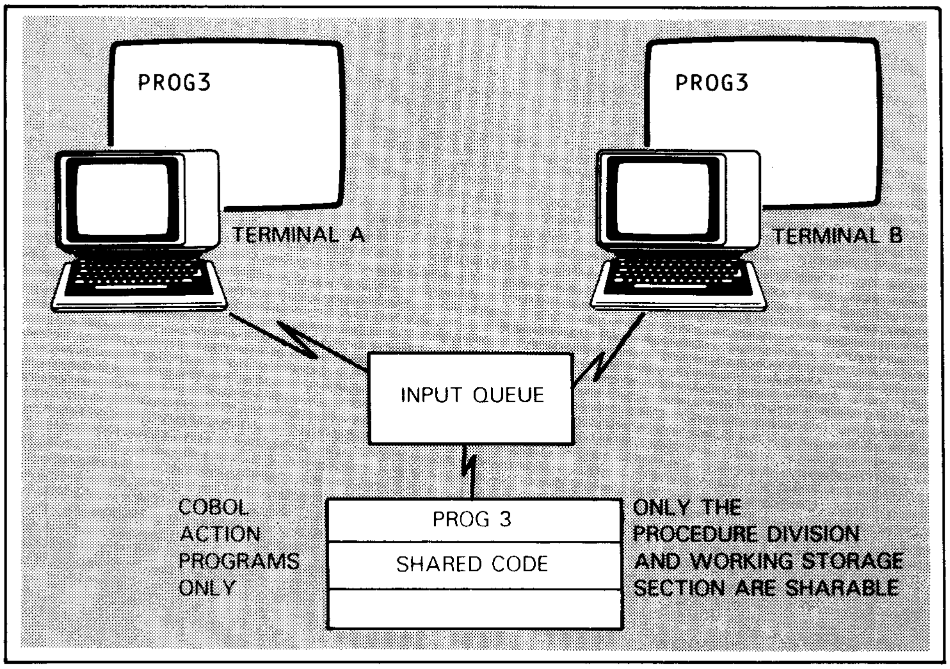
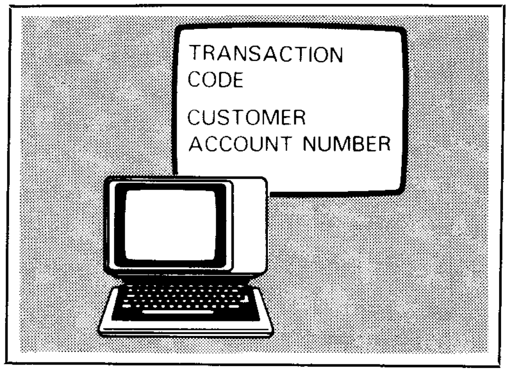

PROG3
PROG3
TERMINAL A
TERMINAL B
INPUT QUEUE
COBOL ACTION PROGRAMS ONLY
PROG 3
SHARED CODE
ONLY THE PROCEDURE DIVISION AND WORKING STORAGE SECTION ARE SHARABLE

Figure 3-3. Action Program Written in Shared Code

3.8. HOW ACTION PROGRAMS ARE PROCESSED

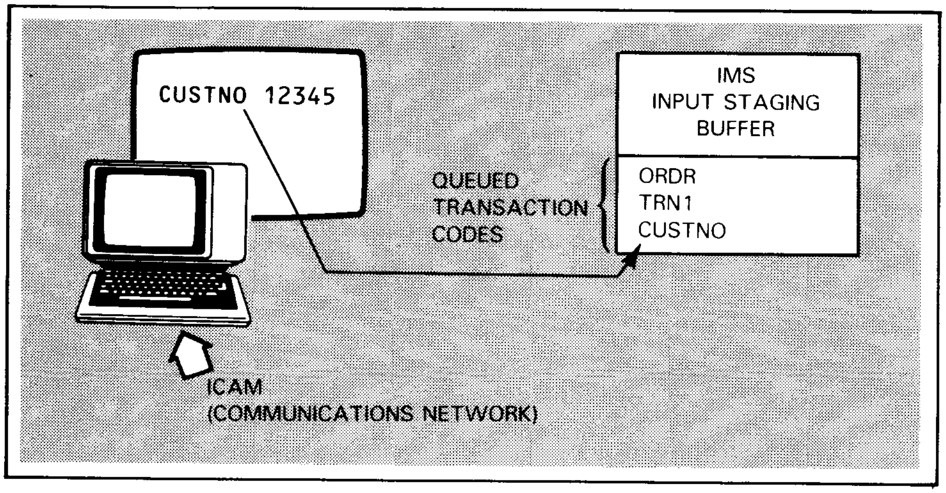
Entering the transaction code

IMS transactions begin when the terminal operator enters a transaction code at a terminal. Once that transaction code – the first input message – is transmitted, things begin to happen.

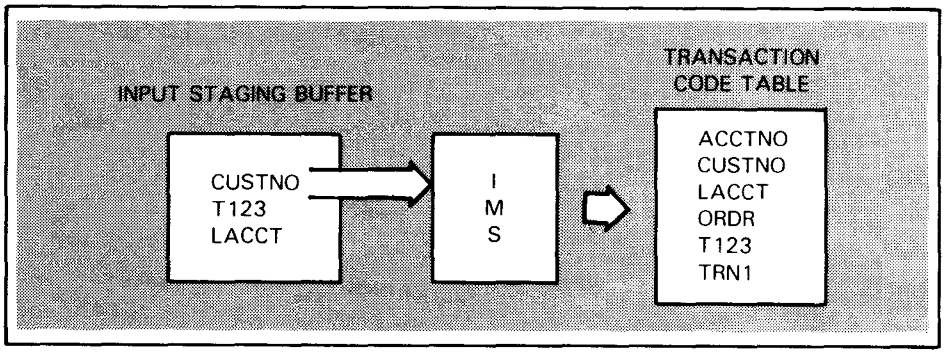


Input staging buffer

Assume the terminal operator enters the transaction code CUSTNO and a customer account number as input. Initially, IMS is only interested in transaction codes. Transaction codes are from 1 to 8 characters in length in multithread IMS and 1 to 5 characters in single-thread IMS. This code comes into the IMS input staging buffer. Messages are queued at the staging buffer until IMS can handle them..



As soon as IMS receives the transaction code (CUSTNO), it validates it by looking at the transaction code table. This table lists all the transaction codes that this IMS configuration supports.



If IMS doesn't find CUSTNO, an error message is sent to the terminal. If it is a valid transaction code, IMS processes the input message.

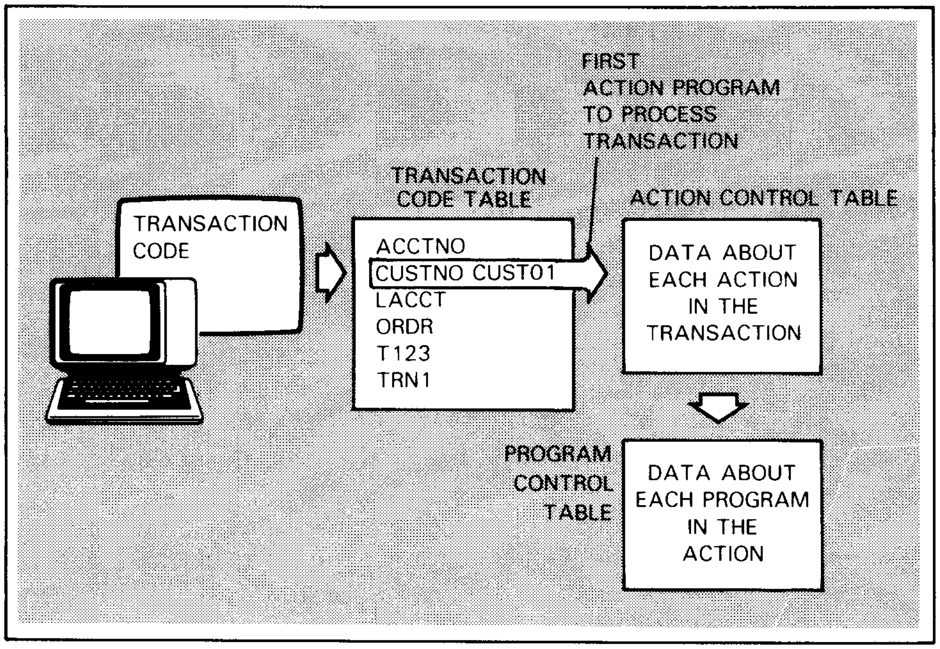
Getting the action name

The transaction code table also provides the next piece of data IMS needs – the name of the first program that processes this transaction. In this case, the name of the first program is also the name of the first action in the transaction. IMS now goes to another table – the action control table – to get more detailed data about this action.

The action control table is stored in the named record file at configuration time and is loaded into main storage at start-up.

Using the action control table

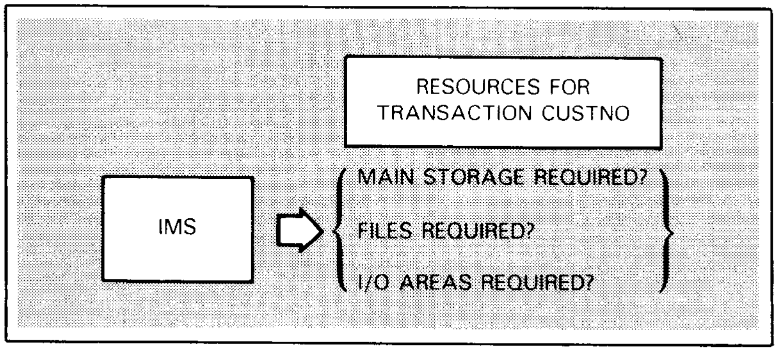
The action control table tells IMS what it needs to know about the action program or programs that process this action. For instance, the action control table describes all the files used in processing the action and the size of the largest output message it can produce.



In multithread IMS, the action control table tells how many users can process an action concurrently. One of the most important pieces of data it contains is the size of the largest nonresident program in this action.

Using the program control table

When IMS gathers all the data it needs, the action control table directs IMS to the program control table. (It is also generated during the configuration process and is part of NAMEREC). The program control table tells IMS about the specific program or programs processing this action and contains data about whether the program is serially reusable, shared, or reentrant, and whether the program permanently resides in main storage.



Assessing IMS resources

While IMS moves from table to table, it's assessing the needs of the action program or programs requested by the transaction code CUSTNO and comparing those needs to the resources available. IMS checks that:

- required files are available;
- there is sufficient main storage for:
  - I/O areas
  - action program
  - activation record

Scheduling the action program

If any of the action program's needs cannnot be met, the transaction code remains in queue until all resources are available. When the resources are available, IMS schedules the first action program to process the transaction and loads it in main storage (unless, of course, it is already there). In addition, IMS allocates and initializes the action program's activation record.

3.9. TRANSACTIONS, TERMINATIONS, AND SUCCESSION

Before we consider the next series of steps in processing an action program, there are some basic concepts and terminology you should become familiar with. A few of these terms are noted elsewhere. Here, we review them quickly and then introduce a few new terms.

The 'action'

As we told you in 1.2, the action is the basic unit of work in IMS. An action occurs when you key in an input message, the appropriate program processes it and a programming function (e.g., data retrieval) is completed, and a response or output message is sent back to the terminal. Figure 3-4 shows an action.

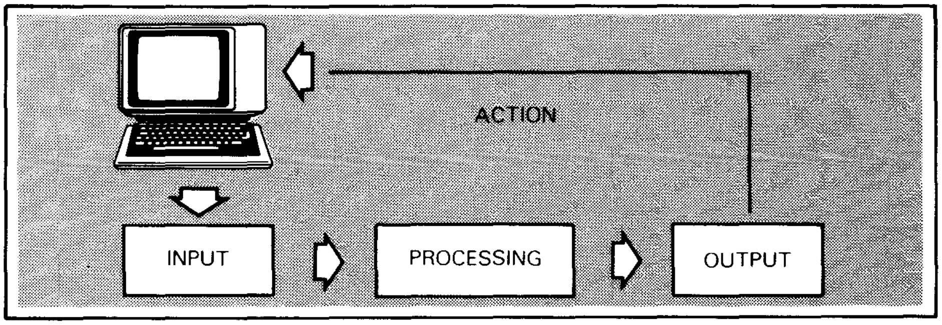


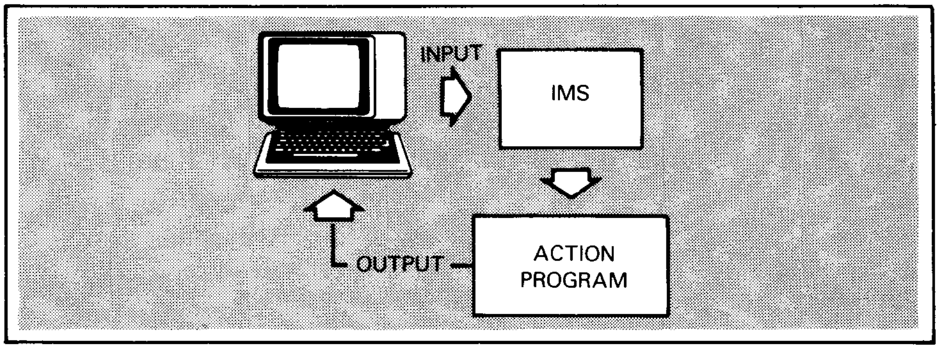
Figure 3-4. An IMS Action Always Consists of Three Parts (Input, Processing, and Output)

Generally, one action program performs one action. Sometimes more than one action program is needed to complete the action.

The 'transaction'

A simple transaction

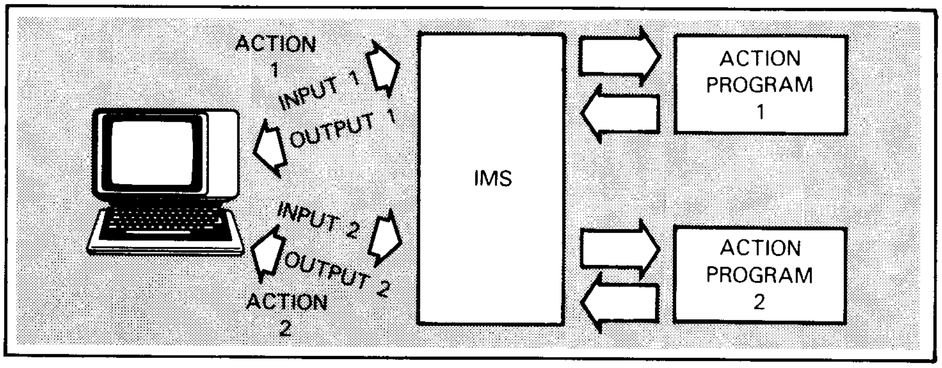
A transaction refers to the whole task you want accomplished. They consist of one or more actions. When one action can accomplish all your processing needs, it's a **simple transaction.** In this case, action and transaction are synonymous.



A Simple Transaction Process (One Action Only)

A dialog transaction

Often, a sequence of two or more actions is needed to accomplish your processing; this is called a **dialog transaction.** Dialog transactions process more than one action program.



A Dialog Transaction Processes Two or More Actions

Transaction structures

These two types of transactions, simple and dialog, are called **transaction structures.** They provide flexibility in designing action programs. In 3.5 we mentioned that action programs can be linked together. This happens by using transaction structures.

'Termination type'

You set up the type of transaction structure you want by specifiying a **termination type.** What this means is that you send IMS, right within your action program, a message. This message tells IMS whether you've completed your task or whether more processing is needed. This message allows IMS to create the transaction structure you need.

3.10. TERMINATION TYPES

There are four termination types used to structure transactions:

Types of termination

1. Normal termination
2. External succession
3. Delayed internal succession
4. Immediate internal succession

MORE processing

All four of the termination types tell IMS that the current program is finished processing. Types 2, 3, and 4 tell IMS that more processing follows.

NO MORE processing

When you specify normal termination, you're telling IMS that the transaction is complete. No more processing will occur. Every transaction ultimately ends with normal termination.

Another program follows

When you specify external, delayed, or immediate succession, you're telling IMS that another action program will follow and resume processing.

Figures 3-5 to 3-8 show how each termination type is used in action program processing.

Normal Termination

Using normal termination

Normal termination tells IMS all processing is complete. You can specify normal termination after a single action program has finished executing or after a series of action programs have completed processing. It all depends upon how you want to structure your transaction.

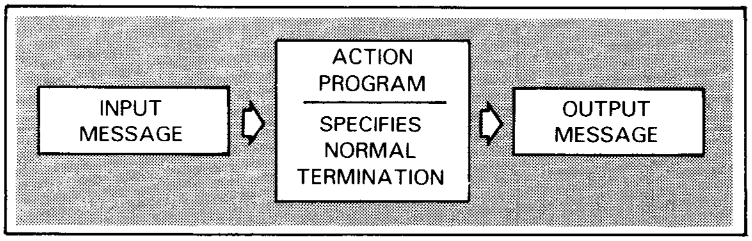


Figure 3-5. Transaction Structure Using Normal Termination

External Succession

Using external succession

External succession tells IMS that the current program is finished processing but there is still more to be done. A successor program follows to pick up where the first program left off.

With external succession, the first action program processes an input message and produces an output message; and so does the successor program. There are two sets of input and output, or in other words, two separate actions.

IMS doesn't schedule the successor program until it receives the second input message. Figure 3–6 shows the sequence of activities when you specify external succession as the termination type.

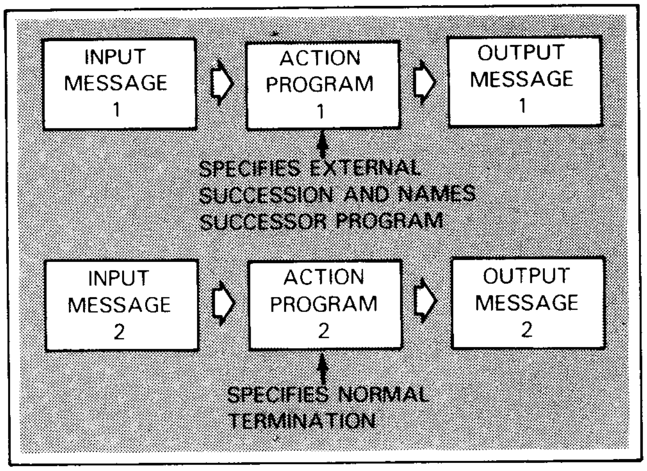


Figure 3–6. Transaction Structure Using External Succession

Delayed Internal Succession

Using delayed internal succession

Delayed internal succession (Figure 3-7) tells IMS that the current program is finished processing but there is more processing to follow. The successor program picks up where the previous program left off.

With delayed succession, the first program processes an input message and produces an output message; but, the output message does not go to the terminal. The terminal operator never sees it. Instead, when the first program finishes processing, IMS takes the output generated and passes it as input to the successor program.

The successor then processes that input and produces an output message which goes to the terminal. Even though only the first input message and the second output message are actually seen at the terminal, internally there are two separate input and output messages and, consequently, two separate actions.

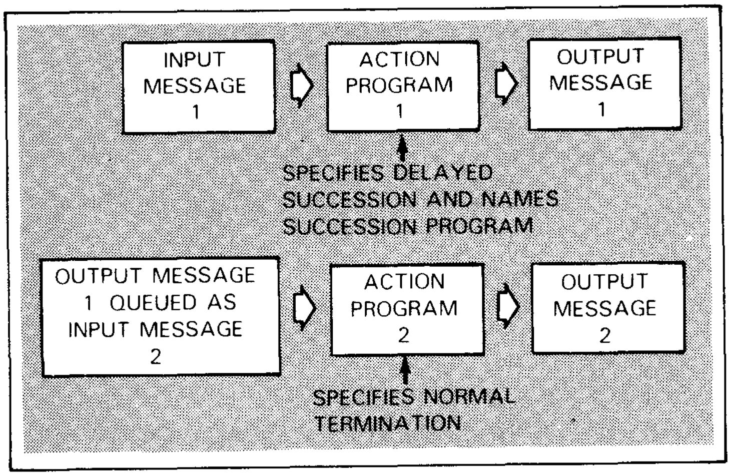

INPUT MESSAGE 1
ACTION PROGRAM 1
OUTPUT MESSAGE 1
SPECIFIES DELAYED SUCCESSION AND NAMES SUCCESSION PROGRAM
OUTPUT MESSAGE 1 QUEUED AS INPUT MESSAGE 2
ACTION PROGRAM 2
OUTPUT MESSAGE 2
SPECIFIES NORMAL TERMINATION

Figure 3-7. Transaction Structure Using Delayed Internal Succession

Immediate Internal Succession

Using immediate succession

Immediate internal succession, like external and delayed internal successions, also says the current program is completed and a successor will follow. In this type of termination, there is an initial input message.

The first action program processes the message and the second program is immediately scheduled to continue the processing. When the second program finishes processing, it generates an output message and terminates.

In immediate internal succession, therefore, there is only one input and one output message. Consequently, there is only one action. Figure 3-8 shows a transaction structure using immediate internal succession.

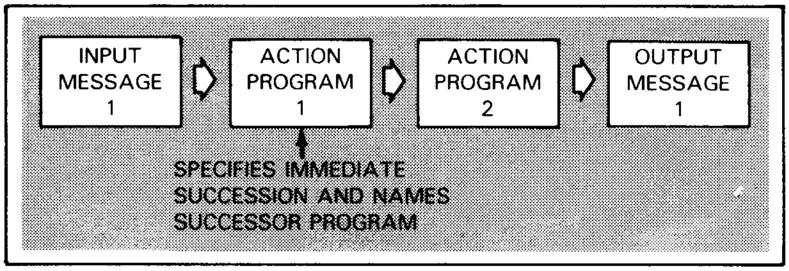


Figure 3-8. Transaction Structure Using Immediate Internal Succession

As you can see, these four termination types allow a great deal of flexibility. Soon we will see how this flexibility is put to use.

3.11. SPECIFYING TERMINATION TYPE AND NAMING SUCCESSOR PROGRAM

Specifying termination type

You specify termination type within your action program. In this way, you inform IMS of the transaction structure you're creating. The only time you don't need to specify a termination type is with normal termination. When you don't specify otherwise, IMS assumes normal termination.

Defining a program information block

Using the termination-indicator field

To specify the termination type, you must set up a program information block area in your activation record. During processing, the action program moves a termination code to the termination-indicator field of the program information block. This code tells IMS how you want to structure your transaction.

TERMINATION CODES

Codes used

N	Normal termination (if you don't specify a code, IMS assumes normal termination)
E	External succession
D	Delayed internal succession
I	Immediate internal succession

Naming a successor program

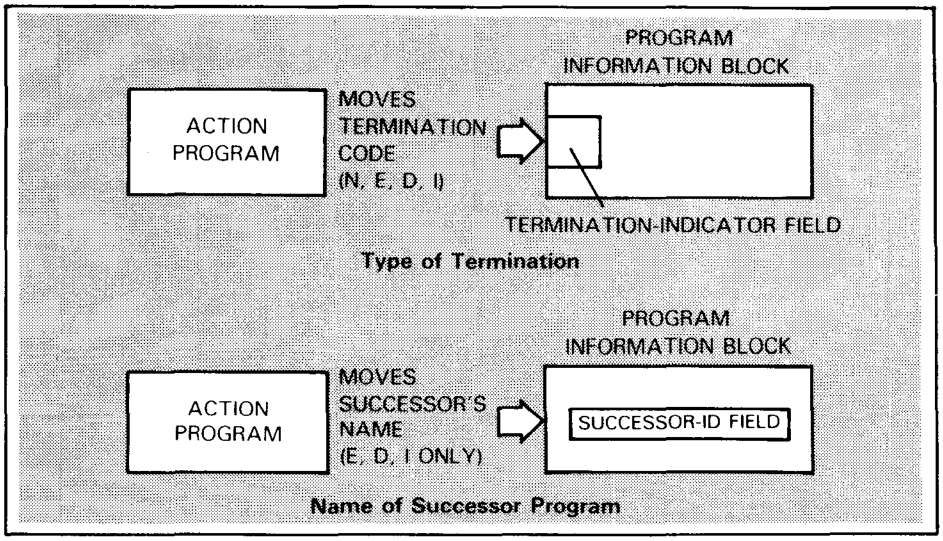
Using the successor-id field

For termination code E, D, or I, you must name the successor program. You enter the name in the *successor-id* field of the program information block. The name can be 1 to 6 characters long and identifies the program that will continue processing the transaction.

A transaction code is not required in the second (or subsequent) input messages of a transaction since the 'successor-id' specified by the previous action program names the program that will process the next message.

Examples

Figure 3-9 illustrates moving the termination code and successor name to the program information block. For detailed information on how you do this, see the action programming manuals.


PROGRAM INFORMATION BLOCK
ACTION PROGRAM
MOVES TERMINATION CODE (N, E, D, I)
TERMINATION-INDICATOR FIELD
PROGRAM INFORMATION BLOCK
ACTION PROGRAM
MOVES SUCCESSOR'S NAME (E, D, I ONLY)
SUCCESSOR-ID FIELD

Type of Termination

Name of Successor Program

Figure 3-9. Action Program Indicates Type of Termination and Name of Successor Program

When the action program finishes executing, IMS checks these fields of the program information block and, on the basis of what they contain, arranges for further processing, if required.

3.12. EXAMINING A SAMPLE TRANSACTION

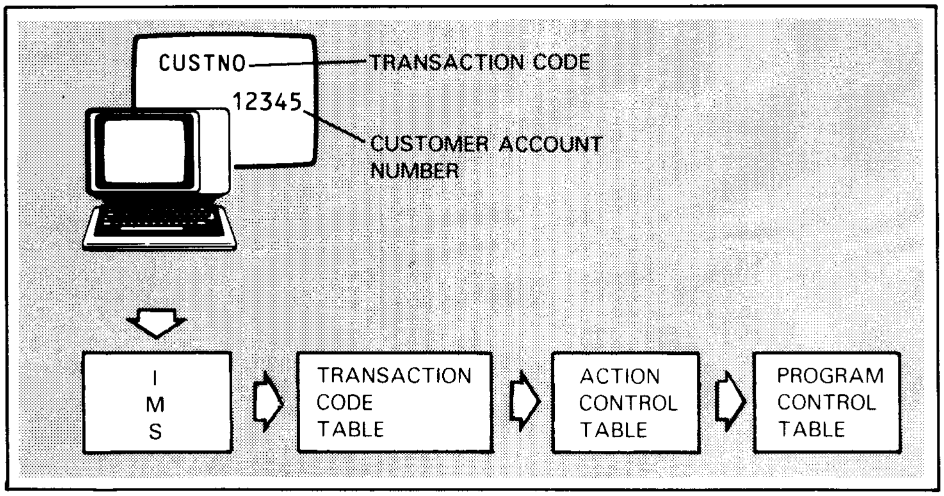
To give you a better feel for how to create a transaction structure, let's look at a sample transaction with which you are already somewhat familiar. You remember that in Section 1, when we discussed dialog transactions, we talked about a transaction that displayed a customer's account balance at the terminal, allowed the operator to credit payments to the account, and displayed the new balance. This series of events was a transaction structure although you didn't realize it at the time. Let's look closely at that transaction – we'll call it CUSTNO – and learn the type of structures it uses.

Sample transaction – CUSTNO

Scheduling the Action Program

Checking IMS tables

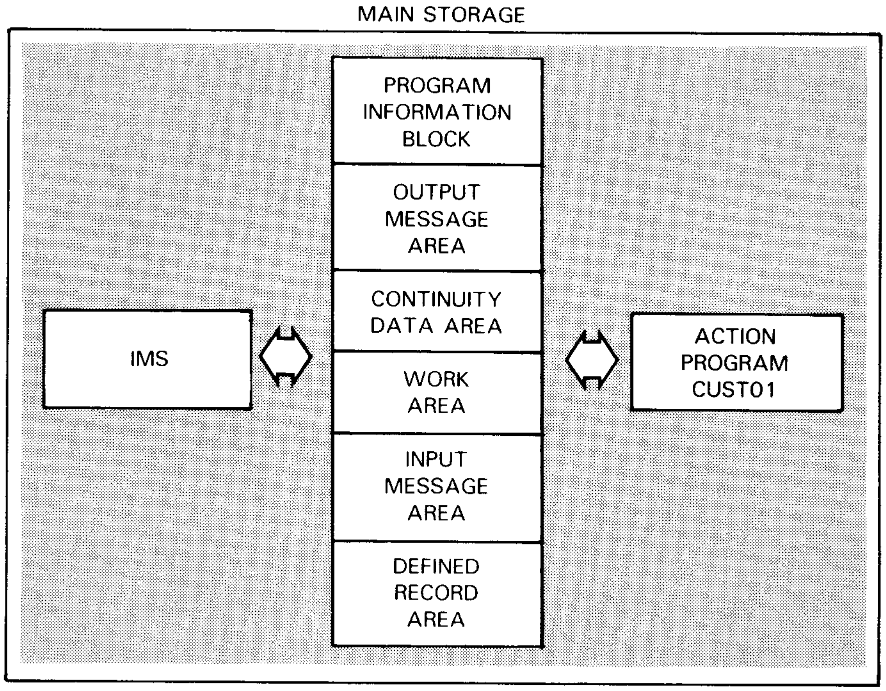
In 3.7 we saw how the transaction code CUSTNO (that initiated the transaction) was entered at the terminal and how IMS checked the transaction, action, and program control tables to determine processing requirements.



Validating an IMS Transaction Code and Determining Transaction Requirements

Loading action program CUST01

When IMS found that it had sufficient resources to meet the transation's needs, it scheduled the first program (CUST01) and loaded it and its activation record into main storage. Let's resume our discussion at this point.



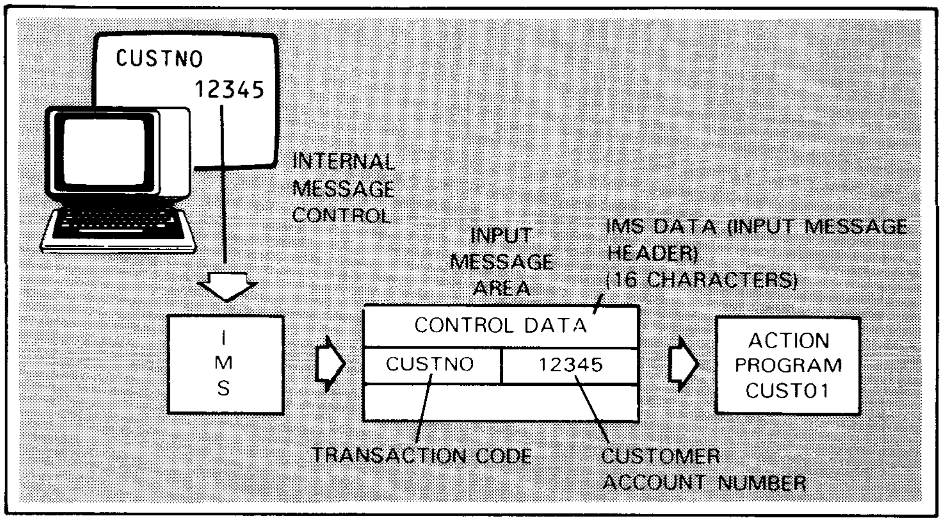
Action Program and Activation Record Loaded in Main Storage

Handling the Input Message

Loading the input message

Using the input message area

CUSTO1 takes the customer number entered as input at the terminal and processes it. As soon as IMS schedules CUSTO1 and loads it and its activation record into main storage, the internal message control component of IMS takes the input message entered at the terminal – transaction code and customer account number – and moves it to the input message area of program CUSTO1. The following illustration shows this.

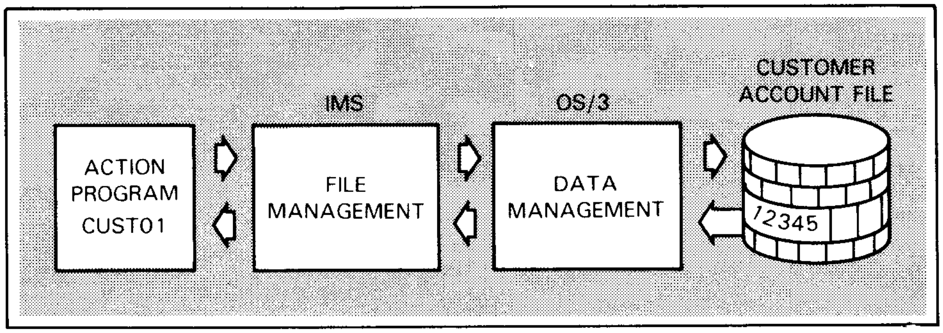


IMS Places Input Message in the Input Message Area

File Processing

Processing the input message

When CUST01 begins processing, it reads the input message area to see exactly what customer account number is requested. Using that number as a key, the program calls upon IMS file management to request OS/3 data management to get the appropriate customer account record.



Action Program Accessing a Data File

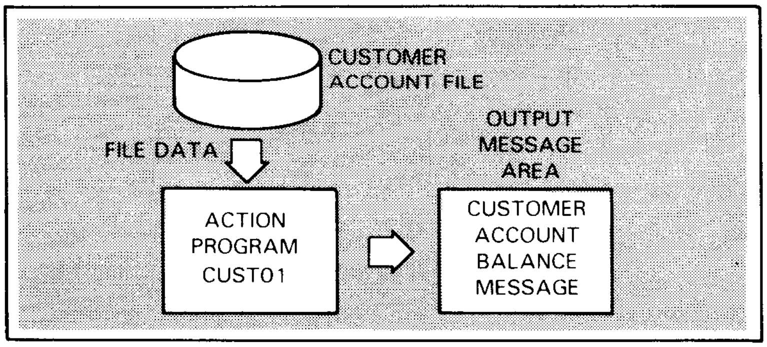
Generating the Output Message

Generating the output message

Once the record is retrieved and the account data is available to the program, CUST01 uses this data to generate an output message to be sent to the terminal. For the purpose of illustration, let's say the customer number is 12345 and the account balance is $4356.33. Using this data, the program creates the output message and its format in the output message area of the activation record.

Using the output message area

Naturally, since CUST01 knows it will generate an output message, it sets up an output message area to handle it. The message is not transmitted to the terminal at this point. That happens when the action program finishes processing. The following illustration shows how the action program generates output.



Action Program Generating an Output Message

Creating the Transaction Structure

Setting up the transaction structure

Before a program terminates, however, it must inform IMS what type of transaction structure it wants to create (Figure 3-10). This means that CUST01 has to let IMS know whether processing for this transaction is complete or not complete.

Using the program information block

CUST01 must move a termination code to the *termination-indicator* field of the program information block. In our sample transaction, there is still more processing to do. So, the code that CUST01 moves to the program information block is E (for external succession).

Specifying termination code

External succession is needed because the transaction requires more input from the terminal to continue processing. External succession is the only type of termination that allows for more terminal input (see 3.10).

Naming successor program

Since a successor program does the processing that follows, CUST01 must identify it by moving its name to the *successor-id* field of the program information block. The successor program is CUST02.

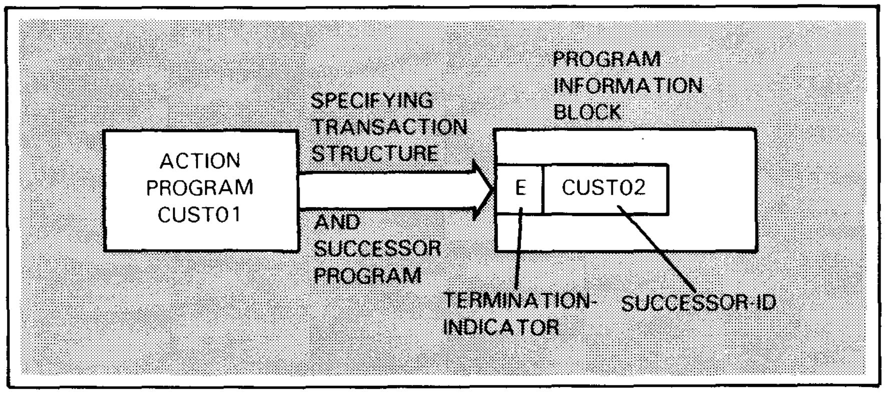


Figure 3-10. Action Program Identifies Termination Type and Successor Program

IMS Functions at Program Termination

Examining the program information block

When CUST01 finishes processing, IMS performs several functions. First, it looks at the program information block to see what transaction structure was specified. It sees the E (for external succession) and knows to expect more input from the terminal. In addition, it sees the name of the successor program, CUST02, and knows that this program will continue processing this transaction.

Examining the output message area

Next, IMS examines the output message area to see if there is a message to be sent to the terminal. Since there is (the customer account balance), IMS internal message control directs the message to ICAM, the communications network. ICAM, in turn, sends it to the terminal. (See Figure 3-11.)

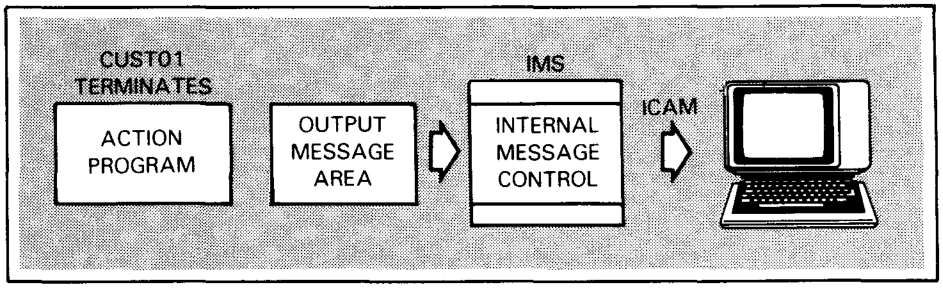
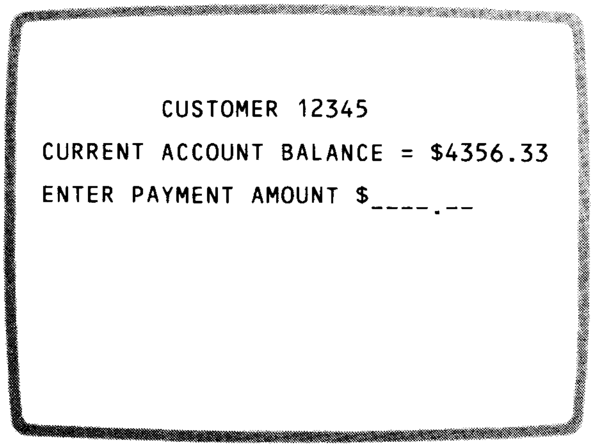


Figure 3-11. Output Message Goes to Terminal when Action Program Terminates

Transmitting the output message

The following screen shows the output message that CUST01 generated. It displays the customer account balance requested and also asks for additional data. This data is the input to the successor program, CUST02.



```
             CUSTOMER 12345
CURRENT ACCOUNT BALANCE = $4356.33
ENTER PAYMENT AMOUNT $____.__
```

Output Message for CUST01

3.13. PROCESSING THE SUCCESSOR PROGRAM

Receiving more terminal input

When the terminal operator enters the payment amount, IMS accepts it and schedules CUST02. This second input message is handled differently from the first. The first input message was the transaction code. It was validated and the appropriate action program scheduled.

Scheduling the successor program

Now the transaction is in progress. IMS knows that the second input message is destined for the successor program. Consequently, as soon as the input is received, IMS schedules the successor, CUST02, sets up an activation record for CUST02, and places the input in CUST02's input message area (Figure 3-12). In this case the input message is data, not a transaction code.

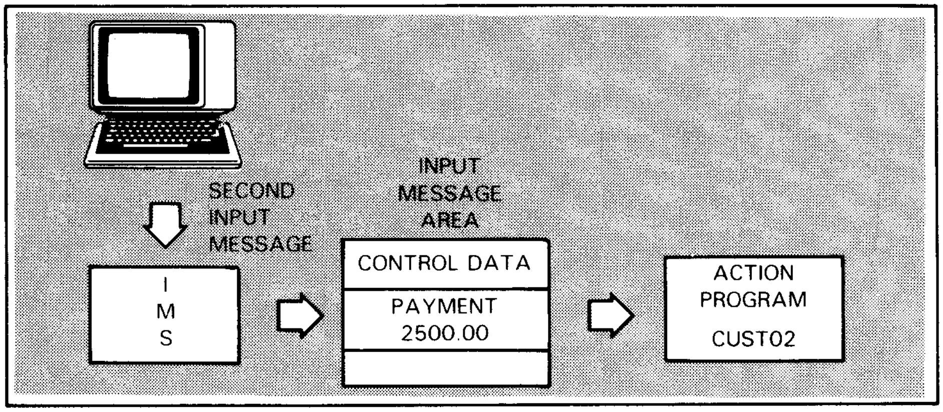


Figure 3-12. Accepting the Second Input Message in an IMS Transaction

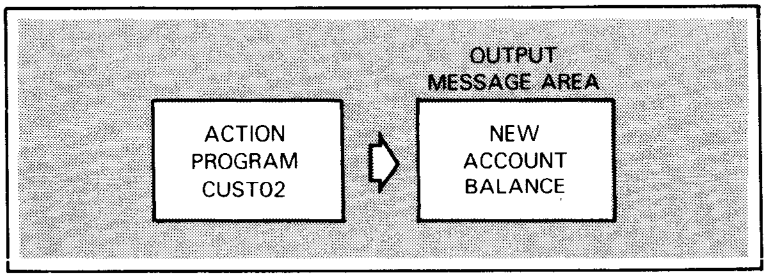
Processing the second input message

File processing

Once CUST02 begins processing, the sequence of events is similar to what happened in CUST01. The program reads the input message area to find out how much was paid against the account balance. CUST02 computes a new account balance and writes it to the file containing customer account records. IMS, by way of OS/3 data management, handles the actual writing of the record.

Generating the second output message

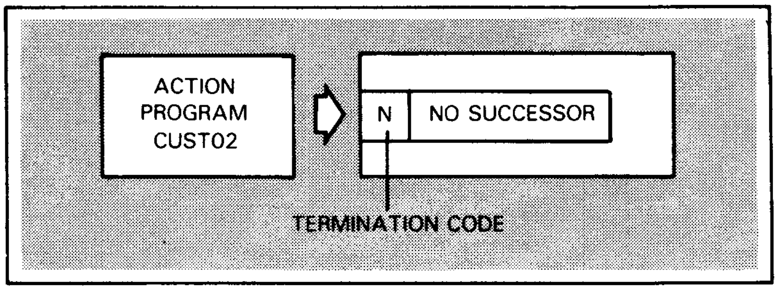
The program then generates an output message stating the new balance. This message is created in the output message area and remains there until the program terminates.



CUST02 Generates an Output Message

Terminating the transaction

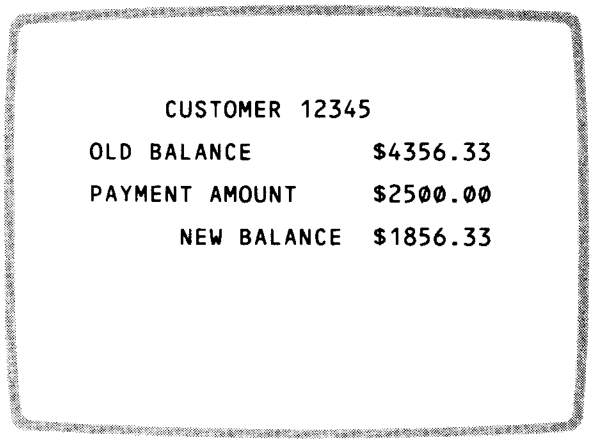
Before termination occurs, however, CUST02 must inform IMS what transaction structure it is creating. Since processing of this transaction is complete, CUST02 moves N (for normal termination) to the *termination-indicator* field of the program information block. If CUST02 doesn't move a value to this field, IMS assumes normal termination anyway. Since there is no successor, the program doesn't move any name to the *successor-id* field.



CUST02 Identifies Termination Type as Normal

IMS closing message

When the program terminates, IMS sends the message in the output message area to the terminal (Figure 3-13) and deallocates all resources held by the program.



```
        CUSTOMER 12345
OLD BALANCE          $4356.33
PAYMENT AMOUNT       $2500.00
         NEW BALANCE $1856.33
```

Figure 3-13. CUST02 Output Message

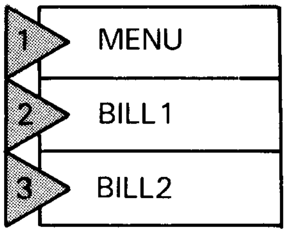
The receipt of the output message signifies that all IMS processing for this transaction is over and the operator is free to enter another transaction code.

3.14. A MORE COMPLEX IMS TRANSACTION

In 3.11 external succession and normal termination are used to create transaction structures. Now, consider a more involved example that uses all the types of transaction structures within one transaction. This may sound very complex; really, it is not. The example we use is lengthy but the processing steps you have already observed are basically the same.

Sample transaction #2

Our sample transaction uses three action programs:



Processing the transaction code

The transaction begins when the transaction code MENU is entered at the terminal. IMS validates this code by checking the transaction code table. If it is valid, IMS determines the first program to process the transaction and what its requirements are.

Loading the action program

(We assume this has all been done and action program MENU and its activation record are loaded in main storage.)

Processing for Program MENU

Processing for program MENU – Step 1

MENU displays a menu screen at the terminal. It does no file processing. As soon as the program receives the input message (the transaction code), it generates an output message in the output message area of the activation record.

Rescheduling an action program

Then, MENU creates a transaction structure. It tells IMS there is more processing to come by specifying the termination code E for external succession. (This code means that more input is required from the terminal operator.)

MENU also identifies the successor program. In this case, the successor is MENU (that is, MENU reschedules itself to continue the processing for this transaction). Figure 3-14 shows the processing for program MENU.

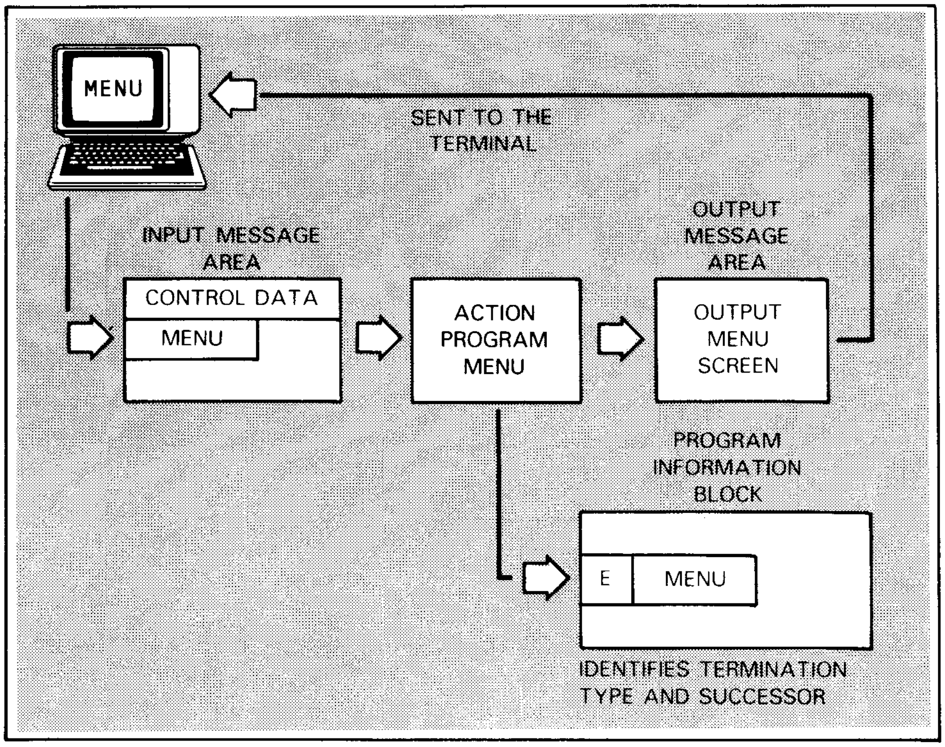
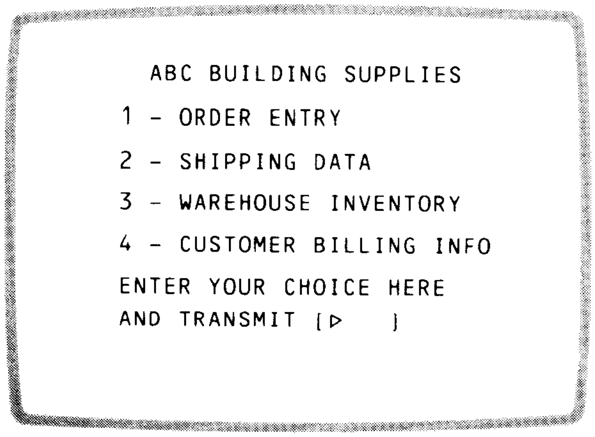


Figure 3-14. Processing for Action Program MENU - PASS1

As soon as the program terminates, IMS examines the termination code and successor name fields and then sends the output message to the terminal.

Output generated by MENU

Figure 3-15 shows the menu screen, output from the program MENU, as it appears at the terminal.



```
    ABC BUILDING SUPPLIES
 1 - ORDER ENTRY
 2 - SHIPPING DATA
 3 - WAREHOUSE INVENTORY
 4 - CUSTOMER BILLING INFO
 ENTER YOUR CHOICE HERE
 AND TRANSMIT [▷    ]
```

Figure 3-15. Output Menu Screen Generated by Program MENU

Entering data on the output screen

The terminal operator must choose one of the menu selections. This choice is the input to the successor program.

Processing the Menu Selection

Input received by successor

As soon as the input reaches IMS, MENU is rescheduled and begins processing for the second time.

Processing for MENU – Step 2

IMS places the terminal input in the input message area of the MENU activation record. This time MENU must determine what choice was made and to set up the transaction structure to process that choice.

Indicating appropriate successor program

We assume that the operator chose menu selection 4, CUSTOMER BILLING INFO. Once the program identifies the choice, it moves the appropriate termination code and successor program name to the *termination-indicator* and *successor-id* fields of the program information block to continue the processing. Figure 3-16 illustrates this process.

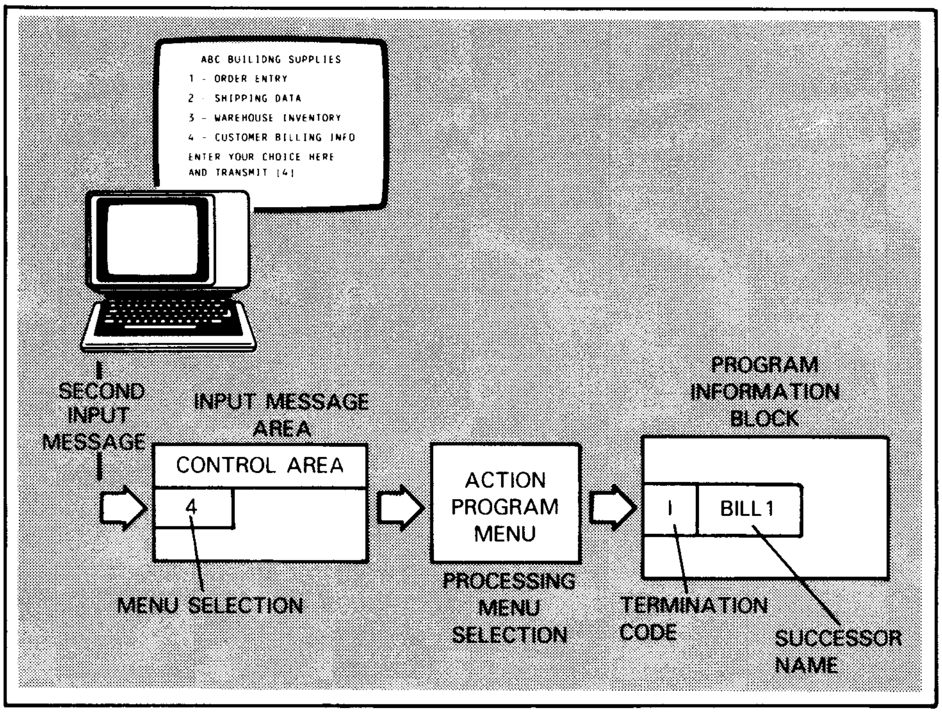


Figure 3-16. Processing for Action Program MENU – PASS2

Using Immediate Internal Succession

Using immediate internal succession

In this example, MENU names BILL1 as the successor program. BILL1 processes customer billing information. The termination code is I (for immediate internal succession).

When an action program uses immediate internal succession, it tells IMS that it does not need any more terminal input to continue processing. And, it wants processing to begin immediately after the current program terminates, without interruption. (See Figure 3-17.)

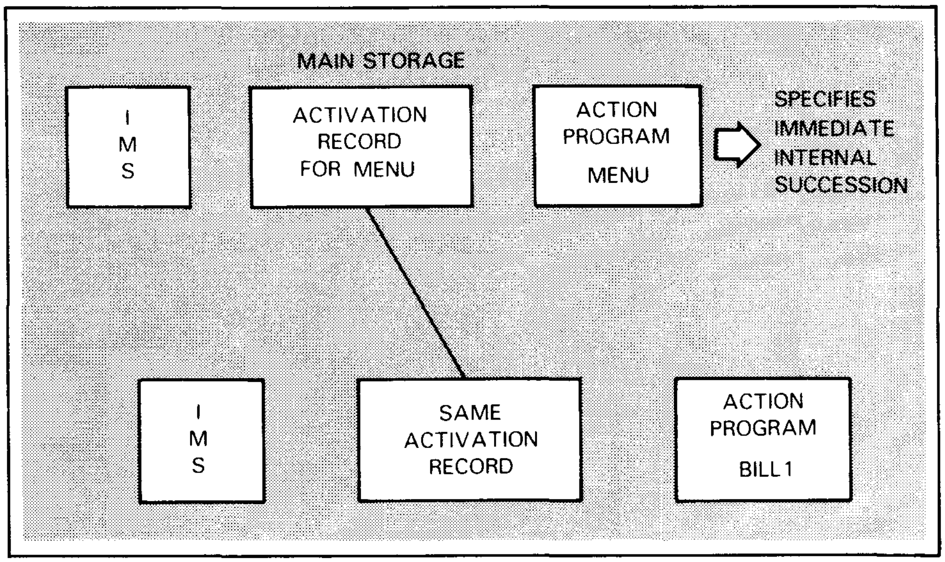


Figure 3-17. Activation Record with Immediate Internal Succession

Using the same activation record

In immediate internal succession, the activation record used by action program MENU remains in main storage and becomes the activation record for the successor, BILL1. This only occurs with immediate internal succession. Consequently, data in the activation record from MENU is immediately accessible to BILL1.

Processing for BILL1

Processing summary for program BILL1

BILL1 is designed to do two things:

1. On the first pass through, it creates a billing screen as output. This screen goes to the terminal when the program terminates

2. On the second pass through, it validates the terminal input entered on the billing screen.

Here's how that happens. . .

BILL1 processing – Step 1

Generating a billing screen

When MENU terminates, IMS immediately schedules BILL1. Then, BILL1 does no file processing but, instead, generates an output message which is a billing screen. This screen requires the terminal operator to enter billing information about a specific customer (Figure 3–18).

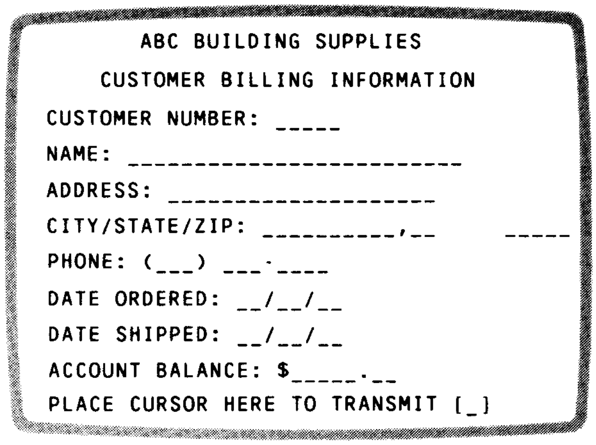
Creating a transaction structure

Transmitting the billing screen

Before the screen is sent to the terminal, BILL1 (like all action programs) creates a transaction structure. Since BILL1 requires more input from the terminal, it specifies E for external succession. And since BILL1 validates the terminal input, it names itself as the successor program.

Input to the billing screen

When the program terminates, the billing screen appears at the terminal. (The dotted lines indicate input to be entered by the terminal operator.)



```
           ABC BUILDING SUPPLIES
        CUSTOMER BILLING INFORMATION
 CUSTOMER NUMBER: _____
 NAME: _________________________
 ADDRESS: ____________________
 CITY/STATE/ZIP: __________,__     _____
 PHONE: (___) ___·____
 DATE ORDERED: __/__/__
 DATE SHIPPED: __/__/__
 ACCOUNT BALANCE: $_____.__
 PLACE CURSOR HERE TO TRANSMIT [_]
```

Figure 3–18. Output Billing Screen Generated by Program BILL1

BILL1 processing – step 2

When this input reaches IMS, BILL1 is rescheduled. When the same program is used to do different types of processing (as MENU and BILL1 are), the program first checks to see what pass it is in. In this way it determines the processing requirements for this run.

Validating billing input

On the second pass through BILL1, the program validates the input coming from the terminal. For instance, it checks to see if the customer name and number already exist on the billing file. If they do, the program returns an error message to the terminal indicating the presence of duplicate records.

Creating a second transaction structure

If all the terminal input is valid, BILL1's processing is complete and it must set up the next transaction structure. This time BILL1 specifies D for delayed internal succession as the termination code and BILL2 as the successor program.

Using Delayed Internal Succession

Using delayed internal succession

You use delayed internal succession when you don't need input from the terminal and you want to use the output message created by the current program as input to the successor. Let's look at how BILL1 and BILL2 do this.

Writing data to the output message area

If all the terminal input is valid, BILL1 takes the data and writes it to the output message area. Then, the program sets up its transaction structure.

Naming the successor program

BILL1 specifies D for delayed internal succession as the termination code and BILL2 as the successor program. Figure 3-19 shows the processing for BILL1.

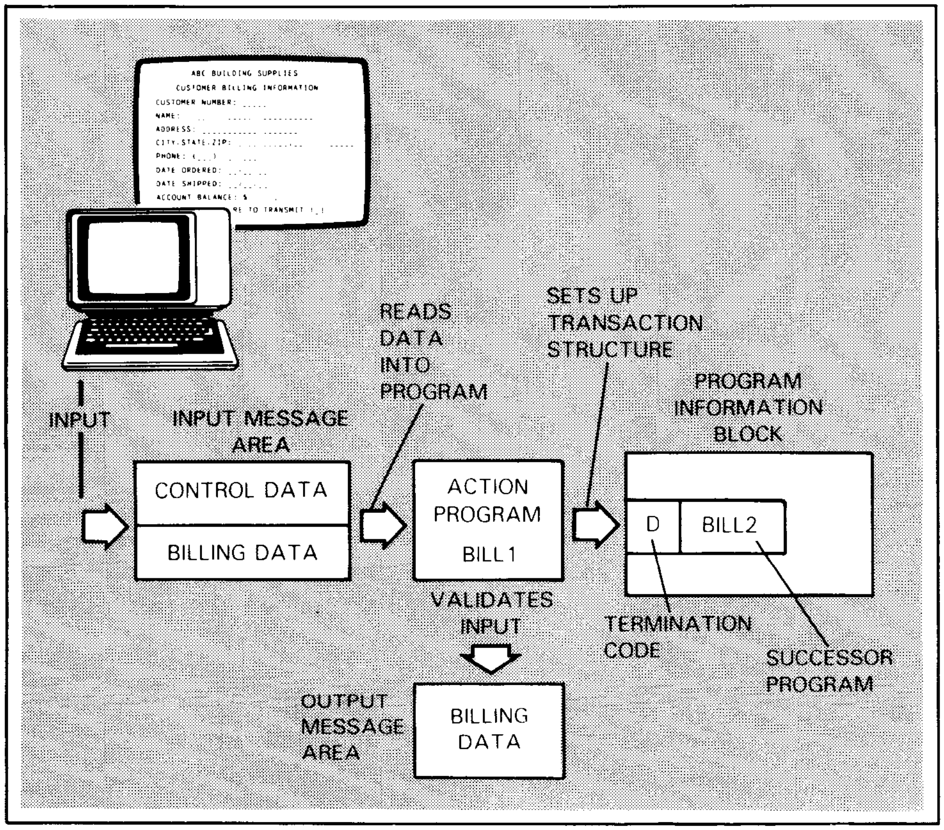


Figure 3-19. Processing for Program BILL1

This is the first time we've seen delayed internal succession used. What does it do? First, it says that BILL1 is generating an output message. But *that* message is not to be sent to the terminal; instead, it is to be queued by IMS as input to the successor program, BILL2.

Handling output with delayed internal succession

When BILL1 finishes processing, IMS sees the termination code D and moves the contents of the output message area to the input buffer until resources are available to schedule BILL2. Once BILL2 is loaded in main storage, the queued message is moved to the input message area of BILL2. In this way, the customer billing data validated by BILL1 is now available to BILL2 (See Figure 3-20).

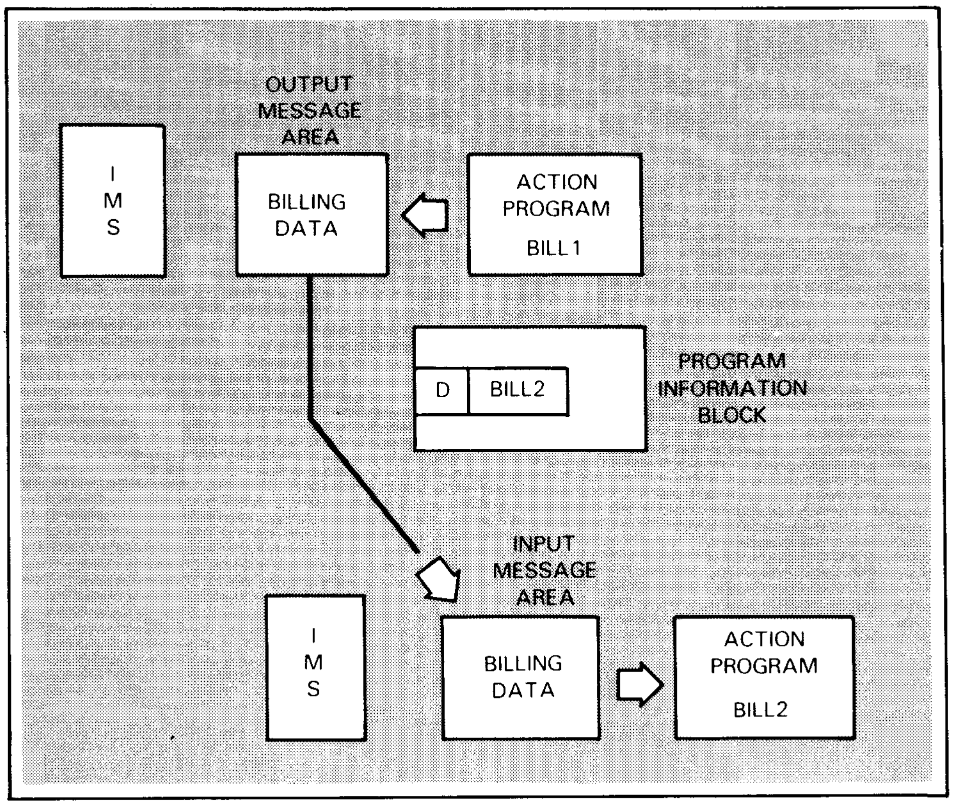


Figure 3-20. Data Available to Successor BILL2

Processing for BILL2

BILL2 begins processing by reading the input message area. Then, the programs calls upon IMS file management to add a billing record or to update the customer billing file.

Terminating the transaction

When updating is complete, BILL2 informs IMS all processing is over. BILL2 can specify N for normal termination or not. In either case, IMS normally terminates the transaction and sends a message to the terminal indicating the transaction is finished. Although IMS sends this message, it is better programming practice to end transactions with your own output message.

IN SUMMARY

We introduced many important concepts in this section. You now know that there are two types of transactions: **simple** and **dialog**. All transactions, no matter what type, are built around the **action**, the unit of work in IMS. An action occurs each time there is **input**, **processing**, and **output**.

Action programs are the vehicle for handling input, processing, and output. They do this by communicating with IMS. All this communication takes place in a special area that action programs set up, called the **activation record**. All data that action programs and IMS exchange passes through the activation record.

One important exchange of data is information about the type of **transaction structure** the action program is creating. How the four types of transaction structures are used determines all transaction processing.

4. Generating an Online IMS

Introducing the configuration process

We've mentioned several times that IMS is a group of software components that you can adapt to your needs. You do this using what is called a **configuration process.** The configuration process, however, is only one step among several needed to generate online IMS.

In this section, we talk about generating your online system. The intent is to provide you with an overview of the steps involved. We talk about creating a communications network and we describe your files and application programs and define all the optional features needed to support online IMS.

Additional information

The current version of the IMS system support functions user guide, UP-8364, discusses the online procedure in detail. Here we provide an overview of the process to familiarize you with the steps involved.

4.1. THE SYSTEM GENERATION STEP (SYSGEN)

Creating a supervisor

The online generation process starts at system generation (SYSGEN). When you first define the capabilities of your computer system, you enter a series of special instructions that, once executed, create a supervisor. The supervisor is a program that controls the activity of other programs operating within the system. It regulates the flow of work to maximize productivity.

Phases of SYSGEN

To use IMS, you must create a supervisor that supports IMS. You do this during three phases of the system generation process:

- SUPGEN
- RESGEN (Applies only to Series 90 systems)
- COMMCT

SUPGEN

SUPGEN phase

During the SUPGEN phase, you specify such things as:

- the number of communication lines
- priority levels for IMS tasks
- number of job slots required
- file locking capabilities (automatically included in System 80)
- in a Series 90 environment DTF, CDM, or mixed data management mode based on the organization of your data files (System 80 environment uses CDM mode only).

RESGEN

RESGEN phase

You use the RESGEN phase to generate you own system resident (SYSRES) disk pack. If you choose, you can use the OS/3 release volume (OS/3REL) as your permanent SYSRES; however, if you decide to generate your own, you must include a RESGEN section in your system generation. These considerations do not apply to System 80 users.

COMMCT

COMMCT phase

You use the COMMCT phase to generate your communications network and an ICAM load module that supports your online IMS. This can be done at the same time you generate a supervisor or later in a separate SYSGEN.

4.2. THE IMS LIBRARY

Online modules

Offline modules

As part of your IMS software, there is a collection of source, object, and load modules that constitute an IMS library. Many of these modules become part of your online IMS. They perform such functions as applications management and internal message control. There are also offline modules that handle recovery and statistical reporting operations.

Pre-online library modules

In addition to the online and offline modules, the IMS library includes the following programs for pre-online processing (Figure 4-1).

- Configuration programs:
  - ZS#CNF for single-thread IMS.
  - ZQ#CNF for multithread IMS.

  The configurator program is executed by an IMS jproc, IMSCONF.

- The NAMEREC file utility (ZP#NRU) initializes the named record file (NAMEREC) and generates passwords.
- The data definition processor (DT3DF) is used to create defined files.
- The edit table generator (ZH#EDT) controls message editing and validation.

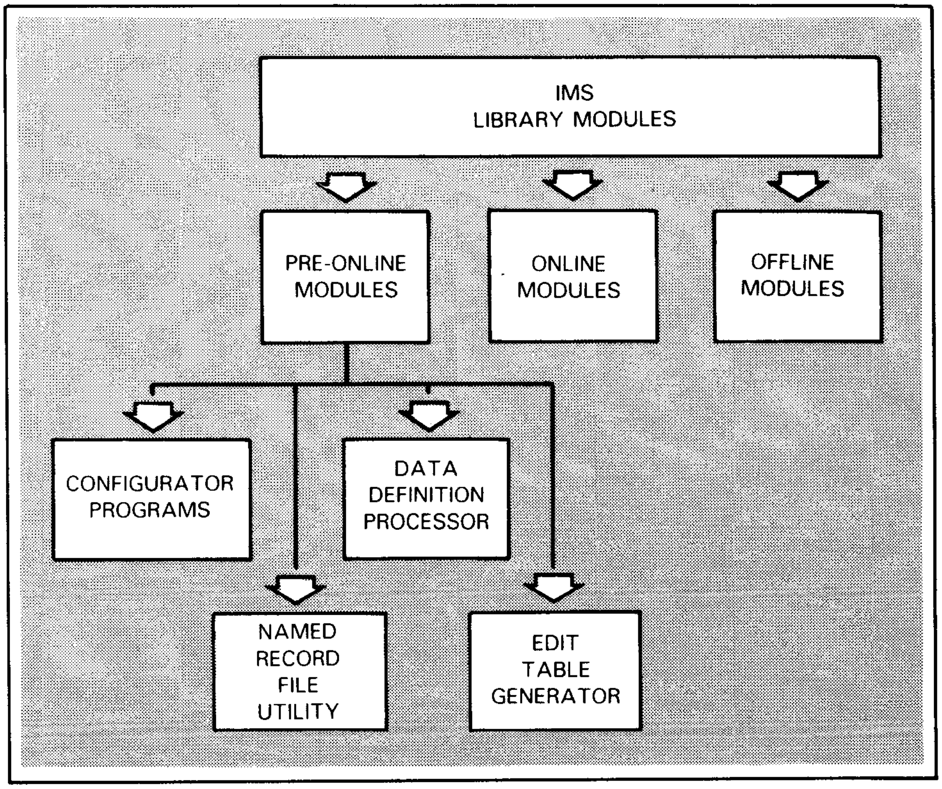


Figure 4-1. Pre-Online Library Module

4.3. PREPARING FOR ONLINE PROCESSING

Pre-online processing

Depending on the requirements of your applications, you must perform some or all of the following functions in preparing for online processing:

- Initialize the named record (NAMEREC) file, which will contain all the tables and records needed by online IMS. This can be done as part of the configuration process or separately with the NAMEREC file utility, ZP#NRU. Required.

- Generate passwords for use with UNIQUE, using the same NAMEREC file utility. Not required.

- Write data definitions for defined files (required for UNIQUE and optional for use with your own action programs). Data definitions are processed by the data definition processor, DT3DF. Not required.

- Write action programs in RPG II, COBOL, or BAL to handle specific applications. These programs are compiled and linked in a process separate from configuration. Required unless you're using UNIQUE only.

- Generate edit tables for use with your action programs, using the edit table utility, ZH#EDT. These tables simplify input message editing and validation. Not required.

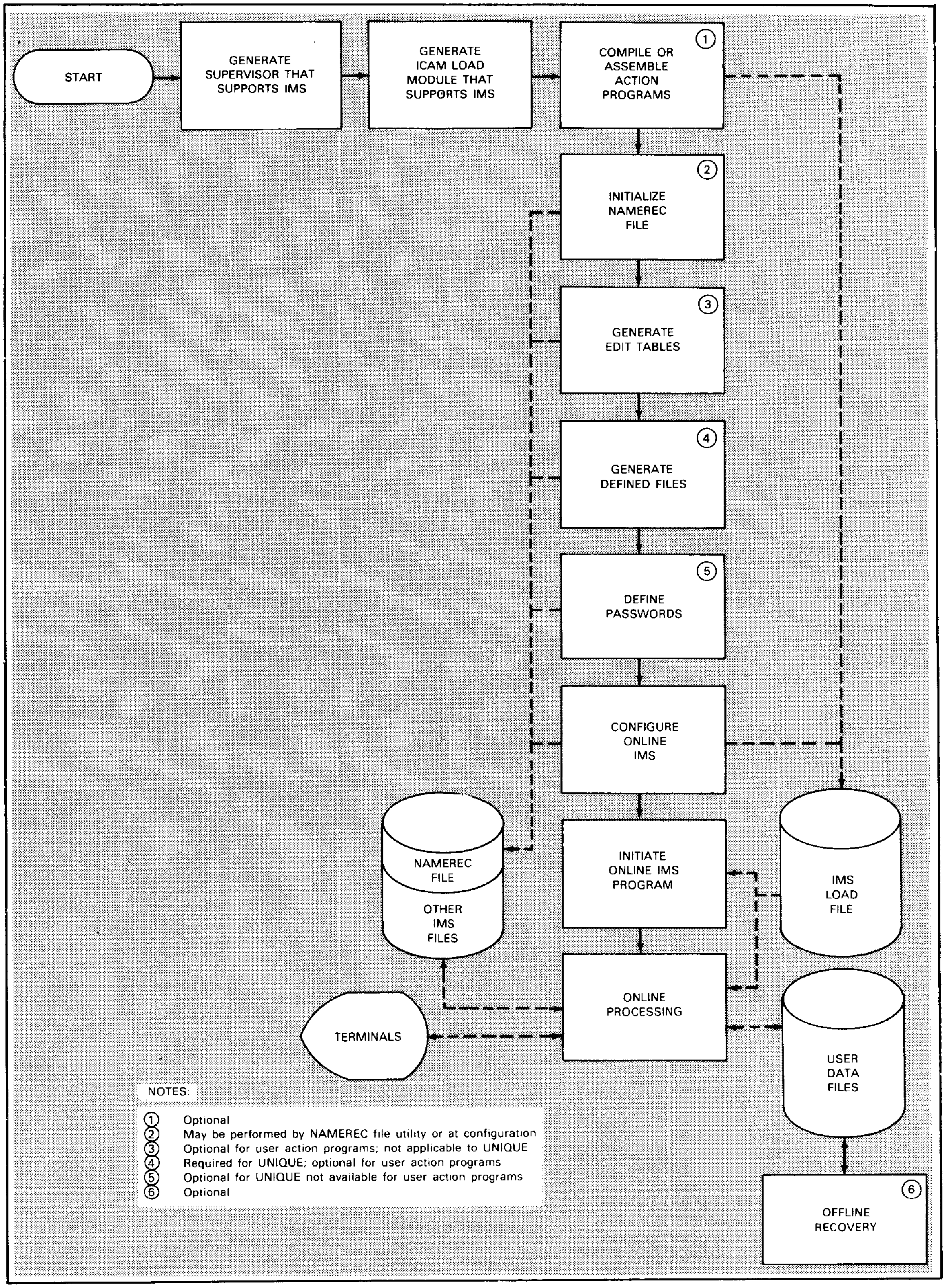
- Configure an online IMS load module, using the IMS jproc, IMSCONF. The configuration process also allocates and initializes the internal files needed by online IMS. Required.

Processing sequence

Figure 4-2 illustrates in flowchart form IMS pre-online and online processing and offline recovery functions.

The pre-online sequence shown in Figure 4-2 is interchangeable, with three exceptions. You must:

- Generate the supervisor as the first step.
- Generate the ICAM load module before configuration.
- Initialize the NAMEREC file before or during configuration and before processing data definitions, passwords, or edit tables. If the NAMEREC file is reinitialized at any time, repeat all pre-online processing, including configuration.



NOTES:

① Optional
② May be performed by NAMEREC file utility or at configuration
③ Optional for user action programs; not applicable to UNIQUE
④ Required for UNIQUE; optional for user action programs
⑤ Optional for UNIQUE not available for user action programs
⑥ Optional

Figure 4–2. IMS Processing

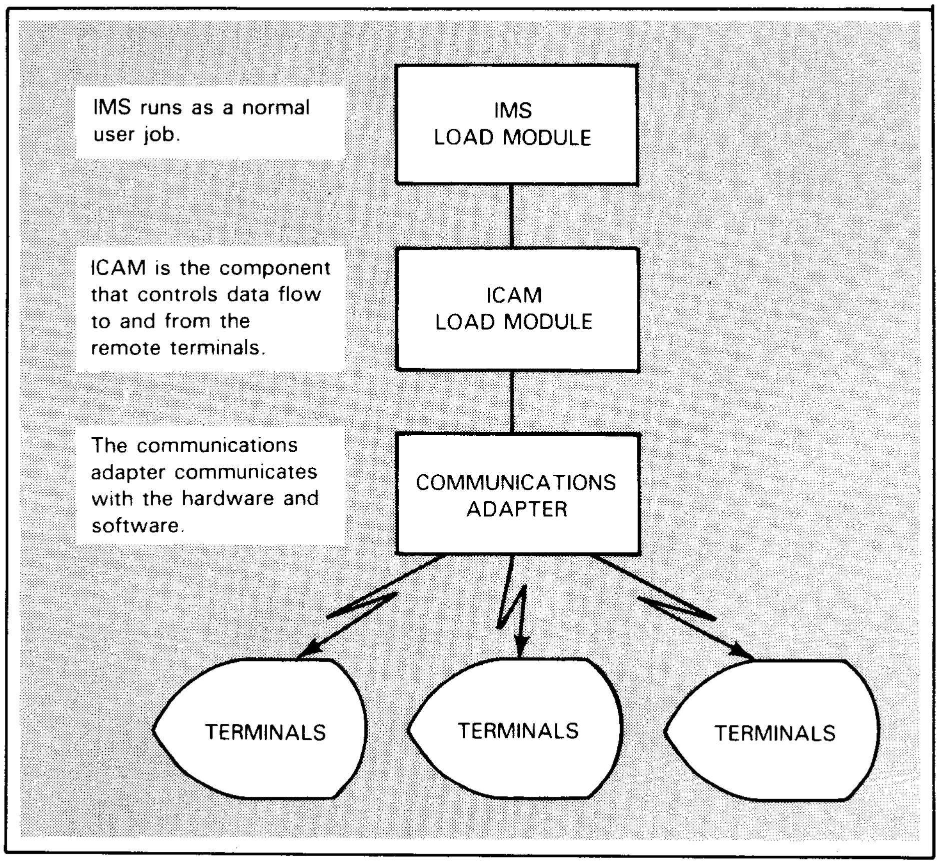
4.4. CREATING A COMMUNICATIONS SYSTEM

Communications environment

When you use IMS online, it is part of a communications system. At one end of the system is your configured IMS load module. At the other end are your terminals. In between is a communications adapter, a hardware device that takes messages coming from terminals and puts them in a form usable by the operating system and vice versa.

Also, there is the integrated communications access method (ICAM). This is a software component that controls the input from (and output to) the terminals.

Because of ICAM, IMS is device-independent. This means that you can have any arrangement of supported communications lines and terminals without IMS or your action programs having to concern themselves with the hardware they're working with.


IMS runs as a normal user job.
IMS LOAD MODULE
ICAM is the component that controls data flow to and from the remote terminals.
ICAM LOAD MODULE
The communications adapter communicates with the hardware and software.
COMMUNICATIONS ADAPTER
TERMINALS
TERMINALS
TERMINALS

IMS and the Communications Environment

Transaction control interface

ICAM makes all communications lines and terminals compatible with IMS. This is accomplished through a special program designed to interact with IMS. This program is called the transaction control interface (TCI) portion of ICAM. This interface was created for use by IMS.

4.5. DEFINING SOME TERMS

There are a number of terms that you hear or see quite regularly whenever the subject is communications. We'll try to acquaint you with a few here.

Defining RESIDENT and TRANSIENT

Your ICAM load module can be transient or resident. If it is **resident**, it remains in main storage at all times. As a result, you get faster processing of transactions. If your ICAM load module is **transient**, it occupies much less main storage then the resident version thereby freeing up main storage for other programs. Transient ICAM is not available with System 80.

Using resident and transient ICAM

Multithread IMS requires a resident ICAM. Single-thread IMS can use either a transient or resident ICAM; but single-thread supporting unsolicited output requires a resident ICAM. The following listing summarizes these residency requirements.

ICAM Residency Requirements

Type of IMS System	Type of ICAM Module
Single-thread (Series 90)	Transient or resident
Single-thread (System 80)	Resident
Single-thread with unsolicited output capability	Resident
Multithread	Resident

Defining dedicated and global

There are two types of ICAM networks that you can define – a dedicated or a global network.

1 A **dedicated** network supports only IMS. All lines and terminals defined for your network are dedicated to IMS for the length of the session.

2 A **global** network supports IMS and other communications programs at the same time.

4.6. SETTING UP THE COMMUNICATIONS ENVIRONMENT

Defining communications support

To create a communications environment for use with IMS, you must define the support you require. You do this with a set of ICAM network definition macroinstructions and system generation message control program parameters. This is not a complicated process.

Message control parameters

The message control program parameters name the ICAM load module, identify the disk volume where the load module will be stored, and define the communications adapters associated with your network.

Network macroinstructions

The ICAM network definition macroinstructions basically define three things:

1. Type of ICAM support network used
2. Network of lines and terminals
3. Input/output requirements

Designating Global or Dedicated Network

CCA macroinstruction

You use the CCA (communications control area) macroinstruction to specify whether you want a global or dedicated network.

Defining the Arrangement of Lines and Terminals

LINE and TERM macroinstructions

You use LINE and TERM macroinstructions to define for ICAM the arrangement of terminals that will be sending and receiving data for IMS transactions.

Defining Input/Output Message Handling

DISCFILE and BUFFERS macroinstructions

You use DISCFILE and BUFFER macroinstructions to define how input and output messages will be handled. These macroinstructions set up the buffers needed to temporarily store the messages. The use of these macroinstructions differs depending on whether you're using a transient or resident ICAM.

All the macroinstructions mentioned plus additional macroinstructions are discussed in detail in the IMS system support functions user guide, UP-8364 (current version).

Producing the ICAM object module

The network definition macroinstructions are entered during the COMMCT phase of SYSGEN. Once executed, these instructions produce an ICAM object module. This object module is the CCA. During the configuration process, the CCA object module is linked and the resulting load module is stored in the library with the configurator program.

The configurator uses the tables in the CCA load module to validate some of the configurator parameters. During the COMMCT phase of SYSGEN, the CCA object module is linked with the MCP parameters to create the ICAM symbiont which is your communications system.

4.7. PRE-ONLINE PROCESSING – SETTING UP IMS

Previously in this section we talked about the IMS library modules you receive as part of your IMS software and the programs they contain. We mentioned that many of these programs become part of your online IMS load module as a result of pre-online processing.

Pre-online processing

Pre-online processing includes:

- allocating and initializing the named record file (NAMEREC), using the IMS utility program ZP#NRU;
- defining passwords, using the ZP#NRU utility program;
- creating data definitions, using the data definition processor DT3DF;
- generating edit tables, using the IMS utility program ZH#EDT; and
- configuring the online IMS system.

Requirements

All pre-online processing, except configuration and setting up the NAMEREC file, is optional. For instance, password definition is not required and is used only with UNIQUE. Data definition is required only if you're using UNIQUE or if your action programs access defined files. Edit table generation is optional and is available to user-written action programs only.

Initializing the named record file

Initializing the named record file is required. You can do it as part of the configuration process or separately by running the utility program ZP#NRU . In either case, you must initialize the named record file before any other pre-online processing can take place because all IMS programs, including the configurator, contribute information to this file.

We briefly discuss the use of the edit table generator and data definition processor and refer you to later sections of the manual where they are covered in greater detail. Here our discussion centers around the required parts of the IMS online generation process – initializing the named record file (NAMEREC) and configuring IMS.

4.8. EDIT TABLES AND DATA DEFINITIONS

Using the edit table utility

The edit table utility (ZH#EDT) provides a means for editing input from terminal operators and doing validation checks. Its use is optional. You define to the edit table generator the form in which you want input data to appear. The data you provide is stored on the named record file (NAMEREC). When input comes in from a terminal, IMS uses the format description stored on NAMEREC to verify the input based on your specifications. A further description and examples of how to use the edit table utility are included in Section 8.

Using the data definition processor

The data definition processor (DT3DF) is used to create defined files. You must create defined files if you intend to use UNIQUE, the IMS-supplied package of programs. UNIQUE accesses defined files only. Action programs that you write yourself can also access defined files.

The data definition records produced by the data definition processor are written to the named record file. For a description of how to create defined files, see Section 10.

4.9. SETTING UP THE NAMED RECORD FILE (NAMEREC)

Function of NAMEREC

As you may recall, we introduced the named record file (NAMEREC) in Section 1 and further described its use in Section 3 when we talked about action programs. By now you are well aware that NAMEREC is an important IMS internal file. It contains the records and tables needed for online IMS processing. These include:

- Configuration tables (transaction, action, and program control tables)
- Data definition records
- Edit tables
- Password definition tables

Initializing NAMEREC

Before NAMEREC can receive this data, you must initialize it. To initialize NAMEREC, you can:

- run the named record file utility, ZP#NRU; or
- specify the INIT parameter of the IMSCONF job control procedure at configuration.

Running the named record utility

When you choose the second option, IMSCONF calls in the ZP#NRU program to initialize NAMEREC as part of the configuration process.

When you run the named record utility separately, you initialize NAMEREC and in addition you can:

- Specify the exact location on disk where the file will reside
- Define passwords and change existing passwords for defined files
- List the contents of the NAMEREC file
- Process data definitons, password definitions, and edit tables before configuration

4.10. DEFINING PASSWORDS

Defining passwords

In addition to uses described in 4.9, you can also use the named record file utility (ZP#NRU) to define passwords. The password definition process, an optional part of pre-online processing, offers data security for defined files when used with UNIQUE. Use of passwords is not available to user-written action programs.

Limited access

When you define passwords for your defined files, terminal operators can access only to those files for which they know the password.

PASSWORD parameter

To define passwords, you insert the PASSWORD function parameter in the job control stream to execute the named record file utility. This parameter names the password, associates it with a particular defined file or subfile, and identifies which terminals can access that file. Using the same parameter, you can also delete an existing password and substitute another in its place.

Other options

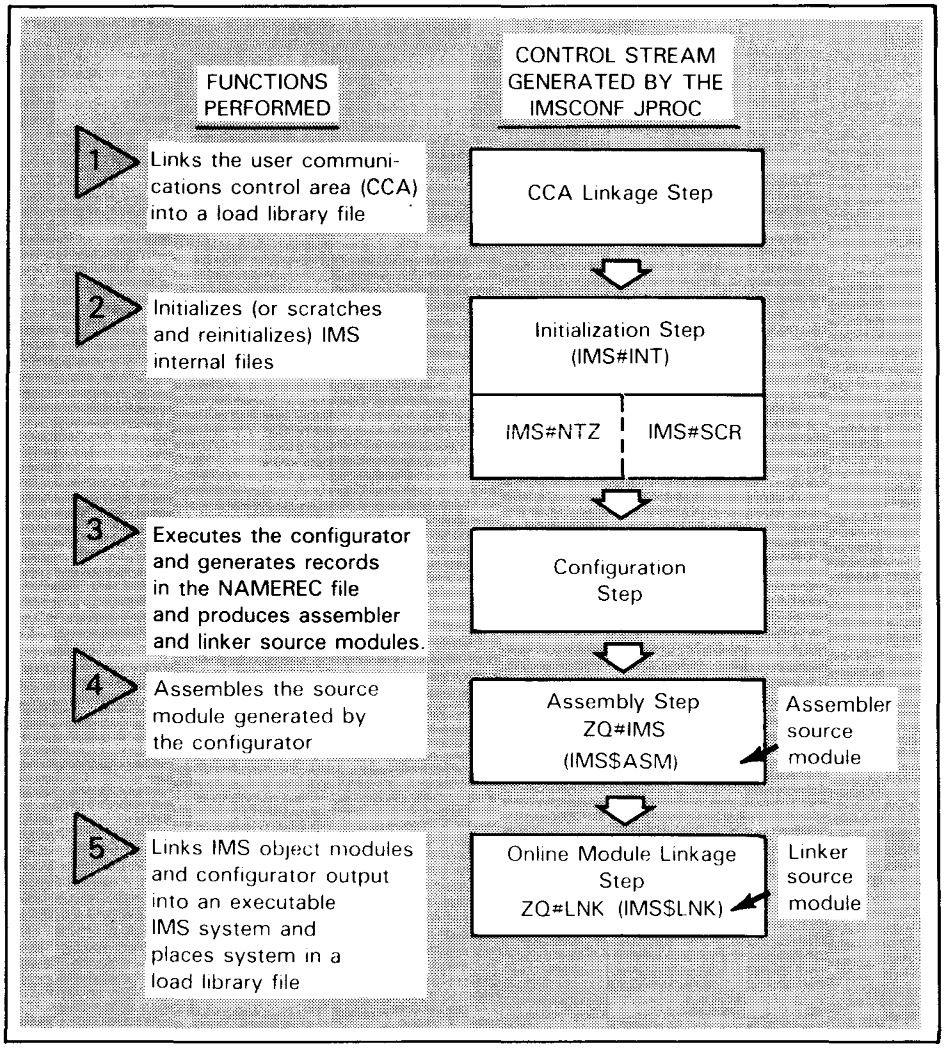
You can also define passwords using the PASSWORD option available through the data definition processor at the same time that you're creating defined files. However, this option doesn't allow you to restrict access to specific terminals, nor to change passwords. For a discussion of how to create passwords using the data definition processor, consult the IMS data definition and UNIQUE user guide, UP-9209 (current version).

4.11. CONFIGURING IMS

Configuration process

Configuring IMS is the most important step in generating your online system. It allows you to tailor IMS to your needs. One configuration process generates a single-thread or a multithread IMS. An IMS-supplied job control procedure (jproc) named IMSCONF generates and executes the IMS configurator program, produces and links the IMS load module, and stores it in the library you specify. The IMSCONF jproc consists of five parts:

IMSCONF jproc


FUNCTIONS PERFORMED
CONTROL STREAM GENERATED BY THE IMSCONF JPROC
1
Links the user communications control area (CCA) into a load library file
CCA Linkage Step
2
Initializes (or scratches and reinitializes) IMS internal files
Initialization Step (IMS#INT)
IMS#NTZ
IMS#SCR
3
Executes the configurator and generates records in the NAMEREC file and produces assembler and linker source modules.
Configuration Step
4
Assembles the source module generated by the configurator
Assembly Step ZQ#IMS (IMS$ASM)
Assembler source module
5
Links IMS object modules and configurator output into an executable IMS system and places system in a load library file
Online Module Linkage Step ZQ#LNK (IMS$LNK)
Linker source module

The functions are further defined:

1. CCA Linkage

CCA linkage

The CCA linkage step links the communications control area object module created during SYSGEN (COMMCT) and stores the output in the load library containing the configurator program. The configurator uses data in the communications control area to validate the NETWORK section of your IMS configurator input.

2. Initialization

Initialization

Initialization allocates and initializes the IMS internal files. We already talked about initializing the named record file separately using the named record file utility. You can also initialize the named record file and other internal files during the configuration. Later in this section we discuss initializing other IMS internal files.

3. Configuration

Configurator program

Configuration executes the configurator program, analyzes and processes input to the configurator (all the options you specify to support your IMS), generates records for the named record file, and creates source modules for the assembler and linkage editor.

4. Assembly

Assembly

The assembly phase takes the configurator output, which includes information on your data files, and converts it into machine-readable code.

5. Online Module Linkage

Online module linkage

Online module linkage takes the output of the configurator (ZQ#LNK, IMS$LNK) and assembly operations and links it with IMS library modules into an executable IMS load module and stores it in a load library.

4.12. USING THE IMSCONF JOB CONTROL PROCEDURE

Using the IMSCONF jproc

To configure IMS you call upon the IMSCONF jproc to generate and execute the control streams that handle the five steps described in 4.11. You call upon IMSCONF as you would any job control procedure. The keyword parameters you specify determine which job control streams are activated.

All keywords are optional and every keyword, except for the INIT keyword, has a default value. When you don't specify a keyword, IMSCONF assumes you accept the default value. Figure 4-3 shows the IMSCONF jproc and the keywords supported. Note the default values (shaded terms).

```
// IMSCONF   [ALTER={NO }] [,CCA=([{cca-name}] [,{REL}])] [,CDM={NO }]
                    {YES}           {IMS4    }     {RES}            {YES}
                                                   {USR}

             [,CNFJCS=([ALL],[CCA],[CNF],[ASM],[LNK])]

             [,IMSFIL[n]=([{lun}][,{vsn   }][,{named-rec-file-id}]
                           {RES}    {SYSRES}    {NAMEREC          }
                         [,{audit-file-id}] [,{cont-data-file-id}]
                           {AUDCONF      }    {CONDATA          }
                           {AUDFILE      }
                         [,{stat-file-id}])]
                           {STATFIL     }

             [,INIT=({I}[,{blksize-namerec}] [,{blocks-namerec}]
                     {R}  {NO             }    {75            }
                     {S}  {3072           }
                          {6144           }
                   [,{cyl-audit-file}][,{cyl-condata}][,{cyl-statfil}])]
                     {NO            }   {NO         }   {NO         }
                     {15            }   {5          }   {10         }
                     {10            }

             [,INPUT={src-module-name}]
                     {CARD           }

             [,LIBL=([{lun}][,{vsn   }] [,{load-lib-name}])]
                      {50 }   {OS3REL}      {$Y$LOD       }
                      {RES}   {SYSRES}

             [,LIBO=([{lun}][,{vsn   }] [,{obj-lib-name}])]
                      {50 }   {OS3REL}      {$Y$OBJ      }
                      {RES}   {SYSRES}

             [,LIBS=([{lun}][,{vsn   }] [,{src-lib-name}])]
                      {50 }   {OS3REL}      {$Y$SRC      }
                      {RES}   {SYSRES}

             [,LOADM=online-load-module-name]

             [,LST={(A)                  }]
                   {NO                   }
                   {([C] [,D][,O][,S])   }
                   {(C,D,S)              }

             [,ZCNF=([{MT}][,{REL}])]
                      {ST}    {RES}
                              {USR}
```

Figure 4-3. Keywords Supported by IMSCONF jproc

Keyword parameters

Each of the keyword parameters that you see in Figure 4–5 and how you use each one is described in detail in the IMS system support functions user guide, UP-8364 (current version). Here is a brief description.

Parameter	Explanation
ALTER	Specifies whether any ALTER job control statements are needed. ALTER statements enable you to change a load module at execution time.
CCA	Identifies the communications control area (CCA) object module created during system generation (SYSGEN). This module describes your communications network. It is linked to create a load module so the configurator can obtain CCA information during configuration processing.
CDM	Designates CDM or DTF mode. Default is DTF.
CNFJCS	Designates specific job control streams to be run. Default runs all job control streams.
IMSFIL	Defines IMS internal files. The named record file (NAMEREC) is required. Depending on whether it is a single-thread or multithread IMS, there are other internal file requirements.
INIT	Generates the control stream for initializing IMS internal files. You can initialize the named record file in a separate process, if you choose.
INPUT	Designates whether configurator input is on cards or in the source library. If in source library, defines source module name.
LIBS, LIBO, LIBL	Direct the configurator program where to store and obtain the modules created during the configuration process. You can also use the LIBS keyword to tell the configurator where your configurator input is stored.
LOADM	Assigns a name to the online IMS load module being configured. You use this parameter to describe your installation's online system.
LST	Specifies various configuration listing options. Default gives you NAMEREC tables, configurator options, and input parameter cards read by the configurator.
ZCNF	Identifies whether you're configuring a single-thread or multithread IMS and identifies the library where the configurator modules are stored.

4.13. INPUT TO THE CONFIGURATOR

Types of input

During the third step of the configuration process, the configurator program processes input parameters. The input is divided into 12 sections:

NETWORK	Defines the ICAM network that supports online IMS and provides additional information about your IMS system. It must be the first section.
GENERAL	Describes general characteristics of the online IMS system, such as standard message size.
OPTIONS	Defines optional IMS features included in the configuration.
TIMEOUTS	Defines various timeout values.
FILE	Describes each user data file accessed by IMS. This parameter is entered once for each file.
TERMINAL	Further defines the terminals specified in the communications network definition.
TRANSACT	Supplies transaction code information to the configurator. It is coded once for each transaction.
ACTION	Describes the actions (units of work) used in this configuration. It is coded once for each action.
PROGRAM	Describes the user action programs included in this IMS configuration. It is coded once for each action program.
LOCAP	Defines a local application program (an IMS system) at a remote computer system. This input section is required with distributed data processing. It is coded once for each remote IMS.
LANGUAGE	Creates a UNIQUE lexicon record in the named record file. This record redefines UNIQUE language elements. For example, it replaces the UNIQUE English commands with French commands. It is coded once for each lexicon record.
DRCRDMGT	Defines the level at which use-defined record management is used. Defined record management handles the processing of defined files.

Options available

Under each input section there are numerous options from which you can choose. The IMS system support functions user guide, UP-8364 (current version) discusses these options in detail.

Action program description

In the FILE, TERMINAL, TRANSACT, ACTION, and PROGRAM sections, you describe the transaction codes, user-written action programs, files used, and terminals supported.

Analyzing input

The configurator program analyzes all this input and creates tables that are written to the NAMEREC file. These tables are read into main storage at IMS start-up time. When a terminal operator keys in a transaction code, IMS uses the data in these tables to determine processing considerations for that IMS transaction.

At this point the groundwork for online processing is complete. In Section 5 we discuss how you initiate your online IMS session.

5. Beginning and Ending IMS Sessions

In Section 4 you learned about the procedures needed to prepare your online IMS. In this section, we talk about how you make the online system operational and how you shut it down when processing is complete.

5.1. BEGINNING AN IMS SESSION

IMS startup

The term for bringing IMS online is **start-up**. There are two steps for start-up:

1 Establish the communications environment

2 Execute the IMS load module

Establish the Communications Environment

Loading ICAM

To do online processing, load the communications module (ICAM) into main storage. (This module was created during the COMMCT phase of the system generation process).

Using the console

You load ICAM from the system console by keying in the name you specified on the *mcpname* parameter in the COMMCT phase of SYSGEN. This name identifies the communications module.

When ICAM is successfully loaded, the message ICAM READY appears on the system console.

Loading a global network

If you're using a global network, you must execute the global user service task program (GUST) after loading ICAM. This program is called ML$$GI. A sample job control stream for executing this program is in the IMS system support functions user guide, UP-8364 (current version).

When the network is loaded, the following message appears on the system console:

```
MC#430 GUST ACTIVE FOR CCA network-name.
```

Now is the time to execute IMS.

Execute the IMS Load Module

Loading IMS

You execute IMS with a standard job control stream entered at the card reader or workstation. You must enter the name in the // EXEC job control statement specified on the LOADM parameter of the IMSCONF jproc at IMS configuration. (Remember that the LOADM parameter uniquely identifies your IMS load module. This is important since you may have several different IMS configurations.)

IMS READY message

When the online IMS module is loaded, the message IMS READY appears at the system console and at static terminals in the IMS network. (Static terminals are those terminals dedicated to IMS for the length of the session.)

The IMS READY message isn't displayed on dynamic terminals (terminals not dedicated to IMS) or if the IMSREADY=NO parameter was specified at IMS configuration. Once IMS is loaded, terminal operators can begin entering transaction codes.

There are special PARAM statements you can include in your IMS execution job control stream.

Special considerations

These statements allow you to specify:

- Whether IMS internal files and tables are to be initialized or information retained from a previous session (multithread IMS only)
- Rollback of data files at system startup
- Closing or extending of a disk trace file from a previous session
- Batch transaction processing in online or offline mode
- Routing characters to identify remote systems where transactions are routed in distributed data processing
- Different values for distributed data processing configuration parameters
- Monitoring of the IMS session to provide additional data when there is a problem with the system (multithread only)

For a discussion of these statements and sample job control streams for executing IMS, consult the IMS system support functions user guide, UP-8364 (current version).

5.2. ENDING THE IMS SESSION

IMS shutdown

The term for ending the IMS session is **shutdown**. You shut down IMS by entering one of the following commands at the master terminal:

- ZZSHD
- ZZHLT

You can also enter these commands at the system console or master workstation if OPCOM=YES was configured.

Orderly shutdown

The ZZSHD command shuts down IMS in an orderly fashion. No further transaction codes are accepted; ongoing transactions continue processing until **timeout** occurs. When no transactions are active, files are closed, communications lines deactivated, and IMS terminates normally.

Emergency shutdown

The ZZHLT command initiates an emergency shutdown procedure:

- No further transaction codes are accepted
- Transactions in progress are stopped
- Files are closed
- Communications lines are deactivated
- IMS terminates with a dump of main storage.

After this type shutdown, file recovery is usually needed.

6. Terminal Operations and I/O Message Processing

6.1. TYPES OF TERMINALS

You know from Section 1 that IMS is a **transaction processing** system. For every message you input, you receive some type of response or output message.

To input a message or receive an output message, you must have some way of communicating with your system. You communicate with your sytem through terminals.

Definition of terminals

A **terminal** is any point in a system or communications network where data can enter or leave. You identify the type of terminals you have by using the TERM macroinstructions when you define your ICAM network.

Use of terminals

Regardless of the terminal type you choose, there are two ways to use terminals:

Static Terminals

Static terminals

Static means that the terminals are attached to IMS for the entire IMS session. When you use static terminals, you cannot access any other program. Static terminals are most frequently used in dedicated networks where all terminals are static. However, you can also use static terminals in a global network. To do this, you define them using SESSION macroinstructions when you define your ICAM network.

Dynamic Terminals

Dynamic terminals

Dynamic terminals are only allowed in a global communications network. The advantage of dynamic terminals is that they're attached to IMS only while your're using them.

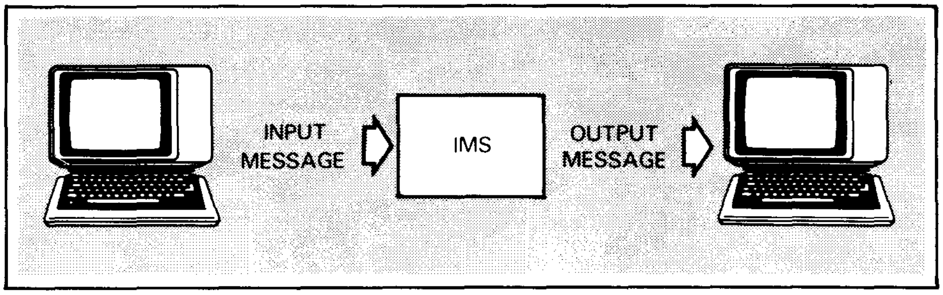
When you want to use IMS, you sign on the terminal to IMS by keying in $$SON. When you want to stop using IMS, you key in $$SOFF. Then you can use the terminal to access other programs that use the same global network.

You incorporate the capability for dynamic session establishment by specifying the GAWAKE operand in the ICAM (CCA) network definition macroinstruction. For more information on dynamic terminals, see the IMS terminal users guide, UP-9208 (current version).

6.2. TYPES OF INPUT AND OUTPUT MESSAGES

Medium for input and output

Regardless of the type of session you're using, whenever you use the terminal, you are either sending or receiving messages. Messages keyed in at the terminal are **input messages** and messages received at the terminal are **output messages.**



Input Messages

Types of input messages

You can generate two types of input messges:

1. Terminal commands
2. Transaction-related messges

Definition of terminal commands

Terminal commands begin with the letters ZZ and perform a variety of functions. There are two types of terminal commands:

Types of terminal commands

- Standard
- Master

Standard commands

You enter standard terminal commands from any IMS terminal. They let you control the flow of messages to and from each terminal.

Master commands

You enter master terminal commands from the master terminal and, in some cases, from the system console or master workstation. They let you control IMS and the communications network.

Transaction-related messages

You're already familiar with transaction-related messages. These include transaction codes that initiate IMS transactions and terminal input to successor action programs.

Output Messages

There are two types of output messages you can receive at the terminal:

1. Action program output
2. IMS output

Defining action program output

Action program output is composed of messages received at the terminal as a result of transaction processing.

Defining IMS output

IMS output is messages showing the status of an IMS transaction or messages related to terminal commands.

6.3. PREPARING THE MESSAGE PROCESSING ENVIRONMENT

To process messages, IMS must be ''up and running''. This task is generally performed by your system administrator and includes the startup procedures listed in Section 5.

When a terminal is ready

Generally, when you reach a terminal, it is sitting idle displaying the message: IMS READY.

You will not see this message if:

Your system administrator suppressed it with the IMSREADY=NO option at configuration time.

Your terminal is dynamic (not dedicated to IMS)

Assuming your terminal is ready, you can begin sending messages. But, before we go any further, let's talk about your terminal's behavior.

Terminal modes

There are two modes your terminal can be in:

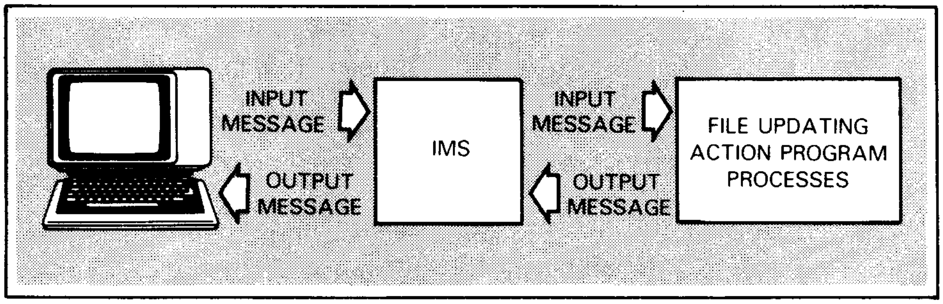
1. Normal mode
2. Test mode

Normal mode

Normal mode is the standard mode for IMS processing.

Normal mode processing

Normal Mode	
IMS	Receives an input message
	Processes the action program and updates files if required
	Sends an output message back to the terminal

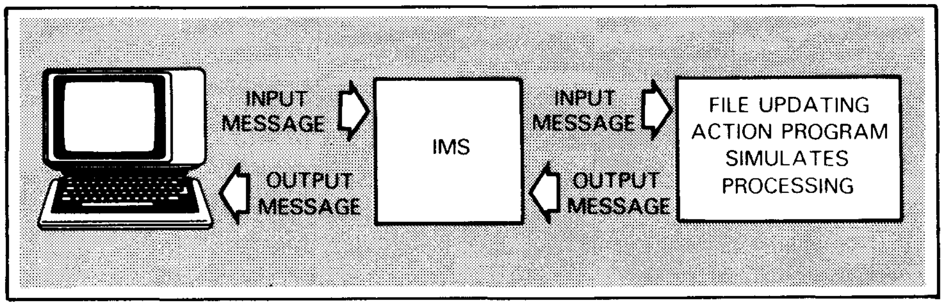


Test mode

In test mode, your IMS terminal simulates transaction processing. You transmit input messages and IMS processes the transactions as if they were actual transactions; however, no physical alteration of your data files occurs. This mode is useful in training new personnel and testing and debugging action programs.

Test mode processing

Test Mode	
IMS	Receives an input message
	Simulates action program processing
	Sends an output message back to the terminal



But your files exist as they were before the transaction occurred.

Types of input messages

The types of input messages you can send are **terminal commands** and **transaction codes**.

6.4. INPUT MESSAGES

Terminal Commands

Terminal commands are one type of input message you can use. They are distinguished from other input messages because they begin with the letters ZZ.

Types of terminal commands

There are two types of terminal commands:

1. Standard
2. Master

In this manual, we give you a brief description of each command and what it does. For detailed information on terminal commands, see the IMS terminal users guide, UP-9208 (current version).

Standard Terminal Commands

Using standard terminal commands

Standard terminal commands control message transmission and terminal operations and testing. You enter them from any terminal including the master terminal. Some standard commands can also be entered from the console or master workstation (if console support is configured).

STANDARD TERMINAL COMMANDS

ZZCNC **CANCELS A TRANSACTION**

Cancels the currently active transaction at your terminal only. It does not cancel a terminal command. It is most commonly used to break a deadlock situation under multithread IMS.

ZZHLD **SUSPENDS OUTPUT TO YOUR TERMINAL**

Suspends all output to your terminal until you enter a ZZRDY command. It is most commonly used when the ribbon breaks or the paper jams on a hard copy device.

▷▷▷

STANDARD TERMINAL COMMANDS

ZZMCH CHANGES THE MASTER TERMINAL

Designates another terminal as the master terminal when the existing master terminal becomes disabled. The new master terminal can be:

- your terminal
- another IMS terminal
- the system console (if console support is configured)

The terminal designated as the new master terminal serves in that capacity only for the remainder of the session. If the old master terminal is revived during the session, it functions as a standard terminal. When a new session starts, the old master terminal is restored.

ZZNRM RESTORES YOUR TERMINAL TO NORMAL MODE FROM TEST MODE

Restores your terminal to normal mode after it operated in test mode. Test mode simulates normal IMS transactions to help train new personnel or debug and test action programs without disturbing online files. In test mode, no update, delete, or add functions are actually performed.

ZZRDY RELEASES THE HOLD ON OUTPUT IMPOSED BY ZZHLD

Returns your terminal to a state where it can receive output messages. This command is used when the output to your terminal is suspended due to a ZZHLD command.

ZZRSD RESENDS THE LAST OUTPUT MESSAGE DISPLAYED AT YOUR TERMINAL

Displays the last transaction-supplied output message received at your terminal. You can use this command only once per message. When you enter a new input message, the preceding output message cannot be resent and IMS displays the response NO MSG IN QUEUE.

ZZTMD PLACES YOUR TERMINAL IN TEST MODE

Places your terminal in test mode from normal mode. Test mode simulates IMS transactions and enables you to test and debug action programs without disturbing online files. When your terminal is in test mode, it appears that transactions are actually taking place, but they are not. The ZZNRM command restores your terminal to normal mode.

Master Terminal Commands

Using master terminal commands

Master terminal commands control and monitor the entire IMS network. They differ from standard terminal commands because you can enter them only from the master terminal. You can enter some master terminal commands from the system console or master workstation when console support is configured. Master terminal commands entered from a standard terminal are invalid.

Using the system console

When the system console is designated as the master terminal, you can enter all master terminal commands from the console or master workstation. The same is true if the master terminal is down and you designate the console as the master terminal using the ZZMCH command.

Master terminal commands

MASTER TERMINAL COMMANDS

ZZALT **ASSIGNS ALTERNATE TERMINAL**

Assigns an alternate terminal for one that is physically or logically down; i.e., messages switched to a down terminal are automatically rerouted to an alternate terminal. When the down terminal becomes operational, rerouting is automatically discontinued.

ZZBTH **INITIATES AND CONTROLS ONLINE BATCH PROCESSING**

Initiates and controls online batch processing. To use this command, you must configure batch processing using the BATCH parameter in the NETWORK section of the configurator and include the PARAM BATCH statement at IMS startup.

ZZCLS **CLOSES A DATA FILE**

Closes a data file so non-IMS users can access it. You are not informed that the file is closed unless you try to access it. This command cannot close a defined file. To reopen a closed file, use the ZZOPN command.

ZZDWN **MARKS DOWN A MALFUNCTIONING TERMINAL**

Logically disables a terminal. It is most commonly used when a terminal in the network is malfunctioning. This improves response time for other terminals in the network.

ZZHLT **IMMEDIATELY TERMINATES AN IMS SESSION**

Terminates the IMS session. Use this only in emergency situations because all transactions in progress are aborted and not rolled back. This necessitates offline recovery or a warm restart for the next IMS session.

▷▷▷

MASTER TERMINAL COMMANDS

ZZOPN OPENS A DATA FILE

Reopens a data file that was closed by the ZZCLS command.

ZZPCH REPLACES AN ACTION PROGRAM IN THE LOAD FILE

Allows you to:

- debug action programs or correct programs and replace the old version with a new one
- add an action program that was missing at startup to the load library

ZZSHD SHUTS DOWN THE IMS SESSION IN AN ORDERLY MANNER

Shuts down the IMS session:

- after all transactions in progress are completed, or
- when a shutdown timeout period expires

ZZTCT DISPLAYS TERMINAL STATUS

Displays the following information about a specific terminal in the network:

- current status
- number of messages, transactions and commands since the start of the IMS session
- name of an alternate terminal (if there is one)

ZZTST TESTS A TERMINAL'S STATUS

Tests a specific terminal to see if it can receive output. To use this command, you must specify unsolicited output in your configuration.

ZZUP MARKS UP A PREVIOUSLY DOWNED TERMINAL

Restores a terminal that was previously marked as inoperative by a ZZDWN command.

Transaction Codes

Defining transaction codes

Transaction codes are messages that identify a transaction to IMS. When you enter a transaction code, IMS finds the action program assigned to that transaction code and processes the transaction. Once you start a transaction, you can't initiate another transaction or issue certain terminal commands until the transaction is complete.

Types of transaction codes

There are two types of transaction codes:

1. Transaction codes for your action programs
2. Transaction codes for IMS-supplied action programs

Transaction Codes for Your Action Programs

Defining user transaction codes

When you write your own action programs, IMS must know the names of all your transaction codes and the action programs related to them. You tell IMS this in the TRANSACT section of your configuration. You can designate a 1- to 8-character code for multithread, or a 1- to 5-character code for single thread, or a function key as your transaction code. You key in this transaction code or press the function key to begin a transaction.

Types of user transaction codes

When you write your own action programs, you must designate transaction codes to access them. There are two types of transaction codes:

1. Alphanumeric messages

   1 to 5 characters in single-thread IMS

   1 to 8 characters in multithread IMS

2. Function keys

Using 1-8 character codes

Whether you use alphanumeric messages or function keys as your transaction codes, you must identify them to IMS in the TRANSACT section of your configuration. When IMS receives the transaction code as input, it accesses the appropriate action program. Your action program must be prepared to receive the transaction code as the first field of its input message.

Function Keys

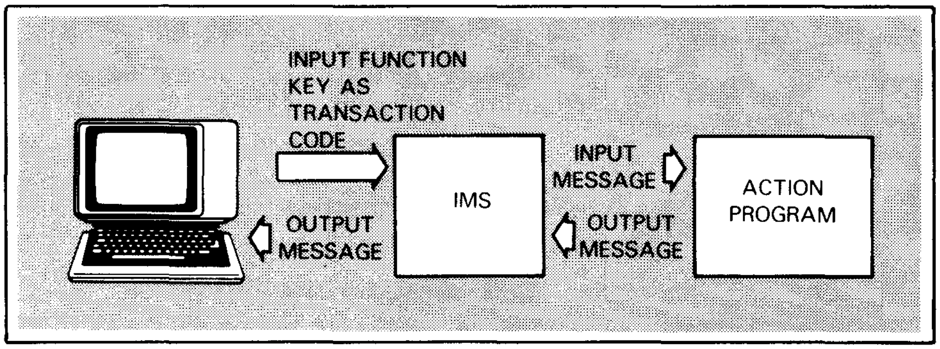
Using function keys

Function keys are fast, efficient, and provide less margin for error than do the alphanumeric messages. Depending on the type of terminal you have, you can use the function keys as follows:

Terminal	Available Function Keys
UNISCOPE Display Terminal	F1 - F4
UTS-400 Terminal	F1 - F22
IBM 3270 Terminal	F1 - F33

Processing function key input

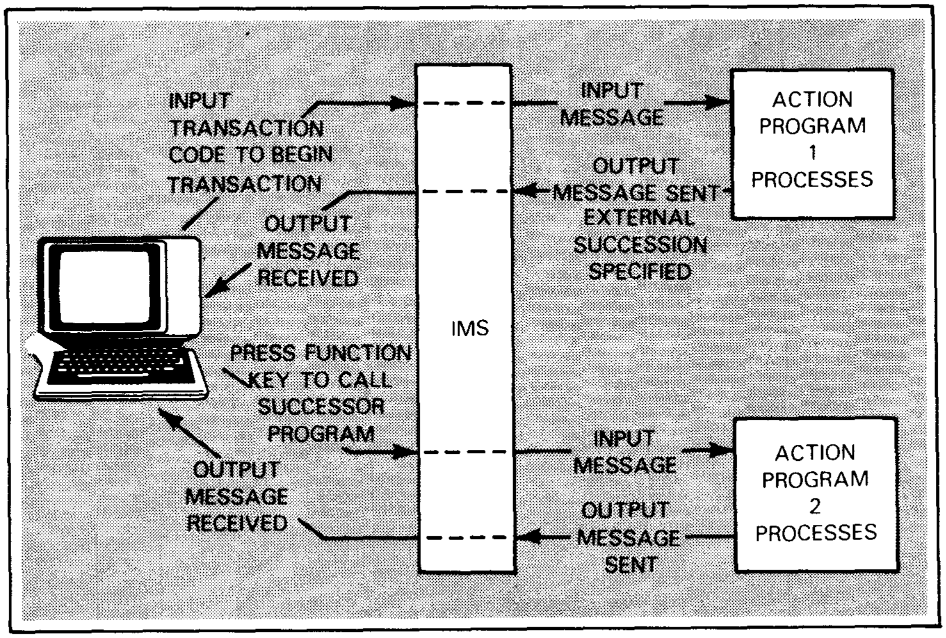
When your terminal is not processing a transaction and is sitting idle, any function key entry is interpreted as a transaction code. When IMS receives your function key as input, it accesses the appropriate action program.



Using a Function Key as a Transaction Code

Defining the function key

In addition to using a function key as a transaction code, you can also use it as a response to an output message. You use a function key in this capacity with external succession. You define the function key in your action program as the input message for the successor program. When the action program sends the output message requiring a response, the operator presses the defined function key and transaction processing continues.


INPUT TRANSACTION CODE TO BEGIN TRANSACTION
OUTPUT MESSAGE RECEIVED
PRESS FUNCTION KEY TO CALL SUCCESSOR PROGRAM
OUTPUT MESSAGE RECEIVED
IMS
INPUT MESSAGE
OUTPUT MESSAGE SENT EXTERNAL SUCCESSION SPECIFIED
INPUT MESSAGE
OUTPUT MESSAGE SENT
ACTION PROGRAM 1 PROCESSES
ACTION PROGRAM 2 PROCESSES

Using a Function Key with External Succession

Transaction Codes for IMS-Supplied Action Programs

IMS supplies you with a number of action programs that perform specific functions. These programs are accessed by special transaction codes that IMS recognizes. There are 7 IMS transaction codes, 4 can be entered from any terminal and 2 can be entered from the master terminal only. The transaction code SWTCH is used for both regular and master terminals.

Enter the following five transaction codes at any terminal:

DITBL	**DISPLAYS DATA IN CONFIGURATOR TABLES** Displays the contents of specified fields in your ACTION and PROGRAM configurator tables.
DLMSG	**DISPLAYS THE LAST EFFECTIVE OUTPUT MESSAGE** Displays the last effective output message sent to your terminal from an action program or UNIQUE. Use this transaction code when: – you cancel a transaction or one terminates abnormally and you need to determine where to resume updating – you want to receive the last effective output message from a specified terminal – you want to receive the last effective output message before a cold or warm restart
DLOAD	**DOWNLINE LOAD A UTS PROGRAM** Loads a user-written UTS program from the IMS load library downline to a UTS 40 or UTS 400 terminal. To use this transaction code, you must specify DLOAD=YES in the OPTIONS section of your configuration.
JI	**INITIATES AN OS/3 JOB** Runs an OS/3 job from an IMS terminal. Initiates a system command that reads a job control stream and schedules the job for execution.
SWTCH	**SENDS A MESSAGE TO ANOTHER TERMINAL OR TERMINALS** Sends a message to other terminals, including the system console. Used when you have an important message to send to other terminals. To use this command, you must specify unsolicited output, using the UNSOL parameter in the OPTIONS section of your configuration.

Enter the following transaction codes only at the master terminal:

SWTCH ALL

SENDS A MESSAGE TO ALL TERMINALS

Sends a message to all terminals in the IMS network.

CHTBL

CHANGES CONFIGURATION TABLES

When you do your configuration, you may make several entries in the ACTION and PROGRAM sections of the configurator. At some time, you may want to change these specifications. The CHTBL transaction code lets you make temporary changes to the internal tables generated by your entries in these sections.

ZSTAT

GENERATES STATISTICAL DATA

Generates statistics about your files, programs, transactions, and terminals. You can send these statistics to:

- the master terminal
- the master terminal and an auxiliary device (terminal, printer, cassette, or diskette)
- a special data file (STATFIL). You print the contents of STATFIL with the statistical file print program.

6.5. OUTPUT MESSAGES

There are two types of output messages:

1. Action program output
2. IMS output

Action program output

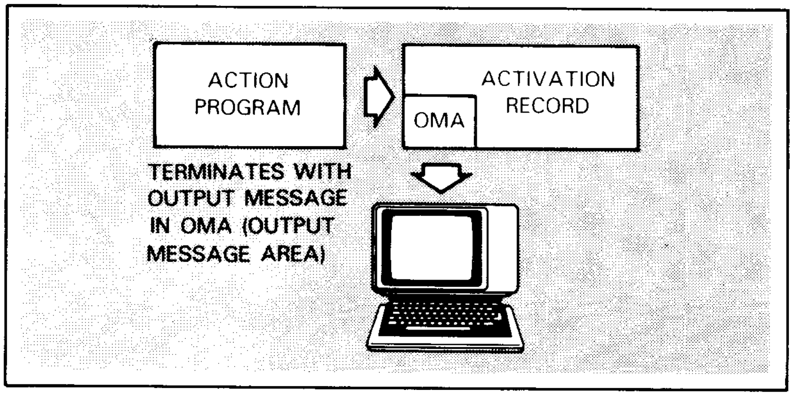
The first type of output is generated by action programs as a result of action processing. This output can be:

- a single message;
- multiple messages;
- a switched message; or
- a special feature message.

Single Message Output

Handling a single output message

With single message output, the action program produces one output message only. This message is created in the output message area of the activation record and is sent to the terminal when the program finishes processsing.



Single Message Output

Multiple Message Output

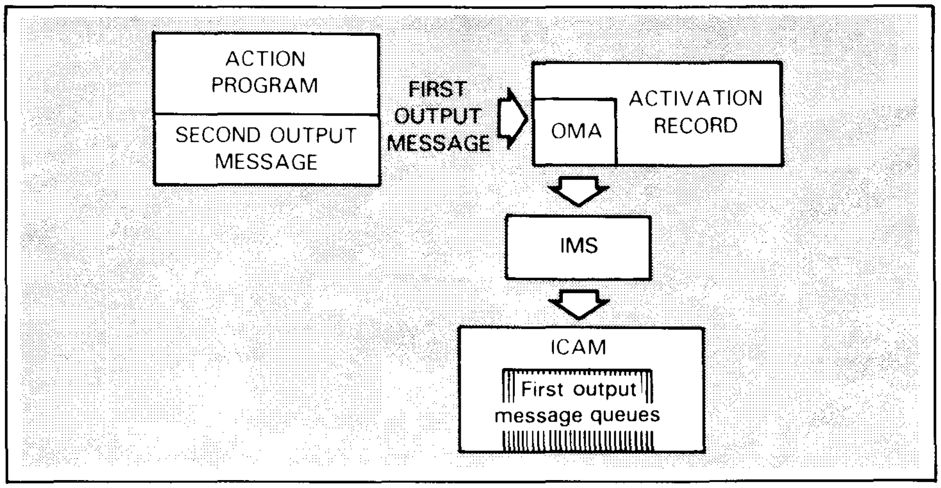
Generating multiple output messages

With multiple output messages, the action program produces more than one output message using SEND functions. An action program does this when the output is greater than screen capacity. (If your message is too long for one screen, the excess data wraps around and deletes the beginning data output.) The program is designed to output only the amount of data that will fit on one screen. This will vary depending on the screen size.

Handling multiple output messages

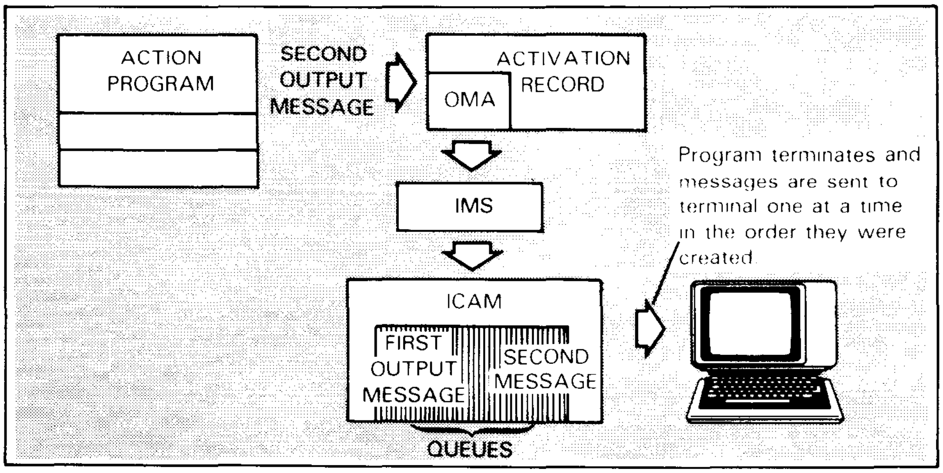
Using an ICAM queue

The first output message created stays in the output message area until the program generates the second message. Since the output message area holds only one output message at a time, something must be done with the first message before the second message is generated. Consequently, IMS takes the first message and moves it to an ICAM queue where it remains until the program finishes processing.



Multiple Message Output

As soon as IMS removes the first message from the output message area, it is ready to hold another output message.



Holding Output Message

Repeating the process

If the action program generates numerous output messages, the process repeats over and over until all processing is complete.

IMS moves each successive output message to an ICAM queue to free up the output message area.

Sending multiple output to the terminal

When the program finishes processing, IMS notifies ICAM; ICAM then begins to send the output to the terminal. The messages are sent one at a time in the order they are created.

Message waiting indicator

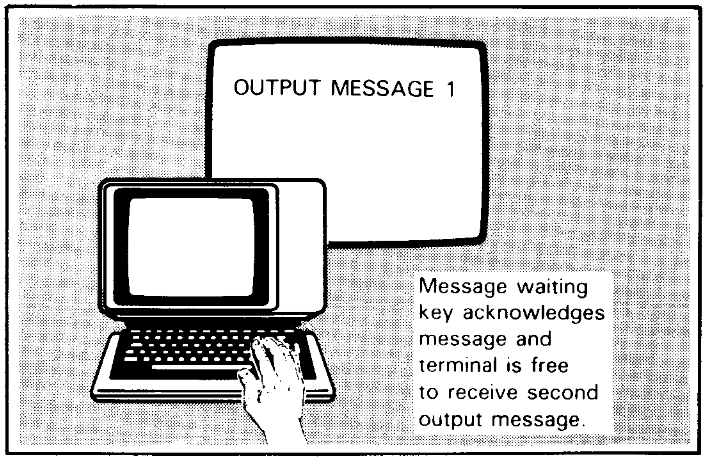
ICAM sends the first output message to the terminal and then notifies the terminal operator of each successive message. This notification is an indicator light or a message.

- If you have a display terminal, this indicator is usually the MESSAGE WAITING light.
- When you have a hard copy terminal, it's the message /CMW or another 4-character message you define in your communications network definition.

For a list of message waiting indicators for specific device types, see the IMS terminal users guide, UP-9208 (current version).

Restrictions with multiple output

When you have multiple output waiting, ICAM displays the indicator before each message. You must accept all queued multiple output messages before you can start another IMS transaction as shown by the following illustration.


OUTPUT MESSAGE 1
Message waiting key acknowledges message and terminal is free to receive second output message.

Accepting Multiple Input

Handling Output Messages

Introducing terms

Before we talk about the other two types of output (switched and special feature output), there are two terms we should introduce. You will see these terms often in the IMS action programming manuals. The terms are concerned with how IMS handles output messages.

- RETURN

CALL RETURN function

Both a single output message or the last output message generated by an action program is handled as a RETURN function in IMS.

- SEND

CALL SEND function

Multiple output (but never the last output message) generated by an action program is passed as a SEND function to IMS. Switched output, as well as several special types of IMS output, are sent using the SEND function. Whenever a program uses this function, you must specify the UNSOL parameter in your IMS configuration.

Switched Output

Defining switched messages

Switched messages are messages sent to your terminal as a result of an input message from another terminal. The input terminal can send a switched message one of two ways:

How to send switched output

1. From an action program using the SEND function
2. From a terminal using the IMS transaction code SWTCH

Queueing switched output

Message waiting indicator

Normally when you receive a switched output message, it's at the end of a transaction because these messages are queued at the terminal until the transaction in progress is completed. Then, you are notified by the applicable message waiting indicator.

(You could also receive the message waiting indicator at the end of each action if you configured UNSOL=ACTION).

Receiving switched messages

When you receive this indicator, you can either:

- ignore it and transmit another input message; or
- accept it by pressing the MESSAGE WAITING key on your display terminal or the CTRL/G and CTRL/C keys on hard-copy devices.

For message waiting responses of specific devices, see the IMS terminal users guide, UP-9208 (current version).

Special Feature Output Messages

Reference material

Since there are a large variety of special output messages, they are discussed in Section 8 of this manual.

IMS Messages

Types of IMS messages

You can also receive IMS messages at your terminal. These include:

- IMS READY message
- Automatic status messages
- IMS error messages
- Responses to terminal commands

IMS READY message

The IMS READY message signals to the terminal operator that IMS is prepared to accept an input message.

Automatic status messages

Automatic status messages notify a terminal operator of the status of the last input message. Multithread IMS generates these messages when there is a delay in processing an input message. The following list identifies the three automatic status messages and their meanings.

Message	Meaning
INPUT IN QUEUE	IMS received the input message but did not pass it to the action program.
INPUT IN PROCESS	The action program is now processing the input.
ROLLBACK IN PROCESS	The action program terminated abnormally and your updated files are being restored to the last rollback point.

Canceling a transaction

Any time after you receive the first status message, you can cancel the transaction by keying in the ZZCNC terminal command. Any other input you enter before receiving an output message is ignored.

IMS error message

IMS error messages result when an action program terminates abnormally. For a complete list of error messages, consult the OS/3 system error messages programmer/operator reference, UP-8076 (current version).

When you enter a terminal command, IMS sends a response message letting you know that the command was processed. For instance, when you enter the ZZTMD command, you receive the message:

```
TERMINAL IN TEST MODE
```

6.6. INPUT AND OUTPUT AT THE SYSTEM CONSOLE

You can also enter and receive messages at the system console and the master workstation (if there is one) when:

Configuration requirements

- You configure OPCOM=YES in the OPTIONS section, or
- You configure the console as the master terminal by not including MASTER=YES in any TERMINAL section

When there is a master workstation, you can enter messages from the console and workstation, but only one message from either source can be processed at a time. Response output messages go to the device at which the message was entered.

The console or master workstation functions as the master terminal when you omit MASTER=YES or when the master terminal is down and you use the ZZMCH command to designate the console as the new master terminal. You must configure OPCOM=YES to designate the console as the master terminal using ZZMCH.

6.7. USING MASTER AND STANDARD TERMINAL COMMANDS FROM THE CONSOLE

Terminal commands available

When the console is the master terminal, you can use all master terminal commands. When you configure console support (OPCOM=YES) but the console is not the master terminal, you can use the ZZSHD and ZZHLT to shut down the IMS session, but other master terminal commands cannot be entered.

You can enter four standard terminal commands from the console:

1. ZZCNC
2. ZZTMD
3. ZZMCH
4. ZZNRM.

Three standard terminal commands cannot be entered:

1. ZZHLD
2. ZZRDY
3. ZZRSD.

6.8. TRANSACTION PROCESSING FROM THE CONSOLE

When you configure console support, you can enter transactions from the console (or master workstation if there is one) just as you do from any terminal. However, there are limitations on the kinds of transactions you can use because of these restrictions on message size:

- The maximum length of an input message is 60 characters.
- The maximum length of an output message is 120 characters.

IMS transaction codes supported

You can use the IMS transaction codes DLMSG and SWTCH; however, UNIQUE is not effective at the console because the output screens are too large. The same is true of the ZSTAT transaction. You can use ZSTAT, but you must direct output to a data file, STATFIL, because the display screens are too large.

User transaction codes supported

A complete discussion of console transaction processing is discussed in the IMS terminal users guide, UP-9208 (current version).

7. File Activity with IMS

IMS internal files

In this section, we talk about both user files and IMS files. As you know from the discussion of preparing an online IMS system in Section 4, there are certain IMS files that handle online and offline IMS processing. To establish processing capabilities, you must initialize these files during IMS configuration. Although we talked about initialization, we didn't talk about the files themselves and what they do. We do this in this section.

User data files

You are also aware from Section 3 that action programs perform file processing. This processing involves user data files or defined files – retrieving, updating, inserting, and deleting records. How this happens is also a subject of this section. Let's begin by talking about file processing in the action program environment.

7.1. ACTION PROGRAM FILE PROCESSING

Action program file processing

The primary activity of action programs is processing input messages and producing output messages. More often than not, this involves some form of file processing. The terminal operator wants to look at a record, add a record, change a record, or perhaps delete it altogether. Each of these activities is a form of file processing.

File management

Action programs themselves don't handle the actual reading and writing of records. IMS handles all I/O. Consequently, each time an action program wants to access a file, it begins by issuing a call to IMS file management. File management, in turn, calls on OS/3 data management which retrieves the desired record and passes it back to IMS. Figure 7-1 illustrates this process.

Files supported

IMS file management makes records available to action programs for processing. File management supports sequential, relative, and indexed files. In addition, it supports defined files in an indexed file organization via defined record management.

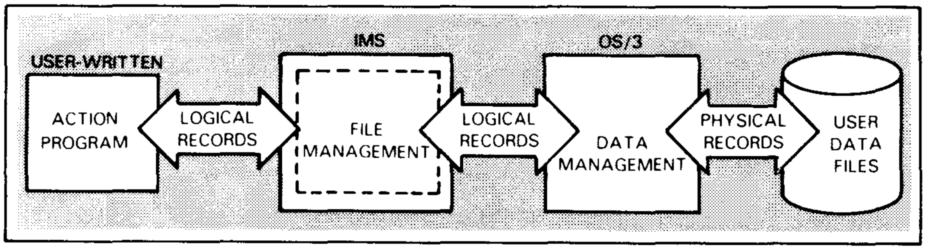


Figure 7-1. How IMS Accesses Your Data Files

COBOL and BAL

COBOL and basic assembly language function calls

COBOL and basic assembly language (BAL) action programs communicate directly with file management in making file requests. They do this through I/O function calls: GET, GETUP, PUT, DELETE, INSERT, SETL, SETK, and ESETL.

These calls are processed by SAM, DAM, ISAM, and MIRAM data management access methods. (To access IRAM files, you must define them as MIRAM files at configuration.) System 80 supports MIRAM files only.

RPG II

RPG II function calls

RPG II programs do not issue function calls. The RPG II compiler converts normal RPG II programming file requests into function calls to IMS file management.

No matter what the programming language, it is the function call (be it from the action program directly or through the compiler) that alerts IMS file management to handle a file request. From that point on, file management interfaces with OS/3 data management to do the actual reading and writing of records.

Summary of files and function calls

Table 7-1 summarizes the file types that IMS supports and the function calls that action programs perform on these files.

Table 7-1. Summary of File Types Supported by IMS File Management

File Organization	Access Mode	Data Management Access Method	Functions Available Through IMS File Management
Sequential	Sequential	SAM/dedicated MIRAM (tape and disk)	Retrieve, Append (write unblocked output)
Relative (nonindexed)	Random	DAM/MIRAM	Retrieve, Update, Insert, Delete
	Sequential	MIRAM	Retrieve
Indexed	Random	ISAM/MIRAM	Retrieve, Update, Insert, Delete
	Sequential	ISAM/MIRAM	Retrieve
Defined File	Random	ISAM/DAM/MIRAM	Retrieve, Update, Insert, Delete
	Sequential	ISAM/DAM/MIRAM	Retrieve

File types

File access

Table 7-1 shows that there are four file organization types: (1) sequential, (2) relative, (3) indexed, and (4) defined files. Each of these file types can access records randomly or sequentially (except for sequential files which can only access records sequentially).

Random and sequential access

Random access means that there is no prescribed order for accessing records. They can be called upon in any order the program desires.

Sequential access means that records are retrieved one after another, in the order they exist on the file.

The discussion of function calls begins with requests made to indexed and relative files.

7.2. INDEXED AND RELATIVE FILES

Indexed files

A key specification characterizes most function calls issued to indexed files. Each record in the file is identified by a specific key.

Relative files

A relative record number characterizes most function calls to relative files. Each record in the file is assigned a number relative to its position in the file (first, second, third, and so forth).

Since the result of function calls to relative and indexed files is the same, we discuss function calls only. However, the actual coding of the function calls for indexed and relative files differs. For a description and explanation of that coding, consult IMS action programming in COBOL and basic assembly language user guide, UP-9207 (current version).

Random Function Calls

Random file access means that you can access records in any order. For indexed files, you define the record by key; and for relative files, by relative record number.

Defining random function calls

The random function calls are: GET, GETUP, PUT, INSERT, and DELETE. Table 7–2 summarizes what these calls do:

Table 7–2. Random Function Calls for Relative and Indexed Files

Function Call	Meaning
GET	Only retrieves the record.
GETUP	Retrieves the record for updating and locks the record from access by other transactions.
PUT	Used with the GETUP function to write an updated record back to the file.
INSERT	Places a new record into the file or overwrites a previously deleted record.
DELETE	Used with the GETUP function to delete a record that was retrieved for updating.

Sequential Function Calls

Sequential file access means that you access records in the order they exist in the file, one after another.

Defining sequential function calls

The sequential function calls are SETK, SETL, GET, and ESETL for COBOL and BAL. Table 7–3 summarizes these function calls and what they do:

Table 7–3. Sequential Function Calls for Relative and Indexed Files

Function Call	Meaning
SETK	Changes the key of reference for multikeyed MIRAM files
SETL	Sets an indexed or relative file into sequential mode and logically positions the file according to your specifications
GET	Retrieves the next logical record in sequential order
ESETL	Changes an indexed or relative file from sequential back to random

For special considerations regarding the use of function calls with indexed and relative files, see IMS action programming in RPG II user guide, UP-9206 (current version) and IMS action programming in COBOL and basic assembly language user guide, UP-9207 (current version).

7.3. SEQUENTIAL FILES

You can only access sequential files sequentially; that is, one record after another as they exist on the sequential file. There are two function calls to sequential files: GET and PUT.

Processing sequential files

There is one point to remember about sequential files in an action programming environment; you can not use the same sequential file for both input and output. It must be either one or the other. Files configured for input can only be accessed by the sequential GET function. Files for output can only be accessed by the sequential PUT function.

In RPG II, you can access sequential files for output only, not input.

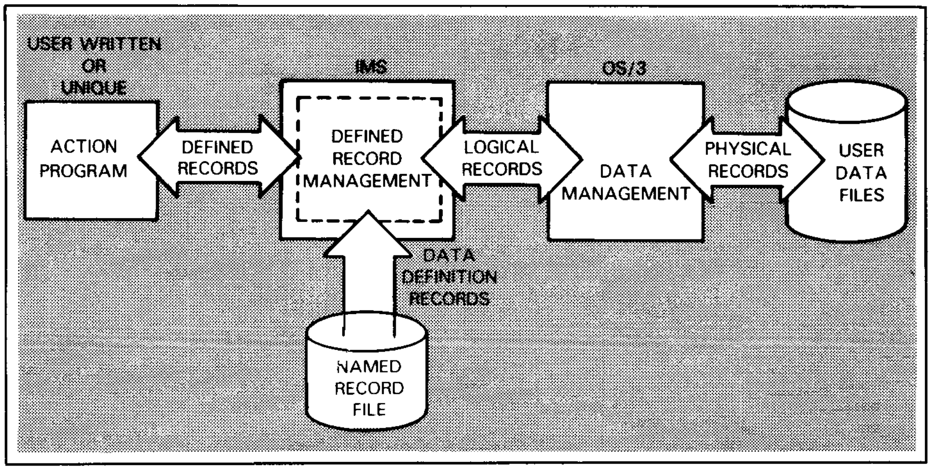
Function calls for sequential files

The sequential GET function retrieves the next logical record in sequential order; and the sequential PUT function writes logical records to sequential files on tape or disk.

7.4. DEFINED FILES

Accessing defined files

Defined record management is a subcomponent of IMS file management. Defined record management handles requests made by action programs to access defined files.



Accessing Defined Files

Function calls supported

Action programs can access defined files using random access functions GET, GETUP, PUT, INSERT, and DELETE or sequential access functions SETL (SETLL), GET, and ESETL (ESETLL). These functions perform the activities described in Tables 7-2 and 7-3.

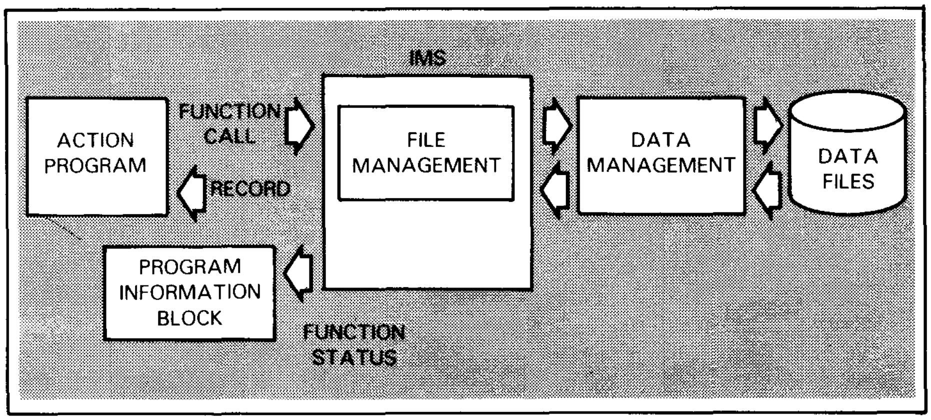
Restrictions

A transaction can access only one defined file during a given action. However, it can also access ISAM, SAM, DAM, or MIRAM files if they're not referenced by the defined file. For detailed information on accessing defined files, consult the IMS data definition and UNIQUE user guide, UP-9209 (current version) and Section 10 of this manual.

7.5. STATUS OF FILE REQUESTS

Results of function calls

After each file request made to IMS file management, IMS returns a value indicating the result of that request. This value is placed in the program information block.



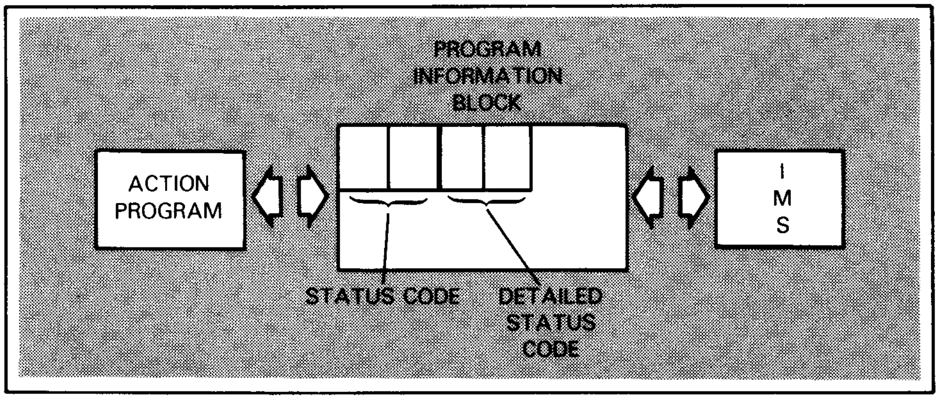
Returning a Status on Action Program Function Calls

Defining a program information block

COBOL and BAL programs must always define a program information block. The RPG II compiler automatically provides a program information block if the RPG II program doesn't define one.

Status code and detailed status code

The first four positions of the program information block contain the status code and detailed status code fields. The status code (positions 1 and 2) defines whether or not the function call was successfully carried out; and if not, why not. The detailed status code (positions 3 and 4) contains specific information about why the function call was unsuccessful.



Status and Detailed Status Code Fields in the Program Information Block

Successful and unsuccessful function calls

If both the status and detailed status code fields contain zeroes, the function call was successful. If one or the other is not zero, some error occurred. The current versions of IMS action programming in RPG II, UP-9206, and IMS action programming in COBOL and BAL, UP-9207, list error codes for all file function calls.

It is good programming practice to design action programs to check these fields after each function call request to determine what values they contain.

7.6. FILE PROCESSING CONSIDERATIONS

Defining user data files

You define all files used by action programs in the FILE section of the IMS configuration. You code a FILE section for each of the data files your action programs access. This FILE section provides to the configurator program detailed information about each file such as the file name, file type, type of record locks to be imposed, whether you want offline recovery for the file, data management interfaces for the file, etc. Single-thread users must specify RECLOCK=YES to obtain record locking.

Opening and closing data files

The file definitions are stored on the named record file in the file control table. At IMS start-up, all files defined by the configurator are opened. They remain open until closed by IMS shutdown procedures.

Files can also be closed and reopened from the master terminal via the master terminal commands, ZZCLS and ZZOPN. No capability exists to open and close files from an action program.

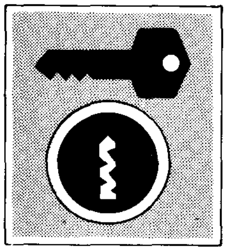
Additional file parameters

If you intend to use UNIQUE for file updating or if your action programs update files, you must also configure FUPDATE=YES in the OPTIONS section of the configuration.

7.7. LOCKING RECORDS

Using record locks

When your IMS transaction updates records in a file, you want to be sure that no other transaction is updating those records at the same time. For this reason, IMS provides a protection feature called **record locking.**



Lock options

You can either lock-for-update or lock-for-transaction. You specify one of these using the LOCK parameter in the FILE section of the IMS configuration. If you don't specify the LOCK parameter, IMS automatically provides lock-for-transaction.

Lock-for-Transaction

Defining lock-for-transaction

When you select lock-for-transaction (LOCK=TR) or omit this parameter:

- IMS locks records being updated until the end of the action.
- Updates are audited. This means that an image of the record before it is updated (before-image) is written to the audit file. If the transaction terminates abnormally, the original image of the record is restored. This process is called **rollback.**

Lock-for-Update

When you select lock-for-update (LOCK=UP):

- IMS locks records only for the duration of the update.
- Updates are not audited and not rolled back if the transaction terminates abnormally.

Additional locking capabilities

In addition to the lock-for-update and lock-for-transaction specifications, there are also special lock rollback indicators and an UNLOCK function call that you can use in your action program to control even further the use of record locks. For more information on use of the lock rollback indicators and UNLOCK, see IMS action programming in COBOL and basic assembly language user guide, UP-9207 (current version) and IMS action programming in RPG II user guide, UP-9206 (current version).

7.8. IMS INTERNAL FILES

Naming IMS internal files

By now, you are well aware that there are several IMS internal files that function either during IMS pre-online processing, online processing, or offline processing.The IMS internal files are:

NAMEREC

The named record file (NAMREC) is a required IMS file. It contains all the control tables generated by the configurator and records from pre-online processing. A NAMEREC file can contain records from as many as 255 different IMS configurations. However, if you have more than one IMS online at the same time, they cannot access the same NAMEREC file.

AUDFILE

The audit file (AUDFILE) is a required file for multithread IMS. It handles online recovery. Before IMS updates any record, it places an exact image of that record on the audit file. If the transaction terminates abnormally or invalid data is entered at the terminal, IMS reproduces the original record using the audit file. The audit file also contains IMS control information recorded when a transaction terminates.

CONDATA

The continuity data file (CONDATA) is a required file for multithread IMS when you use UNIQUE. It also is used when your action programs pass data using the continuity data area, or if you use the terminal output message file TOMFILE.

TRCFILE

The trace file (TRCFILE) is required for both a single-thread and multithread IMS if you want offline recovery capabilities. With offline recovery you can recover your data files at a time when IMS is not online.The trace file, unlike other IMS internal files, can be on disk, diskette, or magnetic tape.

AUDCONF

The audit and continuity data file (AUDCONF) is the single-thread IMS counterpart of the AUDFILE and CONDATA files for multithread IMS. It is always required.

TOMFILE

The terminal output message file (TOMFILE) is part of the single-thread AUDCONF file or of the multithread CONDATA file. During online processing, IMS writes to the TOMFILE the output message generated for each terminal at rollback points and at transaction termination, retaining only one message per terminal on the file. The terminal operator can redisplay the most recent output message which resulted from file updating by entering the transaction code DLMSG. These output messages are also recorded on the trace file for offline recovery if you specify TOMTRCE=YES in the OPTIONS section of the configuration. You must also specify TOMFILE=YES if a warm restart will be used.

STATFIL

The statistical data file (STATFIL) is optional for both single and multithread. It contains statistical data recorded during online processing.

LDPFILE

The fast loader file (LDPFILE) is required when you include FASTLOAD=YES in the OPTIONS section of configurator input. Action programs are copied into this file from the action program load library (LOAD) the first time they are called from a transaction and then are loaded from the LDPFILE.

7.9. ALLOCATING AND INITIALIZING INTERNAL FILES

Assigning and initializing internal files

You can allocate and initialize most IMS internal files while executing the IMSCONF job control stream. You use the IMSFIL and INIT keyword parameters. We discuss these parameters in Section 4. Only the NAMEREC file can be allocated and initialized in a separate process, using the named record file utility. You must also assign the internal files in the job control stream at IMS start-up. For a complete discussion of initialization and start-up procedures, consult the IMS system support functions user guide, UP-8364 (current version).

7.10. RECOVERY

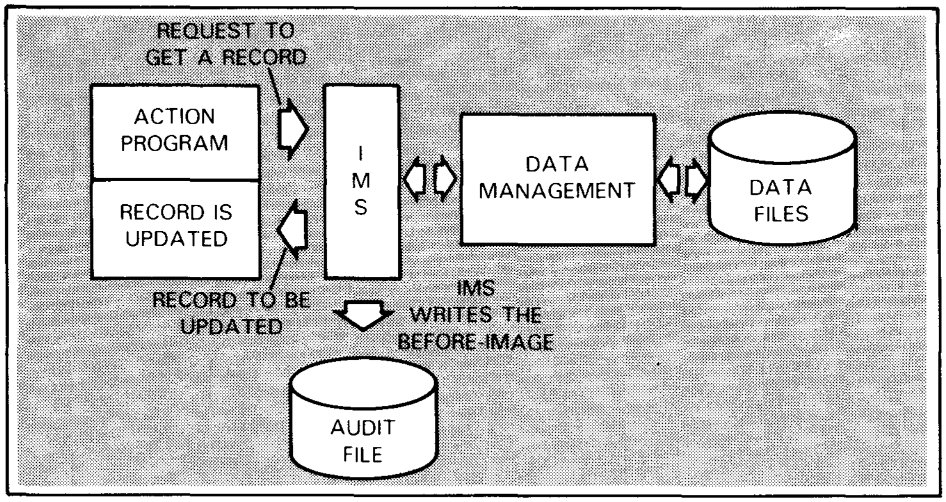
Types of recovery

Whenever you're updating files, you want the security of knowing that if something goes wrong, your files can be restored. IMS provides a complete file recovery system that operates in both online and offline environments to ensure the protection and security of your data files.

Online Recovery

Requirements for online recovery

Online recovery is automatic when you specify FUPDATE=YES in the OPTIONS section of the IMS configuration. You must, of course, have also initialized an audit file (AUDCONF for single-thread IMS and AUDFILE for multithread IMS) during IMS configuration and assigned it in the job control stream at IMS start-up. IMS stores a before-image of each record for updating on the audit file. In this way, records can be restored to their orginal form.



Writing Records to the Audit File

Occurrences of online recovery

Every time a transaction terminates abnormally, if you specified TOMFILE=YES in the OPTIONS section, IMS automatically rolls back your files to the state in which they existed at the start of the transaction or to the last rollback point if you so specified in your transaction.

IMS also rolls back your files at system start-up if you specify a warm restart; in this case, all transactions that were active at the time IMS terminated are rolled back.

In both of these cases, IMS uses the before-image of records stored on the audit file to restore files to their original state.

Offline Recovery

Requirements for offline recovery

Offline recovery is performed by an IMS utility program, ZC#TRC, after all online processing has terminated. To have offline recovery capabilities, you must specify the RECOVERY parameter in the OPTIONS section of configurator input. In addition, you must initialize a trace file and assign it at start-up. IMS records before-images, after-images, or both (depending on what you specify in the RECOVERY parameter) on the trace file so that you have complete recovery capabilities.

The offline recovery program restores files that are physically damaged or that contain inaccurate data.

A second offline utility, ZC#TCP, copies and closes a tape trace file left open by system failure. When a tape file is left open, you must use ZC#TCP to copy it before you can use it for recovery.

Summary of online and offline recovery

Table 7-4 summarizes the online and offline recovery functions, the internal files used for each, and the configuration parameters required.

Table 7-4. Recovery Options

Online/ Offline	Recovery Function	Internal Files Used	Configuration Parameters Required*
Online	Rollback: ■ Online transaction rollback ■ Warm restart	Audit file (AUDCONF) for single-thread. AUDFILE for multithread.	FUPDATE, TOMFILE (TOMFILE required for warm restart only)
	Display last effective output message (DLMSG transaction)	Terminal output message file (TOMFILE partition of AUDCONF or CONDATA file)	FUPDATE, TOMFILE
Offline	Utility programs: ■ Offline recovery utility (ZC#TRC) ■ Tape copy routine (ZC#TCP)	Trace file (TRCFILE)	FUPDATE, RECOVERY
	Recover TOMFILE in addition to data files	TRCFILE, TOMFILE	FUPDATE, RECOVERY TOMFILE, TOMTRCE

*All configuration parameters listed are specified in the OPTIONS section.

Since online recovery happens automatically once you have configured FUPDATE=YES, there is really no additional concern for the user. Consequently, we devote the remainder of our discussion to the offline recovery procedures.

7.11. TYPES OF OFFLINE RECOVERY

Types of offline recovery

The ZC#TRC offline recovery program provides three forms of offline file recovery:

1. Forward recovery
2. Backward recovery
3. Quick recovery

Forward Recovery

Using a backup file

You use forward recovery when a portion of your data files is destroyed. To do this, you must first have a backup copy of the original files. To find out how you create backup files, see the OS/3 system service programs (SSP) user guide, UP-8062 (current version).

Using after-images

The ZC#TRC utility updates your backup file using the after-images recorded in the trace file. If you specify that the terminal output message file, TOMFILE, is to be recovered in addition to your data files, ZC#TRC copies to the TOMFILE the last output message recorded in the trace file for each terminal up to the recovery point.

Backward Recovery

Using before-images

Restoring the TOMFILE

You use backward recovery when your data files are still accessible but may contain invalid data. ZC#TRC uses before-images recorded on the trace file to restore your data files to their original state or to the state they were in at a date and time you specify. You can also recover your TOMFILE during backward recovery.

Quick Recovery

You use quick recovery in the case of system failure or emergency IMS termination. Quick recovery rolls back all transactions active at the time of termination. ZC#TRC uses before-images on the trace file to roll back your transactions.

7.12. RUNNING THE OFFLINE RECOVERY PROGRAM

Recovering active transactions

Before you can use the offline recovery program, ZC#TRC, you must link the recovery object module, ZE#OLREC, with the object module containing your configuration control tables and file definitions and with the appropriate data management modules. The IMS system support functions user guide, UP-8364 (current version) provides sample job control streams for linking ZE#OLREC.

Linking the recovery module

Once you have linked the recovery module, execute the offline recovery program, ZC#TRC. In this job control stream, specify the type of recovery you want (forward, backward, or quick), the point at which recovery starts, and the files to be recovered. The system support functions manual, UP-8364 (current version) also includes sample job streams for doing this.

Executing the offline recovery program

Using the tape copy routine

If you use a tape trace file and (due to system failure or some other emergency) the tape trace file is left open, you must close it before you can use it for recovery. To close the file, use the tape copy routine ZE#COPY. Just as with the recovery module, you must first link the tape copy module, ZE#COPY, and then execute the tape copy routine, ZC#TCP. The IMS system support functions user guide, UP-8364 (current version) also provides job control for accomplishing these activities.

If the trace file is on disk and is left open due to a system failure, offline recovery can close the trace file by using the TRCFILE=CLOSE parameter.

8. Additional Features Available to Action Programs

In Section 3 we talked about action programs. We learned how they process input messages and produce output messages. We also discussed how action programs communicate with IMS to accomplish these activities. In this section we learn about additional capabilities that IMS provides:

Special features

- Continuous output
- Output-for-input queueing
- Line disconnect
- Snapshot dump
- UTS downline load
- Screen format services
- Edit table generator
- Initiating an OS/3 job

Coding for special features

To use these capabilities, you don't need to change the basic design of the action program. You still define the activation record that your program uses to communicate with IMS; and you follow the rules for coding RPG II, BAL, or COBOL action programs. The only difference is that you need to do a little additional coding in your action program to inform IMS that you want to make use of one of these special features.

How to code for these special features is discussed in detail in the IMS action programming manuals. Here, we concentrate on describing what these features are and how you use them.

8.1. CONTINUOUS OUTPUT

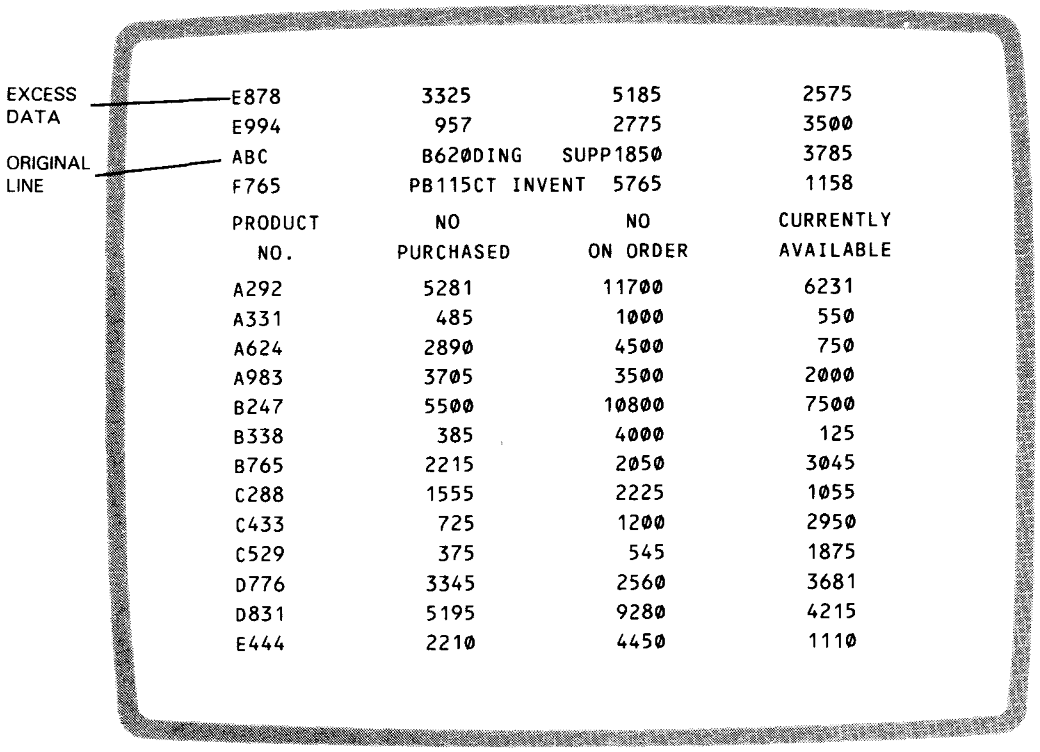
Handling regular output

All output messages generated by an action program are sent to the terminal when the action program terminates - not before. Whether a program produces one or 20 output messages, none goes to the terminal until the program completes processing.

When messages go to a terminal, they are sent one at a time, beginning with the first message. The operator is informed of successive messages by the message waiting light and, when ready, acknowledges that message. This continues until all the output generated by the program is transmitted.

Message size considerations

The primary reason for generating output in sections is that the terminal screen isn't large enough to handle the whole message at once. Any single output message generated by an action program can never be larger than the terminal screen size. Otherwise, it wraps around and is partially lost (Figure 8-1).



```
E878          3325          5185          2575
E994           957          2775          3500
ABC         B620DING    SUPP1850          3785
F765        PB115CT INVENT  5765          1158
PRODUCT        NO            NO        CURRENTLY
  NO.       PURCHASED     ON ORDER     AVAILABLE
A292          5281         11700          6231
A331           485          1000           550
A624          2890          4500           750
A983          3705          3500          2000
B247          5500         10800          7500
B338           385          4000           125
B765          2215          2050          3045
C288          1555          2225          1055
C433           725          1200          2950
C529           375           545          1875
D776          3345          2560          3681
D831          5195          9280          4215
E444          2210          4450          1110
```

Figure 8-1. Output Message Is too Large

Operator acknowledgement

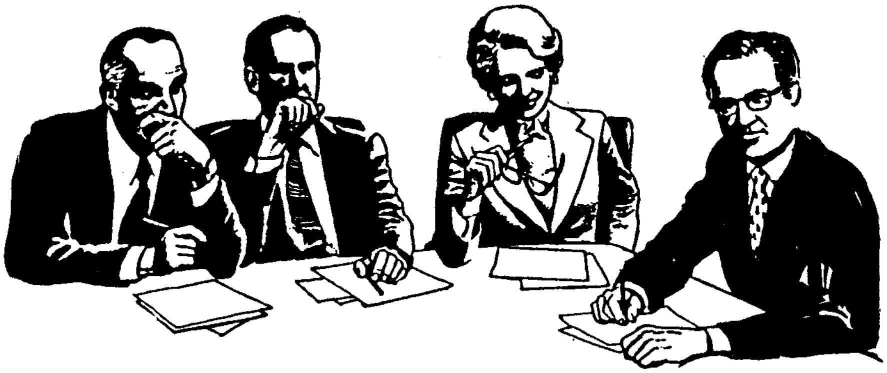
This method of handling output messages works fine if an operator sits at the terminal waiting to acknowledge each message waiting signal, if there aren't too many output messages, and if the data can be analyzed on the spot.

But what happens when a transaction contains large amounts of data for transmission and wants that data available to many users

Example

For example, company ABC wants quarterly sales reports incorporating sales information from 10 regions and distributed to sales managers, supervisors, and company executives.

A series of action programs are developed to gather the statistics and the data is processed until the end of the fiscal quarter. Two days later at the strategy meeting to discuss planning for the next quarter, all interested parties have received the data and have had time to study the report.



The report was easily processed using **continuous output**. Here's how....

Identifying continuous output

...when the action programs were designed to process this report, special codes were included. These codes tell IMS that the output message generated by this program is not regular output. It is special; it is continuous output.

Comparing Regular and Continuous Output

Similiarities

A continuous output message is transmitted when the action program terminates, just like regular output. Also, a continuous output message can't be larger than the size of the terminal screen.

Differences

Continuous output differs from regular output in that an action program can produce only one continuous output message and it must be the last message the program generates. That's an important point to remember.

Figure 8-2 shows how an action program produces three regular output messages and, finally, a continuous output message. When the program finishes processing, the messages are transmitted, in the order they were generated, to the terminals.

Last message transmitted

The continuous output message must be the last message transmitted. The program can produce many regular output messages, but the continuous output message must be the final message.

When using continuous output, it is best to send output messages that need a response to one terminal and send continuous output messages to another terminal. Also, since continuous output messages are limited in size when going to a display terminal, they are usually sent to a printer terminal.

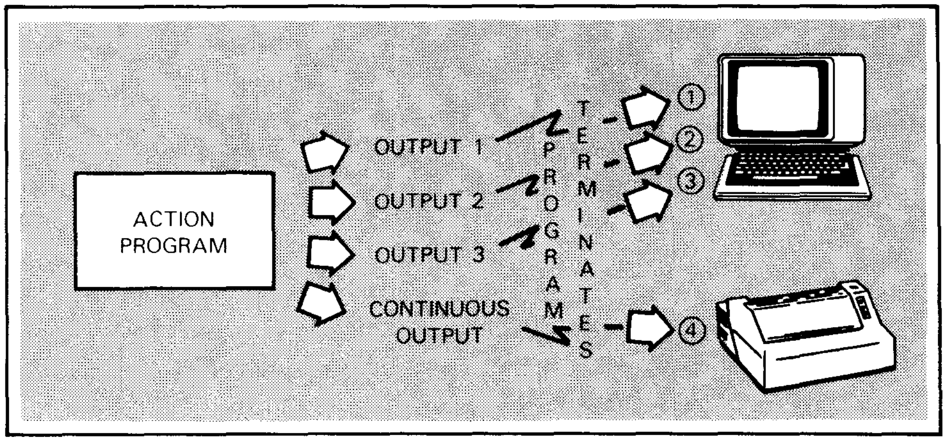


Figure 8-2. Continuous Output Must Be the Last Message Generated

Naming a successor

There's another difference . . . a program that generates continuous output must name a successor program. The reason for this should be obvious. If the first program can generate only one continuous output message and that message can't be larger than the size of the terminal screen, there has to be some way of continuing processing to generate the lengthy report. That's where the successor program comes in.

Example

In Figure 8-3 you see how action program A generates a continuous output message. The program names a successor and terminates. Then, the continuous output message is transmitted to the terminal.

Now, the successor program (B) gets control and generates a continuous output message. It also names a successor program and terminates; and the second continuous output message is sent to the terminal.

The process is repeated with successor program C.

No operator required

Naming a successor program allows a continuous output transaction to continue generating messages for as long as it is required. And this happens without an operator sitting at the terminal acknowledging one message after another. In fact, once the transaction code that started the transaction is keyed, the operator can leave the terminal altogether. IMS transmits each message and schedules the successor program without any operator intervention.

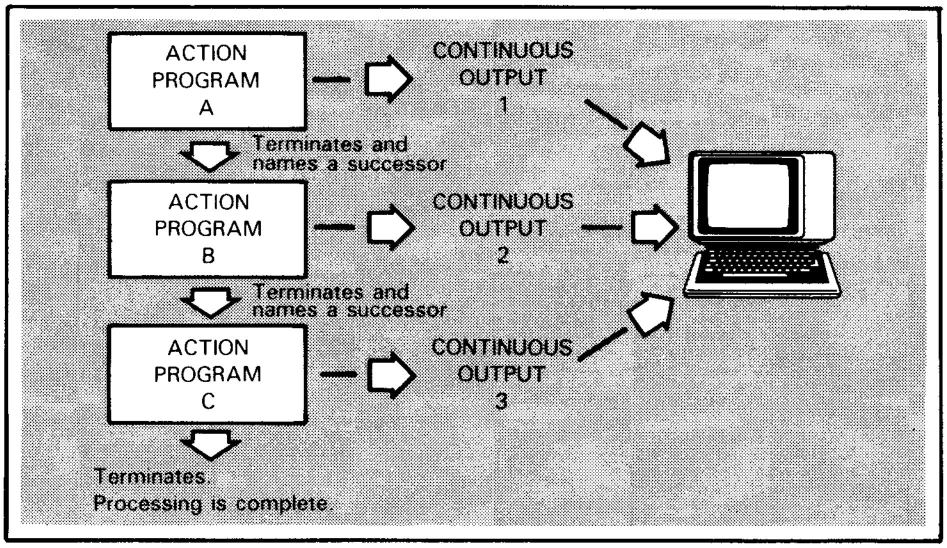


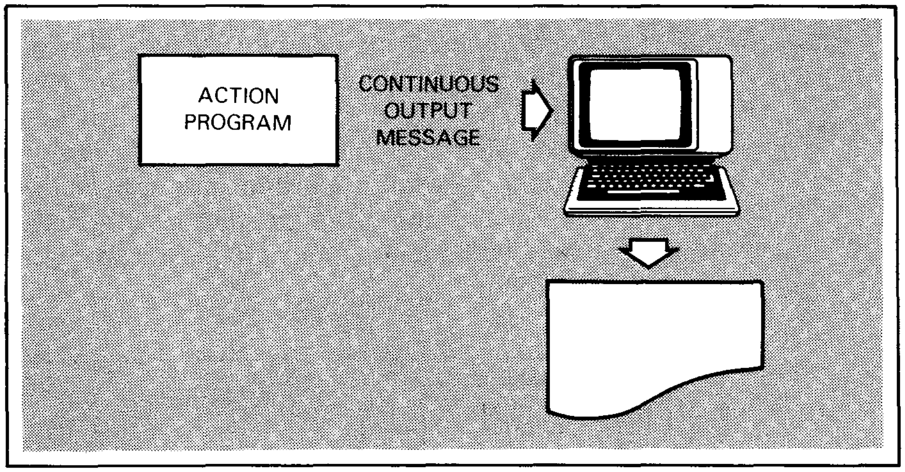
Figure 8-3. Naming Successor Programs to Continue Generating Continuous Output

The final destination for continuous output is usually not the terminal screen. (It's unlikely an operator is going to sit for long periods at the terminal watching screenfuls of data.)

Generating printed reports

Generally, we use continuous output to produce a printed report. The continuous output message appears on the terminal screen but is immediately transmitted to a printer or some other auxiliary device attached to the terminal. It might even be sent to a cassette or diskette or it might be sent directly to a hard copy terminal, such as a teletypewriter.

But remember, whether the message goes to the terminal or to a printer or other auxiliary device, the length of the continuous output message transmitted still cannot exceed screen capacity. The message always passes across the terminal screen first.



Continuous Output Is Used to Produce Printed Reports

How IMS Handles Continuous Output

Let's now look at how IMS handles continuous output as compared to regular output. The handling of regular output should be quite familiar to you by now, but for a review of how IMS handles regular output, see Sections 3 and 7.

Handling regular output messages

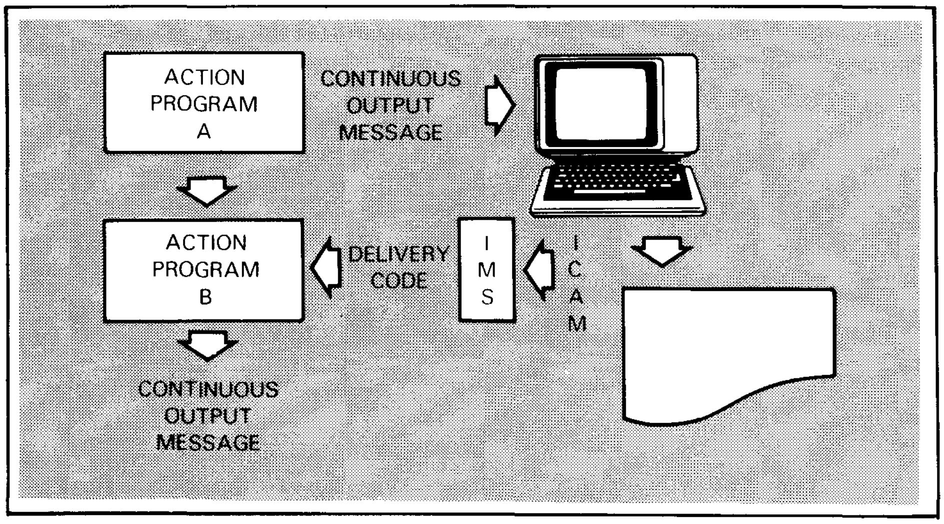
With regular output, the action program terminates and the message is transmitted to the terminal. Then, depending on the type transaction structure the program specified, IMS either waits for more input from the terminal or schedules a successor program.

Handling continuous output

The process for continuous output is slightly different. When the continuous output message is transmitted, IMS doesn't expect any input from the terminal and, more than likely, no operator is there. However, IMS does wait for a response from the terminal receiving the continuous output.

Response from ICAM

The response comes from ICAM, the communications network. It informs IMS whether or not the message was successfully delivered to the terminal or printer. This response is the delivery code. Once IMS receives this code it transmits it to the successor program (B).



Using Continuous Output, ICAM Sends a Delivery Code to IMS

Processing the delivery code

The successor program begins processing by examining the delivery code. If the code indicates successful delivery, the successor program generates its continuous output message. If the code says that the continuous output message wasn't successfully transmitted, the successor doesn't generate a continuous output message. Instead, it tests to see what went wrong and attempts to correct the problem. Once the problem is corrected, the successor program reschedules the first continuous output program so that it can attempt once again to transmit the continuous output message.

Generating more continuous output

When the delivery code says the first continuous output message was successfully delivered, the successor transmits its continuous output message. IMS again waits for a response from ICAM on the status of the message. If there's still more output to be printed, IMS schedules the successor and sends it the delivery code. This process is repeated as long as there is output to be printed.

Devices Supporting Continuous Output

Devices supporting continuous output

The following terminals and auxiliary devices can receive continuous output:

- UNISCOPE 100 display terminals
- UNISCOPE 200 display terminals
- Directly connected workstations
- UTS 10, 20, 40, and 400 terminals
- DCT 500 terminals operating in teletypewriter mode
- DCT 1000 terminals in interactive mode (Series 90 only)
- TELETYPE* models 28, 33, 35, 37, and 38
- Auxiliary devices at the UNISCOPE 100 or 200 Display Terminals or Universal Terminal System 400 (UTS 400). These are (see NOTE):
  - Communications output printers (COPs)
  - Terminal printers (TPs)
  - Model 610 Tape Cassette System (TCS)
  - 8406 Diskette Subsystem

NOTE:

Both TELETYPE and DCT 500 terminals always provide successful delivery codes unless the terminal or line goes down.

When you inform IMS that you're transmitting continuous output to an auxiliary device, there are numerous options you can choose from to specify how it is transmitted. Each of these options has a special code. The action programming manuals describe these options in detail.

**Trademark of Teletype Corporation*

Summarizing Continuous Output

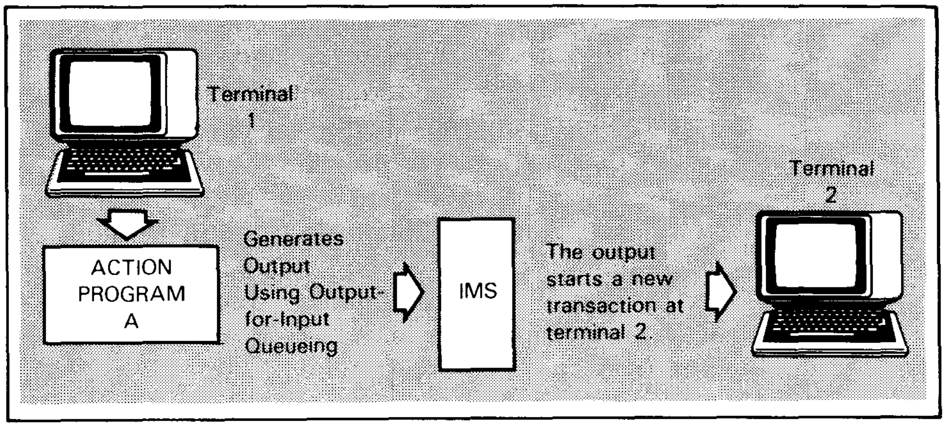
To summarize, continuous output is very useful for generating lengthy reports. Once IMS receives a transaction code that starts the continuous output transaction, IMS and the action program are totally in control. From that point on, no operator intervention is necessary to have one output message after another transmitted to a terminal or auxiliary device attached to the terminal. A continuous output transaction continues as long as each action program names a successor to keep generating continuous output.

To use the continuous output feature, you specify CONTOUT=YES in the OPTIONS section of the IMS configuration. And you include some additional code in your action program to inform IMS that you're generating a special type of output message.

8.2. OUTPUT-FOR-INPUT QUEUEING

Defining output-for-input queueing

Another special feature of IMS is **output-for-input queueing.** This feature is also concerned with output messages; it starts an IMS transaction at another terminal. Let's see how this works:


Terminal 1
ACTION PROGRAM A
Generates Output Using Output-for-Input Queueing
IMS
The output starts a new transaction at terminal 2.
Terminal 2

Using Output-for-Input Queueing to Start a Transaction at another Terminal

Example

Terminal 1 enters a transaction code that calls action program A which begins processing and tells IMS it's generating an output message using output-for-input queueing. IMS takes that message and uses it to start a new IMS transaction at terminal 2.

You're probably wondering why terminal 2 just doesn't start its own transaction by having an operator enter a transaction code there. Well, there are a couple of reasons.

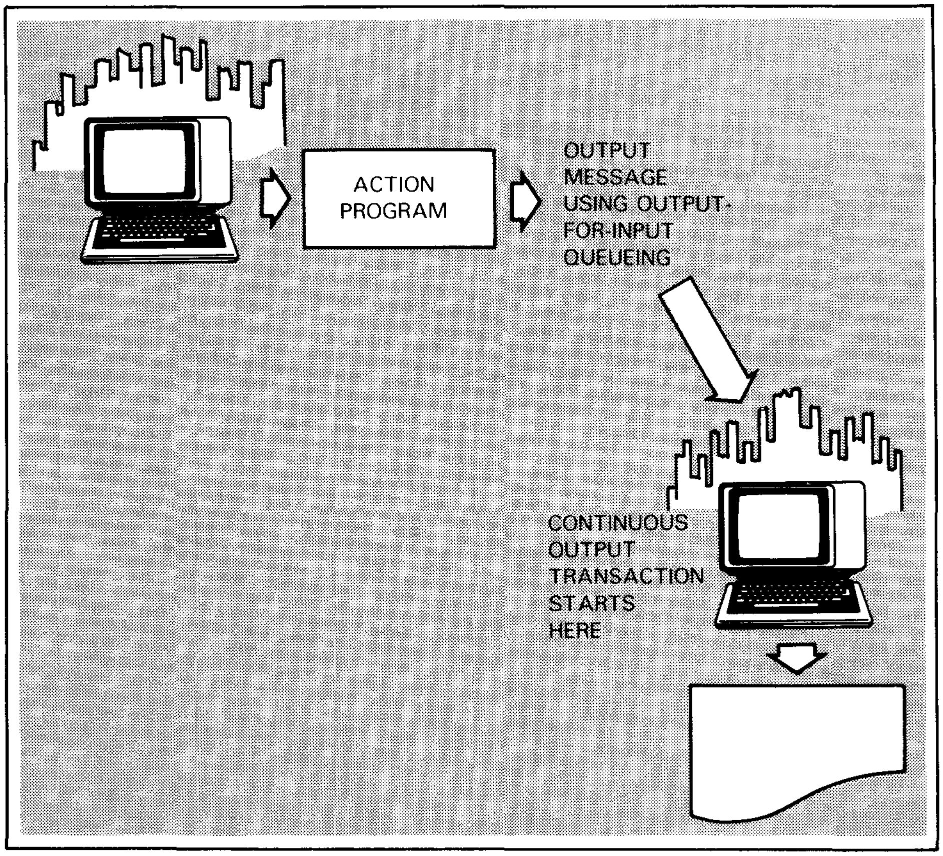
With a screen bypass device

First, some devices, such as the UTS 400 screen bypass device, are defined as terminals but don't have any physical apparatus (such as a keyboard) for entering data. The only way they can receive input is if it is transmitted from somewhere else, such as an action program.

This is one example of when output-for-input-queueing is very useful. One terminal starts an IMS transaction. The action program processing that transaction sends an output message using output-for-input queueing to IMS which, in turn, starts a new transaction at the screen bypass device.

With continuous output

Another situation where output-for-input queueing is frequently used is with continuous output. Using output-for-input queueing, an action program can send an output message that starts a transaction that prints continuous output at another terminal in your installation or at a remote site miles away.


ACTION PROGRAM
OUTPUT MESSAGE USING OUTPUT-FOR-INPUT QUEUEING
CONTINUOUS OUTPUT TRANSACTION STARTS HERE

Using Output-for-Input Queueing to Start a Continuous Output Transaction

Using the Output-for-Input Queueing Feature

Coding for output-for-input queueing

There is special code you include in your action program to use output-for-input queueing. The action programming manuals discuss this in detail. This code informs IMS that the output message generated is to be handled in a special way – as an output-for-input queueing message.

Identifying destination terminal

Identifying transaction code

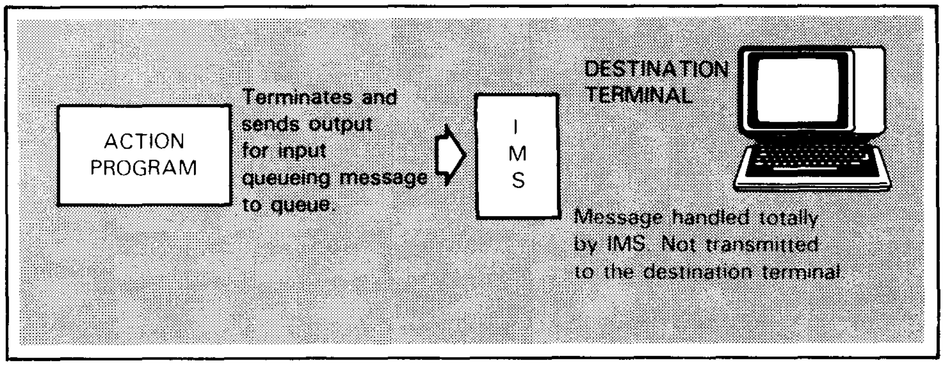
The action program also tells IMS the destination for the message – where it wants the new transaction to begin. In our example, it is a UTS 400 screen bypass device although it could be any terminal in the IMS network. Since this message will start a transaction at the screen bypass device, the program must also provide a valid transaction code – one that tells IMS what action program to schedule to process the transaction.

Handling Output-for-Input Queueing Messages

Special handling

You don't see output-for-input queueing messages at the terminal like you do regular output messages. Because the output-for-input queueing message automatically starts a transaction, there's no reason to display a message; the transaction automatically starts.

When the action program in process sends an output-for-input queueing message, it is queued for passage to the destination terminal. When the action program terminates, the message is sent to the successor action program which starts a transaction at the destination terminal. The following illustration shows how IMS handles output-for-input queueing.

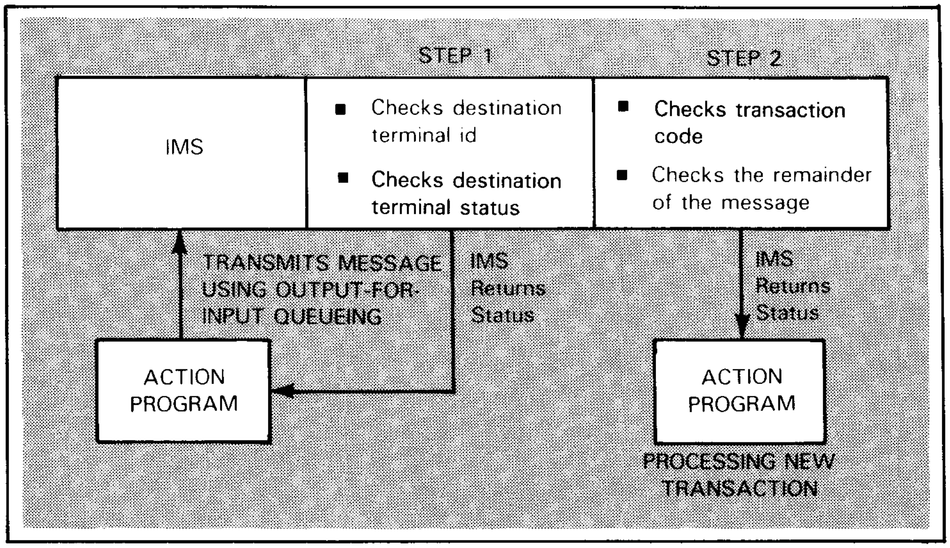

ACTION PROGRAM
Terminates and sends output for input queueing message to queue.
I M S
DESTINATION TERMINAL
Message handled totally by IMS. Not transmitted to the destination terminal.

Handling of Output-for-Input Queueing Output Message

When you think about it, this makes a good deal of sense. If you're using output-for-input queueing to begin with, you either have a screen bypass device as the destination terminal with no display device or you have a terminal with no operator. That is the purpose of using output-for-input queueing in the first place. So, there really is no point in transmitting the output message to the destination terminal. There's no one or no way to acknowledge it. What is important is that the new transaction is to begin there.

IMS internal processing

Here's how that happens. As soon as the action program informs IMS that it is creating an output message using output-for-input queueing, IMS takes that message and begins processing it – before the program ends. The processing is done in two steps:


STEP 1
STEP 2
IMS
■ Checks destination terminal id
■ Checks destination terminal status
■ Checks transaction code
■ Checks the remainder of the message
TRANSMITS MESSAGE USING OUTPUT-FOR-INPUT QUEUEING
IMS Returns Status
IMS Returns Status
ACTION PROGRAM
ACTION PROGRAM
PROCESSING NEW TRANSACTION

Processing a Message Using Output-for-Input Queueing

Step 1

Validating destination terminal id

IMS looks at the destination terminal identification code to determine if the code is valid. Then, it checks the status of the destination terminal.

Is the destination terminal available or is it involved in an IMS transaction?

IMS sends all this data back to the program that generated the message. If the terminal identification is invalid or if the terminal is occupied or down, the program is informed of this and can decide what to do next. Obviously, if any of these conditions exist, no transaction is scheduled at the destination terminal.

Step 2

Validating termination code

IMS checks the transaction code. If the transaction code or any other portion of the output message sent by the action program is invalid, IMS cannot schedule the transaction and an "INVALID TRANSACTION CODE" message is sent to the destination terminal. If the transaction code is valid, IMS schedules the transaction at the destination terminal.

Meanwhile, the program that generated the output-for-input queueing message finishes processing and terminates, and the newly scheduled transaction can begin processing and generating output to the destination terminal.

Summarizing Output-for-Input Queueing

In summary, you use the output-for-input queueing feature to start a transaction at a terminal other than the one you're using. To do this, you enter additional code in your action program that tells IMS you're producing a special type of output.

You identify the terminal where the transaction is to begin and give the transaction code that starts the new transaction.

If the message header and transaction code is valid and the destination terminal is free, IMS starts the new transaction.

If some data is invalid or the destination terminal is occupied or down, the action program is informed of the problem and takes appropriate action.

To use output-for-input queueing, you specify UNSOL=YES in the OPTIONS section of the IMS configuration. If you already specified CONTOUT=YES in the OPTIONS section, you have automatic support for using output-for-input queueing.

8.3. LINE DISCONNECT FEATURE

Defining line disconnect

The **line disconnect** feature allows an action program to disconnect a single-station dial-in line. By doing this, the action program frees the dial-in line for use by another terminal.

Action program HANGUP

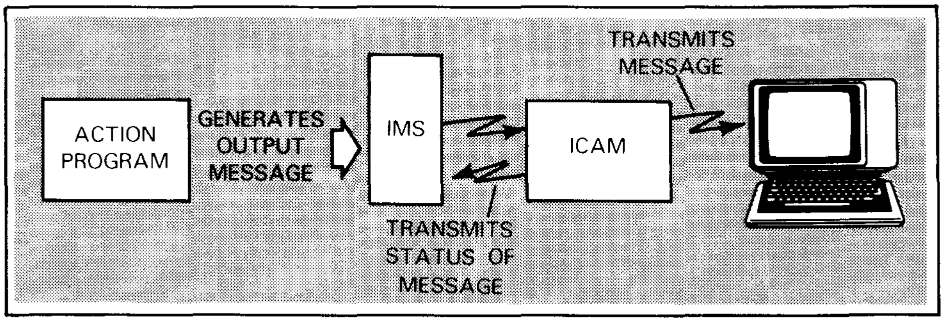
Using the line disconnect feature is easy. IMS supplies an action program that disconnects the communications line for you. The program is called HANGUP. When an action program wants to disconnect a line, it calls on HANGUP to do it.

Action program processing

The action program does its processing and generates an output message. To use HANGUP, the program must specify E for external succession as the termination code and HANGUP as the name of the successor program.

Output and delivery notice

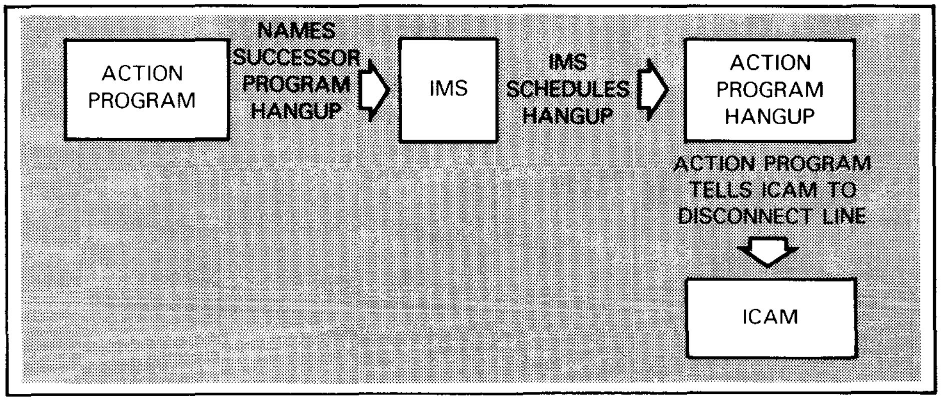
When the current program terminates, IMS sends the output message to the terminal and waits for a delivery notice from ICAM saying whether the message was successfully delivered.



Disconnecting a Line from an Action Program – Part 1

Scheduling HANGUP

Then, IMS schedules HANGUP to disconnect the dial-in line:


ACTION PROGRAM
NAMES SUCCESSOR PROGRAM HANGUP
IMS
IMS SCHEDULES HANGUP
ACTION PROGRAM HANGUP
ACTION PROGRAM TELLS ICAM TO DISCONNECT LINE
ICAM

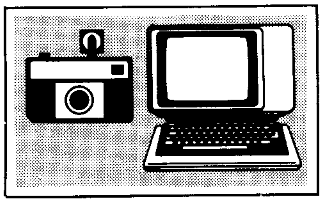
Disconnecting a Line from an Action Program – Part 2

Releasing the line

When the action program HANGUP gets control, it issues a message to ICAM to release the communications line from the terminal presently using it. Thus, the line now becomes available to another terminal.

The line disconnect feature is only available in a dedicated ICAM network. It is not available to global networks. To get the line disconnect feature, specify CONTOUT=YES in the OPTIONS section of the IMS configuration.

8.4. SNAPSHOT DUMP FEATURE



One of the most valuable and frequently used special features provided by IMS is the **snapshot dump.**

Defining a snapshot dump

A snapshot dump gives you a picture of what's happening inside your computer system when your action program is processing. What's even better is that it doesn't show you everything that's going on but just what directly relates to your action program. So, you don't have to sort through a lot of data which has nothing to do with your particular application.

Function

The snapshot dump shows all the areas of the activation record used by the action program, as well as the action program itself. Most importantly, it shows what went wrong with your action program.

To get a snapshot dump, you must run the IMS job with option JOBDUMP or option SYSDUMP job control statements.

A detailed analysis of snapshot dumps is provided in the action programming manuals. Here we explain how snapshot dumps occur and what you find in a snapshot dump.

How You Get a Snapshot Dump

Cause for dump

There are four conditions under which you get a snapshot dump:

1. An action program abnormally terminates due to a program check. This means a program check interrupt occurred during the execution of the action program.

2. An action program abnormally terminates due to a timer-check. This means that the program went into a loop and the program timer ran out.

3. An action program voluntarily terminates by moving the value 'S' to the termination-indicator field of the program information block.

4. An action program or subprogram requests a snapshot dump by issuing the SNAP function call. (COBOL and BAL programs can do this, RPG II programs cannot). The action program then finishes processing.

Abnormal termination

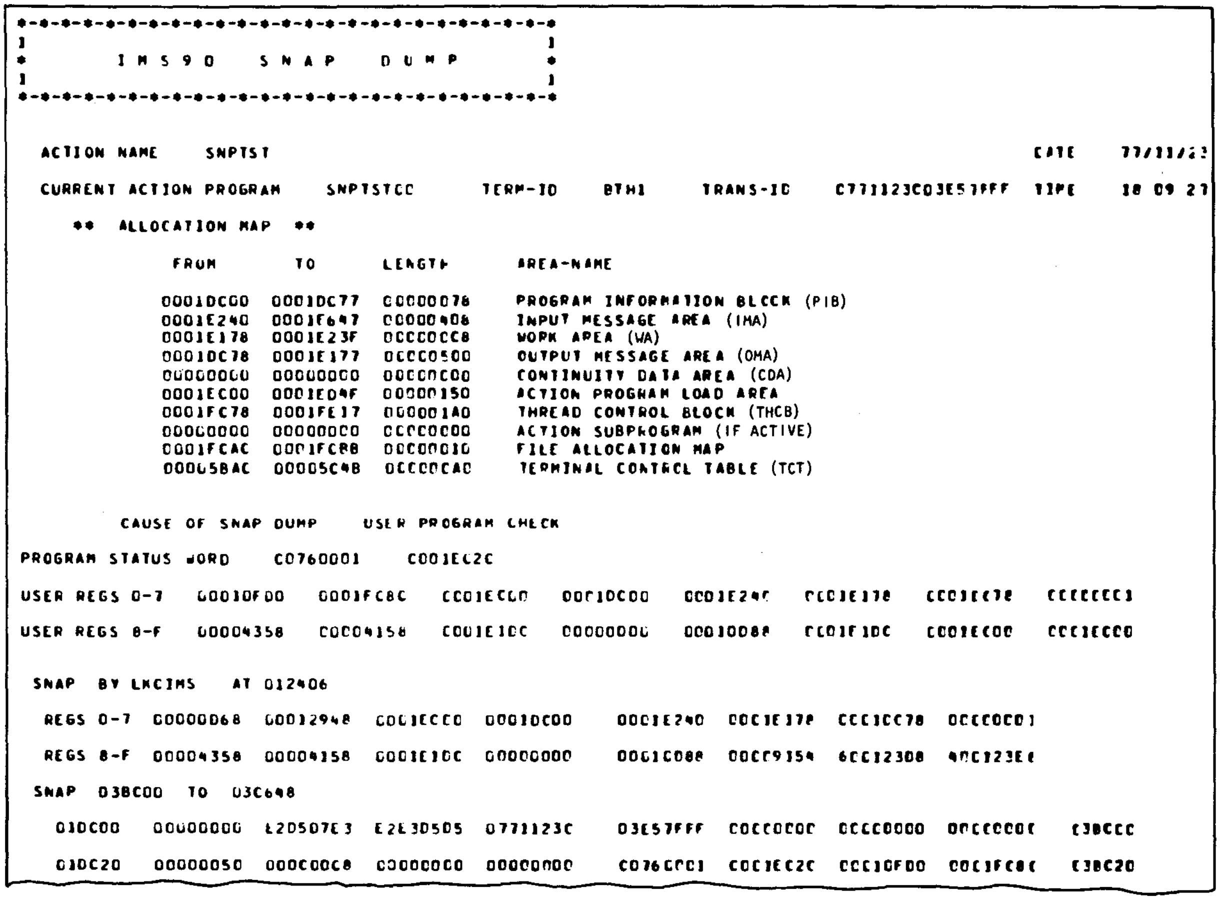
In abnormal termination, the action program has no control over whether the program terminates or not. If an action program creates an error condition related to IMS, IMS terminates the program and sends a message to the terminal identifying what went wrong. That's what happens with program and timer checks. In these circumstances IMS also provides a snapshot dump of the action program.

Voluntary termination

In the case of voluntary termination – the third condition – the action program is in control. It deliberately terminates itself. You must insert special code into the program that tells IMS to dump the program so the programmer can determine what went wrong. The last condition allows an action program to request a snapshot dump and then continue processing.

What a Snapshot Dump Contains

Figure 8–4 is a glimpse of a snapshot dump. A dump can go on for many pages depending on the length of the action program. The figure shows a small portion of the dump for program SNPTST.



```
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
I                                                   I
*        I M S 9 0    S N A P    D U M P            *
I                                                   I
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

  ACTION NAME     SNPTST                                                                  CATE    77/11/23

  CURRENT ACTION PROGRAM     SNPTSTCC       TERM-ID     BTH1      TRANS-IC     C771123C03E57FFF  TIME    18 09 27

     **  ALLOCATION MAP  **

            FROM         TO         LENGTH        AREA-NAME

           0001DC00   0001DC77   C0000078    PROGRAM INFORMATION BLOCK (PIB)
           0001E240   0001F647   C0000408    INPUT MESSAGE AREA (IMA)
           0001E178   0001E23F   0CCC00C8    WORK AREA (WA)
           0001DC78   0001E177   0CCC0500    OUTPUT MESSAGE AREA (OMA)
           00000000   00000000   000C0C00    CONTINUITY DATA AREA (CDA)
           0001EC00   0001ED4F   00000150    ACTION PROGRAM LOAD AREA
           0001FC78   0001FE17   000001A0    THREAD CONTROL BLOCK (THCB)
           00000000   000000C0   0CCC0000    ACTION SUBPROGRAM (IF ACTIVE)
           0001FCAC   0001FCB8   0000001C    FILE ALLOCATION MAP
           00005BAC   00005C4B   0CCC0CAC    TERMINAL CONTROL TABLE (TCT)

        CAUSE OF SNAP DUMP    USER PROGRAM CHECK

PROGRAM STATUS WORD    C0760001    C001EC2C

USER REGS 0-7    00010F00    0001FC8C    CC01EC00    00010C00    CC01E240    CC01E178    CC01EC78    CCCCCCC1

USER REGS 8-F    00004358    C0C04158    C001E1CC    00000000    00010088    CC01F1DC    C001EC00    CCC1ECC0

 SNAP  BY LKCIMS    AT 012406

  REGS 0-7  C0000068  00012948  C001ECC0  00010C00    0001E240  00C1E178  CCC1CC78  0CCC0C01

  REGS 8-F  00004358  00004158  C001E1DC  00000000    00010088  00CC9154  6CC12308  40C123EC

 SNAP  03BC00   TO   03C648

   010C00    00000000  E2D507E3  E2E3D5D5  0771123C    03E57FFF  C0CC0C0C  0CCC0000  00CCC00C    C3BCCC

   010C20    00000050  000C00C8  C0000000  00000000    C076CCC1  C0C1EC2C  CCC10F00  C0C1FC8C    C3BC20
```

Figure 8–4. Sample Snapshot Dump

Contents of snapshot dump

Snapshot dumps can contain six general areas:

Headers

Gives general information about which program was running when the dump occurred and what caused the snapshot dump. If it is an edited snap dump, you also get an allocation map. This is a directory that tells you how to find all the major portions of the dump and gives an address for each section that the dump contains. You do not automatically get this header section when you have a snapshot dump. In multithread you must request it by specifying SNAPED=YES. In single-thread, you receive no headers with the CALL SNAP function.

Register Section

Contains both IMS and action program registers or just IMS registers, depending on what caused the dump.

Activation Record

Shows the contents of the program information block, input message area, output message area, work area, continuity data area, and defined record area when the program terminated.

Action Program

Contains the executing load module as it existed in main storage at the time of the dump.

Thread Control

Contains data used to control the IMS environment. When you get an unedited dump, one without a directory, you can find the location of every section of the snap dump by consulting the thread control block. The thread control block also contains other valuable data for locating the exact cause of program failure.

Terminal Control

Contains data related to the terminal that initiated the action program.

8.5. UTS DOWNLINE LOAD FEATURE

Definition of downline load

You use an action program to load other programs from an IMS library into a UTS 40 or 400 terminal. The advantage of doing this is that you use the processor capabilities of the UTS to process the program, thus freeing main storage of the host computer for other activities.

Using IMS load library

The program to be loaded must be stored in the system load library or in the IMS load library. The actual downline loading of the programs is performed during IMS online processing. There are two ways to downline load a program:

Accomplishing the downline load

1. Enter the transaction code DLOAD. This activates an IMS-supplied action program that does the downline load.
2. Write your own downline load action program in COBOL or BAL. Downline load action programs in RPG II are not supported.

Configuration requirements

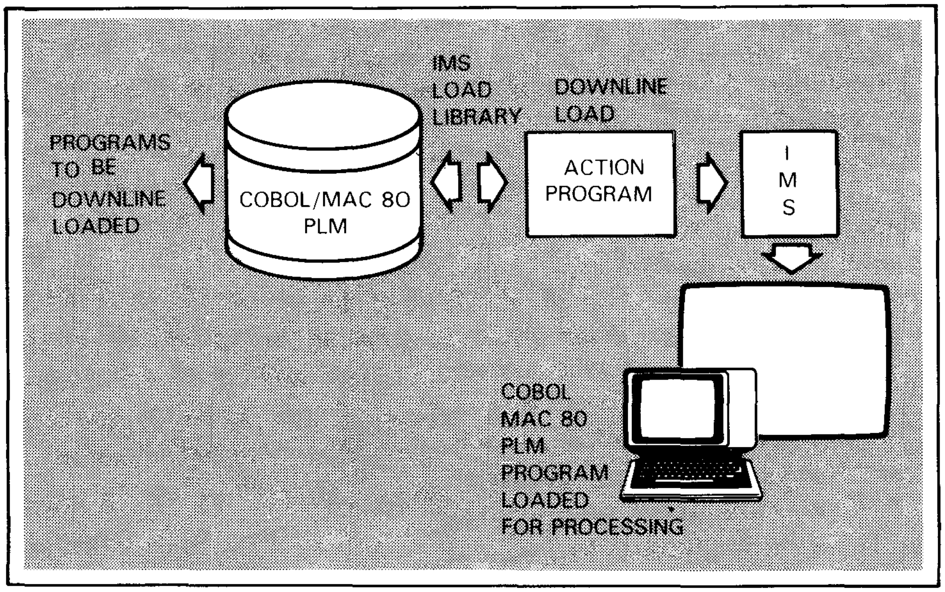
To use the downline load feature, you must generate a resident communications network and must specify DLLOAD=YES in the OPTIONS section of your IMS configuration.

Destination of downline load

When you downline load a program, you load it directly into UTS 40 or UTS 400 main storage or into an auxiliary device attached to the UTS terminal (such as a cassette or diskette). Once it is loaded to an auxiliary device, you read it into the UTS 400 main storage as your application requires. The only special consideration is that when you downline load directly to UTS main storage, the terminal that processes the program must be a master or primary UTS 40 or 400 station.

Programs for downline load

Programs intended for downline load are written in COBOL, MAC 80, or PLM. These programs are often designed to edit or validate IMS transaction codes or other input messages. If the input contains errors, these programs can handle them directly at the terminal, before the input is transmitted to the host computer. The following illustrates downline loading programs to a UTS 400.


PROGRAMS TO BE DOWNLINE LOADED
COBOL/MAC 80 PLM
IMS LOAD LIBRARY
DOWNLINE LOAD
ACTION PROGRAM
I M S
COBOL MAC 80 PLM PROGRAM LOADED FOR PROCESSING

Downline Loading Programs for Processing at a UTS 400

For a detailed description of how to write your own downline load action program, consult the IMS action programming in COBOL and basic assembly language user guide, UP-9207 (current version). The DLOAD transaction code is described in detail in the IMS terminal users guide, UP-9208 (current version).

8.6. SCREEN FORMAT SERVICES

Defining screen format services

Screen format services is a separate software package which allows users to create formatted screens. IMS interfaces with screen format services to allow action programs to use screen formats for input and output messages.

Other formatting techniques

There are, of course, many ways that action programs can format messages. They can use field control characters, field format characters, or device independent control expressions. However, all of these involve considerable coding.

Just the opposite is true when you use screen format services. Generating formatted screens this way is extremely easy. By simply adding a small amount of code to the header portion of your output message, you can display formatted screens at the terminal.

Devices Supporting Screen Format Services

Devices supporting screen format services

You can display screen formats on the following devices:

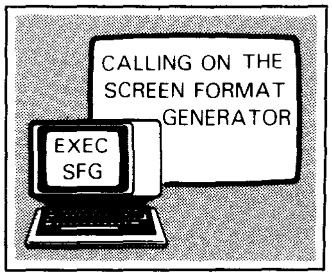
- UNISCOPE 100 (with PROTECT feature)
- UNISCOPE 200 (with PROTECT feature)
- UTS 400 (in UNISCOPE mode or operating in native mode, with PROTECT/FCC switch set to FCC and control page set to XMIT variable)
- UTS 10, 20, 40
- Workstations

How You Create Screens

Using the screen format generator

To use formatted screens, you must create them offline. You do this by executing the screen format generator.

To execute the screen format generator, your operating system must be generated in CDM or mixed mode.



Using screen format menus

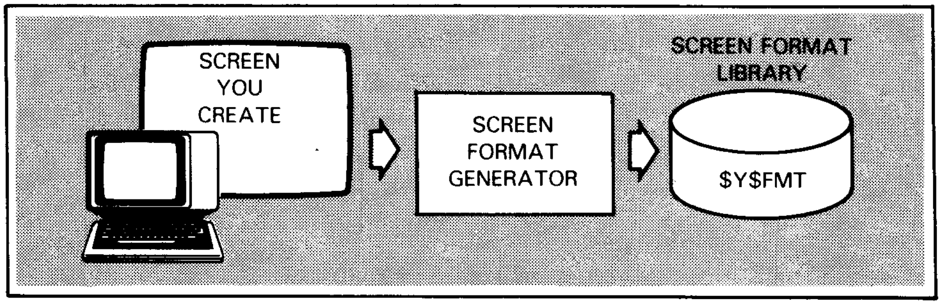
The screen format generator provides a series of menus that direct you in creating a screen. You assign a specific name to each screen you create. You also inform the screen format generator whether it is an input-only screen, output-only, or both.

Among other things, you can define fields as numeric only, or alphanumeric. Based on how you define fields, the screen format coordinator validates data entered for these fields.

Storing formatted screens

The screen format generator stores each screen you create in the system format library ($Y$FMT) or on a disk library you specify.

When an action program requests a specific screen, IMS calls upon the screen format coordinator and it, in turn, retrieves the screen from the format library.



Storing Screen Formats

For more information about screen format services and generating your own screens, see UP-8802.

Using Screen Format Services

Coding required

Using screen format services involves additional coding in your action program. In COBOL and BAL action programs you get screen format services via CALL functions.

The specific CALL functions used for screen formatting are CALL BUILD and CALL REBUILD:

CALL BUILD

- CALL BUILD tells IMS what screen is required to format the output message, what variable data is to be inserted into the formatted screen, and what option indicators are set.

CALL REBUILD

CALL REBUILD tells IMS to build an error screen or to replace input fields with underlines to prompt the terminal operator to make the required entries. In the case of the error screen or the underlines, CALL REBUILD lets you put out an altered screen, without having to generate an entirely new screen. CALL REBUILD cannot be issued for a screen format which uses output option indicators. In this case, a CALL BUILD should be issued with the appropriate indicators set.

For the error screen, the screen format coordinator replaces error fields with blinking characters to show the operator exactly where the input error occurred.

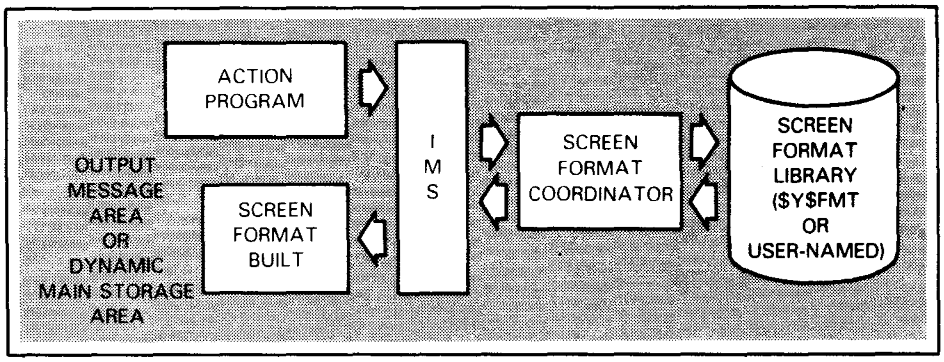
RPG II users

In RPG II action programs, you build an error screen by using option indicators.

Retrieving a screen format

Building a screen

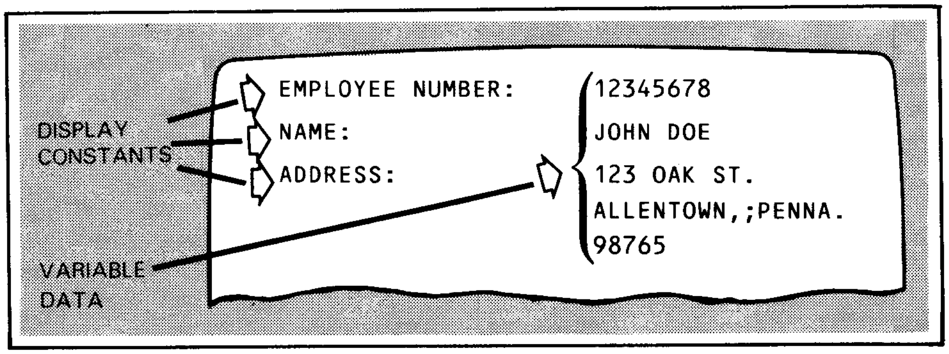
Once an action program requests a screen format, IMS passes the format name, the variable data, and the option indicators to the screen format coordinator, which retrieves the proper format from the format library. IMS then places the format in the output message area of your action program or in a dynamic main storage area.


ACTION PROGRAM
OUTPUT MESSAGE AREA OR DYNAMIC MAIN STORAGE AREA
SCREEN FORMAT BUILT
I M S
SCREEN FORMAT COORDINATOR
SCREEN FORMAT LIBRARY (YFMT OR USER-NAMED)

Retrieving a Screen Format

Displaying a screen

When the screen is complete the screen format coordinator inserts the variable data. When the action program terminates, the formatted screen, display constants, and variable data, all go out to the terminal. Figure 8-5 shows an output screen containing display constants and variable data.

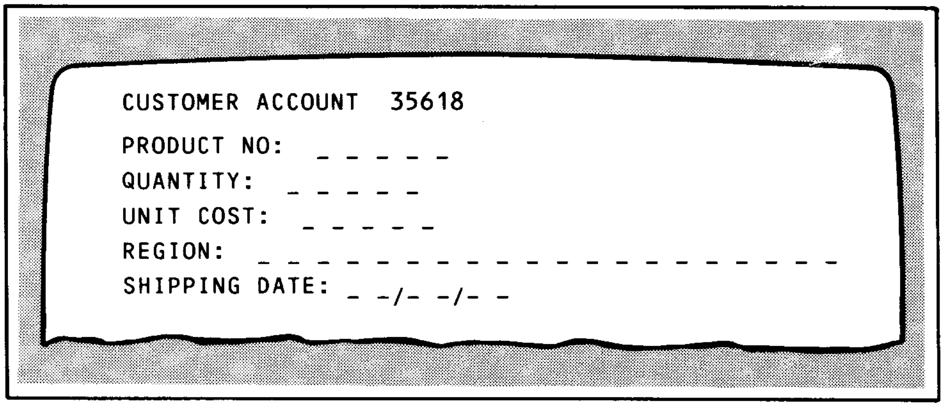


```
EMPLOYEE NUMBER:     12345678
NAME:                JOHN DOE
ADDRESS:             123 OAK ST.
                     ALLENTOWN,;PENNA.
                     98765
```

Figure 8-5. Output Screen Format with Display Constants and Variable Data

Using an input screen

Often a screen generated by an action program requires input from the terminal operator. Figure 8-6 shows a screen that requests specific information from the operator. This information is used to update a customer file.



```
CUSTOMER ACCOUNT  35618
PRODUCT NO:  _ _ _ _ _
QUANTITY:  _ _ _ _ _
UNIT COST:  _ _ _ _ _
REGION:  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
SHIPPING DATE: _ _/_ _/_ _
```

Figure 8-6. Screen Format Requiring Input Data

Validating input

When the terminal operator fills in the data requested, IMS checks the message for terminal commands. Then, it passes control to the screen format coordinator to validate the specific entries made. Once this is complete, IMS passes the validated data to the input message area of the action program named to process it.

Identifying invalid input

When the screen format coordinator detects invalid data, it causes the incorrect entries to blink at the terminal. This way the operator knows which entries to correct. Once the new data is entered and validated, IMS places it in the input message area of the action program to await processing.

Configuration requirements

To use screen format services in your action program, there are certain special considerations to take into account at IMS configuration.

First, you must specify the SFS parameter and RESFMT parameter in the OPTIONS section of the configuration. Also, when specifying the maximum size of the output message area, be sure it's large enough to handle the screen you want to build.

If you don't know the size of the resulting formatted screen, the action program can request the use of dynamic main storage. For more information on this, and other special considerations see the appropriate action programming manual.

For more information on IMS and screen format services, see the system support functions user guide, UP-8364, (current version).

IMS start-up requirements

Also, at IMS start-up, there are certain job control statements which you must include in the job control stream. These are covered in detail in the action programming manuals.

8.7. EDIT TABLE GENERATOR

The last special feature available to action programs that we discuss in this section is the **edit table generator.** As you know, the validity of input data is a major concern in an online system. Coding validation is a complicated and time-consuming process. Using the edit table generator simplifies the procedures.

Defining the edit table generator

The edit table generator is an offline IMS utility available in both single-thread and multithread. You use this utility to generate edit tables that convert freeform input entered by terminal operators into fixed formats required by the action program. Using these edit tables, IMS checks this input for data type, value ranges, and presence of required fields.

Using the Edit Table Generator

Configuration requirements

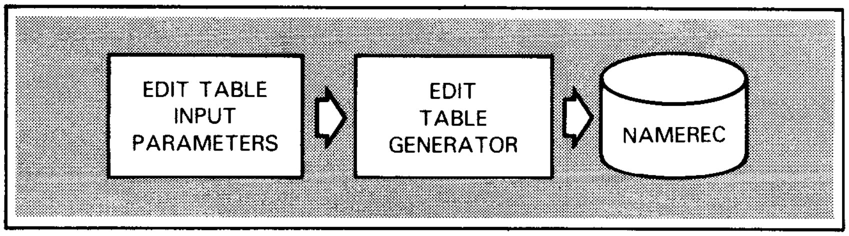
Each edit table is associated with an action. When you generate an edit table for an action, you specify the EDIT=tablename in the ACTION section of your IMS configuration. When the action program is scheduled, IMS loads the edit table (from NAMEREC) to format and validate input to that program.

Using the edit table generateor

The edit table generator utility is run offline. Input to the edit table generator is in the form of keyword parameters. These parameters define the edit table name, each field of the input message to be edited, and the editing criteria for each field.

A detailed explanation of the keyword parameters entered as input to the edit table generator can be found in the action programming manuals.

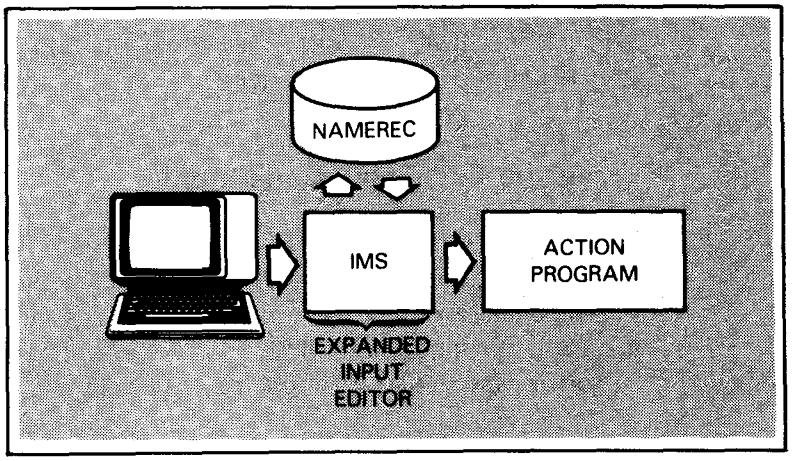
You can run the edit table utility before or after configuration. All output of the edit table generator is written to the named record (NAMEREC) file, from where it is loaded at the appropriate time by applications management.



How Terminal Input Is Edited

When a terminal operator enters a message for which there is an edit table, an IMS component called the expanded input editor validates the data according to the edit table and puts it into the format the action program requires.

The editor checks that the transaction code is the initial field of the input message, that input fields are the correct length, and whether an omitted entry is mandatory.

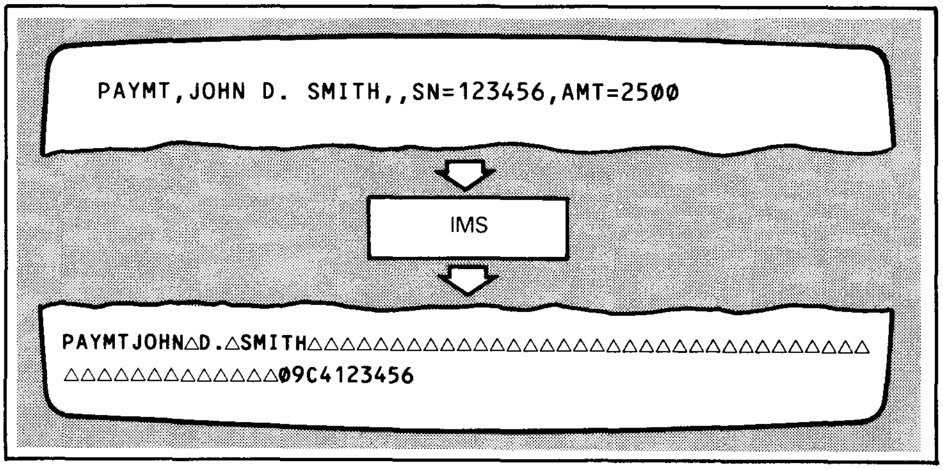


Example

Figure 8-7 shows input entered at a terminal and the same input as delivered to the action porgram. The commas in the terminal input are used as field separators. The two commas between SMITH and SN show that the operation chose not to enter an address field.

The edit table allows this omission since address was not defined as a mandatory field. The editor fills the name field with spaces to the length required by the action program, and also sends spaces for the omitted address field.

Note that the terminal operator reversed the order of the fields SN and AMT when entering them. Consequently, the edit table generator puts them in the order that the action program was designed to receive them. Also, the variable data for the AMT field is changed to a hexadecimal value according to the specifications made to the edit table generator.



```
PAYMT,JOHN D. SMITH,,SN=123456,AMT=25ØØ
```

```
PAYMTJOHN△D.△SMITH△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△△
△△△△△△△△△△△△△Ø9C4123456
```

Figure 8-7. Input Edited According to an Edit Table

8.8. INITIATING AN OS/3 JOB

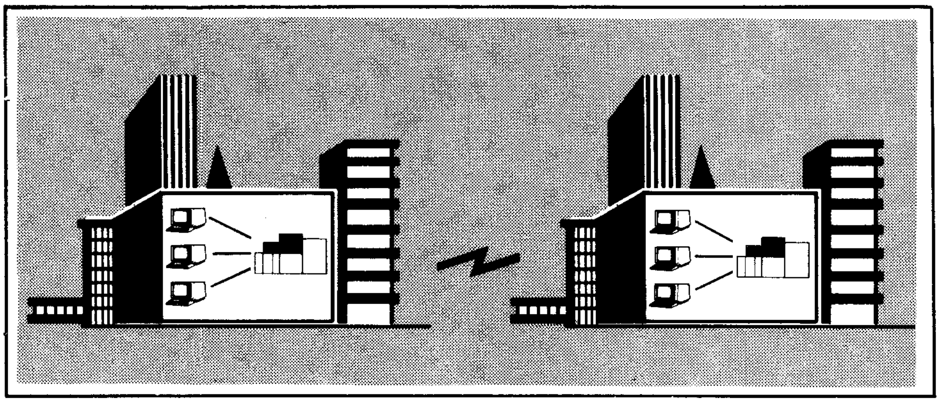
Reading a job control stream

IMS action programs in COBOL and basic assembly language can initiate the reading of a job control stream and scheduling a job for execution. They do this by issuing a RUN function call. For the format of this function call, see IMS action programming in COBOL and basic assembly language user guide, UP-9207 (current version).

9. IMS and Distributed Data Processing

Definition

A **distributed data processing system** allows two independent computer systems to communicate with one another. They are linked together through telecommunications so that each can use the other's capabilities and data. With distributed data processing (DDP), a Series 90 or a System 80 computer can access data on any other Series 90 or System 80 computer or send jobs to it.



DDP System with Electronic Link between Independent Computers

With distributed data processing, you are able to process IMS transactions at a remote computer. We call this capability the **IMS transaction facility**.

IMS transaction facility

Requirements

To use this facility, you must have two multithread IMS systems in operation – one at your site, and the other at the remote site.

9.1. TYPES OF TRANSACTION ROUTING

Types of routing

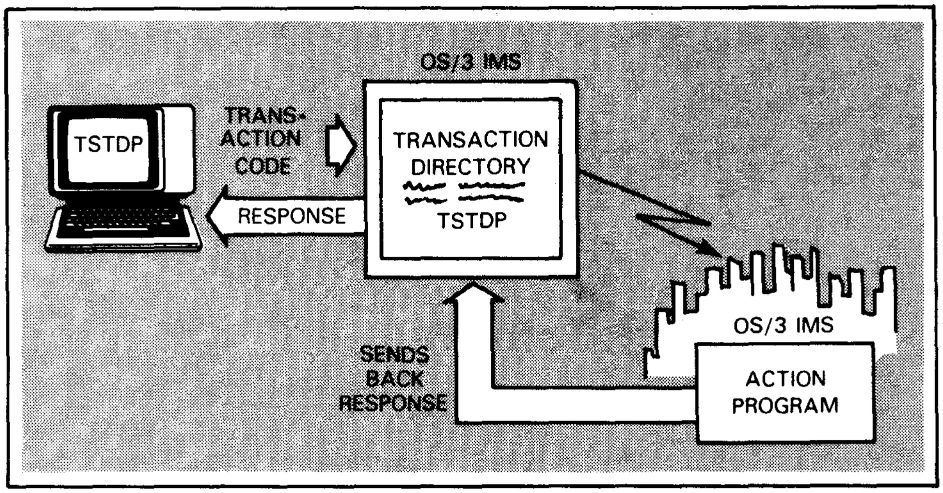
There are three ways in which IMS can route a transaction to a remote system:

1. Directory routing
2. Action program routing
3. Operator routing

Directory Routing

How transaction directory routing works

The terminal operator enters a transaction code that identifies the transaction for processing at a particular remote site. This transaction code and the site where it is to be processed are defined during IMS configuration.



Directory Routing

IMS sends the transaction code to the remote IMS, which handles processing from that point. The remote IMS processes the transaction and sends a message back to the terminal operator.

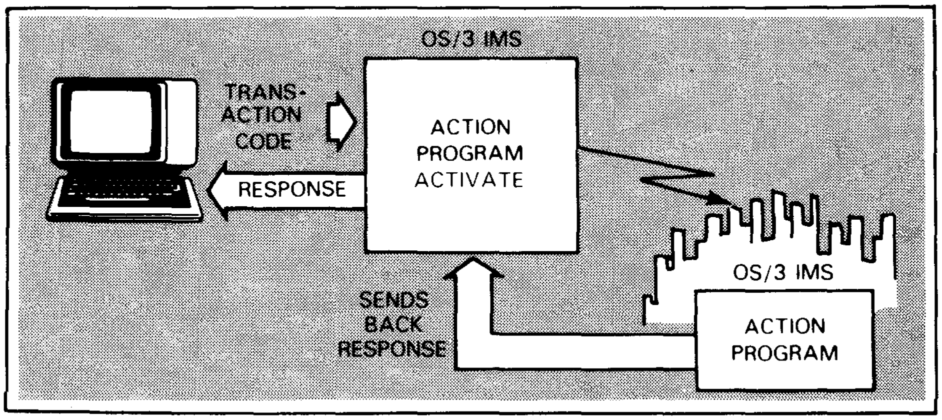
With transaction directory routing, the remote IMS can process dialog transactions as well as simple transactions.

Action Program Routing

How transaction program routing works

The terminal operator enters a transaction code that initiates a transaction at the local IMS system. The COBOL or basic assembly language (BAL) action program that is processing this transaction issues an ACTIVATE function call to IMS. This function call creates a message that initiates another IMS transaction at the remote site.

The remote IMS processes the transaction and sends a message back to the originating action program or its successor program after a CALL RETURN is issued. With transaction program routing, the remote IMS can process only simple transactions.



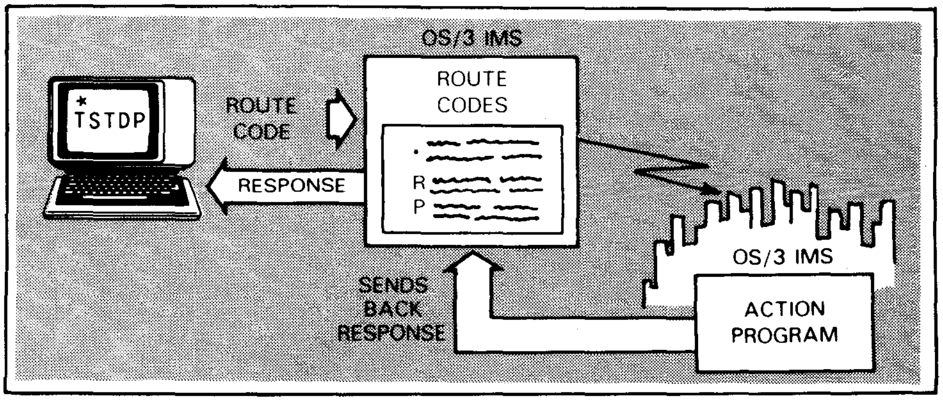
Action Program Routing

Operator Routing

How transaction operator routing works

Operator routing is similar to directory routing. The operator prefixes the transaction code with a routing character followed by a period, which identifies the transaction for remote processing. The special character is defined in the IMS configuration or at IMS start-up and is associated with a remote IMS system.

The remote IMS processes the transaction and sends a message back to the terminal operator. With operator routing, the remote IMS can process dialog transactions as well as simple transactions.



Operator Routing

9.2. CONFIGURING FOR THE IMS TRANSACTION FACILITY

To use the IMS transaction facility, you must generate an ICAM and IMS for the remote and local sites that support distributed data processing.

Configuring for distributed data processing

There are special considerations when configuring IMS. You must include:

- **LOCAP Section**

  This section defines remote systems.

- **DDPBUF and DDPSESS Parameters**

  Specifies the size required for DDP buffers and the number of concurrent DDP sessions in the GENERAL section.

- **TRANSACT Section**

  Use the LOCAP parameter in this section for a DDP transaction directory routing.

- **RCHAR Parameter**

  A routing code that defines or redefines a remote host for operator routing. You can include the RCHAR parameter in the LOCAP section or in the // PARAM LOCAP statement during IMS start-up. If you do both, the PARAM statement at start-up overrides the RCHAR statement just for that IMS session.

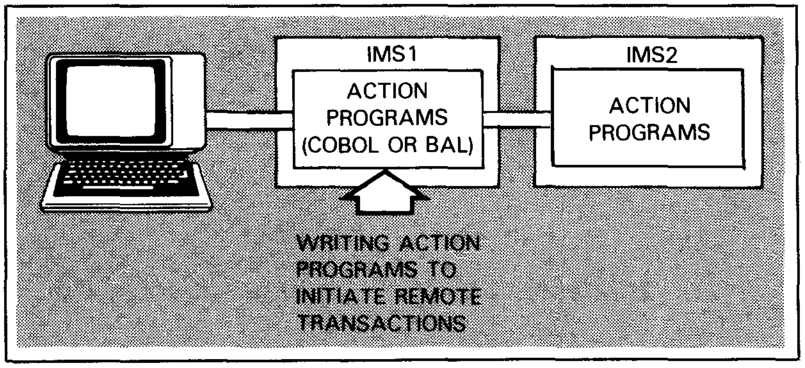
Further information

For specific network definition, configurator, and start-up requirements, see the IMS system support functions user guide, UP-8364 (current version). To write action programs for use in a DDP environment, refer to the action programming manuals.

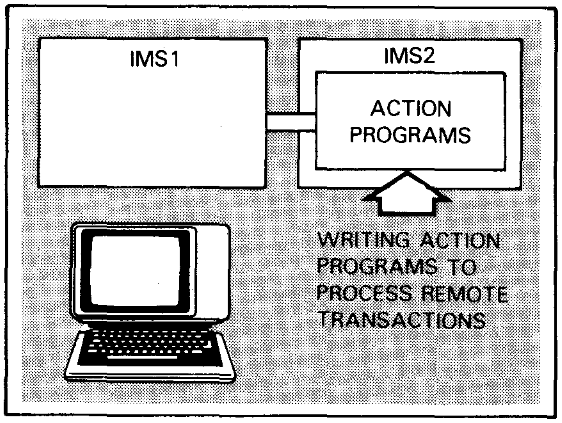
9.3. PROGRAMMING FOR DISTRIBUTED DATA PROCESSING

There are two aspects of programming for distributed data processing transactions:

1. You may be writing action programs at the primary IMS (the system where a remote transaction is initiated) to initiate a remote transaction through action program routing. Only COBOL and BAL action programs can initiate remote transactions.


IMS1
ACTION PROGRAMS (COBOL OR BAL)
IMS2
ACTION PROGRAMS
WRITING ACTION PROGRAMS TO INITIATE REMOTE TRANSACTIONS

2. You may be writing action programs at the secondary IMS (the system where a remote transaction is processed) to process a remote transaction initiated by operator, directory, or action program routing. COBOL, RPG II, and BAL action programs can process remote transactions. So can UNIQUE.



Initiating a Remote Transaction from an Action Program

In a program-initiated remote transaction, you make the decision whether to route the transaction to a remote system on the basis of some data the terminal operator enters or perhaps something you discover when you access your files or make computations.

Initiating remote transaction

External succession required

You initiate a remote transaction by identifying the remote IMS system *(locap-name)* in the output message header, building a message containing a transaction code in your output message area, and issuing an ACTIVATE function call. You must terminate your action program externally, naming a successor program at your local IMS system. Of course, you can reschedule the same action program as the successor.

Processing response message

Action programs at the remote IMS system process your message and send a response. Your successor program receives the response message in its input message area. You can then send an output message to the originating terminal, or you can issue another ACTIVATE call to initiate another remote transaction.

Processing a Remote Transaction at the Secondary IMS

Similar to processing local transaction

There is little difference between the way you process a remote transaction and the way you process a local transaction. You can use the same action programs to process both local and remote transactions.

Receiving input message

When the transaction begins, you receive an input message starting with a 1- to 8-character transaction code, just as with a local transaction.

Determining source of remote transaction

You can determine which remote IMS system initiated the transaction by testing the *source-terminal-id* field of the input message header. You determine whether the transaction was initiated by an operator (either operator or directory routing) or by an action program by testing the *DDP-mode* field of the program information block.

Operator-initiated transaction

You can use any type of action program succession with operator-initiated transactions. Once the transaction begins, the IMS transaction facility establishes a communications link which stays in effect until the transaction ends. When you use external succession, the terminal operator receives and responds to your output messages without entering any additional codes.

Program-initiated transaction

When the remote transaction is initiated by an action program, you send an output message to the originating action program's successor, which in turn sends an output message to the terminal. You must use normal termination when returning the output message.

General restrictions

There are a few general restrictions on processing remote transactions:

SEND function restriction

1. You cannot use the SEND function to output a message to the originating terminal (or any terminal at the remote IMS). However, you can use the SEND function to output a message to a terminal at your local IMS.

Continuous output restriction

2. You cannot send continuous output to the originating terminal. Again, you can use the SEND function to initiate continuous output at a local terminal using output-for-input queueing.

Auxiliary device restriction

3. You cannot send output to an auxiliary device attached to the originating terminal. However, you can output to local auxiliary devices using the SEND function.

Screen formatting restriction

4. You can send screen-formatted messages to the originating terminal in operator or directory routing, but not to the originating action program in action program routing.

9.4. PROCESSING REMOTE TRANSACTIONS AT THE TERMINAL

Operator-initiated transactions

There are two types of operator-initiated transactions:

1. Directory-routed transactions
2. Operator-routed transactions

Directory routing

You initiate a directory-routed transaction the same way as other transactions, because the transaction code itself tells IMS where to route the transaction for processing. The transaction code is associated with the remote system by the LOCAP parameter in the configurator TRANSACT section.

Operator routing

In operator routing, you decide which remote system will process the transaction by entering a route code, followed by a period, then the transactions code. For example, to initiate a UNIQUE transaction, you might enter:

```
*.OPEN CUSTFIL
```

The route code is associated with the remote system by the RCHAR parameter in the configurator LOCAP section or the PARAM LOCAP statement at start-up time.

Action program routing

The third type of remote transaction routing – action program routing – is not initiated by the terminal operator. You initiate a local transaction, which in turn initiates the remote transaction. However, you should be aware that a remote transaction may be in process while you are waiting for a response to your input message.

Using Terminal Commands in Remote Transaction Processing

Standard terminal commands

Six standard terminal commands affect remote transaction processing:

ZZCNC	Cancels a remote transaction
ZZTMD	Puts your terminal in test mode so that files at the remote system are not affected by update functions
ZZNRM	Restores your terminal to normal mode
ZZRSD	Resends the last output message from the action program at the remote system
ZZHLD	Suspends output to your terminal
ZZRDY	Releases the hold on output to your terminal

Master terminal commands

Two master terminal commands affect remote transaction processing:

ZZSHD	Shuts down the local IMS system. Remote transactions are allowed to continue during the shutdown time-out period the same as local transactions. At the end of the time-out period, they are aborted.
ZZHLT	Terminates the local IMS system immediately. Local and remote transactions are aborted. Files at both the local and remote systems may need to be recovered.

Terminal commands that can't be used

Other master terminal commands and the ZZMCH standard terminal command, cannot be used in the remote transaction processing environment.

IMS-Supplied Transactions

Remote UNIQUE transactions

As we mentioned before, you can route UNIQUE transactions to a remote system by entering a route code, followed by a period and the OPEN command. Once you start the UNIQUE transaction, you process it the same way as a local UNIQUE transaction. You don't enter another route code unless you enter the CLOSE or ZZCNC command to end the transaction or the transaction abnormally terminates.

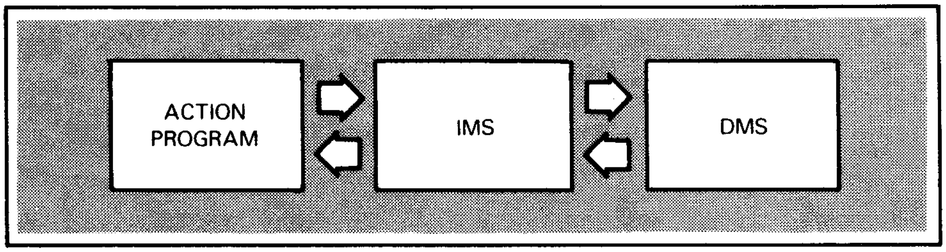
Other IMS transactions not supported

Except for UNIQUE, you cannot route IMS-supplied transactions to remote systems.

10. IMS Access to DMS Data Bases

IMS/DMS interface

The IMS/DMS interface allows IMS users to access OS/3 data base management system (DMS) data bases. This provides IMS users with the structural flexibility and powerful access mechanisms available to data base management systems.



Setting up a data base

Of course, to access a data base, you must first have set one up. The procedures for doing this are described in the DMS system support functions user guide/programmer reference, UP-8272 (current version).

Data base start-up

In addition, there are statements you include in the job control stream to start up your data base management system when it interfaces with IMS. For a complete discussion of these statements, as well as a detailed description of the IMS/DMS interface, consult the IMS/DMS interface user guide, UP-8748 (current version).

Here, we familiarize you with how you, as an IMS user, can take advantage of data base capabilities.

10.1. COMMUNICATING WITH A DATA BASE

Data manipulation language

COBOL action programs communicate with the data management system (DMS) through a special data base access language, called the data manipulation language (DML). DML statements resemble conventional COBOL programming statements. You embed DML statements in COBOL action programs.

RPG II and basic assembly language (BAL) action programs can access a data base only through IMS defined files. You include special clauses in the data definition to access a data base. You can also use UNIQUE to access a data base through defined files.

10.2. COBOL/DML ACTION PROGRAMS

Required entries

You write a COBOL action program that accesses a DMS data base as you would a conventional COBOL action program with these additions:

- A schema section with an INVOKE statement in the data division.
- A working-storage section in the data division.
- Data manipulation language (DML) statements in the procedure division.

INVOKE Statement

Data base terminology

The INVOKE statement identifies the schema, subschema, and device media control language modules. They describe the physical and logical structure of the data base as set up by the data base administrator.

Defining a schema

The **schema** is the global view of the data base's logical structure. It is a total picture of the logical relationships shared by data in the data base.

Defining a subschema

The **subschema** is a logical view of a segment of the data base. It is a portion of the schema used by a specific application.

Defining a device media control language

The **device media control language (DMCL)** module describes the physical structure of the data base. It identifies the location of data in the data base.

Working-storage Section

Defining a data base management communication area

To communicate with DMS, the INVOKE statement provides for a data base management communications area (DMCA) where control information between IMS and DMS is exchanged. This communications area is set up in the working-storage section or linkage section of the action program. DMS also inserts record, set, area names, and other data into this section.

DML Statements

Entering DML statements

The DML verbs that access the data base are placed in the procedure division of the action program. Because you're combining the capabilities of an IMS action program and the DMS data manipulation language (both having procedure division requirements), you must observe a combination of procedure division rules that conform to both systems. For a complete description of these rules, consult the IMS/DMS interface user guide, UP-8748 (current version).

Initiating data base communication

The first DML statement in your initial action program following the COBOL MOVE statement must always be an IMPART statement. This statement registers the run unit with the data base management system. With the successful execution of the IMPART statement, the data base management system opens the applicable data base areas.

DML formats

Once this is done, you can use nearly all the DML verbs to accomplish the record accessing capabilities you require. You'll find the formats for data manipulation language verbs in the DMS data manipulation language user guide/programmer reference, UP-8036 (current version).

Ending data base communication

Communication with successor program

When the IMS transaction is complete, you use the DEPART statement to end communication with the data base management system. If, however, you want to continue communication with the data base beyond the length of a single action program, you terminate the current program with an UNBIND statement. The succeeding action program then issues a BIND statement to continue IMS/DMS intercommunication without interruption.

Status and rollback function

In addition to the DML verbs, you also include in the procedure division the sections for handling DMS status reports and DMS rollback. The status section provides the capability of determining whether or not data base access was successful. The rollback section provides for recovery of data base files.

10.3. DATA DEFINITION USING A DATA BASE AS SOURCE

Creating defined files

You write a data definition to create a defined file in IMS. User-written action programs and UNIQUE can access defined files. The defined record management component of IMS constructs defined files from elements of existing disk files.

Using data base files

The IMS/DMS interface expands the capabilities of defined record management. It allows IMS to use a DMS data base subschema as a source for creating defined records. In this way, defined record management can construct defined records from DMS data base records, as well as from conventional files.

Action programs and terminal operators are not aware that the defined records are derived from data base records. The user-written action programs and UNIQUE do not change; they still access defined records. But now, defined records, in turn, access a data base.

Creating a DML/data definition

To create defined records which access a data base, you must construct an IMS data definition. Embedded in the data definition are special DML clauses that build data base relationships (such as owner and member) into the data definition.

Like the COBOL/DML action program, the data definition issues an INVOKE statement that identifies the schema, subschema, and device media control language module. In the definition division of the data definition you describe the defined record. The formats include special data definition clauses for the IMS/DMS interface. For a complete discussion of these formats, consult the IMS/DMS interface user guide, UP-8748 (current version).

10.4. PREPROCESSING

Preprocessing requirement

When your COBOL action programs and data definitions contain data manipulation language statements or syntax, they must be preprocessed before they are submitted to the COBOL compiler or data definition processor.

Figure 10-1 illustrates the preprocessing and compilation order required before linking your COBOL/DML program. Figure 10-2 illustrates the preprocessing and data definition processing for the data definition.

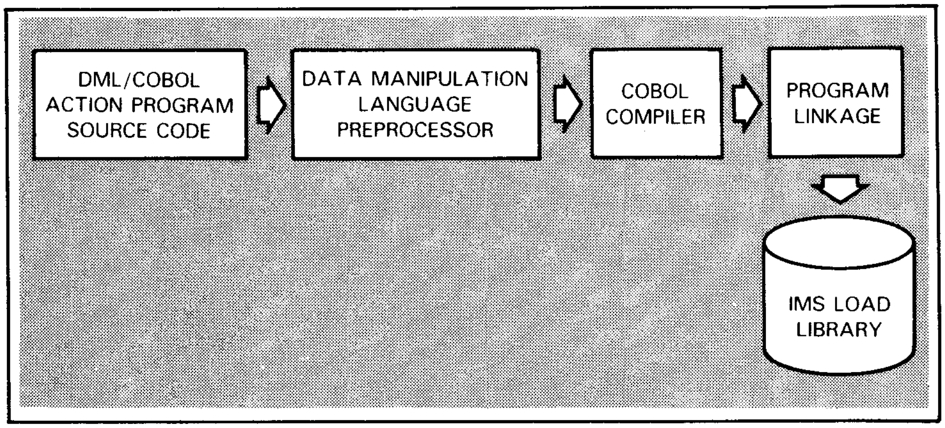

DML/COBOL ACTION PROGRAM SOURCE CODE
DATA MANIPULATION LANGUAGE PREPROCESSOR
COBOL COMPILER
PROGRAM LINKAGE
IMS LOAD LIBRARY

Figure 10–1. Sequence for Preprocessing and Compiling a COBOL/DML Program

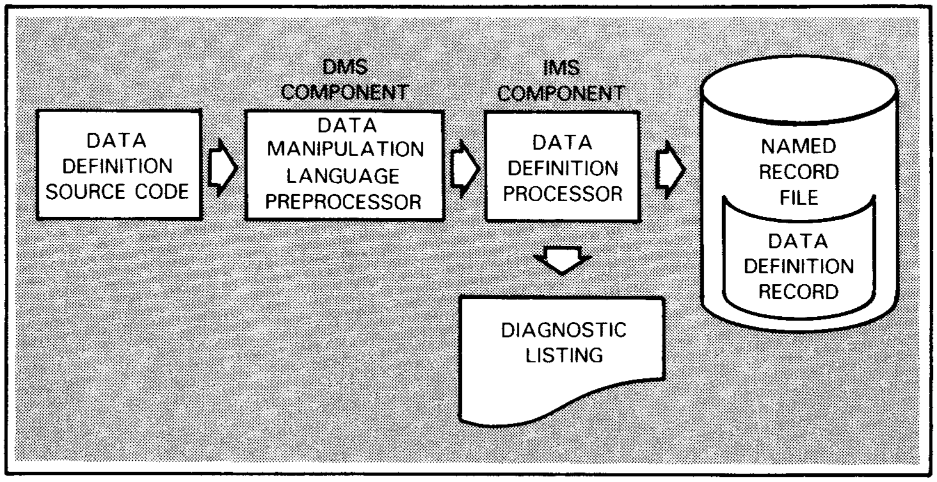

DMS COMPONENT
IMS COMPONENT
DATA DEFINITION SOURCE CODE
DATA MANIPULATION LANGUAGE PREPROCESSOR
DATA DEFINITION PROCESSOR
NAMED RECORD FILE
DATA DEFINITION RECORD
DIAGNOSTIC LISTING

Figure 10–2. Creating a Data Definition Record

10.5. CONFIGURATION CONSIDERATIONS

To communicate with a data base, there are certain parameters to include in your IMS configuration. You must:

Required parameters

- Link the DMS modules to IMS by specifying DMS=YES in the OPTIONS section of configurator input.

- Configure COBOL action programs as shared for better performance in the multithread environment; specify TYPE=SHR in the PROGRAM section and SHRDSIZE=n (shared program size) in the ACTION section of configurator input.

11. Batch Processing of Transactions

Definition

The batch processor is an optional component of IMS. It enables you to enter input for IMS transactions in card format instead of through a display terminal and provides direct output to a printer or printer file. You can process batch transactions online (with other IMS programs using the normal terminal communications network) or offline (without an active terminal network).

Uses for batch processing

Batch processing can be a very useful tool. You can use it to:

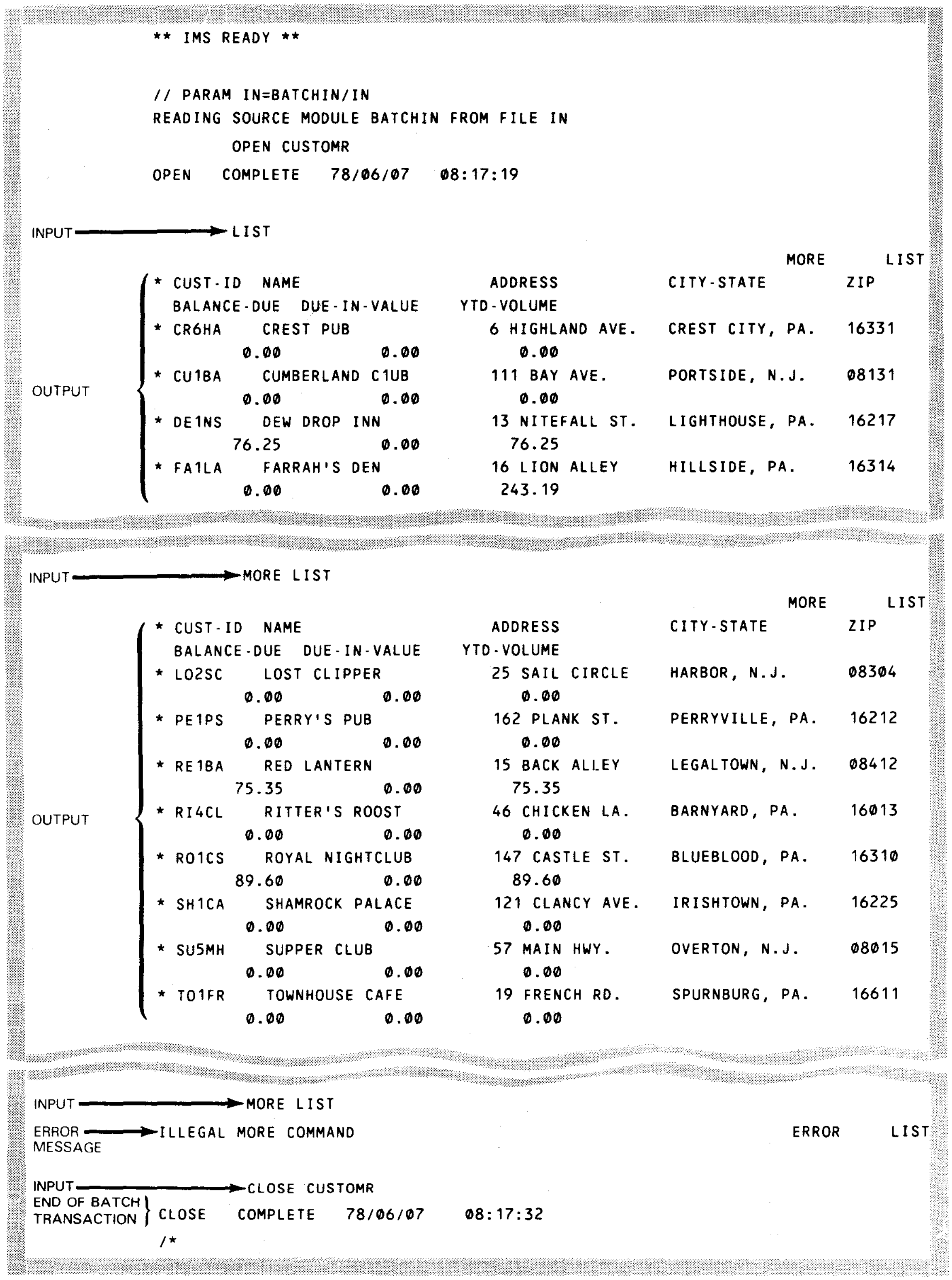
- Print a file at the end of a day's production. This is particularly convenient for reviewing files that undergo massive updating.
- Print a defined file that you created from your user file. This activity is not practical in normal operations from a display terminal.
- Test UNIQUE-based file query and update dialogs designed for routine use at production terminals, as well as test user-written action programs that your operators initiate as transactions during normal production. Printed output listed by the batch processor reproduces all input and output messages and provides a permanent hard-copy of each transaction.

Example

Figure 11-1 illustrates the printed output resulting from a batch transaction using UNIQUE.

Note that both input and output messages are displayed, as well as an error message when the operator attempts another MORE LIST command after the entire contents of the file has already been displayed.

BATCH PROCESSING



```
** IMS READY **


// PARAM IN=BATCHIN/IN
READING SOURCE MODULE BATCHIN FROM FILE IN
        OPEN CUSTOMR
OPEN   COMPLETE   78/06/07   08:17:19

LIST
                                                                MORE      LIST
* CUST-ID  NAME                   ADDRESS           CITY-STATE        ZIP
  BALANCE-DUE  DUE-IN-VALUE     YTD-VOLUME
* CR6HA    CREST PUB              6 HIGHLAND AVE.   CREST CITY, PA.   16331
         0.00          0.00        0.00
* CU1BA    CUMBERLAND C1UB        111 BAY AVE.      PORTSIDE, N.J.    08131
         0.00          0.00        0.00
* DE1NS    DEW DROP INN           13 NITEFALL ST.   LIGHTHOUSE, PA.   16217
        76.25          0.00       76.25
* FA1LA    FARRAH'S DEN           16 LION ALLEY     HILLSIDE, PA.     16314
         0.00          0.00      243.19

MORE LIST
                                                                MORE      LIST
* CUST-ID  NAME                   ADDRESS           CITY-STATE        ZIP
  BALANCE-DUE  DUE-IN-VALUE     YTD-VOLUME
* LO2SC    LOST CLIPPER           25 SAIL CIRCLE    HARBOR, N.J.      08304
         0.00          0.00        0.00
* PE1PS    PERRY'S PUB            162 PLANK ST.     PERRYVILLE, PA.   16212
         0.00          0.00        0.00
* RE1BA    RED LANTERN            15 BACK ALLEY     LEGALTOWN, N.J.   08412
        75.35          0.00       75.35
* RI4CL    RITTER'S ROOST         46 CHICKEN LA.    BARNYARD, PA.     16013
         0.00          0.00        0.00
* RO1CS    ROYAL NIGHTCLUB        147 CASTLE ST.    BLUEBLOOD, PA.    16310
        89.60          0.00       89.60
* SH1CA    SHAMROCK PALACE        121 CLANCY AVE.   IRISHTOWN, PA.    16225
         0.00          0.00        0.00
* SU5MH    SUPPER CLUB            57 MAIN HWY.      OVERTON, N.J.     08015
         0.00          0.00        0.00
* TO1FR    TOWNHOUSE CAFE         19 FRENCH RD.     SPURNBURG, PA.    16611
         0.00          0.00        0.00

MORE LIST
ILLEGAL MORE COMMAND                                            ERROR     LIST

CLOSE CUSTOMR
CLOSE   COMPLETE   78/06/07    08:17:32
/*
```

Figure 11-1. UNIQUE Transaction Listed by Batch Transaction Processor

11.1. PROCESSING IN THE BATCH ENVIRONMENT

Creating pseudoterminals

Batch transactions are processed as if they originate from one or more actual terminals. Based on the input you provide to the cofigurator in the BATCH parameter, IMS creates one or more pseudoterminals to initiate batch processing.

When processing begins, the batch processor responds to input from the pseudoterminals and lists output on a print file assigned to each pseudoterminal. This output is a step-by-step record of each transaction initiated.

Types of transaction supported

The batch processor handles (in online or offline mode) the following four types of transactions:

1. UNIQUE dialogs – Initiated by the OPEN command and comprising any of the UNIQUE command repertoire
2. Other transactions – Initiated by a 1- to 8-character transaction code, such as those initiating user-written action programs
3. Two of the standard terminal commands: ZZTMD and ZZNRM
4. SWTCH transaction – To send output to an online terminal

11.2. PREPARING FOR BATCH PROCESSING

Configuring IMS

BATCH parameter

To include batch processing modules in your IMS configuration, specify the BATCH parameter in the NETWORK section. This parameter has two forms:

1	BATCH=YES
2	BATCH=n

When you specify BATCH=YES, IMS generates one batch pseudoterminal to initiate batch transactions. When you specify BATCH=n, IMS generates the number of pseudoterminals designated by n.

Single and multithread batch operations

In a single-thread environment, you can only process one batch transaction at a time. Consequently, BATCH=YES or BATCH=1 is your only option. On the other hand, in a multithread environment, it is possible to process up to four batch transactions simultaneously.

Executing IMS

Considerations when executing IMS

In addition to specifying the BATCH parameter at IMS configuration, there are certain modifications you must make to the IMS execution run stream. They are:

- Assigning source module input files or embedding input source data in the control stream.
- Assigning printer files.
- Inserting the PARAM BATCH statement and optional PARAM IN statements to control the batch processor (these must always follow any other PARAM statements present).

Input

Card input

You can enter input to the processor on cards if you choose; simply include card-images directly in the IMS execution control stream. You place the cards after the PARAM BATCH statement. They can be interspersed among the PARAM IN statements.

Disk input

Sometimes it is more convenient to record the card-images of your input messages on disk and place them in disk library files for easy reference. When you choose this option, you must name the source module input files during the IMS execution job.

Using the PARAM IN statement

You name them using the PARAM IN statement which identifies the module name and the file where it resides. PARAM IN statements follow the PARAM BATCH statement. There can be as many PARAM IN statements (or none) as you choose.

Printer Files

Purpose of printer file

You must have one printer file for each batch pseudoterminal. Each printer file records the output of batch transactions initiated at its pseudoterminal.

PARAM Statements

Using the PARAM BATCH statement

You specify the PARAM BATCH or PARAM BA statement to designate the type of batch processing you require (NO, OFFLINE, or ONLINE). The default is PARAM BA=NO which indicates batch processing is not supported.

Offline batch processing

Offline batch processing is processing done without an active communications network.

Online batch processing

Online batch processing is processing with an active communications network.

11.3. CONTROLLING BATCH PROCESSING

Offline Processing

Controlling offline processing

Offline IMS batch processing occurs without an active communications network. All processing is controlled by the IMS execution job control stream. You specify the PARAM BATCH=OFFLINE statement followed by PARAM IN statements identifying input source modules and/or embedded data sets. The order in which these appear in the control stream determines the order in which the batch processor processes them.

Online Processing

Controlling online processing

Online batch processing requires an active communications network. You control online batch processing from the master terminal using the ZZBTH master terminal command. By entering this command, the terminal operator controls which source modules are processed and when.

Performance considerations

Batch processing conducted online can adversely effect the performance of other IMS terminals. It is best to confine batch processing to slow periods or just prior to IMS shutdown. The number of batch transactions in progress also effects IMS performance.

Parameter specifications

For online batch processing specify the PARAM BATCH=ONLINE statement. It is optionally followed by PARAM IN statements and embedded data sets. No batch processing is performed until the first ZZBTH master terminal command is entered.

Forms of the ZZBTH command

The ZZBTH command has several forms. They are:

```
ZZBTH ⎧module-name,file-name⎫
      ⎪*[,ALL]              ⎪
      ⎨CANCEL               ⎬
      ⎪PAUSE                ⎪
      ⎩RESUME               ⎭
```

Module name format

You must identify at the terminal the name of the input source module.

ZZBTH ALL format

Specifying the ZZBTH *[,ALL] format allows you to use source data identified by PARAM IN statments and/or embedded data sets. Only this form allows for source input from the job control stream. Otherwise, source input must be specified using the ZZBTH module-name.file-name format.

When you enter the ZZBTH command at the master terminal and the batch processor recognizes batch input, the message BATCH INITIATED is sent to the master terminal and processing begins.

Single and multithread processing

With single-thread IMS, only one ZZBTH command can be entered at a time. When processing is complete, you can enter another ZZBTH command. In multithread IMS, several ZZBTH commands can be processed sumultaneously, up to the maximum number specified in the BATCH=n keyword parameter (11.2).

Diagnostic message

If the master terminal operator attempts to enter a ZZBTH command in excess of the maximum number of batch modules that can be processed concurrently (or a single-thread operator enters a second command before the first is complete), IMS returns a diagnostic message:

```
TOO MANY ZZBTH COMMANDS ENTERED - COMMAND IGNORED.
```

ZZBTH CANCEL format
ZZBTH PAUSE format
ZZBTH RESUME format

The ZZBTH CANCEL command cancels a batch transaction currently in progress; the ZZBTH PAUSE command allows you to suspend batch processing temporarily; ZZBTH RESUME restarts the batch transaction after a suspension. For further details on using the ZZBTH master terminal command, consult the IMS terminal users guide, UP-9208 (current version).

11.4. ERROR MESSAGES AND RECOVERY

Error messages

In addition to the error messages generated by IMS (which the batch processor includes in its output listing to each pseudoterminal), the batch processor creates a set of its own error messages. It sends these messages to the master terminal or includes them in the output lising for the pseudoterminal. For a complete listing of batch processing error messages, see the IMS system support functions user guide/programmer reference, UP-8364 (current version).

Recovery

For file recovery during batch processing, you enter a DLMSG input message via cards. You supply one DLMSG card-image for each batch pseudoterminal. The DLMSG command provides in print the last output message generated by each transaction. Using this output, you can decide where the system aborted and determine where to restart your batch processing.

12. Defined Files and Data Definitions

12.1. COMPARING CONVENTIONAL AND DEFINED FILES

Describing conventional files

You are obviously familiar with **conventional files.** These are the data files you normally access in your application programs. You designed these files; you know what records they contain and the size and location of fields in the records. In addition, you know their physical location – that is, exactly what disk they're on. Consequently, whenever you want to retrieve a record, you simply specify what disk it is contained on and its file name. IMS file management, in conjunction with data management, locates the record and brings it into your program.

Purpose of defined files

Defined files are built from conventional files. They provide a means of creating a file for a particular application from existing data without having to physically create the file.

Example

Figure 12–1 shows that the defined record PAYROLL was derived from the conventional record PERSNEL. The defined record PAYROLL, contains only four of the eight fields present in the conventional file record. Thus, we have drawn records from the conventional file PERSNEL to create a new defined file called EMPINFO.

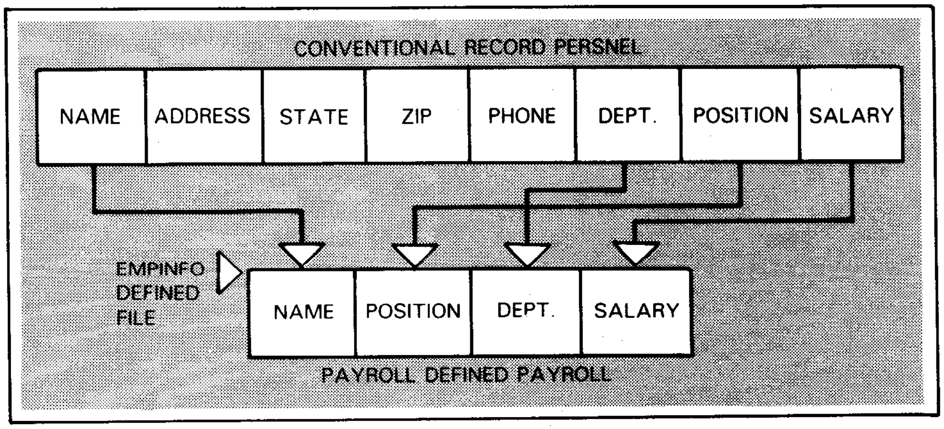


Figure 12–1. Creating a Defined Record from a Conventional Record

What makes a defined file different? The difference is that the defined file never physically exists.

No physical location

Description of the defined file

The defined file in Figure 12-1 doesn't occupy any physical location on disk. The only thing that exists is a description of the defined file which you provide using what is called a **data definition**. This description provides IMS defined record management with enough data to go out to your conventional files and pull in the fields required to make up the defined record.

Advantages of defined files

Defined files can be very useful. For one thing, they give you the flexibility of data base management without the tremendous amount of time and effort that creating a data base requires. Also, by using defined files, you can avoid unnecessary repetition of data and waste of disk space. And, at the same time, you're able to combine data from several conventional files and manipulate that data so that it is best suited to a particular application.

12.2. CREATING DEFINED FILES

Creating a data definition

To use defined files you must create a data definition. The data definition is a description of the defined file. It tells IMS what conventional file or files you're drawing data from.

Using the data definition language

Running the data definition processor

You write a data definition using the IMS data definition language. This language resembles COBOL. Once written, you submit the data definition to an IMS utility program called the data definition processor. (Figure 12-2).

The data definition processor:

Output of the data definition processor

- Creates a data definition record in the NAMEREC file.
- Produces a printed description of the defined file and a diagnostic listing.

Preparing the NAMEREC file

Since the data definition is stored on the NAMEREC file, you must initialize it (using the NAMEREC file utility or the configurator before running the data definition processor).

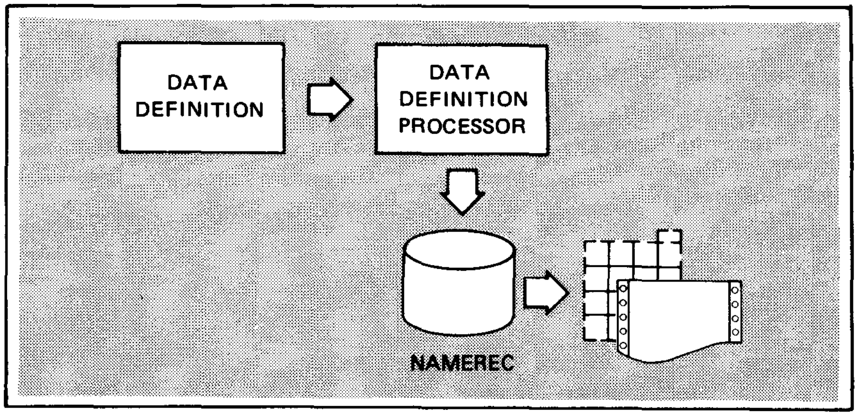


Figure 12-2. Creating a Data Definition

12.3. DESCRIBING DEFINED FILES AND DEFINED RECORDS

In the data definition, you describe the structure of the defined file and the defined record it contains.

Types of defined files

A **defined file** can be structured in various ways. It can:

- Be identical to an already existing conventional file. The most common reason for doing this is for use with UNIQUE.
- Describe the conventional file differently.
- Combine data from more than one conventional file.

With each run of the data definition processor you create one defined file. This defined file can contain several different types of defined records.

Types of defined records

A **defined record** can be structured in various ways. It can consist of:

- All or part of a record from one conventional file
- All or parts of several records from the same conventional file
- All or parts of several records from different conventional files.

Assigning an identifier

For each type of defined record you describe, you include an identifier. The identifier is a portion of the record key of the conventional record from which the defined record is being created (Figure 12-3). These identifiers (or mini-keys) serve to locate the data in your conventional files.

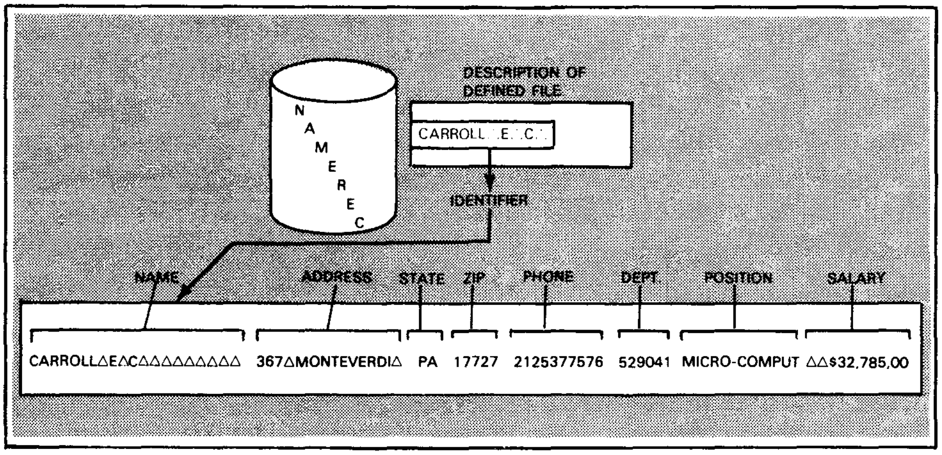


Figure 12–3. Using the Identifier to Locate the Conventional Record

File restrictions

Because IMS is dependent upon linking defined records to conventional records through the use of record keys, you can build defined files from indexed files (ISAM), multiple indexed random access method (MIRAM), or a database subschema using the IMS/DMS interface. You can only use nonindexed files (DAM or MIRAM) when they are combined with indexed files or a subschema.

12.4. ACCESSING DEFINED RECORDS

Function of defined record management

Each time an action program calls a defined record, defined record management (a component of IMS file management) checks the NAMEREC file for the description of the defined record. The identifier statement, which is part of your data definition, is used to locate the conventional records that contain the data used in building the defined record. Figure 12-4 illustrates this process.

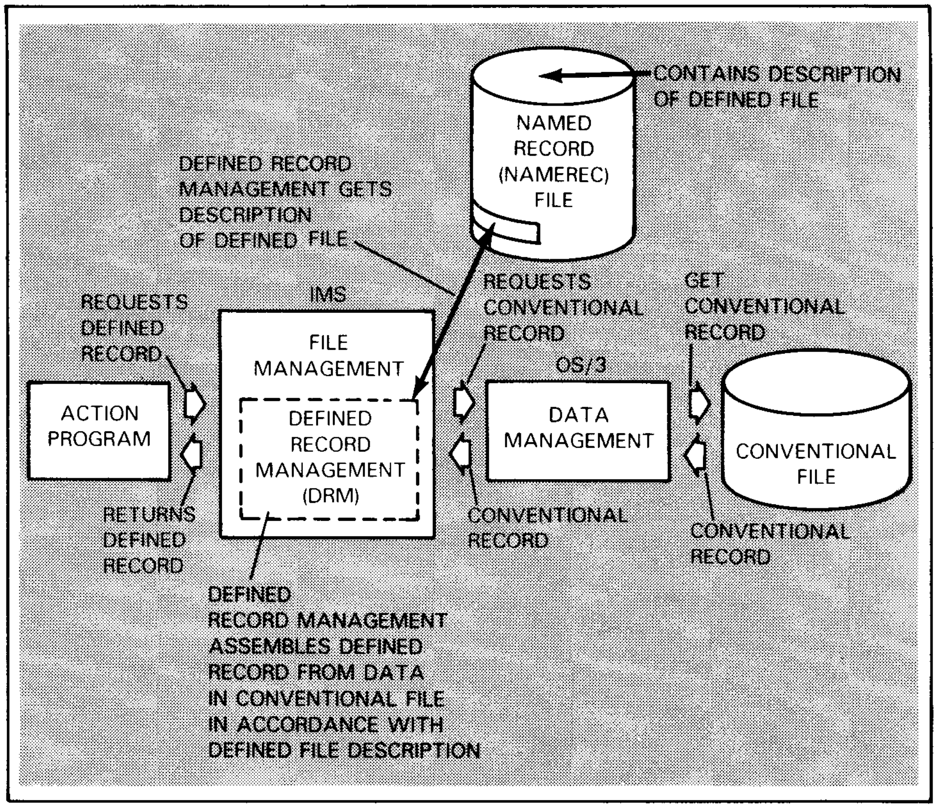

CONTAINS DESCRIPTION OF DEFINED FILE
NAMED RECORD (NAMEREC) FILE
DEFINED RECORD MANAGEMENT GETS DESCRIPTION OF DEFINED FILE
IMS
REQUESTS DEFINED RECORD
FILE MANAGEMENT
REQUESTS CONVENTIONAL RECORD
GET CONVENTIONAL RECORD
OS/3
ACTION PROGRAM
DEFINED RECORD MANAGEMENT (DRM)
DATA MANAGEMENT
CONVENTIONAL FILE
RETURNS DEFINED RECORD
CONVENTIONAL RECORD
CONVENTIONAL RECORD
DEFINED RECORD MANAGEMENT ASSEMBLES DEFINED RECORD FROM DATA IN CONVENTIONAL FILE IN ACCORDANCE WITH DEFINED FILE DESCRIPTION

Figure 12-4. Accessing a Defined Record from an Action Program

Using the defined record area

Defined record management uses the defined record area as a work area during this process of retrieving defined records.

The defined record area is part of the activation record. When an action program intends to access defined records, it must set up a defined record area. Once the defined record is retrieved, defined record management makes it available to the action program for processing.

12.5. DEFINED FILES AND UNIQUE

Defined files required by UNIQUE

UNIQUE is the IMS-supplied set of action programs for file retrieval and updating. It accesses defined files only. Consequently, to use UNIQUE you must create defined files. Often, to get online quickly, you can create defined files that are identical to your conventional files. Thus, you can perform limited transaction processing with UNIQUE while writing and testing your own action programs.

For a detailed description of how to use UNIQUE, consult the IMS data definition and UNIQUE user guide, UP-9209 (current version).

12.6. CREATING A DATA DEFINITION

Structure of a data definition

Figure 12-5 shows the overall structure of a data definition. It contains:

- **Identification Division**
  Assigns a unique name to the data definition

- **Data Division**
  Describes the conventional files from which the defined file is extracted

- **Definition Division**
  Describes the structure of the defined file

```
IDENTIFICATION DIVISION.
PROGRAM-ID.data-definition-name.
[AUTHOR.comment-entry.]
[INSTALLATION.comment-entry.]
[DATE-WRITTEN.comment-entry.]
[DATE-COMPILED.comment-entry.]
[SECURITY.comment-entry.]
[REMARKS.comment-entry.]

DATA DIVISION.
FILE SECTION.
FD file-name-1.record-description
               [record-description]...
[FD file-name-2.record-description
                [record-description]...]...

DEFINITION DIVISION.
defined-file-definition
```

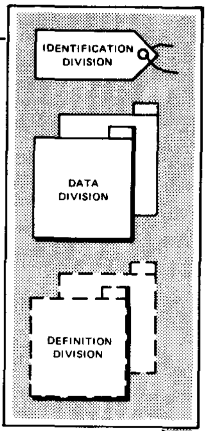
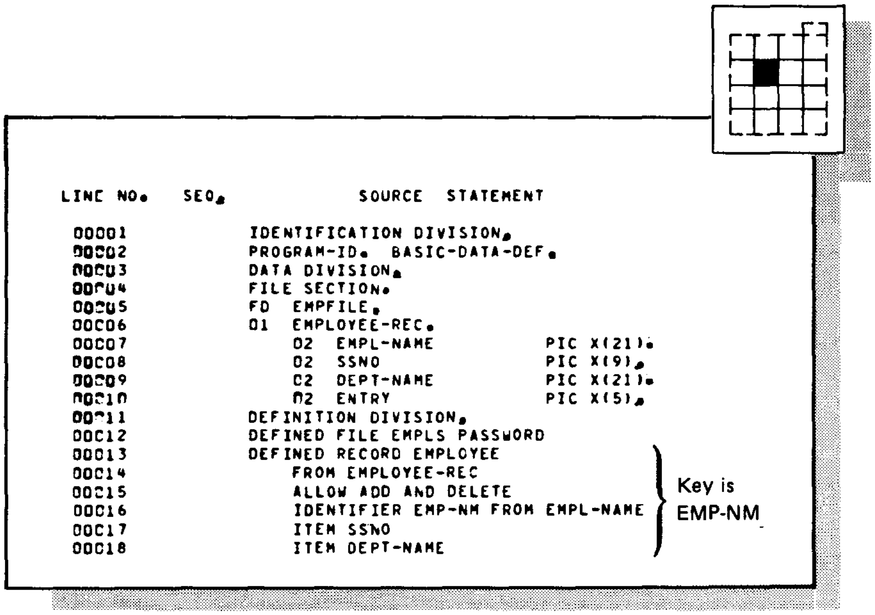


Figure 12-5. Overall Format of a Data Definition

The identification and data divisions are similar to those in a COBOL program. While the definition division is unique to IMS, its syntax is similar to COBOL.

Sample data definition 1

To describe how you set up a data definition, refer to Figure 12-6. It shows a sample data definition for the defined file EMPLS.



```
LINE NO.   SEQ.            SOURCE  STATEMENT

                     IDENTIFICATION DIVISION.
                     PROGRAM-ID.  BASIC-DATA-DEF.
                     DATA DIVISION.
                     FILE SECTION.
                     FD  EMPFILE.
                     01  EMPLOYEE-REC.
                         02  EMPL-NAME            PIC X(21).
                         02  SSNO                 PIC X(9).
                         02  DEPT-NAME            PIC X(21).
                         02  ENTRY                PIC X(5).
                     DEFINITION DIVISION.
                     DEFINED FILE EMPLS PASSWORD
                     DEFINED RECORD EMPLOYEE
                         FROM EMPLOYEE-REC
                         ALLOW ADD AND DELETE
                         IDENTIFIER EMP-NM FROM EMPL-NAME
                         ITEM SSNO
                         ITEM DEPT-NAME
```

Key is EMP-NM

Figure 12-6. Data Definition for Defined File EMPLS

Identification Division

Data definition name

Note that in the identification division the data definition name is BASIC-DATA-DEF.

Data Division

Conventional file(s)

The data division describes the conventional file from which the defined file is built. The conventional file name is EMPFILE. It contains EMPLOYEE-REC records and each record contains four fields - EMPL-NAME, SSNO, DEPT-NAME, and ENTRY.

Definition Division

Defined file name

The definition division describes the structure of the defined file (EMPLS).

The defined file statement begins a defined file definition and names the file. The name distinguishes the defined file from other defined files which may exist on the named record (NAMEREC) file.

Other uses of the defined file name

You use the defined file name to refer to the defined file in a number of places outside the data definition:

- Keyword parameters DFILE and DDRECORD in the ACTION section of the configuration
- Keyword parameters FN and DDN in the password definition input to the NAMEREC file utility
- The defined-file-name parameter in action program function calls to defined record management
- The defined-file-name and data-def-rec-name fields in the program information block for COBOL, BAL, and RPG II action programs

The IMS system support functions user guide, UP-8364 (current version) describes IMS configuration and the NAMEREC file utility. Action programs are discussed in the current version of the IMS action programming in COBOL and BAL user guide, UP-9207 and the IMS action programming in RPG II user guide, UP-9206.

Specifying a password

The PASSWORD parameter allows access to the defined files. Passwords are used only in conjunction with UNIQUE action programs.

Use with UNIQUE

Creating passwords with NAMEREC file utility

When you specify PASSWORD, terminal operators enter the *defined-file-name* as a password in the UNIQUE OPEN command to access a defined file. If you choose, you can omit PASSWORD and define a password for use with UNIQUE by running the NAMEREC file utility. This allows you to limit defined file access to specific UNIQUE terminals and use multiple passwords to access the same defined file.

Defined record name

The defined record statement names the defined record.

Source of the defined record

The FROM clause identifies the conventional record from which the defined record is built. In Figure 12-6 the conventional record is EMPLOYEE-REC. Since this is the only conventional record described, the FROM clause seems irrelevant. However, many data definitions describe several conventional files and many defined records. In such a case, it is important to link each defined record to the conventional records from which it is built.

You write a separate defined record definition for each defined record type in the defined file.

Allowing for updating

The ALLOW ADD AND DELETE clause permits terminal operators using UNIQUE or user-written action programs to add or delete defined records.

Specifying the identifier

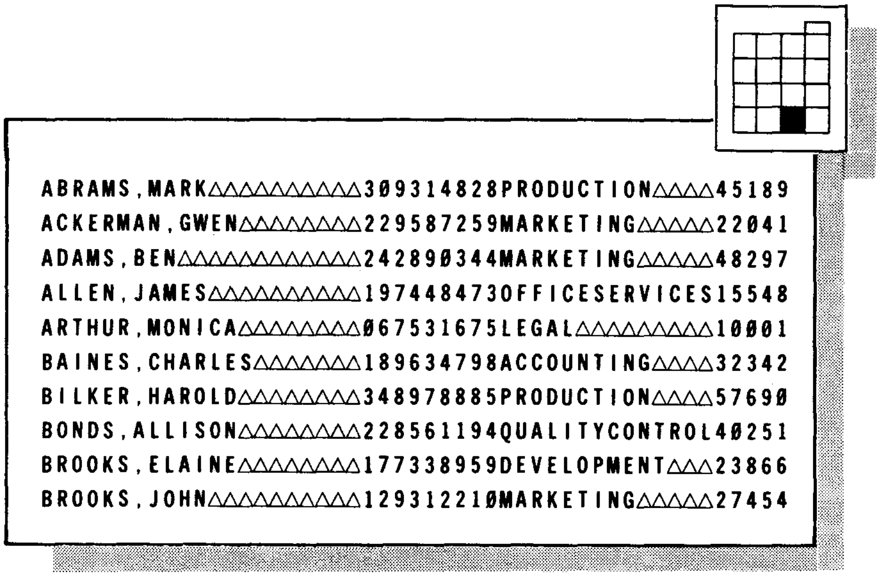
The identifier statement locates the data in the conventional file. The identifier is part of the record key of the conventional file (Figure 12-3). Because identifiers are derived from key fields, defined records can only be built from records in:

- indexed files; or
- a data base subschema.

Defining other fields

The item statement describes other fields in the conventional record which are to be included in the defined record.

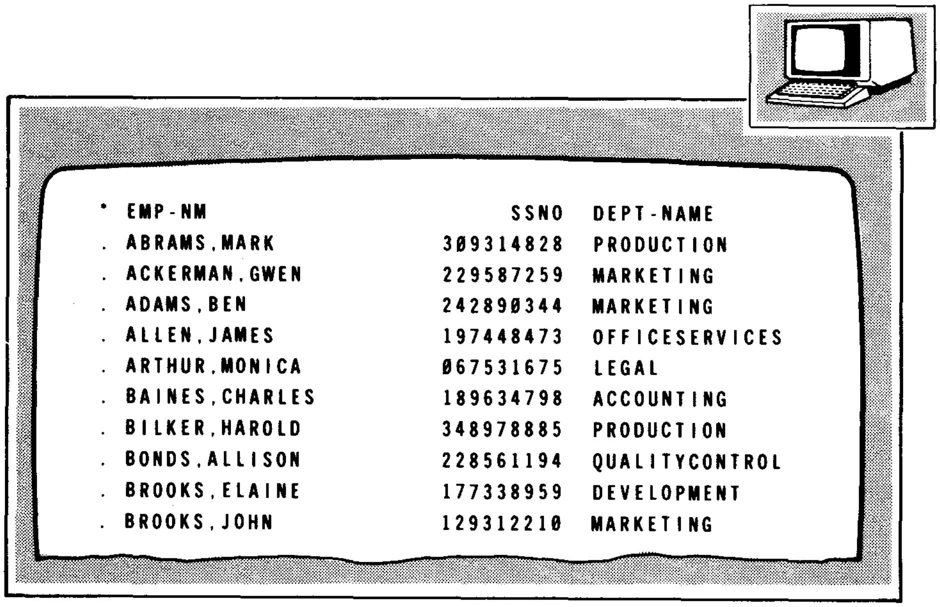
At this point, the data definition for defined record EMPLOYEE is complete. Figure 12-7 illustrates conventional records EMPLOYEE-REC.



```
ABRAMS,MARK△△△△△△△△△△△3Ø9314828PRODUCTION△△△△45189
ACKERMAN,GWEN△△△△△△△△△229587259MARKETING△△△△△22Ø41
ADAMS,BEN△△△△△△△△△△△△△24289Ø344MARKETING△△△△△48297
ALLEN,JAMES△△△△△△△△△△△197448473OFFICESERVICES15548
ARTHUR,MONICA△△△△△△△△△Ø67531675LEGAL△△△△△△△△△1ØØØ1
BAINES,CHARLES△△△△△△△△189634798ACCOUNTING△△△△32342
BILKER,HAROLD△△△△△△△△△348978885PRODUCTION△△△△5769Ø
BONDS,ALLISON△△△△△△△△△228561194QUALITYCONTROL4Ø251
BROOKS,ELAINE△△△△△△△△△177338959DEVELOPMENT△△△23866
BROOKS,JOHN△△△△△△△△△△△12931221ØMARKETING△△△△△27454
```

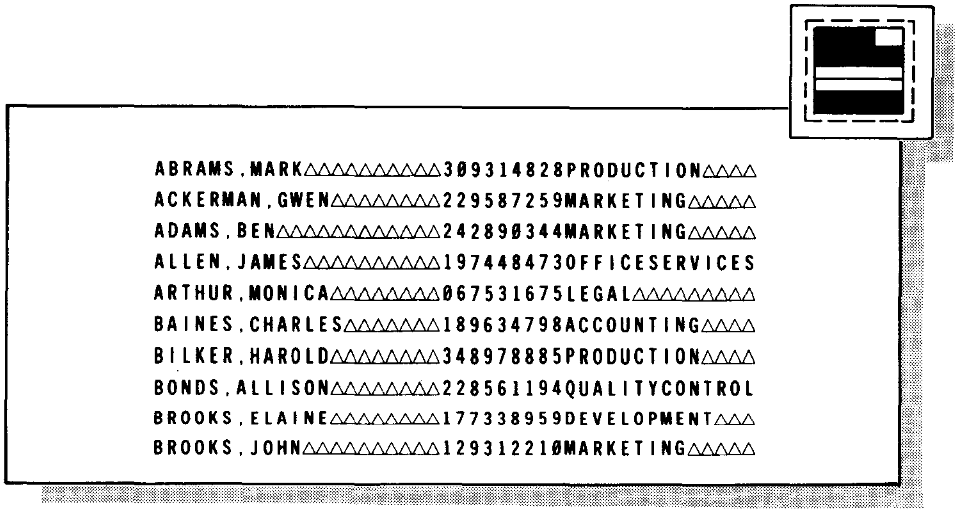
Figure 12-7. Excerpt from EMPFILE Showing Conventional EMPLOYEE-REC Records

Figure 12-8 shows the resulting EMPLOYEE defined records displayed at the terminal in response to a UNIQUE LIST command. Figure 12-9 shows how the same 10 records are delivered by defined record management to an action program.



```
• EMP-NM                    SSNO   DEPT-NAME
. ABRAMS,MARK          3Ø9314828   PRODUCTION
. ACKERMAN,GWEN        229587259   MARKETING
. ADAMS,BEN            24289Ø344   MARKETING
. ALLEN,JAMES          197448473   OFFICESERVICES
. ARTHUR,MONICA        Ø67531675   LEGAL
. BAINES,CHARLES       189634798   ACCOUNTING
. BILKER,HAROLD        348978885   PRODUCTION
. BONDS,ALLISON        228561194   QUALITYCONTROL
. BROOKS,ELAINE        177338959   DEVELOPMENT
. BROOKS,JOHN          12931221Ø   MARKETING
```

Figure 12–8. First Few EMPLOYEE Defined Records as Listed at a Terminal by UNIQUE



```
ABRAMS,MARK△△△△△△△△△△△△3Ø9314828PRODUCTION△△△△
ACKERMAN,GWEN△△△△△△△△△△229587259MARKETING△△△△△
ADAMS,BEN△△△△△△△△△△△△△△24289Ø344MARKETING△△△△△
ALLEN,JAMES△△△△△△△△△△△△197448473OFFICESERVICES
ARTHUR,MONICA△△△△△△△△△△Ø67531675LEGAL△△△△△△△△△
BAINES,CHARLES△△△△△△△△△189634798ACCOUNTING△△△△
BILKER,HAROLD△△△△△△△△△△348978885PRODUCTION△△△△
BONDS,ALLISON△△△△△△△△△△228561194QUALITYCONTROL
BROOKS,ELAINE△△△△△△△△△△177338959DEVELOPMENT△△△
BROOKS,JOHN△△△△△△△△△△△△12931221ØMARKETING△△△△△
```

Figure 12–9. First Few EMPLOYEE Defined Records as Delivered to an Action Program

The data definition we've just described is a simple one. It creates one defined file with one type of defined record; and that defined record accesses one conventional file only. Now we're ready to look at a slightly more involved example. Figure 12-10 illustrates a sample data definition for defined file EMPDEPS.

```
LINE NO. SEQ              SOURCE STATEMENT

   ØØ1         IDENTIFICATION DIVISION.
   ØØ2         PROGRAM-ID. EMP-AND-DEP.
   ØØ3         DATA DIVISION.
   ØØ4         FILE SECTION.
   ØØ5         FD EMPFILE.
   ØØ6         Ø1 EMPLOYEE-REC.
   ØØ7            Ø2 EMPL-NAME                 PIC X(21).
   ØØ9            Ø2 DEPT-NAME                 PIC X(21).
   Ø1Ø            Ø2 ENTRY                     PIC X(5).
   Ø11         Ø1 DEPENDENT-REC.
   Ø12            Ø2 EMPL-NAME                 PIC X(21).
   Ø13            Ø2 DEP-NAME                  PIC X(21).
   Ø15            Ø2 D-SSN                     PIC X(9).
   Ø16            Ø2 FILLER                    PIC X(12).
   Ø17         DEFINITION DIVISION.
   Ø18         DEFINED FILE EMPDEPS PASSWORD
   Ø19         DEFINED RECORD DEPENDENT
   Ø2Ø            FROM EMPLOYEE-REC
   Ø21            ALLOW ADD AND DELETE
   Ø22            IDENTIFIER EMP-NM FROM EMPL-NAME
   Ø24            ITEM DEPT-NAME
   Ø25         SUPPLEMENT DEPENDENT-TRAILER
   Ø26            FROM DEPENDENT-REC
   Ø27            POINTER IS EMP-NM
   Ø28            ITEM DEP-NM FROM DEP-NAME
   Ø29            ITEM D-SSN
```

Figure 12–10. Data Definition for Defined File EMPDEPS

More complex conventional file

In the identification division, the data definition name is EMP-AND-DEP. The data division describes the conventional file EMPFILE. In this instance EMPFILE contains not one but two record types, EMPLOYEE-REC and DEPENDENT-REC. For EMPLOYEE–REC three fields are described; and for DEPENDENT-REC, four. Note that the field EMPL-NAME appears in both records.

Allowing access with UNIQUE

In the definition division, the defined file name is EMPDEPS. The PASSWORD parameter indicates that UNIQUE users have access to this defined file by entering at the terminal the defined file name along with the OPEN command.

Describing the defined record

The defined record DEPENDENT is built from the conventional record EMPLOYEE-REC. It allows for addition and deletion of records and has two fields: EMP-NM, the identifier or mini-key for the defined record, and DEPT-NAME. Up to this point the definition division closely resembles the one illustrated in Figure 12-6.

Creating a supplement to the defined record

Here, however, the similarity ends. We have added a supplement to the defined record DEPENDENT. The supplement allows you to add fields to the defined record, in this case DEPENDENT, by drawing data from another conventional record in the same file or in a different file. Figure 12–11 shows how this happens.

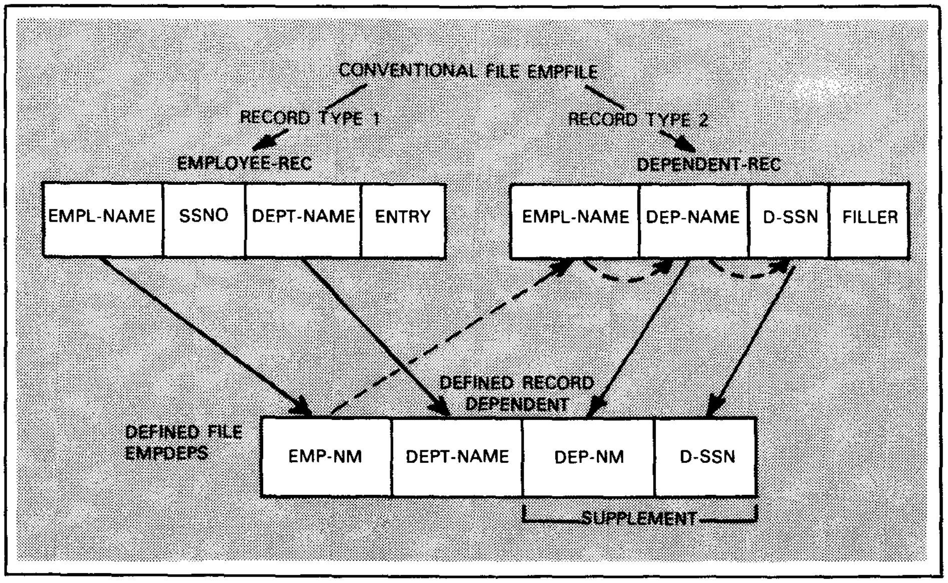
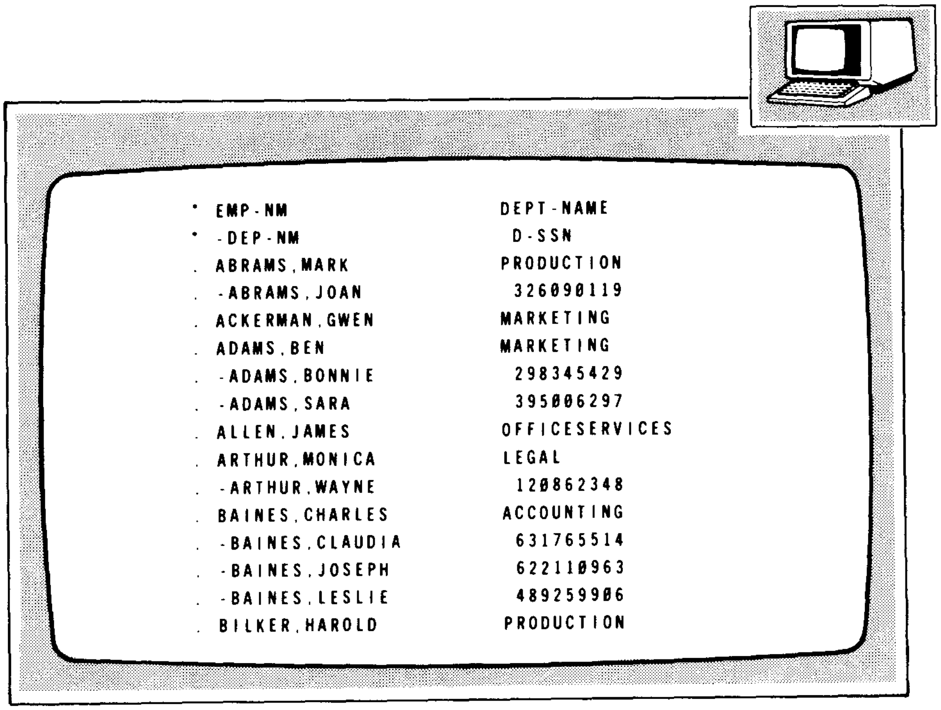


Figure 12–11. Creating a Defined Record with Supplement

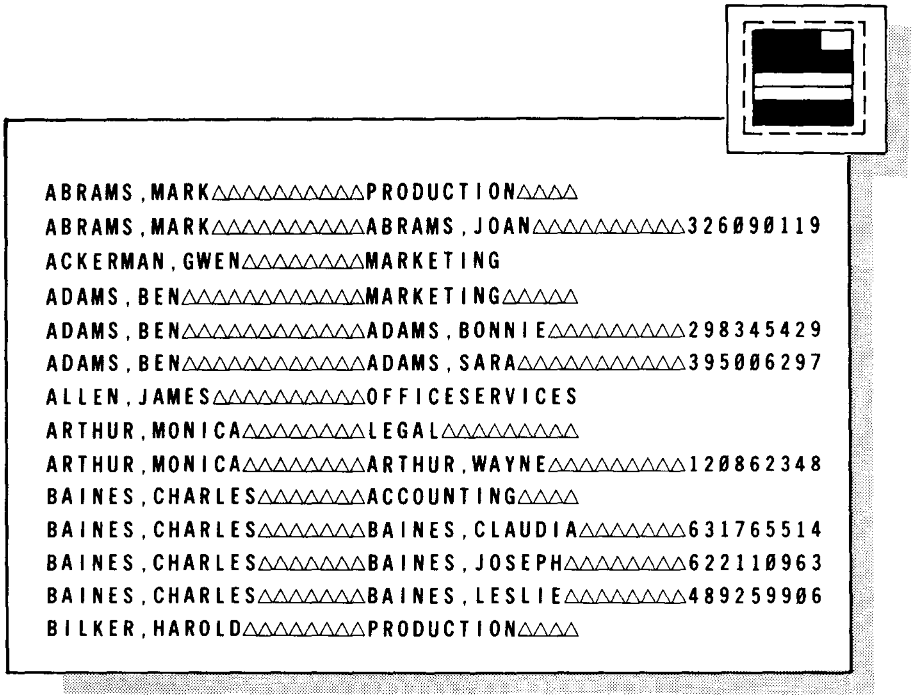
Figure 12–11 shows how we supplement the initial defined record DEPENDENT by adding the fields DEP-NM and D-SSN drawn from the conventional record DEPENDENT-REC. Note that the field EMPL-NAME links the two conventional records and allows us to access data in both records.

Figures 12-12 and 12-13, respectively, show the first few EMPDEPS records as listed in a UNIQUE transaction and as they are delivered to an action program.



```
* EMP-NM                  DEPT-NAME
* -DEP-NM                  D-SSN
. ABRAMS,MARK             PRODUCTION
.  -ABRAMS,JOAN            3260Ø0119
. ACKERMAN,GWEN           MARKETING
. ADAMS,BEN               MARKETING
.  -ADAMS,BONNIE           298345429
.  -ADAMS,SARA             395ØØ6297
. ALLEN,JAMES             OFFICESERVICES
. ARTHUR,MONICA           LEGAL
.  -ARTHUR,WAYNE           12Ø862348
. BAINES,CHARLES          ACCOUNTING
.  -BAINES,CLAUDIA         631765514
.  -BAINES,JOSEPH          62211Ø963
.  -BAINES,LESLIE          4892599Ø6
. BILKER,HAROLD           PRODUCTION
```

Figure 12-12. First Few DEPENDENT Records as Listed at a Terminal by UNIQUE



```
ABRAMS,MARK△△△△△△△△△△PRODUCTION△△△△
ABRAMS,MARK△△△△△△△△△△ABRAMS,JOAN△△△△△△△△△△3260Ø0119
ACKERMAN,GWEN△△△△△△△△MARKETING
ADAMS,BEN△△△△△△△△△△△△MARKETING△△△△△
ADAMS,BEN△△△△△△△△△△△△ADAMS,BONNIE△△△△△△△△△298345429
ADAMS,BEN△△△△△△△△△△△△ADAMS,SARA△△△△△△△△△△△395ØØ6297
ALLEN,JAMES△△△△△△△△△△OFFICESERVICES
ARTHUR,MONICA△△△△△△△△LEGAL△△△△△△△△△
ARTHUR,MONICA△△△△△△△△ARTHUR,WAYNE△△△△△△△△△12Ø862348
BAINES,CHARLES△△△△△△△ACCOUNTING△△△△
BAINES,CHARLES△△△△△△△BAINES,CLAUDIA△△△△△△△631765514
BAINES,CHARLES△△△△△△△BAINES,JOSEPH△△△△△△△△62211Ø963
BAINES,CHARLES△△△△△△△BAINES,LESLIE△△△△△△△△4892599Ø6
BILKER,HAROLD△△△△△△△△PRODUCTION△△△△
```

Figure 12-13. First Few DEPENDENT Records as Delivered to an Action Program

By creating a series of defined records and supplements in your data definition, you can achieve tremendous flexibility without ever altering your user files.

Creating hierarchical defined records

If your applications should require even more sophisticated file manipulation, the data definition also provides the capability of creating a defined file with a hierarchy of relationships among its defined records. You accomplish this by establishing parent-child relationships among defined records. For detailed information on how to set up these hierarchical relationships, consult the IMS data definition and UNIQUE user guide, UP-9209 (current version).

13. The IMS File Processing Package – UNIQUE

13.1. A LOOK AT UNIQUE

Definition

UNIQUE is a set of IMS-supplied action programs that allow you to access your files. With UNIQUE you can perform retrieval and update functions, as well as calculate statistics on your files.

Advantages

Since UNIQUE is a packaged set of programs, it has certain limitations. If you want to tailor IMS to your individual programming needs, you should write your own action programs. However, if you want to get online quickly and see how IMS operates, UNIQUE is an excellent way to do it.

UNIQUE illustrates very well the conversational nature of IMS. For every UNIQUE command entered at the terminal, the operator receives a response from IMS. In addition, UNIQUE furnishes a speedy means of displaying and updating files without doing any programming at all.

Reference manual

In this section we provide a brief overview of UNIQUE and its commands. For a detailed discussion of how to use UNIQUE, consult the IMS data definition and UNIQUE user guide, UP-9209 (current version).

13.2. USING UNIQUE

File support

A UNIQUE conversation is a series of commands (input) and responses (output) dealing with a particular defined file. Whenever we talk about files in conjunction with UNIQUE, we're talking about defined files, since you can use UNIQUE only with defined files. Figure 13-1 illustrates terminal operator access to defined files.

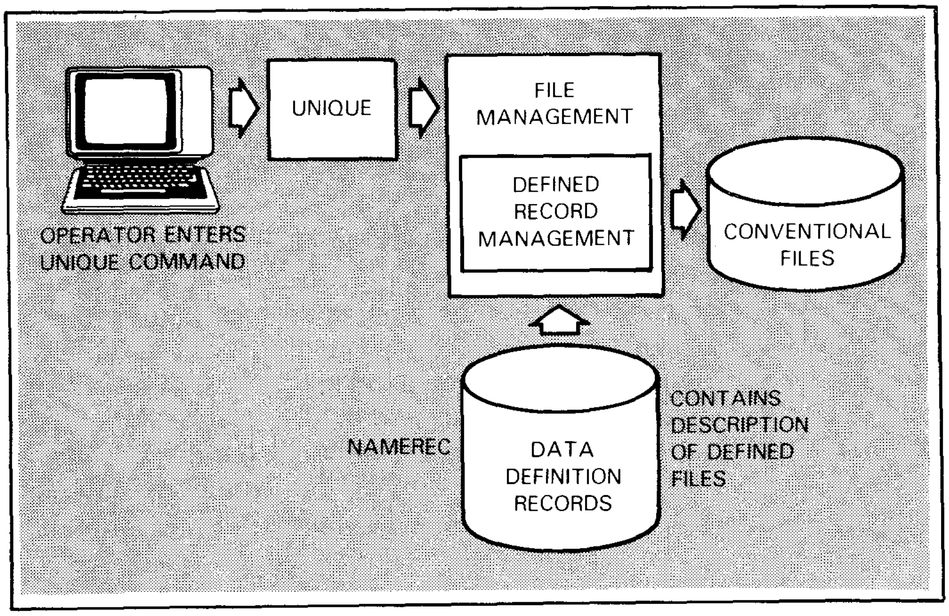

UNIQUE
FILE MANAGEMENT
DEFINED RECORD MANAGEMENT
OPERATOR ENTERS UNIQUE COMMAND
CONVENTIONAL FILES
NAMEREC
DATA DEFINITION RECORDS
CONTAINS DESCRIPTION OF DEFINED FILES

Figure 13-1. UNIQUE Uses Defined Files

Accessing a defined file

To access a defined file with UNIQUE, you must know the password for that file. A password, defined in the data definition, is the same as the defined file name and all configured terminals can use it.

Defining passwords

Passwords can also be defined by the IMS administrator using the NAMEREC file utility. The administrator can restrict a password to specific terminals and can change passwords, even voiding a password defined in the data definition. The NAMEREC file utility is described in detail in the IMS system support functions user guide, UP-8364 (current version).

13.3. UNIQUE COMMANDS

Communicating with UNIQUE

Once you know the password for a particular defined file, you communicate with UNIQUE through a series of commands entered at the terminal. These commands tell IMS what file activities to perform. The UNIQUE commands are:

OPEN	Initiates the UNIQUE transaction and opens a dialog with a defined file identified by its password.
CLOSE	Ends the UNIQUE transaction.
DISPLAY	Displays the contents of a record.
NEXT	Selects the next identifier from the most recent DISPLAY, DELETE, ADD, or CHANGE command and performs the same function.
DELETE	Displays a record, which you can then delete by entering the OK command.
OK	Complete an update function – DELETE, ADD, or CHANGE.
CANCEL	Cancels an update command – DELETE, ADD, or CHANGE.
ADD	Initiates a series of inputs and responses that result in adding a record to the defined file.
CHANGE	Initiates a series of inputs and responses that result in changing a defined record.
LIST	Lists all or selected portions of a file and performs statistical functions.
MORE	Displays the next screenful of data from the previous LIST or DETAIL command.
DETAIL	Gives a secondary listing without interrupting LIST command processing.
SHOW	Displays the format of records in the defined file, the most recent LIST and DETAIL commands, and any outstanding DISPLAY, DELETE, ADD, or CHANGE command.

NOTE:

These commands may be changed by using the LANGUAGE section of the IMS configurator. Listed are the default UNIQUE lexicon commands.

13.4. UNIQUE APPLICATION

OPEN and CLOSE commands

The first OPEN command you enter starts the UNIQUE transaction and opens a dialog with a defined file. Additional OPEN commands automatically close the dialog with the current defined file and open dialogs with other defined files. The UNIQUE transaction ends when you enter a CLOSE command.

Command formats

You can enter UNIQUE commands at display or hard copy terminals. The format for most UNIQUE commands is the same, regardless of the terminal you're using. Only the ADD and CHANGE commands have different formats. You can enter UNIQUE commands in uppercase or lower case letters; but UNIQUE output is always displayed in uppercase.

Defining terms

When output appears at the terminal, fields in the defined record are displayed as column headers. The first field in the defined record is called the identifier. The other fields are called ITEM-NAMES. Let's say there is a defined file, EMPLOYE, which contains PERSONAL records. These records provide general information about each employee in company ABC. Each PERSONAL record has fields called EMP-NAME, ADDRESS, CITY, ST, ZIP, PHONE, and S-S-NO. Figures 13-2 and 13-3 show this correspondence.

Displaying a record

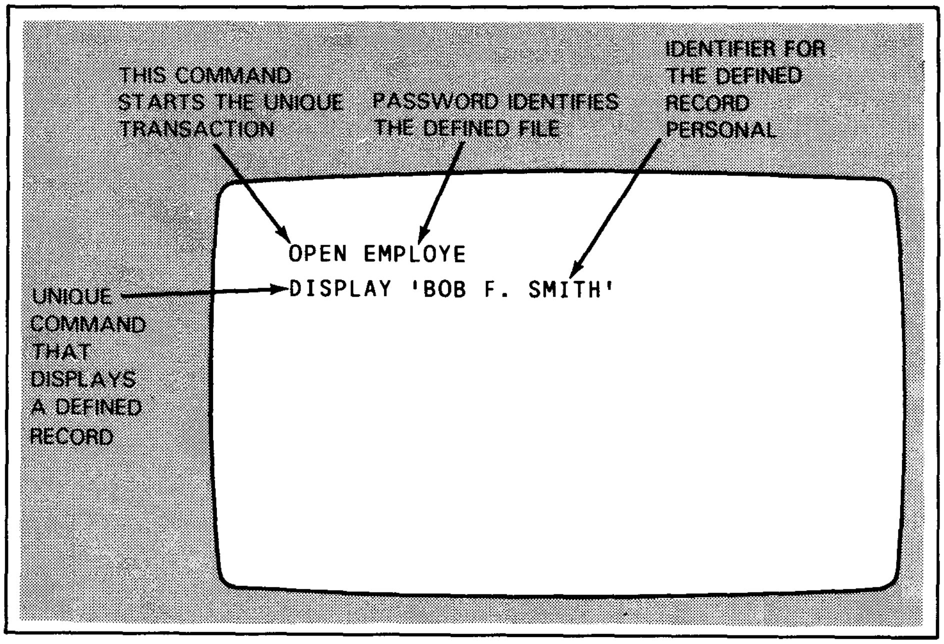
The OPEN command, followed by the password EMPLOYE (in this case, the password is the same as the defined file name), signals the start of the UNIQUE dialog with the EMPLOYE file. The operator then identifies the operation to be performed (DISPLAY) and the record to be displayed (BOB F. SMITH) which is the identifier name. You always use the identifier name (the first field in the record) to display or update a record.

Screen format

Figure 13-3 shows that the fields in the PERSONAL record are all displayed as column headers and the data about Bob F. Smith is filled in under those headers. Bob F. Smith is the identifer for the record and the other fields are item-names.

Reference manual

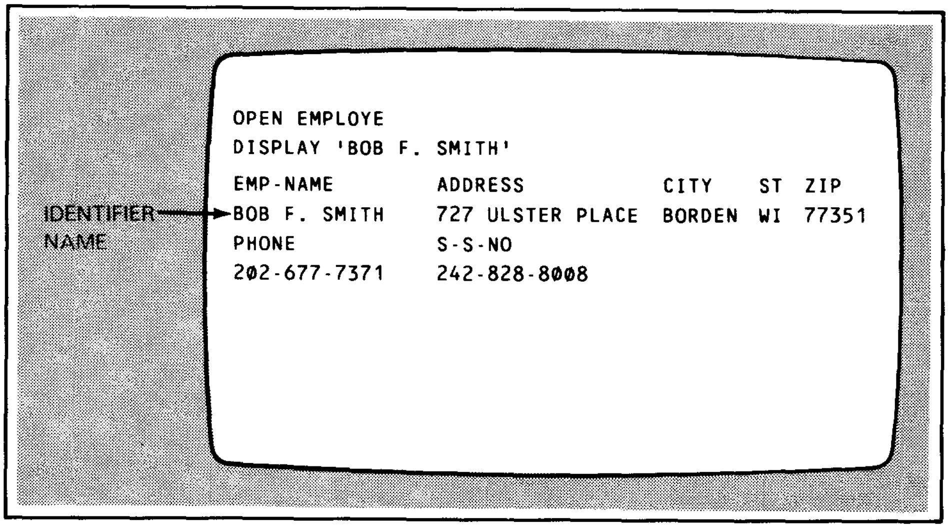
For an in-depth discussion of the use of these and other UNIQUE commands, see the IMS data definition and UNIQUE user guide, UP-9209 (current version).



```
OPEN EMPLOYE
DISPLAY 'BOB F. SMITH'
```

THIS COMMAND STARTS THE UNIQUE TRANSACTION
PASSWORD IDENTIFIES THE DEFINED FILE
IDENTIFIER FOR THE DEFINED RECORD PERSONAL
UNIQUE COMMAND THAT DISPLAYS A DEFINED RECORD

Figure 13–2. An Input Screen to be Processed by UNIQUE



```
OPEN EMPLOYE
DISPLAY 'BOB F. SMITH'
EMP-NAME        ADDRESS            CITY    ST  ZIP
BOB F. SMITH    727 ULSTER PLACE   BORDEN  WI  77351
PHONE           S-S-NO
2Ø2-677-7371    242-828-8ØØ8
```

Figure 13–3. Output Screen Generated by UNIQUE

Glossary

A

action
The basic unit of work in IMS. An action consists of an input message, the processing of that message by one or more action programs, and the output of at least one message.

action control table
Table that describes to IMS the action program or programs that process a specific action. Contains information on the files used in processing the action and the sizes of the largest input and output messages.

action program
A program that works with IMS to access, retrieve, add, change, or delete data from your files. One or more action programs perform an action.

action program area, snapshot dump
Area of snapshot dump that contains the executing load module as it existed in main storage when the program terminated.

action program routing
In distributed data processing, one way that IMS can route a transaction to a remote system. The terminal operator enters a transaction code that initiates a transaction at the local IMS system. The COBOL or basic assembly language (BAL) action program processing this transaction issues an ACTIVATE function call to IMS, which creates a message that initiates another IMS transaction at the remote site. The remote IMS processes the transaction and sends a message back to the originating action program or its successor program.

action scheduling
A component of applications management that schedules an action for each input message by allocating main storage and other resources needed by that action. Also handles termination of action programs.

ACTION section, configurator
Section of configurator input that describes the actions used in the configuration.

activation record
A main storage interface area that contains and passes control information and data between the action program and IMS. Can contain the continuity data area (CDA), defined record area (DRA), input message area (IMA), output message area (OMA), program information block (PIB), and work area (WA).

activation record area, snapshot dump
Area of snapshot dump that shows the contents of the activation record when the program terminated.

ADD command, UNIQUE
UNIQUE command that initiates a series of inputs and responses that change a defined record.

after-image
The image of a record after updating occurs.

applications management
A major component of IMS that controls the execution of action programs. Includes action scheduling and file management.

assembly, step in IMSCONF
Step in IMSCONF jproc that takes the configurator output, plus information in your data files, and converts it into machine-readable code.

AUDCONF (audit and continuity data file)
IMS internal file that is the single-thread counterpart of the AUDFILE and CONDATA files for multithread IMS.

AUDFILE (audit file)
IMS internal file that handles online recovery and contains IMS control information recorded when a transaction terminates.

automatic status messages
Messages automatically generated by multithread IMS to notify the terminal operator of the status of the last input message.

B

backward recovery
A type of offline recovery that uses before-images recorded on the trace file to restore data files to their original state or to the state they were in at a specified date and time.

BATCH parameter, configuration
Parameter that enables you to designate batch processing modules in your configuration.

batch processing
A method of processing groups of transactions in card format instead of through a display terminal. You can process batch transactions online or offline without an active terminal network.

batch processor
An optional component of IMS that enables you to enter input for IMS transactions on cards instead of interactively and to direct output to a printer or printer file.

batch transaction
A transaction entered in card format instead of interactively.

before-image
The image of a record before any updating occurs.

C

CALL function (function call)
A request to IMS for input/output or other services. In COBOL action programs, a CALL function is in the form of a CALL statement. In BAL action programs, a CALL function is in the form of a CALL or ZG#CALL macroinstruction. RPG II action programs do not use CALL functions, but the compiler converts normal RPG II file requests into function calls.

CANCEL command, UNIQUE
UNIQUE command that cancels an update (DELETE, ADD, or CHANGE) command.

CCA linkage, step in IMSCONF
Step in IMSCONF jproc that links the communications control area object module to the configurator modules to create the IMS configurator program.

CHTBL, IMS transaction code
Transaction code that lets you make temporary changes to the internal tables generated by your entries in the ACTION and PROGRAM sections of the configurator.

CLOSE command, UNIQUE
UNIQUE command that ends a UNIQUE transaction.

COMMCT
Phase of system generation that generates the communications network and an ICAM load module that supports online IMS.

communications adapter
Hardware device that controls and coordinates the message flow between ICAM and the terminals.

CONDATA (continuity data file)
IMS internal file required for multithread IMS when you use UNIQUE, when your action programs pass data using the continuity data area, or when you use the terminal output message file.

configuration
The process where the IMSCONF jproc executes the configurator program, analyzes and processes input to the configurator, and generates records for the named record file.

configurator
Component of pre-online processing that, through the job control procedure IMSCONF, enables you to create an online IMS that performs the functions you want.

configurator tables
Tables created by the configurator and stored on the named record (NAMEREC) file. These tables are the transaction code table, action control table, and program control table.

continuity data area (CDA)
Area in the activation record that holds data an action program wants to save and pass to the action program that follows it.

continuous output
An output message transmitted in a continuous series of sections without intervening input messages. The continuous output message must be the last message transmitted by the action program.

D

data definition
A description of an IMS defined file.

data definition processor
An IMS utility program that creates a data definition record in the named record file and produces a printed description of the defined file and a diagnostic listing.

data division, data definition
Section of a data definition that describes the conventional files from which the IMS defined file is extracted.

data management, OS/3
The interface between IMS file management and your data files.

data manipulation language (DML)
A special data base access language that resembles COBOL and is used in action programs to communicate with the data base management system (DMS).

dedicated network, ICAM
A communications network in which all lines and terminals are dedicated to IMS for the length of the session.

defined file
A logical file that IMS builds from records in your existing data files.

defined record
A record that redefines records in your existing files.

defined record area (DRA)
Area in the activation record used by defined record management when action programs use defined records.

defined record management
The IMS component that handles requests from UNIQUE and action programs for defined records.

definition division, data definition
Section of a data definition that describes the structure of the defined file.

delayed internal succession
Termination type in which one action program processes an input message and then queues an input message to a successor action program without sending that message to the terminal.

DELETE command, UNIQUE
UNIQUE command that displays a record, which you can then delete by entering the OK command.

delivery code
ICAM's response to a continuous output message informing IMS whether or not the message was successfully delivered to the terminal or printer.

DETAIL command, UNIQUE
UNIQUE command that gives a secondary listing without interrupting LIST command processing.

detailed status code
A two-byte field in the program information block that contains specific information about why a function call was unsuccessful. Supplements the status code by giving more detailed information.

device media control language (DMCL)
Language that describes the physical structure of a data base, identifying the location of the data in the data base.

dialog transaction
A transaction that consists of a sequence of two or more actions.

directory routing
In distributed data processing, one way that IMS can route a transaction to a remote system. The terminal operator enters a transaction code that identifies a transaction for processing at a particular remote site. IMS sends the transaction code to the remote IMS, which processes the transaction and sends a message back to the terminal operator.

disk buffer file
A file that holds holds input messages for ICAM to reduce the amount of main storage taken up by network buffers.

DISPLAY command, UNIQUE
UNIQUE command that displays the contents of a record.

distributed data processing
System that allows two independent computer systems to communicate with one another. They are linked together through telecommunications so that each can use the other's capabilities and data.

DITBL, IMS transaction code
Transaction code that displays the contents of the fields in your action and program configurator tables.

DLMSG, IMS transaction code
Transaction code that displays the last effective output message sent to your terminal from an action program or UNIQUE.

DLOAD, IMS transaction code
Transaction code that loads a user-written UTS program from the IMS load library downline to a UTS 40 or UTS terminal.

DRCRDMGT section, configurator
Section of configurator input that defines the level at which you intend to use defined record management.

dynamic terminals
Terminals that are attached to IMS only while you are using them.

E

edit table generator
An offline IMS utility that uses edit tables to convert freeform input entered by terminal operators into fixed formats required by the action program. IMS uses these edit tables to check the input for data type, value ranges, and presence of required fields.

external succession
Termination type in which both the first and the successor action programs process an input message and produce an output message.

F

file management
A component of applications management that provides all action program I/O functions and provides access to your data files through OS/3 data management. It interfaces with IMS internal files and your data files.

FILE section, configurator
Section of configurator input that describes each user data file to be accessed by IMS.

forward recovery
A type of offline recovery that uses after-images recorded on the trace file for recovery purposes when a portion of your data files is destroyed.

function key
A terminal key that can be used as a transaction code when your terminal is not processing a transaction or as a response to an output message when you are using external succession.

G

GENERAL section, configurator
Section of configurator input that describes general characteristics of the online IMS system.

global network, ICAM
A communications network that supports IMS and other communications programs at the same time.

H

headers section, snapshot dump
Area of snapshot dump that gives general information about which program was running when the snapshot dump occurred and what caused it.

I

identification division
Section of a data definition which assigns a unique name to the data definition.

identifier
A portion of the record key of the conventional record from which the defined record is created. It is used to locate the data in your conventional files.

immediate internal succession
Termination type in which one action program processes the message and schedules another program to continue the processing. When the second program finishes processing, the program generates an output message and terminates.

IMSCONF job control procedure
An IMS-supplied job control procedure that generates and executes an IMS configurator program creating an online IMS.

information management system (IMS)
An interactive transaction processing system that enables you to easily access and modify your data files.

initialization, step in IMSCONF
Step in IMSCONF jproc that allocates and initializes IMS internal files.

input message area (IMA)
Area in the activation record that receives input from the terminal.

integrated communications access method (ICAM)
Communications software that handles all input and output between your terminals and IMS.

internal message control
An IMS component that processes input and output messages, executes terminal commands, and controls message editing.

INVOKE statement, in COBOL/DML action programs
A statement that identifies the schema, subschema, and device media control language modules describing the physical and logical structure of a data base.

L

LANGUAGE section ,configurator
Section of configurator input that creates a UNIQUE lexicon record in the named record file. This lexicon record redefines UNIQUE commands.

LDPFILE (fast loader file)
IMS internal file required when you specify FASTLOAD=YES. The first time they are called by a transaction, action programs are copied into the LDPFILE from the action program load library (LOAD). After that, the action programs are loaded from the LDPFILE.

library, IMS
A collection of source, object, and load modules that are part of your IMS software.

line disconnect
A feature that allows an action program to disconnect a single-station dial-in line, enabling another terminal to use the same dial-in line.

LIST command, UNIQUE
UNIQUE command that lists all or selected portions of a file and performs statistical functions.

LOCAP section, configurator
Section of configurator input that defines a local application program to a remote computer system.

lock-for-transaction

Record locking capability that locks records until the end of an action and audits updates. A before-image is written to the audit file and, if the transaction terminates abnormally, the original image of the record is restored.

lock-for-update

Record locking capability that locks records only for the duration of the update. Updates are not audited and are not rolled back if the transaction terminates normally.

M

master terminal commands

Commands that control and monitor the entire IMS network. You can enter these commands only from the master terminal or, when designated, the system console or master workstation.

MORE command, UNIQUE

UNIQUE command that displays the next screenful of data from the previous LIST or DETAIL command.

multiple messages

Type of action program output where the program produces more than one output message when the output is greater than screen capacity.

multithread processing

The processing of actions concurrently for different transactions.

N

NAMEREC file (named record file)

An internal IMS file that contains all the tables and records needed by online IMS. These tables include configuration tables, data definition records, edit tables, and password definition tables.

NAMEREC utility

Component of pre-online processing that initializes the named record (NAMEREC) file.

network definition macroinstructions

A set of macroinstructions that define the type of ICAM support required, the network of lines and terminals, and the input/output requirements.

NETWORK section, configurator

Section of configurator input that defines the ICAM network to support your online IMS and provides additional information about your IMS system.

NEXT command, UNIQUE

UNIQUE command that selects the next identifier from the most recent DISPLAY, DELETE, ADD, or CHANGE command and performs the same function.

normal termination

Termination type that tells IMS that processing is complete after a single action program or a series of action programs have finished executing.

O

offline processing

Processing that consists of operations that recover damaged or inconsistent files. The two recovery programs available with offline processing are the offline recovery utility and the tape copy program.

offline recovery

Type of recovery that restores damaged or inaccurate files after online processing has terminated.

offline recovery utility

Program that reconstructs your files when online recovery fails or cannot correct the problem.

OK command, UNIQUE

UNIQUE command that completes an update function - DELETE, ADD, or CHANGE.

online module linkage, step in IMSCONF

Step in IMSCONF jproc that takes the output of the configurator and assembly operations, links it with IMS library modules into an executable IMS load module, and stores it in a load library.

online processing
Processing that is transparent and functions internally to enable you to process transactions. The five components of online processing are start-up modules, applications management, internal message control, shutdown modules, and action programs.

online recovery
Type of recovery that restores damaged or inaccurate files during online processing.

OPEN command, UNIQUE
UNIQUE command that initiates a UNIQUE transaction and opens a dialog with a defined file identified by its password.

operator routing
In distributed data processing, one way that IMS can route a transaction to a remote system. The terminal operator prefixes the transaction code with a special character that identifies the transaction for remote processing. The remote IMS processes the transaction and sends a message back to the terminal operator.

OPTIONS section, configurator
Section of configurator input that defines optional IMS features to be included in the configuration.

output-for-input queueing
Output message that starts an IMS transaction at another terminal.

output message area (OMA)
Area that holds the output message that the action program generates.

P

PASSWORD parameter, in data definition
Parameter that protects defined files by limiting access to them.

password
A name associated with a particular IMS defined file or subfile for data security purposes.

pre-online processing
Processing that prepares and configures IMS to define your online IMS environment.

program control table
Table that tells IMS about the specific program or programs processing an action. It contains data about whether the program is serially reusable, shared, or reentrant, and whether or not the program permanently resides in main storage.

program information block
Area in the activation record that passes control information between IMS and the action program during program processing.

PROGRAM section, configurator
Section of configurator input that describes the action programs to be included in the IMS configuration.

Q

quick recovery
A type of offline recovery that uses before-images recorded on the trace file to roll back transactions when there is a system failure or an emergency IMS termination.

R

random function calls
Call from an action program that alerts IMS file management to handle a random access file request. Random function calls are GET, GETUP, PUT, INSERT, and DELETE.

recovery
A method of protecting your data files so that, if something goes wrong during updates, your files can be restored.

reentrant code
A type of coding that enables an action program to be used concurrently by two or more users. Programs written in reentrant code are completely shareable. Because more than one action program is using the program, no parts of the program can change during action programming.

register section, snapshot dump
Area of snapshot dump that contains both IMS and action program registers or just IMS registers, depending on what caused the dump.

RESGEN, phase of system generation

Phase of system generation that allows you to generate your own system resident (SYSRES) disk pack.

resident ICAM

An ICAM load module that remains in main storage at all times.

rollback

The process of restoring the original image of a record when a transaction terminates abnormally.

S

schema

The global view of a data base's logical structure. It is a total picture of the physical and logical structure of the data base.

screen format services

A separate software package that allows users to create formatted screens. IMS interfaces with screen format services to allow action programs to use screen formats for input and output messages.

serially reusable code

A type of coding that allows only one user at a time to access the action program. Another terminal requesting the same program must wait until the first user is finished.

sequential function calls

Call from an action program that alerts IMS management to handle a sequential access file request. Sequential function calls fro random files are SETL, SETK, GET, and ESETL. Sequential functions for sequential files are GET and PUT.

shared code

A type of coding that enables an action program to be used concurrently by two or more users. Programs written in shared code are only partially shareable, and only COBOL action programs can use shared code.

SHOW command, UNIQUE

UNIQUE command that displays the format of records in the defined file, the most recent LIST and DETAIL commands, and any outstanding DISPLAY, DELETE, ADD, or CHANGE command.

shutdown, IMS

The process of ending an IMS session.

simple transaction

A transaction that consists of a single action.

single message output

A type of output in which an action program produces only one output message, which is sent to the terminal when the program finishes processing.

single-thread processing

The processing of one action at a time.

snapshot dump

A picture of what goes on internally when your action program is processing. The snapshot dump shows the program itself, the areas of the activation record used by the action program, and what, went wrong with the program.

standard terminal commands

Commands that help you control message transmission and terminal operations and testing. They can be entered from any terminal and, if console support is configured, from the console or master workstation.

start-up, IMS

The process of bringing IMS online.

STATFIL (statistical data file)

IMS internal file that contains statistical data recorded during online processing.

static terminals

Terminals that are attached to IMS for the entire session.

status code

A two-byte field in the program information block that defines whether or not a function call was successfully carried out and if not, why.

subschema

A logical view of a segment of the data base. It is a portion of the schema useable by a specific application.

successor program

An action program that follows another action program and resumes processing of a transaction.

supervisor
A program that controls the activity of other programs operating within the system.

SUPGEN, phase of system generation
Phase of system generation in which you specify the number of communication lines, priority levels for IMS tasks, number of job slots required, file locking capabilities, and DTF, CDM, or mixed data management mode.

switched message
A message sent to a terminal as a result of an input message from another terminal.

SWTCH, IMS transaction code
Transaction code that sends a message to another terminal or terminals, including the system console.

system generation (SYSGEN)
Phase in online generation process which creates a supervisor that supports IMS. The three phases of SYSGEN are SUPGEN, RESGEN, and COMMCT.

T

tape copy program
Program that recovers a tape trace file that is left open as the result of a system failure. This program then produces a new trace file that you can use for offline recovery.

terminal
Any point in a system or communications network where data can enter or leave.

terminal control table, snapshot dump
Area of snapshot dump that contains data related to the terminal that initiated the action program.

TERMINAL section, configurator
Section of configurator input that further defines the terminals already included in your communications network definition.

test mode, terminal state
Mode in which your terminal simulates transaction processing but no physical alteration of your data files occurs.

thread control block, snapshot dump
Area of snapshot dump that contains data used to control the IMS environment.

TIMEOUTS section, configurator
Section of configurator input that defines the various time-out values.

TOMFILE (terminal output message file)
An internal IMS file on which IMS writes the output message generated for each terminal at rollback points and at transaction termination, retaining only one message per terminal in the file.

TRANSACT section, configurator
Section of configurator input that supplies transaction code information to the configurator.

transaction
One action or a sequence of actions that complete a desired function.

transaction codes
1- to 8-character messages that identify to IMS the first action program in a transaction.

transaction code table
Table that lists all the transaction codes the IMS configuration supports. It also provides IMS with the name of the first program that processes the transaction.

transaction facility, IMS
Capability to process IMS transactions at a remote computer.

transaction structures
The two types of transactions, simple and dialog.

transient ICAM
An ICAM load module that occupies main storage only when required, thereby freeing up main storage for other programs.

TRCFILE (trace file)
An internal IMS file used with offline recovery to recover data files when IMS is not online.

U

uniform inquiry update element (UNIQUE)

A set of IMS-supplied action programs that perform a variety of retrieval and updating functions for defined files.

UTS downline load feature

Using an action program to load other programs from an IMS library into UTS 40 or 400 terminal main storage.

W

work area (WA)

Area in the activation record used by COBOL and basic assembly language (BAL) action programs to hold logical records during file processing and as a working storage area.

working-storage section, COBOL/DML action program

Section of a COBOL/DML action program in which a data base management communications area (DMCA) is set up for the exchange of vital information between IMS and DMS.

Z

ZSTAT, IMS transaction code

Transaction code that generates statistics about your files, programs, transactions, and terminals.

Index

SPERRY UNIVAC

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

(Document Title)

(Document No.) *(Revision No.)* *(Update No.)*

Comments:

From:

(Name of User)

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

Cut along line.

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY UNIVAC

ATTN.: SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

CUT

FOLD